Coding Places

Uneven Globalization of Software Work in Rio de Janeiro, Brazil

by

Yuri Vladimirovich Takhteyev

B.S. (Stanford University) 2000
M.S. (Stanford University) 2000

A dissertation submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Information Management and Systems

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor AnnaLee Saxenian, Chair
Professor Paul Duguid
Professor Peter Evans
Professor Coye Cheshire

Spring 2009

Coding Places: Uneven Globalization of Software Work in Rio de Janeiro, Brazil

Yuri Vladimirovich Takhteyev

Abstract

Coding Places: Uneven Globalization of Software Work in Rio de Janeiro, Brazil

by

Yuri Vladimirovich Takhteyev

Doctor of Philosophy in Information Management and Systems

University of California, Berkeley

Professor AnnaLee Saxenian, Chair

The dissertation looks at the practice of software development in Rio de Janeiro, Brazil — a place at the periphery of the global world of software development. Software development is often conceptualized as a placeless field of endeavor. At the same time software work is highly concentrated, with "right" and "wrong" places to do it. Ethnographic data from a "wrong place" helps challenge some of the common approaches to the globalization of work and knowledge. My account shows problems with the trivialized notions of the "flattening" of the world, demonstrating not only that place continues to matter, but also that globalization can strengthen some of the existing asymmetries. On the other hand, I document the extent and importance of transnational connections in modern work, uneven as such connections may be.

I look at how cultural identities, geographic distances, differences of language,

national boundaries and geopolitical tensions are negotiated in the face of increasingly "global" knowledge and technology, and how "global" technology, practices and culture are made to work in specific "local" places. I present globalization as an active process and as comprised of numerous individual globalization projects. Drawing on contemporary ethnographic data and historical accounts, I show how individuals build alliances in personal pursuits of globalization, stressing the need to understand such pursuits through a combination of cultural and economic perspectives. I show how such alliances transform local contexts, first establishing landing strips that enable the arrival of additional elements of remote practices, then gradually synchronizing local contexts with remote ones. I discuss the challenges faced by such alliances, stressing the partial and uneven state of globalization, and the combination of isolation and hyper-connectedness that sometimes emerges in the process.

The dissertation is based on one year of fieldwork in Rio de Janeiro, which included participant observation of software projects and semi-structured interviews with around one hundred people. One of the cases explored in depth is Lua, a programming language developed in Rio de Janeiro. The theoretical framework developed in the dissertation draws on the literature on communities of practice, ethnography of work, and science and technologies studies.

Table of Contents

Acknowledgements

This dissertation would not have been possible without the many people who have in various forms provided support, guidance and inspiration. A brief mention in the acknowledgements cannot repay all the debts nor is it possible to acknowledge everyone who has helped me complete this work.

Paul Duguid's *Social Life of Information* introduced me to many of the questions that this dissertation addresses even before I was aware of the existence of the School of Information. Paul's arrival the School in 2004 has been a true blessing for me. While Paul's own work has led me to expect the intellectual stimulation that I have found in my many conversations with him, it did not prepare me for his generosity and his dedication to the success of this project. Working with Paul has been an amazing experience.

I am also tremendously indebted to the other members of my dissertation committee, who have all inspired me with their own work and have put much time into helping me improve the dissertation. Anno Saxenian's work on transnational connections has set an example for me from my early days at the School of Information and has encouraged me to do international research. Peter Evans's work introduced me to the history of Brazilian IT policy. His continuous insistence on hearing "the point" of my work has helped me sharpen my arguments. Coye Cheshire has helped me think about the

relative strengths and weaknesses of the methods used in my work.

Peter Lyman, under whose supervision I started my program at Berkeley, must be mentioned alongside my committee. I am grateful to Peter for convincing me to come to Berkeley, for introducing me to ethnography as a research method, and for securing funding for my research. He is and will be greatly missed.

Many other Berkeley faculty members have contributed to my learning at Berkeley. I particularly want to thank Jean Lave, Suzanne Scotchmer, Michael Buckland, Nancy Van House, Ray Larson, Ted Egan, and Raka Ray for introducing me to many new ideas and providing suggestions for my work.

Luisa Farah Schwartzman has seen more revisions of this dissertation than anyone else (with the possible exception of Paul Duguid). She has helped me reduce the amount of technical jargon in my writing, and to increase the number of definite articles. Having a native speaker of Portuguese around and ready to help decipher the most enigmatic passages in my recordings undoubtedly qualifies as an unfair advantage. Working on our dissertations at the same time, we gradually found more and more intersections in our bibliography, despite the seemingly unrelated topics. We have jokingly argued about who stole whose ideas. I may never be able to fully appreciate Luisa's impact on my work.

Over one hundred people have generously volunteered their time to tell me about their lives and work. This dissertation would not be possible without their courage to share their stories with a stranger. Many of those people also helped me feel at home in a new city. Some have become friends. Many of my interviewees have volunteered to read

chapters of this dissertation, finding factual errors, misinterpretations, inconsistencies, and often simply grammatical mistakes. I am particularly grateful to “Rodrigo” for his willingness to talk openly about the many challenges facing his project, for introducing me to many of the people on whose stories this dissertation relies, and for tolerating my straddling of fieldwork and friendship. I thank Roberto Ierusalimschy and Luiz Henrique Figueiredo for our discussions about the past, present and future of Lua and for their comments on the dissertation. I thank the members of the Kepler team and Alta’s developers for tolerating a resident ethnographer in their midst.

Analysis of the interviews would be much harder without access to transcription. I have done just enough transcription myself to know how difficult this work is. I thank Siobhan Hayes, Eva do Rego Barros, Rosa Paiva, Eliodora Besser, Patricia Martinez Alzueta, and Mariana Timponi for their work. I thank Marcelo Besser and LoGoS Traduções e Consultoria for organizing the process.

Conversations with Brazilian scholars have helped me better understand the local context and Brazil’s history. I am particularly thankful to Paulo Tigre, Ivan da Costa Marques, Sidney Oliveira de Castro, Henrique Cukierman, Antonio Botelho, Nelson Senra, and Simon Schwartzman. I thank the Institute of Economics of Universidade Federal do Rio de Janeiro for hosting me in 2005.

Fellow students at the School of Information have helped make my time here enjoyable and productive. The diversity of their interests has introduced me to a variety of ways of thinking about information technology, compensating for the narrow

specialization inherent in a doctoral program. I thank Jens Grossklags, Paul Laskowski, Joseph Hall, danah boyd, and Mahad Ibrahim for helping me stay on track. They have been a great cohort. I thank Dan Perkel, Megan Finn, Ryan Shaw, Christo Sims, Rajesh Veeraraghavan, and Bob Bell, for attending numerous practice talks, providing suggestions, and being a great group to be around. I thank Arthur Law and Sean Savage for helping me remember that there is life outside the doctoral program and that San Francisco is a wonderful city.

I thank my family for their support over the years. My parents have helped lay the foundation of my own globalization projects. My father's global dreams and his ability to imagine his children's future education abroad even before the fall of the Berlin Wall has helped bring me from Vladivostok to California. He introduced me to my first computer and I often remember our long conversations about the nature of information nearly two decades ago. My mother has contributed greatly to my international adventure by helping me develop a passion for foreign languages.

This dissertation is based upon work supported by the National Science Foundation under Grant No. SES-0724707, by the Berkeley Fellowship, and by Yahoo! Research Key Technical Challenges Grant.

To my parents, who laid the foundation for my global projects,

and to Luisa, who has accompanied me.

Introduction

Why would you come from California to Rio de Janeiro to study software developers? The question was asked in a friendly tone, with just a touch of suspicion. It would not send blood rushing through my veins if not for the place where it was asked. I was stooping behind a small window, in the midst of explaining to a US consular officer why a Russian citizen born in Vladivostok would be seeking an American visa in Rio de Janeiro, nearly at the exact opposite side of the world from where I was supposed to be applying for it. I was in the wrong place, and a good explanation was due, lest my personal world would suddenly become far from flat. Saying that I had come to Brazil to study *software developers* was a sure way to raise eyebrows further. Indeed, why would anyone come from California to Rio de Janeiro to study software developers?[1]

I hope to show in this dissertation that we have much to gain from looking at

1 In this dissertation, I use the term "software developers" to refer to people who create software and whose role in its creation requires some understanding of the inner working of this software. Thus, I exclude from this definition people whose involvement in creation of software is limited to purely managerial roles or auxiliary functions (e.g., secretaries in software companies). Apart from that, however, I use the term inclusively, considering that people who create software may do so for a number of reason (money, fun), may do so in different organizational settings (individually, as employees of firms, as members of open source projects, as university researchers), may have different positions in organizations ("assistant developer," "system analyst," "software architect," "associate professor"), may have different types of formal training (if any), and can be judged as having different levels of skill ("newbies," "hackers," "gurus"). No single English or Portuguese term comfortably covers this range, and I use the term "software developers" with realization that some of the readers may feel that certain other terms apply much better to some of the specific people. See chapter 2.2 for additional discussion of the different terms used by software developers and my rationale for using the term "software developers."

software development in such an unlikely place. The experience of software developers working in a place like Rio de Janeiro teaches us a lot about *place* and its persisting importance in today's "knowledge economy." Software development presents us with a quintessential case of "knowledge work," putting in clear view some of the contradictions inherent in our understanding of this concept. Software development is commonly seen as a global profession, freed from the constraints of geography by the immaterial nature of its inputs and outputs (text in, text out), rendered mobile by the new information technologies that software developers have mastered like nobody else, having been the ones who designed them.[2] Yet there seem to be "wrong" places to study software development. If we talk about the importance of place in software, the place in question is expected to be one of a small number of locations where software development is strong, such as Silicon Valley or Bangalore. Needless to say, wrong places to study software development are in some ways also wrong places to engage in it. In this dissertation I look at people engaged in this global practice in one such wrong place and try to understand how they overcome the limitations imposed by the place, helping the practice of software development appear so naturally universal.

A Practice Perspective on Software and Place

While the precise number of "software developers" in Rio de Janeiro is unknown,

2 Software developers and computer science researchers were among the earliest people to adopt many of the tools without which modern "knowledge work" is unimaginable. Donald Knuth, one of the most revered figures in computer science, *stopped* using email in 1990, concluding that "15 years of email [since 1975] is plenty for one lifetime" (Knuth n.d.). Software development thus offers us a chance to look ahead at what may be coming later in other industries. (See also Castells 1996/2000, p. 92–94, for a discussion of IT industry's use of its own innovation.)

the larger category of "computer professionals" accounts for about 20,000 residents of the city, or about 0.2% of its population of around 11 million people.[3] In comparison, the San Francisco Bay Area in the United States employs around 200,000 computer professionals, who comprise more than 3% of the population—clearly a much larger number. This already substantial difference is amplified tremendously if instead of counting people we look at such metrics as market capitalization of software firms based in each place. The value of publicly traded "software development" and "computer services" companies headquartered in the San Francisco Bay Area adds up to nearly half a trillion dollars. The only publicly traded software company based in Rio has a market capitalization of about half a billion—about one-thousandth as much, an equivalent of a handful of San Francisco Bay Area startups. Even within Brazil, Rio loses to the larger São Paulo, itself a rather small dot on the map of global software industry, but definitely more noticeable than Rio.

This concentration of valuation, which far surpasses the differences in the number of people employed, is indicative of the difference in the *kind* of software work that gets done in different places and the geography of control. Many of the developers working in San Francisco and in some of the other centers of the software world apply their efforts to software intended for broad use, aiming to solve big problems that would bring their companies big rewards. Almost no such work is done in Rio de Janeiro, where developers dedicate their efforts almost exclusively to the smaller problems faced by local organizations. The differences in the perceived importance of work give places like

3 See chapter 2.2 for a longer discussion of the degree of geographic concentration in the software industry and the explanation of the numbers stated in this paragraph.

Silicon Valley tremendous symbolic power in the cultural world of software development. Software developers working in Rio typically recognize their city as a peripheral place in the larger world of software, acknowledging the dominance of the foreign centers.

Yet it is precisely this peripheral position in a remarkably concentrated industry that makes Rio an interesting place to investigate software work. Despite the startling concentration of the software industry, both in terms of the actual concentration of work, and especially in terms of geographic centralization of corporate control and the cultural influence of a small number of places, and despite Silicon Valley's status as a text-book example of an "industry cluster" (Saxenian 1996), software development is itself often hailed as a premier example not only of a "flattened" field of endeavor, where geography has ceased to matter and where anyone can come and play, regardless of where they live, but also as the very technology which brings about this flatness (Cairncross 1997, Friedman 2005/2006).[4]

The notion of increasing irrelevance of place is often linked with that of "knowledge work" and "knowledge economy" (again see Friedman). Whereas traditional industries convert material inputs into material outputs, and moving those inputs and outputs costs money, "knowledge work" is understood as transforming "knowledge," an entity that can be easily imagined as perfectly mobile—at least as long as our idea of "knowledge" is modeled largely on computer files. And while this crucial resource could in theory be hoarded by the privileged few, in practice it is often seemingly rendered free

4 Friedman starts *The World is Flat* (2005/2006) with his trip to Bangalore and an interview with the CEO of Infosys. While the book later jumps between a large number of industries, the rise of Bangalore's software industry provides the organizing metaphor.

for all by the collective generosity of "communities of geeks," which Friedman sees as an example of a broader phenomenon that he calls "uploading."

There is some truth to those ideas. Software work indeed requires *relatively* little capital investment and behemoth companies such as Google had their offices in garages just a few years ago. (To be more precise, *certain kinds* of software work require little capital—an issue explored in more detail in later chapters.) To some extent, one just needs a computer, stable electricity and Internet access—all of which are available in places like Rio de Janeiro even for the relatively poor.[5] Rio software developers tend to agree that there is no shortage of "uploaded" knowledge. They often emphatically talk about the Internet as "the world's greatest library," full of "all the imaginable and unimaginable resources,"[6] stressing the presence of both explicit documentation and actual source code that can be studied or simply re-used.

As a result, a substantial number of people in Rio de Janeiro do engage in this line of work, and as we will see in later chapters, they do so in ways that are often quite similar to those of their Silicon Valley counterparts. Connected to the Internet and often quite capable of reading English, they write their software using the same programming languages, the same development tools and the same hardware. Their discussions are full of English words, and they often explain their decisions by citing the same adages as

5 Tigre (2003) argues that Brazil might be best understood as two societies: "The first is a relatively wealthy population of about 30 million, which has the income, education, and infrastructure to participate in the modern information economy. The second is a poor population of about 140 million, which lacks income and access to the necessary infrastructure to participate" (p. 33). According to the 2005 household survey (PNAD 2005), 32.1 million Brazilians (about 22% of the population) have accessed the Internet in the three months prior to the survey (Olinto 2007).

6 See appendix C, "Original Interview Quotes" (Jaime, October 2005, "A maior biblioteca mundial"). See also appendix B ("Transcription and Quoting Method") for a description of the quoting methods.

American programmers, frequently using the original English. And the developers themselves often argue, at least when talking in general terms, that technological knowledge flows freely over the Internet and that their work is no different from that of their American counterparts. "A server is a server," they say.[7] When talking about specifics, however, the fact that "this is not Silicon Valley" comes up as a frequent explanation.

The practice of software development thus appears to be simultaneously remarkably placeless and starkly placed. This paradox can perhaps be grasped most clearly by considering the case of Google: the company whose search engine is often mentioned as the greatest leveler by Brazilian programmers, but which itself arose—and perhaps could only have arisen—in a highly predictable place, biking distance to Silicon Valley's Sand Hill Road.

Reproduction of Practice

I could approach the paradox described above by trying to understand why software development remains so concentrated in the era of unrestricted knowledge flows. (Some of the possible answers to this question are mentioned in chapter 2.2.) For most of this dissertation I take a different approach, however. Instead of assuming that technical knowledge ought to be naturally fluid and trying to understand what keeps the practice of software development so concentrated, I take its concentration as a given and try to understand how the practice of software development moves in space *at all* and

7 See the glossary for definitions and explanations of technical terms. Some of such terms are underlined in the text.

investigate the effort that is needed to do this work away from the few places where it is well established. Understanding how the seeming universality of software work is achieved *in spite* of this geographic concentration then becomes a key question.

In doing so, I drop the term "knowledge" for the sake of another one: "practice." To understand how knowledge travels we must look at it in conjunction with all other things that must be in place to support its power—the social arrangements that provide the "tracks" along which technical knowledge travels (Latour 1988a). While this expansion of scope could be done by arguing for a broader notion of "knowledge," I switch to a different term partly to draw on the body of theory from which this term originates and which links this concept with that of "culture" (see chapter 1.1), and in part because I feel that the tendency to think of "knowledge" as similar to the content of computer files is so strong today that I cannot expect the reader to ever fully leave behind this metaphor.

The concept of "practice" provides us with a useful analytic layer between the more abstract, propositional notions of knowledge and the practical details of real life. I understand "practice" as a system of activities, a collective way of doing certain things, organized through material tools, persistent patterns of interactions, and shared systems of meaning. Looking at the practice of software development, I thus look at the *doing* of software development, the people and groups that engage in this doing, and the relationships between them. Focusing on activities, and especially on *systems* of activities makes it easy to see why the practice of software development would cluster in a handful of places, as it helps us recognize the many different pieces that would need to be put

together to recreate the practice in a new place. For someone who understands this perspective (presented in chapter 1.1), the problem becomes that of comprehending how a living practice could *ever* move to new places. To put the same question differently, we can ask how "uploaded" knowledge and other elements of the practice, removed from their original context, are put together and made to work in a new place. This dissertation attempts to answer those questions through a close examination of such processes of reproduction and re-embedding (Giddens 1990) in a particular place and time.

Understanding the way a universal practice is made to work in a concrete place requires looking at the many peculiarities of this place, the specific configurations of resources that are available to the actors who inhabit that place and the specific history that has led to those configurations. It is for this reason that I focus on a single city and present it as something concrete, rather than sampling software developers from a wide range of places and losing the concreteness. In looking at a specific place, however, I seek to show relations that I believe exemplify patterns that we can find in many other places. Every place has a history and every place has local context. In every place concrete work must be done to turn abstract knowledge into a living practice.

While I believe that patterns that I explore in this dissertation could have been shown using many other cities, the choice of the specific place can make a difference. Different degrees of peripherality would bring into focus different parts of the reproduction process. Focusing on a place where the practice of software development is yet to take roots would help us see the earliest steps in this process, but would shorten the history available to exploration. Picking a place that is secondary today but could have

become the main center of information technology had the history of the twentieth century gone just a little differently (for instance Cambridge or Berlin) would highlight the importance of contingencies, but would give us little insight into the future possibilities. I believe that my choice of place gives us a good balance: a city present on the world map, yet not quite one of the "global cities," in a developing country that seems to be gaining momentum, yet doing so at a pace that allows for some reflection, and with a history of IT policy that goes back a few decades—putting some of the most important events far enough back to allow for critical analysis.[8]

Open Source

My study also focuses disproportionately on specific kinds of software practice, which the reader may find atypical. In particular, two of my three cases involve (in somewhat different ways) production of "open source" software. Open source software is distributed in a manner that allows the users to modify it, and then redistribute it to others without paying royalties to the original author. This approach can make production of software more efficient, but it also creates obvious challenges, since open source products cannot be easily sold on the market, requiring other ways of funding the work. Over the last fifty years, a complex system of institutions has evolved that makes this method of software production viable (Schwarz & Takhteyev 2009).

---

8 At a workshop in Berkeley in May 2006, after listening to two Berkeley economists talk enthusiastically about the prospects of Chinese and Indian economies, Antônio Barros de Castro, a Brazilian economist and the former president of the Brazilian Development Bank (BNDES), commented that similar projections were made about Brazil in the days of "the Brazilian miracle," which ended abruptly for reasons not fully understood to this day. In this sense, Brazil's "longer" history of information technology is a valuable resource.

Open source software development presents in about the clearest form the paradox mentioned earlier. Open source communities are intentionally open and the apparent generosity of those "communities of geeks" provide much of the motivation for Friedman's discussion of "uploading" as one of the key factors the "flattening" of the world. Such communities are also remarkably dispersed and rely predominantly on computer-mediated interaction, with members often having little idea where on the planet other participants happen to be. At the same time, however, the geographic concentration of those communities rivals that of the software industry, with rare projects that originate in "wrong places" often quickly moving their centers to the West Coast of the United States. (The case of Linux, for example—see chapter 2.2.) The global culture of such communities is largely based on the "hacking" culture that originally developed in American universities (especially UC Berkeley and MIT, see Levy 1984) and is supported by business models pioneered by American companies and optimized for the situations they face. English is almost always the working language of such communities, even as they might strive to create software products that support every last script on the planet. As I will try to show, participation in such projects involves a complex negotiation of culture, language and geography, and is often *harder* than engaging in other forms of software practice, as it requires *more* fluency in foreign culture and demands *more* of the resources that may be hard to find in places like Rio de Janeiro.

Open source development contributes to globalizing the practice of software development. It is important, however, to avoid trivializing this relationship and to consider the local work that mediates it. Open source development creates an opportunity

—and a challenge—to participate in new ways in projects based far away. To take this opportunity and respond to this challenge, however, Rio developers must learn quite a bit more about foreign practices and must find more of the missing pieces of the practice.

On a more abstract level, open source development also simply represents a *new* way of developing software, and thus highlights the challenge of keeping up with the evolving practice based far away—what I call "synchronization work." Looking at how Rio developers respond to this challenge may therefore help us understanding how people engaged in other practices respond to changes in the practice.

Themes and Significance

My discussion of practice in place focuses on several themes. The first is *the process of disembedding and re-embedding* (Giddens 1990) involved in its reproduction across space: people engaged in a practice that is based somewhere else often have to re-assemble the practice around imported elements, substituting for missing pieces what happens to be available. (And if they want to get involved more centrally, *extending* the practice, they will have to find ways to thoroughly disembed their own innovations, to make them mobile and useful in the remote places where the practice is strongest.) The second theme is *the cumulative and parallel nature of the reproduction process*. I look at the local practice of Brazilian software developers as a partial replica of the American software practice and frame my observations as a particular moment in the history of the reproduction of this practice—a moment when many elements of the practice had already been brought in and reassembled (hence the need to look at the history, see chapter 2.2),

while others are still missing. I also look at this reproduction as one of many parallel efforts to reproduce foreign practices. Third is the theme of *a "diasporic" situation of the peripheral practitioners,* who engage simultaneously in two cultures: the local mainstream culture and the foreign culture of the practice. (Those engaged with the practice at its centers may face this issue as well, but the gap between the two cultures is usually not as wide.) In particular, I look at how commitments to those two cultures come in conflict and how such conflicts are negotiated. Closely related to this is *the complex relation between individual and collective efforts of reproducing foreign practices:* the local practitioners must often make a decision whether to cast their lot with local colleagues or to focus on their individual connections to the remote centers. The fourth theme is *the interaction between the cultural and economic layers of the practice,* and the need to look at the two simultaneously, considering the situations when one of those layers is present and the other is missing (culture-looking-for-job vs job-looking-for-culture). Finally, underlying all of those themes is the one of *actors' reflexive understanding of the world,* the possible futures they can imagine individually and collectively, and the factors that influence this imagination (Appadurai 1996).

By bringing to light the work that peripheral practitioners must do to give software development its seeming universality, I hope to give them the credit that they deserve (and all too often deny themselves), touching upon the question of why software development remains centralized. While I do not see this centralization as a puzzle *per se*, I do believe that there are many explanations that are wrong and self-serving, and that such explanations may themselves contribute to the persistence of centralization. The

discussions of the geography of software work (or other types of "knowledge work") and the feasibility of developing "the next Silicon Valley" in this or that place quite often arrive at the importance of attracting "smart people."[9] While not doubting that smart people are important for a successful software industry (as for many other types of work), researchers and policy makers sometimes seem too quick to assume that places that lack strong software industry lack smart people. In fact, if one assumes that technological knowledge flows naturally between capable minds and is sufficient for recreation of a knowledge industry, then the concentration of software development in a handful of places would seem to imply that other places lack smart people, smart governments, smart investors or all of the above. Unfortunately, such judgments are often internalized by the peripheral actors themselves, who might sometimes consider themselves an exception to the rule, but too often assume that the mediocrity of their fellow citizens limits what they can achieve. Highlighting the work that went into bringing about the current state of affairs and the achievement inherent in that, I hope will present a brighter picture and in turn facilitate local cohesion.

I also hope to show how such peripheral work contributes to the continued dominance of the remote centers. Software development is in many ways an economy of attention (Goldhaber 1997, 2006), or "mindshare" as some developers call it. By fixing their gaze solidly on foreign technology and investing efforts into making it work locally, Brazilian developers often deny to local projects the attention that such projects may

9 For instance, Paul Graham writes in his essay entitled "How to be Silicon Valley" (2006):

> The exciting thing is, *all* you need are the people. If you could attract a critical mass of nerds and investors to live somewhere, you could reproduce Silicon Valley. And both groups are highly mobile. They'll go where life is good. So what makes a place good to them?

need. Such lack of attention and, more importantly, lack of *trust* in local projects is ironically the opposite of what has been credited for making Silicon Valley the success that it is—the strong networks of personal relations and personal trust (e.g., Saxenian 1996). Unlike in the San Francisco Bay Area, in Rio being local carries a stigma and the local place works against the practitioners. The local developers are thus themselves involved in replicating the asymmetries from which they suffer.

Such observations should not be interpreted as suggesting that peripheral participants and the regulators should either turn away from foreign technology or desist altogether in light of the challenges. As Brazil has learned in the past, isolationism can be a dangerous strategy and nuanced solutions are needed. I do not make specific policy recommendations, but I do hope that this work will help inform policy by providing an alternative image of work at the periphery of technical communities, giving policy makers an opportunity to visit a world that they govern (in part) but do not always understand, to see the challenges faced by people who inhabit this world and to consider how helping them face those challenges may contribute to the larger developmental agenda. I hope in particular that the case of Kepler (chapter 3.4), read together with the two alternatives (chapters 3.1 – 3.3), will be useful for thinking about innovation policy.

I do not present Kepler as a simple model ready for imitation—the reader may find many mistakes in the decisions made by the participants. It does, however, represent an imperfect attempt at the sort of innovation that I believe would be crucial for reducing the asymmetry between the centers and the periphery of the software practice—innovation that is simultaneously local and global, engaging with the remote elements of

the practice without having to disengage from the local place. I hope that my in-depth account of the project provides a starting point for thinking of policies that would support such innovation. I also hope that my critical (though sympathetic) account will help software developers in Rio and other "wrong" places look at their practices from the side and perhaps make some adjustments. If nothing else, I hope it will help them understand that many of the problems and dilemmas they face are also faced by others.

The Choice of Method

The dissertation relies on qualitative analysis of data collected through participant observation and semi-structured interviews with around one hundred people. I rely on such methods for several reasons.

The simplest and perhaps the least important of those is the fact that no appropriate quantitative data was available or could be realistically collected that would illuminate the questions that I aim to investigate. While statistical data could be very powerful, it must be collected through proper methods to be useful. Without a proper sample frame, a survey of Rio software developers or software firms would lack the reliability that one expects of quantitative data. In contrast to frame-less surveys, the qualitative methods that I use have an important advantage of explicitly recognizing the lack of representativeness and finding ways to deal with those limitations.

More importantly, however, qualitative methods are uniquely fit for research that aims to understand the role of *place* and the ways in which abstract ideas are made to

work in concrete circumstances. To understand this interaction between the abstract and the concrete, we must capture and represent the concrete. This means taking time to understand circumstances of a particular place and, further, the work and life histories of particular individuals. This requires *in the very least* conducting lengthy interviews that allow the participants to tell the researcher in substantial detail about their background, their career history, their aspirations, allowing the researcher to start seeing connections between them. In my experience, an hour long interview only begins to scratch the surface of the individuals' history of involvement with a practice like software development. In addition to being lengthy, the interviews cannot be overly structured if the researcher wants to capture the diversity of experiences and to let the interviewees bring up factors that the researcher might not have considered relevant.[10] Capturing diversity is crucial for capturing concreteness. An overly restricted interview guide that starts with an assumption that each interviewee is an instance of the same phenomenon, the numeric parameters of which the researcher seeks to measure, solicits generic responses which lead us to a statistically abstracted individual and away from

10 The term "semi-structured interviews" is typically used to describe the interviews of the type that I used in my work. Such interviews are contrasted with "structured" interviews, in which a standardized questionnaire is used with all participant, who are then asked specific questions from the questionnaire and must typically pick a response from a constrained set of options. Researchers who use "semi-structured" interviews typically also come prepared with a set of questions, but such questions are usually phrased in a broader way and the questionnaire is normally *not* used during the actual interview. While researchers rarely describe their interviews as "unstructured," examples of techniques that involve even less pre-defined structure would be informal conversations conducted in the course of participant observation and "non-directive" (or "reflective") interviews in which the researcher asks a single question or suggests a topic, and later only asks clarifying questions, as well as repeating to the interviewees what they just said and asking for elaboration (Rogers 1945). My fieldwork involved plenty of informal conversations. To avoid ambiguity, however, I do not refer to them as "interviews" when describing my work. The use of reflective interviews is rare in social science, though in my experience this method sometimes works well as a part of semi-structured interviewing, as an elaboration technique. Used this way it bears certain resemblance with the use of "markers" as described by Weiss (1994).

understanding the concrete relationships between the abstract technical knowledge and place. Qualitative methods, on the other hand, allow us to focus on the individuality of the participants.

Use of long semi-structured interviews precludes quantitative analysis. First, the substantial time that it takes to arrange, conduct and analyze each interview necessarily limits the size of the "sample." This study relied on around one hundred interviews, a large number by the standards of qualitative studies, but minuscule by the standards of survey research. Additionally, due to the lack of a rigid structure, the interviewees do not always answer the same questions and to the extent that they do, reducing participants' answers to the least common denominator is a non-trivial task. More importantly, I make no pretense as to the randomness of the sample. Instead, my sample has features of what qualitative researchers call either "a theoretical sample" or "a snowball sample"—terms that in theory refer to different approaches to sampling, but in practice frequently apply to the same samples. (My sampling strategy and the rationale for it are described in the next section.) For those reasons, I avoid subjecting my interviews to any quantitative analysis, believing that such analysis would likely be misleading.[11] Appendix A describes the people whom I interviewed to allow the reader to consider what my analysis may be omitting. That summary should not be used as representative of software developers in Rio. While I avoid attaching numbers to my analysis, I occasionally use quantifiers such

11 Some qualitative researchers do perform quantitative analysis on qualitative data and I believe there are ways to do such analysis correctly. If there are good reasons to believe that the sample is unbiased relative to the variable that is being investigated and the interviewing procedure does not influence those variables, then the interviews can later be analyzed using quantitative methods. Such methods are frequent in linguistics, for example, though even then the preference is usually for corpora collected without the involvement of an interviewer (for instance by asking people to record all of their conversations for a period of time—see Biber 1993, Crowdy 1993).

as “some,” “many” or “most.” I use such quantifiers sparingly to identify patterns that I see among my interviewees that I also believe to be true of the general population based on the sum of my knowledge of the field.

For the same reasons I avoid excessive formalism in my analysis. Such formalism, common in some variations of “grounded theory” method, in my view attempts to create an illusion of the positive scientific method, which I believe is impossible without proper sampling.[12] Instead, I take an interpretive approach to both my interviews and ethnographic observations, relying primarily on the memoing method described by Emerson et al. (1995). I strive to stay true to my data and be precise when presenting what other people say (see, for example, appendix B, “Transcription and Quoting Method”), and I qualify my presentation of cases that I know to be atypical. At the same time, I try to go beyond simple summaries of what is typical, looking at each individual’s story as a potential “case” through which their society can be understood, rather than a bundle of parameters to be reduced to a row in a table. For this reason, I subject only a subset of my interviews to detailed analysis, though always keeping in mind the others and frequently looking at them for counter examples. I present in this dissertation an even smaller number of cases. Despite having a large number of interviews to draw upon, I prefer to try to reuse stories from the same small number of people, who in most cases are connected to one another. I believe this choice helps preserve the interconnections between the individuals and the different contexts that those individuals inhabit. It should also help

12 Strauss & Corbin (1990) and Glaser’s response (1992) mark the official split of Grounded Theory (Glaser & Strauss 1967) into “Straussian” and “Glaserian” versions. The complicated issue of which one is more “formal” and which one is more “positivist” is not relevant for our purposes here.

the reader keep track of the "cast of characters" and understand them as concrete individuals.

Interviews have important limitations from the point of view of qualitative research. The first and the most obvious problem, is that interviewees may not always tell the truth, for example exaggerating their accomplishments. This problem can typically be avoided by asking interviewees to be specific (Weiss 1994) and not being overly gullible.[13] Second, and more importantly, the answers that the researcher obtains are often limited by the questions he or she asks. Even when the questions are seemingly open-ended, the interviewees often form an idea of what factors the interviewer wants them to talk about and may choose to stick with those factors. A yet more important limitation is that the factors that are of most interest to the researcher may be things that are normally "not said," especially to an outsider. Longer interviews help establish more trust and convince the interviewee that the researcher really does want to hear about their experiences. The longer length and greater detail also increases the chance of the interviewee "tripping" and stepping outside standard public discourse. Such *faux pas* can indicate the tensions underlying the public discourse (for instance, the tensions underlying the notion of software development as a "global community"), but they do not usually allow us to understand such tensions in detail. A researcher interested in understanding what is not said must ultimately look for additional methods. For those reasons, in the second phase

13 Asking interviewees to be specific is not only common wisdom in qualitative interviewing. In my experience leading a team of software developers in California and having to interview new applicants I also found that asking for detail is crucial for noticing when people are not telling the truth. Doing so in research is a higher form of art, since the interviewer does not have the same luxury of power imbalance as the interviewing manager. On the other hand, people are typically a lot more frank in research interviews then they are while applying for a job and less fact-checking is often needed.

of my fieldwork I changed my method to rely heavily on participant observation and semi-structured interviews with members of the groups in which I was involved. The data collected in the course of this second phase makes up the core of the dissertation, while the interviews conducted in the first phase provide a background against which I do my analysis.[14]

Participant observation is an ethnographic technique that involves studying people by taking part in their activities, typically for an extended period of time. It may involve rather different degrees of participation, though it always presumes, at the very least, being there when the activity takes place, rather than asking people to talk about the activity after the fact. On one extreme, the researcher may choose to simply be there and observe. On the other extreme, he or she may become actively engaged in the activity, to the point that the activity comes to depend on their participation.[15] Such active engagement, which characterized my involvement with Kepler (the open source project that is the subject of chapter 3.4), implies abandoning all pretense at being a disinterested observer. It also creates a risk of "going native"—becoming so involved in the activity and so successfully acculturated to the community that the activity that was originally the object of investigation becomes more important than the activity of studying it. My

14 The data collected in the second phase includes both my field notes and the additional fifty interviews conducted during the second phase, most of which were directly related to the three projects that I was observing. While such interviews are on the surface similar to the interviews that I conducted in the first phase, they are actually very different, since in most cases I either arrived to the interviews with deep knowledge of the group and its activities or could later compare what I was told with direct observation. Such knowledge changes the dynamic of the interview. On the one hand, it makes it easy to focus on details and to elicit frank discussion. (Knowing half the story goes a long way towards convincing the interviewee to tell the other half.) On the other hand, it sometimes makes it harder to discuss the other aspects of individual's life.

15 Jorgensen (1989), for example, discusses the different degrees of involvement by a participant observer.

engagement with Kepler brought me close to that point, requiring that I draw clear lines.[16] It is during that time, I believe, that I gained some of the most important insights about the project that I was studying.

Participant observation ironically has an advantage over interviews in terms of representativeness of the results: the data collected through participant observation is so obviously specific to the sites that the researcher studied, that there is usually less temptation to assume it to be generalizable. I explicitly warn the reader, however, that neither of the organizations that I studied is quite typical even relative to what I myself know of software in Rio de Janeiro. "Alta," the company described in chapters 1.2 and 3.1 is *somewhat* typical in the sense that it pursues a strategy that is similar to that of many other companies that employ software developers in Rio. It seems quite clear to me, that Alta has pursued this strategy more successfully than most software firms operating in the city and I believe this makes Alta a more instructive example. Kepler and Lua (chapters 3.2 and 3.4) are clearly exceptional projects. Both are examples of what rarely happens in places like Rio. Understanding those cases, however, gives us insights into why such projects are rare and into the space of possibilities that peripheral participants face.

16 As I started to recognize my growing involvement in the project, I decided to use writing of fieldnotes as the test for my continued commitment to the research activity. By that logic, the researcher actively engaged in the activity that he or she studies have not "gone native" as long as the time dedicated to the activity does not prevent them from keeping detailed fieldnotes describing the events. While fieldnotes became a chore during a part of my fieldwork, I maintained them throughout the time when I did participant observation.

The Project

As Van Maanen (1988) points out, ethnographers use different approaches to present their observations. Some tell "realist tales": accounts that simply present what happened, taking as given the ethnographer's ability to know what happened and to interpret it. Others tell "confessional tales": accounts that focus on the observer as much if not more than they do on the observed.[17] They normally do so out of realization that the observer inevitably influences what is observed and that the process of observation and interpretation is often fragile and its success is contingent on many factors. The inclusion of the observer in the account helps the reader better understand what was observed by being told who did the observing and how. It also helps the ethnographer consider their own biases, as it encourages him or her to think more closely (and explain to the reader) about their own role in the events.[18] (Such reflexivity forms the foundation of what Burawoy 1998 calls "reflexive science.") Though I try to include myself in the account whenever appropriate, I avoid the extremes of confessional ethnography, feeling that such presentation would distract too much from the argument that I want to make. In particular, the order of the chapters is dictated by the logic of the argument rather than by the chronology of the project.[19] To compensate, I present such chronology in this section.

This section also expands the discussion of method started above. I present the

17 Van Maanen also describes a third type: "impressionist tales," which "present the doing of the fieldwork rather than simply the doer or the done" (p. 102), and focus on drawing the reader into the scene rather than on reflection or analysis.

18 As Van Maanen points out, such a "confession" is of course also another way of establishing the authority of the author by explaining the process by which they arrive at their conclusions.

19 For example, most of the events presented in chapter 3.4 precede those in chapter 3.1. My interviews with Jason (chapter 2.1) also occurred very late in my fieldwork.

more detailed discussion of my method in the context of the projects' history (and separately from the theoretical justification of the method), because some of those details may be important for the reader, but cannot necessarily be explained from the point of view of what would have made most sense to do.[20]

The Early Question

In the summer of 2003, after spending three years working as a software developer in Mountain View, CA, the heartland of the region known world-wide as "Silicon Valley" (but typically referred to locally as just "South Bay" or "the peninsula") and before starting my Ph.D. program at Berkeley, forty minutes away by car, I spent a month in my home town in Russia, on the other side of the Pacific, in Vladivostok, a city known to many Brazilians primarily as a base of attacking Alaska in "War," a board game based on the American game "Risk."[21] While there, reconnecting with old friends and meeting new people, I saw a world that I had started to forget during my years in California. I was in a place that was in some ways very parochial, while at the same time much more global than Mountain View. One could not find in Vladivostok California's diversity of cuisine or languages, yet the existence of the external world was much more apparent than it ever was in California. Many of my conversations revolved around places outside Russia, and in particular the United States, which seemed to be present in Vladivostok in the way no country is in California. Some of my friends worked in research, leading to discussions of

---

20 To paraphrase Søren Kierkegaard, an ethnographic project can only be understood backwards, but must be carried out forward.

21 The game is distributed in Brazil under an English name ("War") by a company based in São Paulo.

how their work related to similar efforts in the United States. I thus became interested in understanding how academic researchers in countries like Russia learn about what happens in their field abroad and in particular in the United States. A trip to Brazil in January 2004 and a month in Finland the following summer, combined with my participation in a short-lived research project dedicated to "consumption of information," solidified this interest, while also generalizing it to "knowledge work" beyond academic research.

By September 2004 I decided to dedicate my dissertation to understanding how Brazilian software developers access software knowledge from abroad, choosing Brazil over Russia with the hope of avoiding the methodological difficulties inherent in native ethnography. Over the course of the year, I framed this question as a part of a larger one: How do people go about acquiring professional knowledge when they are separated by distance from the areas where this knowledge is most abundant? At the moment, it did not occur to me to ask whether Brazilian software developers were in fact in a place where locally-generated software knowledge was in short supply and whether they actually tried to access knowledge from such places like the Silicon Valley. Both assumptions were reasonable and turned out to be correct—the developers I later interviewed typically saw Rio as no match for Silicon Valley as far as software goes, and they most certainly did seem focused on keeping up to date with what was happening abroad. Such assumptions, however, hid many of the questions that later came do dominate my dissertation and to which I will turn shortly.

My initial approach to the project drew on several traditions. Under the influence

of Orr (1996), Brown & Duguid (2001), Saxenian (1996, 1999), as well as what I now see as insufficiently careful reading of Lave & Wenger (1991), I worked with an assumption that place and face-to-face interactions matter a great deal for learning, and that physical distance could not be ignored. In line with this assumption, I believed that the questions I wanted to answer could not be addressed without face-to-face contact with the people whose practices I wanted to understand. On the other hand, I believed, to some extent *contra* Brown & Duguid, that digital documents could go quite far in bridging this gap, and that the ideas of communities of practice (Lave & Wenger 1991) can be reconciled with economic approaches to knowledge and innovation, as well as the ideas of social epistemology.[22]

During that year I also got my first exposure to the history of Brazil's "market reserve" policy aimed to create a local computer industry, mostly through Evans (1995). This led me to start thinking about both the benefits and dangers of autarkic policies and about the different ways of engaging with foreign technology.

A Round of Interviews

I arrived to Rio de Janeiro in June of 2005 for a six months stay and started building my sample of "software professionals." I defined the term loosely, including in it people who were trained to write software, whether or not they actually wrote it as a part of their job, and people who actually wrote software, regardless of whether they were

22 I was taking solace in Cowan et al.'s (2000) argument that most "tacit knowledge" is potentially codifiable, even if much of it had not been codified for lack of proper incentives, and that while learning through participation may at times be a more economical way to acquire knowledge, it cannot be the only way to do so in theory. In terms of social epistemology, I was primarily influenced by Goldman (2002).

trained to do so.[23] My sample combined elements of a "theoretical sample" and a "snowball sample." The term "theoretical sample" describes an approach to sampling that involves the researcher seeking "cases" that they hope will challenge their preliminary assumptions and lead to further development of the theory (Glaser & Strauss 1967). Such sampling technique often aims to increase the diversity of the sample, to compensate for its small size. In my case, I attempted to include among my interviewees every type of software developer that I could identify, "oversampling" atypical individuals. For that reason, I made sure to interview not only developers graduating from top universities, but also their professors, continuing my quest for the ultimate "alpha-geek" until my interview with Roberto Ierusalimschy, one of the authors of the Lua programming language, to which I dedicate chapter 3.2. While 4% of the people I interviewed in 2005 were inventors of a programming language with a substantial international following,[24] one quite obviously should not generalize this statistic to the population of Rio's "geeks." I similarly attempted to include developers with as little education as I could find. I interviewed people from a range of work environments: small companies, large local companies, multinationals, university research labs, people officially employed by their companies and those hired as contractors, employees of the public and private sector. I talked to people of different ages,[25] and made an attempt to include a larger number of women than I believe a random sample would have drawn.

---

23 Later in the project I switched to the term "software developers" for reasons discussed in chapter 2.2. See footnote 1 on page 1 for my current definition.

24 Two interviewees out of fifty, who worked jointly on the same programming language (Lua).

25 Interviewees' ages ranged from 21 to 51 in this first phase, though most interviewees were 25 to 40.

A lengthy interview requires a substantial time commitment, which people are not always willing or able to offer. As has often been demonstrated for surveys, people who are willing to participate in research are often systematically different from people who are not (Creswell 1994). In my own experience, for example, socio-economic status was an important factor: I found it easiest to recruit people who saw me as a peer, could recruit people of higher status with only a little bit of difficulty, but had to struggle to recruit developers of lower socio-economic status.[26] To mitigate such effects, it is important to maintain fairly high response rate. I used a common a technique: progressive referral, also known as "a snowball sample." With this technique, the researcher starts by recruiting a number of subjects (often with great difficulty) and then asks each interviewee to recommend other people who could be interviewed. Needless to say, a referral increases the chances of a positive response dramatically, giving the researcher a chance to include people who would not be willing to offer an hour of their time to a complete stranger.

The snowball sample does not eliminate bias, since people typically recommend those who are similar to themselves and the researcher can easily end up staying in the

26 While many factors may be behind this, I believe one of the most important ones is the simple difference in participants' control over their time. Developers who had levels of education comparable with mine typically exercised substantial control over how they spend their day (much like I did in my earlier work as a software developer) and could give me time if they chose to do so. Those of higher socio-economic status could do so as well, but often faced more requests, which made it harder for me to get their attention. Developers of lower socio-economic status often had long and constraining jobs. When time could be found, space often became a problem. While developers in higher positions often did not hesitate to bring me to their company's offices, those with less education nearly always preferred to meet somewhere else, which typically meant crowded public spaces (bars, fast-food restaurants), often closer to where they lived and quite far from downtown Rio de Janeiro. Apart from those constraints, homophily clearly played a role as well. Highly educated developers were often happy to treat me as one of them, while some of my other interviewees clearly saw me as more similar to their bosses than to themselves.

same network. However, this method makes it easier to keep track of bias, and to compensate for it. It also helps build a theoretical sample, since after building up a network the researcher can afford to become "picky" and ask for specific types of referrals. For example, at different points in my research I found myself asking my interviewees if they could suggest a female software developer, a developer who contributes to open source, a developer who works for a large company,[27] the smartest developer they know. The snowball sample has another important advantage: the fact that the participants are often related (as friends or colleagues) to each other makes it possible to see connections between them and to do certain forms of cross-check. For this reason, for the fifty subjects interviewed in 2005 I alternated between asking interviewees for "near" and "far" referrals. For "near" referrals, I asked them to recommend colleagues or friends, which helped me get a second and third image of their organization. For "far" referrals I asked them to recommend people with particular characteristics, which typically allowed me to move to a different clique or organization along a "weak tie." I typically aimed to interview around three people in each organization.

I came to Rio with some knowledge of Portuguese, though not quite ready to conduct interviews in Portuguese comfortably. My earliest interviews were thus conducted in English, while I was also taking private classes to prepare for interviews in Portuguese. I started conducting such interviews in the beginning of my second month, at first resorting to Portuguese only when talking to interviewees who could not speak

27 My original difficulty in interviewing people working for large companies had in part to do with such companies' relatively small share of the Rio labor market but even more so with their more closed nature.

English. As my Portuguese fluency improved, I conducted more interviews in Portuguese, eventually using English only with the developers who spoke fluent English and preferred to talk to me in it.

During my second month in Rio I learned a methodological lesson that affected greatly the rest of my project. Most developers that I interviewed up to that point assured me that they never discuss technology outside work, which I found quite surprising. I brought this up during an informal post-interview chat with a developer who did mention discussing technology and work with friends. He suggested that the other interviewees were simply not willing to admit it. Talking about work, he explained, is simply not considered cool in Rio—young men are expected to talk about soccer and women, not computers. He assured me that my other interviewees do talk about technology with friends, and that I just had to know how to ask. As I soon came to realize, small differences in wording and intonations did indeed affect greatly the interviewees' readiness to talk about talking about technology. The incident also made me realize, however, for the first time, the subtle incongruence between the local culture and the seemingly global software practice. Furthermore, it led me to start paying attention to not only what my interviewees were telling me, but also why they were telling me that, as well as to things that were unsaid, or sometimes half-said. (Another point that I learned in this and other similar interactions was the importance of drawing on "ethno-methods"—developers I interviewed in Brazil became a great source of explicit advice on how to interview other Brazilians.[28])

28 I see this methodological observation as extending Giddens's (1979) argument that the actors know a good deal about the conditions of social reproduction. This is especially true when "actors" in

In late August 2005, when discussing my plans with a senior official of the Ministry of Science and Technology, I got reprimanded for trying to understand Brazilian reality in isolation from Brazil's history. I believe this dissertation shows that I took this suggestion seriously. Though my investigation into Brazil's history and its relation to the current practices did not come together until after my return from Brazil, I did use my visit to learn more about history. Luckily, I was affiliated with Institute of Economics of the Federal University of Rio de Janeiro, as a visiting scholar with Prof. Paulo Tigre, whose 1983 book documented the growth of Brazil's computer industry in 1970s. My conversations with Paulo Tigre, as well as with his student Sidney de Castro Oliveira, and also with Ivan da Costa Marques and Henrique Cukierman helped me gain a better understanding of Brazilian history from a perspective that is not very popular today, especially among the younger of my interviewees. As chapter 2.2 ("Software Brasileiro") shows, I did not fully agree with them—nor, for that matter, with their critics. However, their perspective gave me a point of comparison that helped me question the idea of "global technology" and start looking at how the global nature of technology is constructed through local work.[29]

---

question are highly educated professionals. When using the term "ethno-methods" here, I do not use it in its narrow ethnomethodological sense (e.g., Garfinkel 1967), but rather in the broader sense of social science–like "methods" used by the actors themselves.

29 I was particularly influenced by Marques's observation (2005) about the ambivalence inherent in the position of the peripheral practitioners: "The ambivalence around simultaneously copying and rejecting the models they imitate often brings those located in the contact zone between colonized and colonizers (such as Brazilian computer professionals) into a kind of impasse. They simultaneously imitate and are hostile to the models they imitate. They copy to the extent that they accept the standards diffused by modernity" (p. 150). Of course, the term "colonized" should be applied with much caution to Brazilian programmers, most of whom have predominantly European ancestry.

A Year at Berkeley

In January of 2006 I returned to Berkeley to analyze my data and to prepare for my qualifying exam before returning to Rio for another five months. During that time my interest increasingly shifted from the mechanics of how my interviewees kept in touch with foreign technology, to the tensions and contradictions in some of their accounts. I came to see those contradictions as reflecting underlying conflicts between their commitments to the local place and to the "global" (but often also quite foreign) technological practice. I also started recognizing in those tensions the different images of the world that the developers had.

In September 2006 I exchanged a few email messages with "Rodrigo Miranda," one of my first interviewees in 2005, a coordinator of an open source project, called "Kepler," aiming at building a web development platform based on Lua, the programming language developed in Rio de Janeiro that I referred to earlier. When I mentioned to Rodrigo that I was planning to return to Rio in early 2007, he asked me if I would like "to participate in the Lua adventures," adding that he might be able to find funding to pay me to work on some parts of Kepler. I declined the job offer, but took time to learn more about Kepler and Lua—projects that I earlier treated as too atypical for serious investigation. As I learned more about Kepler and Lua, I found myself puzzled and surprised at every step. I was also starting to get a better understanding of the more typical cases. I then decided to dedicate half of the second phase of my fieldwork to Lua and Kepler, reserving the other half for a study of a more typical case—some company building custom Java web applications for local clients.

Participant Observation

In February I joined the Lua mailing list, spent some time reading its archives and did six interviews with Lua users in California.[30] I then went to Rio to start this second round of fieldwork in the last days of the month, having already secured not only Rodrigo's invitation to study Kepler and Roberto Ierusalimschy's blessing for studying Lua, but also a desk in Rodrigo's office at Nas Nuvens, a company that sponsored Kepler. I thus jumped into my study of Kepler right away, leaving my study of a "typical" company for the later part of my stay. As it turned out, I arrived at the right time. (See chapter 3.4.)

Despite having seemingly open access to the project and getting a chance to meet most participants early on, I soon confirmed my suspicions that mere physical observation does not go very far when observing software work: one mostly gets to see people staring at their screens, typing, and occasionally swearing. Such observation gets even more complicated when the participants do their work in different parts of the city, which cuts the amount of time dedicated to water cooler conversations even further (replacing them, for example, with instant messaging, a more private medium). I tried to compensate for this with interviews, but my conversations with the developers often seemed too removed from what they were actually doing. In fact, after a few weeks, I started experiencing doubts that the project even existed. I had already gotten involved in helping Rodrigo with the project website, but felt that even this was giving me too second-hand of a view. Meanwhile, our discussions arrived at the conclusion that we

30 Five face to face in San Francisco Bay Area and one by phone—see chapter 3.2.

needed a wiki for managing the documentation. After we went through a number of options for wiki software, I made a fateful decision to write my own wiki in Kepler, which was after all a platform for developing web applications such as wikis. While writing a simple wiki only took a few days, it immediately changed my place in the project. As the first public application built on the platform, the wiki generated immediate interest. As I started spending time making improvements, my conversations with the developers changed. I was now one of them, a member of the project and the larger "Lua community."

I found in such active participation a solution to many of the problems of studying software work that troubled me at first. Software work often moves between many contexts, especially in the case of an open source project. Without literally looking at the developers' monitors over their shoulders, both at work and at home, keeping track of their solitary work, private emails and IM conversations, cell phone calls and face-to-face chats, one can hardly see all the work that goes into the creation of the software project.[31] While no method can reconstruct the project in its entirety, active *participant* observation provides the ethnographer with a partial solution: a situated and integrated picture that weaves together *some* private emails and IM conversations, *some* late night conversations over pizza, as well as quite a few hours alone in front of the monitor making sense of

31 Latour & Woolgar (1979/1986) provides, in jest, a description of what a social scientist would have to do to their subjects if they were to aim for the same degree of rigor as the biologists in the lab that he studies:

> [W]e would require about a hundred observers of this one setting, each with the same power over their subjects as you [the biologists] have over your animals. In other words, we should have TV monitoring in each office; we should be able to bug phones and the desks; we should have complete freedom to take EEGs; and we would reserve the right to chop off participants heads when internal examination was necessary. With this kind of freedom, we could produce hard data. (p. 256).

debug traces.

Such engagement also creates a number of challenges. Apart from requiring a certain amount of skill, which I was lucky to have (note that either too little or too much skill can become a challenge), it also presents the methodological challenges of getting involved without "going native." A certain degree of re-socialization is of course a crucial aspect of the ethnographic experience, hence many ethnographers believe in doing ethnography far enough from home to achieve isolation from the home environment (see Van Maanen 1988). Too much involvement, however, can really strain the ethnographer. It also raises serious questions of commitment. I got asked, on quite a few occasions, whether my participation in the project was "serious" or "just a research project." To be a participant, was to be involved seriously, treating the activity as meaningful and important on the same terms as the other members. Faced with this choice, I decided to get involved seriously.

In traditional ethnography, the obvious need to physically return home provides ethnographers with a natural end to the involvement—and hopefully keeps them from making unrealistic commitments before that. Virtual projects done over the Internet create an opportunity—and in the view of some members an *obligation*—for the ethnographer to maintain commitment to the project through continued remote participation. While I left Brazil in August 2007, I stayed involved with the project since, contributing work, observing online interactions and conducting occasional phone interviews with the participants.

In June 2007, I moved my base from Rodrigo's office to the office of "Alta," a

software company building Java web applications for local clients. My time at Alta was shorter than my engagement with Kepler, though having interviewed a number of Alta's employees before starting to work there (while still negotiating an arrangement) helped somewhat. My study of Alta was also substantially less participant, as I continued to be involved with Kepler and conducting interviews related to Kepler and Lua. As it soon became clear, my commitment to Alta was insufficient for the company to depend on me —especially when its obligations to clients were at stake. To say it differently, my participation would not have been sufficiently "serious." I had to settle for a relatively passive role: spending time in the office, chatting with the employees, conducting sit-down interviews with them, sometimes spending time watching their work over their shoulder, poking around in the code repository, but not actually doing their work together with them. Despite such limited participation, the six weeks spent in Alta's shiny office provided me with an opportunity to better understand a different and in many ways more typical work environment.

I returned to California in August 2007, bringing with me 150,000 words of fieldnotes (the length of this dissertation with all the appendices), not counting notes and recordings for the additional fifty interviews conducted during that period (as well as the fifty interviews from 2005).[32] I spent the following year sifting through this material and theoretical literature, looking for ways to put the two in a conversation with each other. The next section outlines the result of this process.

---

32 Most of the interviews conducted in 2007 were with people involved in different roles with Lua, Kepler or Alta. Some were follow up interviews with people interviewed earlier and a small number was with new people.

The Chapters that Follow

The dissertation is organized into three parts. **Part 1, "Work as a (Global) Practice,"** provides the theoretical foundations. It consists of three chapters: a review of the existing literature on work and practice, an ethnographic chapter that presents challenges to this literature, and a theoretical chapter that attempts a resolution of those challenges. Parts 2 and 3 explore in more detail the different contexts that I studied, adding more depth to the argument summarized in part 1. **Part 2, "Histories and Maps,"** combines a discussion of an individual entry into the world of software with a look at the history and geography of this world. **Part 3, "The Roads Ahead,"** looks at three ongoing projects based in Rio de Janeiro, which illustrate three of the possible configurations of global and local factors available to the software developers in the city. This allows for a discussion of the future possibilities for software development in Rio de Janeiro — and other places at the periphery of the software world.

Part 1: Work as a (Global) Practice

In part 1, **chapter 1.1** ("Work as Practice") provides a review of a theoretical tradition upon which I base my discussion of work as practice, synthesizing the somewhat diverging perspectives and accommodating some of the common criticisms. My review focuses on the Chicago School literature on work (Hughes, Becker, Van Maanen and Barley, Orr) and Lave & Wenger, presented historically. I start with Hughes (e.g., 1958) and Becker (e.g., 1953, 1963), then punctuate my presentation by introducing the Marxist

critique (Braverman 1974) and possible resolutions (Burawoy 1972, Willis 1977). I then read Van Maanen & Barley (1984) as in part a reply to Braverman, and follow it up with a discussion of the Science & Technology Studies literature and the literature on boundaries (e.g., Gieryn 1983). I arrive at Lave & Wenger (1991) armed with the prior discussion of work as labor and of boundaries, highlighting the way those ideas are represented in Lave & Wenger. I close this section with a discussion of Orr's (1996) ethnography of Xerox technicians which, I argue, presents an image of work that encourages an overly-situated reading of Lave & Wenger, and concluding that a new picture of work is needed.

Chapter 1.2 ("The Global Tongue") presents a collection of vignettes from my ethnography that shows the limitations of the understanding of practice developed in chapter 1.1. The chapter focuses on the complex role played by English in the work of Brazilian software developers, and the ambiguity between its status as the language of software and the language of the United States. I look at the use of language because it provides the clearest examples of a more general phenomenon: the juggling of multiple cultures in which the developers must engage, an issue that is explored more fully in later chapters.

I take up those theoretical challenges in **chapter 1.3** ("Practice in Space"), looking at how practice is reproduced across space. I start by trying to theorize synchronization of practice between two abstract places, without considering the larger world of which those places are a part. I do so by drawing a distinction between concrete (and typically local) communities of practitioners (as presented by Van Maanen & Barley), and the more

abstract “categories” (such as “software developers”) that are often treated as universal by their members. I argue that such categories are both real and crucial for our understanding of any practice. I draw on Giddens’ theory of structuration (1979) to explain how abstract categories come to affect practice, looking at such categories as resources that structure local action. I consider two ways of understanding abstract categories: “external” (categories as defined by the place the practitioners play in the division of labor) and “internal” (categories as seen by the practitioners), showing how both can be understood through the prism of structuration. I present those two understandings not as alternative explanations but rather as different “moves” that actors can employ. I then illustrate some of those moves using the history of computer and software engineering in Brazil. In doing so, I stress the need to combine cultural and material explanations, arguing that practitioners are simultaneously tied together by production relations, formal knowledge and common culture.

My model extends Giddens’ notion of structuration to “structuration across space” —the idea that, rather than using their knowledge of their own society to structure local action, the actors may use their knowledge of a foreign society. I make this difference explicit in the second half of the chapter. I transition from this to a discussion of Appadurai’s (1996) notion of imagination, stressing that people’s actions are influenced by what they can imagine both individually and collectively, and introducing the notion of “subvocal imagination”—things that can be imagined but are too “crazy” to be defended publicly.

In the last part of the chapter I turn explicitly to the notion of peripherality,

introducing the notion of "Meccas" from the literature on social worlds (Levine 1972). I argue that we must consider not only how actors understand local and remote structure (and use this understanding as a structuring resource), but also how they see the world as a whole and their place in it. I aim for a discussion of peripherality that focuses on how peripherality is understood without questioning that it exists objectively.

Part 2: Histories and Maps

Part 2 presents two historic chapters, one focusing on individual biography and another on the larger history of information technology in Brazil.

Chapter 2.1 ("Nerds") asks how individual software developers in Brazil enter this line of work. I argue that software development is understood by many software developers in Rio (and elsewhere) as more than an occupation and that employers seek developers who "live" software, having "fallen in love" with it early in their lives. The chapter then looks in detail at the biography of a particular software professional, "Jason," starting with his earliest steps into the world of software when he was eight years old. I stress that like Jason many developers become "nerds" before they become software developers. That is, they join the software world as a culture before they enter it as a profession. I provide a broader discussion of what it means to be a "nerd," looking at how different nerd practices overlap with the practices of the software world. (The shift from "work" to "practice," articulated in chapter 1.1, becomes crucial here.) In particular, I look at how such practices require orientation towards foreign culture and systems of knowledge, and often provide early motivation for learning English, linking to the

discussion in chapter 1.2.

Teenager's entry into the culture of software development is only the first step: a youth who decides to pursue software as a *career* will have to learn to navigate the software world not only as a culture but also as a set of economic relations. I close the chapter with a discussion of this transition.

Chapter 2.2 ("Software Brasileiro") looks at the geography of the larger world of software development and the history of its entry into Brazil. The first part of the chapter considers the geographic organization of the software practice. I argue that while the software world is characterized by common practices and culture, as well as active circulation of technical knowledge and artifacts, such global flows have a clear geographic structure, with strong asymmetry between the "central" and "peripheral" areas, and with most of platform work done at the centers. I focus more narrowly on the differences between the US and Brazil, drawing on a number of data sets from each country. The second part of the chapter looks at how the software world got to Rio de Janeiro, using the history of Brazil's "enrollment" (Latour 1988a) into the software world to look at the myriad ties that link Rio to other parts of that world. I investigate how such ties were built and the work that went into their construction. The historical account is based primarily on the existing literature, supplemented with my own interviews with computer professionals who worked in Brazil in the 1960s and the 1970s. While I introduce few unknown facts about the history of Brazil's involvement with IT, the chapter adds flesh to the theoretical skeleton presented in chapter 1.3, and presents a new interpretation of this familiar history. Linking the history of Brazilian "informatics" to the

global history and geography of computing, I argue that even Brazil's protectionist IT policy of 1970s and 1980s (the so called "Market reserve") was at its essence an approach to synchronizing local practices with global models. The name of the chapter is thus an intentional misnomer.

Part 3: The Roads Ahead

In part 3 we turn to the opportunities faced by Rio software professionals today, starting with **chapter 3.1**, to which I give a Portuguese title ("Aplicações Corporativas" —"Enterprise Applications") to note the importance of local ties for the developers who take this common route. The chapter looks at "Alta"—a relatively typical (if better-than-average) software company in downtown Rio. I introduce here a new level of analysis, looking more closely at a small firm as a mediating actor between individual software developers as representatives of the software world and the local clients and illustrating the compromises involved in doing work for local clients. (There is money to be made in local work, but you have to accept work that does not fit well with your idea of what good software work ought to be about. At the same time, local success depends crucially on being able to project a global image.) I also introduce a new set of actors: young engineers educated at PUC (one of Rio's most prestigious universities), who have enough resources to start their own company, and whose experience can be contrasted with those of the less privileged developers.

Chapter 3.2 (" Porting Lua") introduces a rather different kind of project: Lua, a programming language developed in Rio that has been mentioned on many occasions in

the previous chapters, but has not been discussed in detail up to this point. This chapter focuses on the "disembedding" that Lua had to achieve to succeed, and the way such disembedding simultaneously frees Lua for its success abroad and makes it foreign in Rio. I look at this disembedding step-by-step, tracing Lua's history from a highly situated and local project to its current position. This chapter is based on my interviews with Lua's authors and its users in Rio and San Francisco, as well as mailing list archives.

Chapter 3.3 ("Fast and Patriotic") looks at Lua's relationship to Rio de Janeiro and Brazil in the recent years. I discuss the tensions between Lua's adoption in Brazil and its status as a global programming language. Looking at the conflicting positions on the possibility of viewing Lua as a "patriotic" artifact allows us to see the complex relationship between developers' commitments to Brazil and to the world of software development.

Chapter 3.4 ("Glocal Dreams") introduces us to people whose work bridges the two different worlds: the mostly local world inhabited by Alta and the mostly global community inhabited by Lua. It thus presents in the most stark form the contradictions of peripheral participation. This chapter draws on the most extensive and engaged part of my ethnography and is in some sense the key chapter of the dissertation, as it helps us to move to the discussion of what is possible. The chapter focuses on "Rodrigo Miranda," a PUC graduate who refuses to work for companies like Alta (though the thought crosses his mind regularly), instead trying to build an open source web development platform based on Lua. Instead of pursuing it as an academic research project (as he attempts during his short stint as Roberto Ierusalimschy's Ph.D. student), Rodrigo joins his brother

in an attempt to develop the platform as a startup company. The challenges they face reveal the danger of applying foreign recipes too literally and validate the common opinion that Brazilian companies need to stick with services. The project's path to open source reflects simultaneously their better understanding of their local situation and foreign rules of the game. Rodrigo's story shows the difficulties of making a "global" project at the periphery, and the balancing act between local and global resources. Rodrigo must build stronger global links, drawing on foreign technical labor. He must also find ways to deploy in the global arena the locally bound resources that he has at his command. This chapter also has the clearest implication to the discussion of open source.

1. Work as a (Global) Practice

1.1 Work as Practice

In this chapter I review an approach to studying work that I call "work as practice." (I borrow the term from Orr 1996, the last of the authors I review in this chapter.) In doing so, I aim to construct a foundation for a later discussion of how a technical practice such as software development spreads and stays synchronized across space. The literature that I bring together under the heading of "work as practice" shares important insights, as it looks at work as intertwined with individual identities and membership in social groups. At the same time, the literature brings together several theoretical traditions, with two being of particular importance: the Chicago School of Sociology (exemplified here most clearly by Hughes and Becker) and the Marxist theories of practice (exemplified by Lave & Wenger, 1991). My synthesis relies on the Chicago School tradition as the basis, bringing in elements of the theory of practice to the extent that they become necessary. (One of those elements is the concept of "practice" itself.) I do so in part because I find that the down-to-earth approach that characterizes the Chicago School fits more naturally with an ethnographic project than does the often highly abstract discussion of practice theory. This approach also helps me link my project

with a longer tradition of studying modern work ethnographically.

I present in this chapter a conservative synthesis of the existing literature, aiming to organize it, to discuss some its problems, and look at some of the solutions to those problems, arriving at an approach that I believe is defensible against the criticism to which some of the work in this tradition has been subjected. I do not aim to *extend* the theory of "work as practice" in this chapter. In the two chapters that follow, I subject this synthesis to my own criticism—for failing to provide a theory of practice in space that would work at a world scale. I do so first by presenting ethnographic observations that challenge the initial theory (chapter 1.2), and then by providing a theoretical critique and proposing a resolution (chapter 1.3), drawing on additional ideas from practice theory (Giddens in particular). Parts 2 and 3 of the dissertation then illustrate and expand the theory presented in chapter 1.3.

I present my review chronologically, starting with 1950s. The reader may wonder if books written half a century ago can help us understand such modern occupations as software development. I intend to show that they do, in three ways: by reminding us of certain ideas that are too often forgotten, by putting more recent literature in perspective, and by showing the ways in which peculiarities of software development represent not a break from the past, but rather a radicalization of phenomena observed for many decades.[33] I also believe that understanding the ideas expressed by the earlier authors, allows us to focus on the specific contributions of their intellectual heirs. Putting each

33 I borrow the concept of "radicalization" from Giddens (1990), who presents a more general argument that the social transformations of the recent decades are best understood not as an advent of a new era of "post-modernity," but rather a "radicalization of modernity."

author in their historical context also makes it easier to see how their writings respond to the challenges leveled at their predecessors.

An important problem that has characterized some of the literature in the "work as practice" tradition is the insufficient attention to the economic side of work. Many kinds of modern work cannot be understood without considering them as a form of cultural practice, that is a system of activities that is linked with the collective of people who engage in such activities and which cannot be understood without considering how individual practitioners become members of this collective, how they establish their claims to membership, and how their membership interacts with their identity. As I argue in chapter 2.1, for example, the work of software developers cannot be understood without considering the developers' identity as "nerds" and the process through which they become "nerds." At the same time, however, it is important to avoid looking at work as *just* a matter of culture. Work is also a form of participation in a system of production and is entangled with *economic* relationships. Such relationships are nearly always invested with power and the goals of the workers may not always align with those of the organizations that employ them. I refer to this aspect of work as "work as labor." Authors writing in the "work as practice" tradition have been often accused of politico-economic naiveté by those of the "work as labor" conviction (e.g., Braverman 1974). I take this charge seriously and attempt in my review to show where the literature on practice may in fact be guilty of ignoring the "labor" side of work, point out the places where this side is acknowledged and incorporated, then bring out and expand those elements. As I attempt to show in the later chapters, using software development as my case, work must be

understood *simultaneously* as a matter of both practice *and* labor. Such simultaneous analysis is especially important for understanding how practice is reproduced in space, since the reproduction of labor relationships may either run ahead or fall behind the reproduction of the cultural practice.

From Knowledge to Practice

Software development is often described as "knowledge work," and sometimes presented as the quintessential example of it—something that can help us understand other types of modern work. The immaterial, informational elements of software development are easily apparent and are quite clearly important for understanding globalization of software work. In this chapter and this dissertation, however, I take a different approach to software development, looking at it as *practice*. Such approach does not reject the importance of knowledge, but avoids reducing software work to manipulation of "knowledge." I do so out of a belief that such reduction inevitably proceeds to further reduction of "knowledge" to "information" and then to a purely informational understanding of work. The concept of "practice" helps us buttress against such a reduction. The rest of this section briefly reviews some conceptualizations of knowledge and presents an argument for looking at "knowledge work" as practice.

Modern understanding of knowledge is strongly influenced by what has been called the "codification view" of knowledge.[34] This tradition, which goes back to Diderot

34 The term "codification" is common today, see citations later in the text on the discussion of "codified" vs. "tacit" knowledge. The term "codification" was introduced by Polanyi (1966). This approach is also sometimes known as "the conduit metaphor" (Reddy 1979). See also Lakoff & Johnson (1980) for

and other Encyclopedists of 18th century, has stressed the understanding of technical knowledge as a collection of reproducible facts and techniques. Under this "codification view," knowledge possessed by skilled artisans can—and should—be converted into textual form to then be made available to all who need it. New information technologies developed in the 19th century (such as microfilm photography, developed in 1839), inspired by the end of the century a wave of excitement about the potential for reproduction and free dissemination of knowledge, led by the Documentation movement of Paul Otlet (e.g., 1925/1990) and supported by figured such as H.G. Wells (1938). Some of those ideas were later popularized by Vannevar Bush in his articles on *Memex* (e.g., 1945), and went into the Rapid Selector (Shaw 1949). With the diffusion of the digital computer after World War II, the dream of freely reproducible knowledge started to take a concrete shape. Work of early information retrieval researchers such as Luhn (e.g., 1961) and early networks researchers (e.g., Davis et al, 1967) laid the foundation of what we now know as "googling."[35] Doug Engelbart's 1962 report and Ted Nelson's visions of Xanadu (a hyper-text system later described in Nelson 1974/1987) provided a revised version of the Memex. The spread of Internet access and the development of the World Wide Web technologies in 1990s made this view even more appealing. In my own interviews with software developers, questions about how they learn what need to know to do their work often led to the same short response: "the Internet."

The codification view of knowledge has been quite influential in economics and

a discussion of the conduit metaphor as a metaphor.

35 Needless to say, information retrieval and networking are but two of the many technologies that go into making the modern Internet experience possible. However, many of the other important technologies (e.g. hard drives) become available in roughly the same time frame.

business. Since Drucker's (1969) realization that of the effect of knowledge and innovation on productivity and long term development, there has been growing interest in understanding how knowledge could be "managed." While business literature on knowledge management has sometimes acknowledged the fact that knowledge cannot be fully reduced to information in computer files, much of it bears the imprint of the codification view. When the work of Paul Romer (1990, 1993), established knowledge as a legitimate topic in economics, it brought with it a codification perspective on knowledge.[36] Romer's discussion focused on the sharing of "designs," which he described as "non-rival"—that is, in principle usable by any number of people at the same time.[37] Consequently, in Romer's (1990) view, designs/knowledge "can be accumulated without bounds on a per capita basis" (p. S75)—an assumption that established a new paradigm for economic treatment of knowledge.[38] Romer then argued (1993) that the misfortunes

---

36 The topic of knowledge was largely ignored in economics until the 1950s, at which point it entered the literature (e.g., Schmookler 1954, Solow 1957) but remained quite peripheral until the early 1990s. (The two widely cited papers on knowledge by Hayek—1937 and 1945—deal with actors' knowledge of the market and the knowledge carried by the market itself, rather than technical knowledge.) In 1965 Schmookler writes (p. 333):

> During the last dozen years or so economists have shown that the production, diffusion, and use of new knowledge are more important to the growth of output per head than is the accumulation of physical capital. It seems safe to say that the discovery occasioned more surprise among economists than among educated men generally. The differential surprise is an instructive example of how damaging to the understanding professional knowledge can be sometimes.

After brief excitement about knowledge in 1960s, however, it was abandoned as a topic of research once again. For the discussion of the marginal role of knowledge in economics in 1970s and 1980s see Romer (1990).

37 More precisely, Romer (1990) argues that "designs" are non-rival (i.e., any number of people can be using them at the same time without cost to each other), but potentially *excludable* through an artificial mechanism of intellectual property protection—people can be banned from using knowledge by legal means. They are thus not a pure "public good" since traditional public goods (such as clean air for breathing) are both non-rivalours and non-excludable.

38 Of course, while in principle knowledge is reproducible without limits, such reproducibility can in reality be limited by law to allow creators of knowledge to earn money by selling rights to reproduction. Economics of innovation often works with an assumption that no innovation would

that afflict many countries would be best understood by considering the "idea gap" between them and the more successful countries like the United States. Such "idea gaps," Romer argued, are "relatively easy to solve," and consequently "idea gap explanation for persistent poverty offers a more optimistic picture of the potential for rapid development" (p. 546).[39]

The increasing popularity of the codification vision led to a critique of this view. The question of whether all knowledge can actually be codified in principle (apart from the technical issues of storage, retrieval and transmission explored by engineers) first attracted the attention of philosophers. Ryle (1949) proposed a distinction between "knowing that" and "knowing how," and Polanyi (1952, 1966) introduced the concept of

happen without such monetary incentives and that use of innovative ideas must thus be restricted by law. (See Gallini & Scotchmer 2002 for a discussion of prizes and contracts as alternative to intellectual property.) This is a point at which the economic and the technological visions of knowledge split paths, since many proponents of the engineering vision have increasingly stressed the creation of replicable knowledge "just for fun."

39 The persisting patterns of inequality (which continue today, years after Romer's paper), present an obvious problem for this optimistic view—a problem that Romer does recognize and address briefly. One of the reasons he mentions is the possible lack of "the basic institutional infrastructure required for market exchange" (p. 547). The recognition of the importance of "infrastructure" is an important insight, similar to Latour's (1988a) discussion of "tracks" on which knowledge travels (see below). Romer, however, trivializes the idea of infrastructure, treating it as a singular hurdle and suggesting that ideas are easily replicable once some *basic* infrastructure is achieved. Romer's second explanation (which he stresses as more important) is "an aversion in some developing countries to any contact or exchange with firms from industrial economies" (p. 548). While this may be true of a small number of countries (and may have been true of a larger number in the past), I believe such "aversion" is less common than is often assumed. Even in case of Brazil, a country notorious for its protectionist policy in regards to information technology, it would be wrong to talk about an "aversion" to exchange with foreign firms. As I show in chapter 2.2, Brazil's protectionist policy should be best understood not as an attempt to isolate the country from foreign firms and their knowledge, but rather (among other things) an attempt to *buy* technical knowhow from those companies that were willing to sell it, offering select firms access to the Brazilian as a form of payment. (Note that Brazil lacked money to pay for "knowledge" directly and the major computer companies, e.g., IBM, were not interested in selling their know-how anyway.) Even if we see protectionist policies of 1970s and 1980s as a sign of "aversion" rather than as a matter of haggling over prices (or of attempting to reproduce at home what was simply not available for sale), such policies are now in the past. Since the early 1990s, it would be hard to see any signs of aversion to foreign companies or their technology in the Brazilian government. The same is true of software professionals, who in fact often appear more interested in learning about technology developed abroad than about what happens in Brazil.

"tacit dimension" of knowledge, arguing that "we know more than we can tell" (1966, p. 4). The idea of tacit knowledge was then developed further by scholars in the Science and Technology Studies tradition, such as Collins (1974) who looked at the diffusion of knowledge regarding a laser design (and later, in Collins 2001, at the difficulties experienced by British physicists attempting to replicate a Soviet experiment), and MacKenzie & Spinardi (1995) who discussed "the uninvention of nuclear weapons" in the United States. Some of the discussion of 'tacit knowledge' focused on the simpler observation that communication always relies heavily on prior common ground and typically references the physical environment.

The term "tacit knowledge," however, suffers many of the shortcomings of the codification view of knowledge. The use of the word "knowledge," a noun, positions tacit knowledge as another type of "knowledge," somewhat of an evil twin of the "normal" (codifiable) knowledge. (See, for example, Cowan et al. 2000.) The term "tacit knowledge" invites us to continue thinking of knowledge as a "thing" and something an individual (or a group) can "have."[40] Yet when we ask where it is, tacit knowledge proves to be an illusive, mysterious concept. Additionally, while authors who argue for tacit knowledge typically introduce a distinction between "knowledge" and "information," such distinction is often forgotten or employed inconsistently when the ideas developed by those authors are popularized. I suspect that the term "knowledge" is today so strongly connected to the idea of replicable designs and electronic transmission of files, that I

40 As Duguid (2005) points out in responding to Cowan et al. (2000), Polanyi (1966) argues for a tacit *dimension* of knowledge, rather than for "tacit knowledge" as a type of knowledge. Ryle (1949) similarly talks about "knowledge that" and "knowledge how" as interdependent.

cannot expect my reader to simply put those associations aside.

Instead, I use a different term, and look for a different theoretical foundation. I find such a foundation in the work of Becker (1953, 1972), Lave & Wenger (1991) and other authors discussed below, and systematize it using Lave & Wenger's term "practice." I define "practice" as a system of activities, a collective way of doing certain things. Understood this way, "practice" subsumes both tacit and codified dimensions of knowledge. In this process, "tacit knowledge" disappears as a distinct category. That is, I do not find it useful to draw a distinction between "tacit knowledge" and "practice" and limit myself to saying that "tacit knowledge" is a part of practice. In contrast, knowledge codified in documents becomes highlighted as an *element* of practice—an important one, yet one of many. I focus on what people *do* and how they come to change how they do things, and then approach the specific knowledge-processing activities—such as reading, writing, and other forms of communicating—*in the context of this doing*, considering them side-by-side with the many aspects of practice that cannot be reduced to acquisition of information.

Looking at practice rather than acquisition of knowledge, however, means more than just looking at what people *do*. To follow Becker (1953, 1972) and Lave & Wenger (1991) is to recognize that practice is a deeply collective and relational phenomenon which transforms the individual, making her or him a member in a collective of practitioners, enmeshed in a new web of relationships. Looking at engineers, for example, it is not sufficient to ask how someone acquires the knowledge of engineering or starts to engineer things. Instead, we must consider the full implications of *becoming an engineer*

—the individual's entry into a network of socio-technical relationships, which will transform how they see other people, objects and themselves, and how others will see and interact with them. Learning, according to Lave & Wenger, is "the historical production, transformation and change of persons" (p. 51). Such "relational view of the person and of learning" (p. 53) implies that "activities, tasks, functions, and understandings do not exist in isolation; they are part of a broader systems of relations in which they have meaning" (p. 51). To understand practice and learning, we will therefore have to look at them in the context of such broader systems, considering also how the participants themselves reflect on those broader systems. This view also implies that newcomers do not merely acquire the knowledge that was out there before they arrived, but rather become a part of a system, changing it by their very arrival.

The argument that learning must be understood as a matter of progressive engagement in a community of practitioners was most famously presented in Lave & Wenger's (1991) *Situated Learning: Legitimate Peripheral Participation*.[41] This important book has been widely read in a number of different fields, and interpreted in a number of ways. I look at Lave & Wenger in the context of a particular tradition, the "Second Chicago School" of sociology,[42] which I believe gives it its fullest meaning, while also

41 As was suggested in the previous section, this argument involves two analytical moves: the change of focus from acquisition of knowledge to practice, and from practice as something that people *do* to something of which they *become a member*. Both of those moves are necessary, since making the first move without the second merely replaces a cognitivist explanation with a behaviorist one. Lave & Wenger then make a third analytical move, from practice in the narrow sense to practice in the "historical, generative sense," which I will discuss below as well.

42 Fine (1995) uses the term "Second Chicago School" to refer to the generation of sociologists associated with the Department of Sociology of University in Chicago in the decades following the end of the World War II, whose interests, theories and methods differ substantially from those of the pre-war Chicago School tradition (most strongly associated with Blumer, who left the University of Chicago for Berkeley in 1952, and Park). As other contributors to the volume, Fine cites Howard

helping us see the work's most important limitations. While certain important elements of *Situated Learning* do not fit in this tradition, looking at Lave & Wenger in connection with the (second) Chicago school makes it possible to link their theoretical discussion to a wider body of ethnographic work, as well as to very similar theoretical accounts.

In this chapter I review the work of Hughes, Becker, Van Maanen & Barley, Lave & Wenger, and Orr, providing my synthesis of their respective theories of work and practice. (Later, in chapter 1.3, I also incorporate some insights from the work of Anselm Strauss and other authors who contributed to the Chicago school literature on "social worlds.") While discussing their respective takes on the relationship between learning, practice and communities, I consider the fact that they use different terms to talk about similar, but not identical, concepts. Hughes writes of "work" or "occupations" (1958), Becker of "activities" (1953) or "groups" (1963) or without using any technical terms at all (1972),[43] Van Maanen & Barley (1984) talk of "occupations," Orr (1996) or "work" and "practice," and Lave & Wenger (1991) talk of "practice." "Work," "occupation" and "practice" are distinct but intertwined concepts. I look at work as a kind of practice, and one of the most important ones to study, since work is in many ways central to the functioning of society (see the discussion of Hughes below). Yet we cannot fully understand modern work without looking at how it relates to other types of practice—

Becker, Erving Goffman and Anselm Strauss as the most prominent icons of the "Second Chicago School," pointing to the importance of Everett Hughes as the linking figure. (Hughes was a student of Park and a mentor of Becker, Goffman and Strauss.) Coser (1994) attributes to Hughes the same role. John Van Maanen, Stephen Barley and Julian Orr do not belong to the same cohort of sociologists (nor are Barley and Orr "sociologists" in the narrow sense of the word), but their work fits closely in the tradition of study of work started by the "Second Chicago School."

43 Becker's writing often shows a clear desire to avoid introduction of "technical" terms that would suggest an attempt at "Theory" with capital "T." Becker thus often talks about "jobs", "trades" or "occupations" but strives to use those as lay terms, without assigning to them any special meaning.

those that we would hesitate to call "work"—and understand what it has in common with those practices. In case of software development, we must find a way to account for the fact that some of the world's most important infrastructure today runs on an operating system much of which was written either "just for fun" or out of moral calling.[44]

The research tradition that I trace from Hughes has been subjected to an important critique by Marxist scholars of work, who have come to dominate sociology of work in 1970s (see Simpson 1989 and the discussion below). Similar critiques have been later applied to the more recent authors in this tradition (see Contu & Willmott 2003). I believe that the two traditions can be reconciled and that we must see work as being a matter of participating *simultaneously* in a cultural community and system of labor relations. Without attempting to work out such reconciliation fully in this chapter, I try to make explicit the disagreements between the traditions, setting up ground for a more balanced discussion in the later chapters.[45]

Occupations as Named Parts in a System of Division of Labor

Everett Hughes, a transitional figure between the "first" and "second" Chicago

44 The GNU/Linux operating system (commonly known as just "Linux"), which today is used by a wide range of companies (from Google and IBM to Microsoft), as well as the US Army consists of two key layers of software. One of those layers—the Linux kernel—was allegedly written "just for fun" (the author's own account was published under the title "Just for Fun"—Torvalds 2001). Another layer—the GNU system—came out of a determined and lengthy effort (1984 to the early 1990s, and continuing since) to provide the community of software enthusiasts with a freely available alternative to the proprietary UNIX operating system. Richard Stallman, the leader of the GNU project stresses that not all of that was fun, and that developing GNU was a matter a moral stand (see, for example, Williams 2002, Stallman 2002).

45 It should be clear that my own approach to the issue leans much more heavily on the Hughesian tradition than it does on the Marxist one. My goal, therefore, is not to merge the two approaches fully, but to bring certain elements of the Marxist critique into what overall is a Hughesian framework.

schools of sociology, is often recognized as a founding figure in the modern sociology of work.[46] Hughes offers a theory of occupations that highlights the relationship between work and identity (at least for skilled work), looking at occupations as social groups, while at the same time considering their relationship with other occupations and their position in the societal division of labor. This second aspect of Hughes writing is often downplayed in the work of his followers, who often focus more fully on occupations as cultural groups, rather than units in the larger society. Revisiting the work of Hughes will help us establish a balance between "work as practice" and "work as labor" later in the chapter.

Hughes writes about "work" without defining this term but seemingly using it in a way that does not differ much from the vernacular usage. Unlike Marx, who sees "work" as a fundamental relationship between man and nature (see below), Hughes looks at work mostly in the context of contemporary industrial society. He associates "work" with participation in the "labor force" and "formally organized work activities... named and categorized in payrolls, organization charts and union-management contracts, and in income-tax, licensing, and social-security legislation" (1959/1994, p. 21-22).[47] He also

---

46 See Simpson (1989) and Ritzer's (1989) response for an overview of the history of "sociology of work" in the 20th century. Ritzer in particular sees Hughes's course on "Sociology of Work," first taught in 1939 (p. 595) as establishing the field. As most reviews of the field's history, Simpson and Ritzer both point to the work of Elton Mayo (e.g., 1933) as the precursor of the 1950s literature. (Burawoy's 1979 review of "anthropology of industrial work" does this as well.) To save space, I start my review with Hughes.

47 Hughes's notion of work appears to exclude unpaid work done by women, and housework in particular. The importance of such work has been since discussed by feminist scholars (e.g., Hochschild 1989). For the purposes of my argument it does not matter whether or not the concept of "work" includes unpaid work. Whether the work is recognized as "work" in the society is important, however. Invisible work of the kind described by Hochschild comes without a "calling card" (see Hughes's quote below), which has implications for how it affects the identity of the people who do this work.

contrasts “work” with “leisure” (p. 22-23). Hughes sees “work” as often defining the course of an individual’s life: “A man’s work is as good a clue as any to the course of his life, and to his social being and identity” (1958, p. 7). The type of work the individual does is the most important determinant of social status in the industrial society: “Race, sex, marital status, and other characteristics formerly determined civil estate quite directly; now it is work that counts... and the other characteristics take their importance by virtue of their influence on one’s place in the labor force” (1959/1994, p. 22).

Hughes focuses on skilled work, which he sees as divided into *occupations* (or more specifically *named occupations*). An occupation is defined simultaneously by a particular type of work activities, a group of people who carry them out, and their role in the division of labor. Hughes’s stress on the division of labor *between* occupations (drawing heavily on Durkheim’s notion of “organic solidarity”) is important to note here, since it is largely left out by much of the later “Chicago School” literature on work and practice. Specifically, Hughes writes that “an occupation, in essence, is not some particular set of activities; it is the part of an individual in any ongoing system of activities” (1959/1994, p. 24). He then re-defines an occupation as “a more-or-less standardized one-man’s part in some operating system” concluding from this that an occupation “cannot be described apart from the whole” (p. 30). In introducing the notion of “*a more-or-less standardized one-man’s part*” (my emphasis) in the larger division of labor, Hughes draws a link between the division of labor *between* individuals in a specific enterprise (as in Adam Smith’s famous example of the pin factory), and the division of labor *between* occupations in the context of a larger industrial society. He suggests, in

other ways, that the division of labor between individuals is not organized on an ad hoc basis in every enterprise. Rather, the work tends to be divided in more or the same roles, which allows us to say that people who take the same role in different enterprises comprise the same "occupation."[48]

At the same time, Hughes stresses the personal significance and the cultural aspects of occupations. He sees many occupations as characterized by shared "culture" and "technique." Hughes ties the technique on the one hand to the culture shared by those who engage in the occupation, and to "a set of collective representations, more or less peculiar to the occupation and more or less incomprehensible to the community." He stresses that the practitioners of occupations have a distinct way of relating to certain objects and activities: "The technique of the physician is in relation to the human body, which must be for him a different sort of object from what it is for the layman. To the layman it is a sacred thing, and an object of sentiment." (1958, p. 35.) Members of the occupations see familiar objects in different ways. Even more importantly, they typically have a distinct way of seeing *the activity itself*.

Named occupations "are a combination of price tag and calling card" writes Hughes (1951/1958, p. 42). The "calling card" metaphor links two sides of the occupation: an occupation is a crucial source of individual identity, but it is important as a source of identity largely because it identifies the individuals' role in the industrial society. This metaphor and the term "*named* occupations" (my emphasis) bring together what I later (chapter 1.3) call the "external" and the "internal" view of "named"

48 The relationship between occupations (or more narrowly "professions") is later developed further by Abbott (1988), who talks about "a system of profession."

categories of practitioners.

Insiders and Outsiders

Hughes's notions of "culture" and "technique" are closely tied to those of *membership* and *license.* "An occupation consists, in part, of a successful claim of some people to *license* to carry out certain activities which others may not, and to do so in exchange for money, goods and services," writes Hughes (1958, p. 78). The "license"[49] to perform certain activities is given to a *group.* The individual who seeks permission to engage in those activities must gain membership in the group that has the license: "the practitioners, by virtue of gaining admission to the charmed circle of colleagues... exercise the license to do things others do not do" (1958, p. 79). This leads to a question of how membership is granted—a question that Hughes, to the best of my knowledge, does not answer. The work of Hughes's student and co-author Howard Becker, however, helps shed light on this issue.

Howard Becker, one of the most well-known figures of "the Second Chicago School," started his career with a study of Chicago jazz musicians. His 1951 paper on the subject (Becker 1951) introduces the study as a case of conflicts experienced by "members of service occupations" (p. 136) and Hughes incorporates Becker's study as an example of an occupation (1951/1958, one of Hughes's early publications on the subject of occupations). However, Becker later reprints the study in *Outsiders: Studies in the*

49 Hughes uses the word *license* figuratively, except when talking about "professions," where members are literally licensed.

Sociology of Deviance (1963). This re-conceptualization of the practice of Chicago jazz musicians from "an occupation" to membership in "a deviant group" may seem puzzling, but in fact provides us with an opportunity to get a deeper understanding of Hughes' discussion of occupations as groups.

While viewing jazz musicians as first an occupation and then as a "deviant group" may seem strange, we should remember that one of the key aspects of an occupation for Hughes is that its practitioners *do things that other people do not do.* Becker's notion of "deviance" does not correspond precisely to Hughes's (and early Becker's) "occupation," but those two kinds of groups share enough similarities to warrant a comparison. Becker defines as "deviant" activities that are labeled as such by groups that do not engage in them.[50] He explains his inclusion of Chicago jazz musicians among "deviant groups" (in a chapter following one about marijuana users) by the fact that "their culture and the way of life are sufficiently bizarre and unconventional for them to be labeled as outsiders by more conventional members of the community" (1963, p. 79). Both types of groups are thus characterized by "a culture" and engage in activities that other people do not do. The question of whether the musicians form "an occupation" or a "deviant group" would thus presumably rest on whether they have a *license* to engage in the activities that they engage in. There is a marked difference, however, between Hughes and Becker as to *whose licenses is relevant*. Hughes talks abstractly about "the society" allowing people to do certain things. Becker is careful to be more specific. The jazz artists do not generally lack

50 This later became known as the "labeling theory" of deviance and crime. More specifically, Becker writes: "I mean, rather, that *social groups create deviance by making the rules whose infraction constitutes deviance,* and by applying those rules to particular people and labeling them as outsiders" (1963, p. 9, original emphasis).

society's license to play: their activities are legal and they have audience that enjoys their music. Rather, they are labeled as outsiders by specific (if powerful) other groups.[51]

Becker's insight has important consequences for the discussion of occupation. Becker urges us to avoid looking at rule-breaking without asking who creates the rules and when they choose to enforce them. Deviance becomes a matter of group boundaries and their enforcement. Applied to studies of work this suggests that we might want to question Hughes distinction between activities for which society does or does not give license to certain groups (and more generally the idea of a monolithic "industrial society"), and instead look at people who engage in certain activities, form a group (recognized by themselves and often by outsiders) by virtue of such engagement, have a distinct "technique" for performing the activity, and a special way of relating to this activity and the relevant objects. Looking at it this way, we quickly realize important similarities between Hughes's doctors and Becker's marijuana users.

Becker's study of deviant groups (1953, 1963) develops Hughes's notions of "culture" and "technique" and presents a model of the process by which a newcomer enters the group, presaging some of the ideas of Lave & Wenger (1991). Becker's theory of progressive engagement in the activity (read with the hindsight assisted by Lave & Wenger) can be summarized as follows. The novice starts off with a limited interest in the activity, perhaps seeing little point in it (or worse, seeing it negatively). If they were to try to engage in the activity by themselves anyway, they would likely fail to obtain the expected results, lacking the necessary *technique*. (A first-time user of marijuana will

51 Such commitment to being clear about who the actors are is something that Becker not only practices in his own writing but has also taught to others through his methods books, e.g. Becker (1986, 1998).

likely burn her throat by inhaling incorrectly. A wannabe programmer will likely write code that will not run.) If they do obtain the results, they might fail to recognize or appreciate them. (A novice marijuana user might fail to recognize being "high" and if she does might fail to find it enjoyable. A novice programmer might not recognize the joy of getting the code to compile.) Becker writes that the user must learn to *define* the effect of the activity as enjoyable, acquiring what Becker calls "a conception of the meaning of the behavior, and perceptions and judgments of objects and situations that make the activity possible and desirable" (p. 235). (I will further use the terms "perceptions and judgments" or simply "perceptions" as short-hand.) Experienced practitioners become important here, as they help the novice learn the basic technique and to develop the necessary "perceptions and judgments." This is not to suggest that they become the primary source of learning: rather they help the novice make early steps into the activity, and it is from the combination of their advice and own experience that the novice learns the perceptions and the technique.[52]

The shared techniques and dispositions are for Becker (1963) the *culture* of the group united by those activities.[53] While the novice acquires the culture of the group by engaging in the activity, this should not be understood as two distinct steps. As Becker shows, engagement in the activity is not a matter of a single decision, but is often a sequence of steps, each step leading the novice to changes in dispositions that make the next step sensible. Becker illustrates this best when he describes (Becker 1998) a

52 This is quite similar to Vygotsky's "Zone of Proximal Development" (Vygotsky 1978).

53 Note that Becker includes "technique" in "culture" while Hughes talks about "culture and technique." Becker's inclusion of technique in "culture" (together with dispositions) makes sense, considering that he sees acquisition of technique as acquisition of proper dispositions.

deliberately extreme case of gradual changes that might lead a man to a sex change operation. Deepening engagement in the activity thus happens in lockstep with adoption of the culture of the group that practices that activity. We might add to this, that the entry into a subculture of the group defined by an activity may well precede direct contact with the actual activity.[54]

Becker (1963) sees this shared culture as arising from shared problems: "Many people have suggested that culture arises essentially in response to a problem faced by a group of people, insofar as they are able to interact and communicate with one another effectively" (p. 81). He notes, however, that not all activities develop a culture around them, since some activities are carried out in isolation, perhaps because they are stigmatized so much that those who engage in such activities choose to share this fact with nobody. (Becker uses the kleptomaniac as an example.) It is also clear that some activities, on the other hand, are so common that they do not lead to distinct group culture —instead the techniques and dispositions necessary for such activities are already present in the mainstream culture of the place.[55]

---

54 For example, Desmond (2007, p. 69) writes about firefighters:

> The process of becoming a firefighter begins long before young people join firecrews; it begins during their childhoods with thousands of experiences specific to their upbringing. In this way, firefighters acquire many of the skills and attitudes needed to chase smoke long before they step onto the fireline.

I discuss this process for software developers in chapter 2.1 ("Nerds").

55 Becker (1963) talks about "the mainstream culture" generally, but it is important to keep in mind that there is no such thing as "the" mainstream culture. Understood as a shared set of techniques and dispositions, a culture is always linked with a group and is typically mutually constituting with it. (See the discussion of Van Maanen & Barley 1984, 1985 later in this chapter.) "Mainstream" is then always a relative term. One culture could be "mainstream" for another culture in the sense it the first culture corresponds to a group that forms a superset of the second culture's group. ("The American culture" is mainstream in relation to the culture of American jazz musicians.) This becomes particularly important in later chapters, as we take an international perspective on practice and consider that

Becker's theory, as summarized so far, illuminates the process by which the individual may come to engage in an activity, in a way that can be clearly applied to the process of entry into what Hughes would call occupations. It mostly leaves out, however, Hughes's stress on the division of labor between occupations. It seems clear, that to understand the continuous use of marijuana in the United States or the continuous practice of jazz, software development, or medicine, we would need to not only look at how an individual comes to engage in those activities, but also at how such activities interact with other activities, considering, for example, the relations between users of marijuana and dealers or police. Becker raises this question in his first paper about the Chicago jazz musicians (1951), looking at their interactions with the clients, but limits his discussion to the musicians conflicted feelings about the clients.[56]

Work as Labor

While most sociology of work in 1950s and 1960s shows influence of Hughes and other Chicago School authors, the field was transformed in 1970s under the influence of a different tradition: Marxism.[57] Although the Chicago School relied heavily on Durkheim

---

occupational groups may span large areas and in this sense may be more "mainstream" than the local "mainstream" cultures from which they diverge. (For instance, the choice of English or Portuguese for software documentation can be described as "mainstream" depending on the frame of reference.) In this dissertation I use the word "mainstream culture" for cultures that dominate specific geographic places, cutting across occupations.

56 Becker later explores the division of labor between practices in *Art Worlds* (1982).

57 The transformation started in 1960s as many scholars started searching for the more formal alternatives to the ethnographic approach of the Chicago School, partly under the influence of Parson's ascending structural-functionalism (see Simpson 1989, Ritzer 1989). Such search for formal and macro accounts may have paved the way for neo-Marxist approach to work. I do not discuss this decade to save space.

(for instance, the ideas of "organic solidarity" and "industrial society") and Weber, and while Durkheim and Weber had in turn been influenced by Marx (see Giddens 1971), Chicago School sociologists do not draw on Marx directly and their approaches differ from Marx's substantially, usually lacking interest in radical criticism, longer-term historical change and relationships of production. Marxist literature similarly developed up to 1970s independently of American sociology.[58] Braverman's (1974) *Labor and Monopoly Capital* (subtitled "The Degradation of Work in the Twentieth Century") and other Marxist writings of 1970s, however, had a great impact on American sociology of work.

Braverman, a Marxist coming from the socialist movement rather than academia, explicitly enters a debate with the literature on occupations, arguing that under capitalism work is increasingly deskilled and alienated. Alienation of labor in the capitalist society is one of core principles of Marx's thought, which Marx contrasts with what he sees as the natural (unalienated) labor. In Marx's writing, labor is understood as human efforts to transform their material environment to fit their needs for survival. Such efforts are defining of the human existence and serve as the foundation of any human society. In other words, no human society can be understood without paying attention to how it organizes efforts towards producing things that people need for survival. Unalienated labor involves collective effort to transform the material environment to serve the needs

58 Indicative of this fact, Burawoy (1972) talks about "sociology" and "Marxism" as two distinct lines of thought. While he points out the influence of Marxism on sociology ("sociology has borrowed much from Marx and emerged in part through a debate with him," p. xiii), this influence is presented as a matter of one tradition influencing another. In contrast, today's canon—as taught in the United States—includes Marx as a *founder* of sociology, not an external figure who influenced it (see, for example, Giddens et al. 2007).

of the group. Under capitalism, however, the workers do not transform their material environment towards *their* needs. Instead, they offer their labor to the capitalist in exchange for money. Labor then becomes "alienated" (alternatively "estranged" or "externalized"[59]), that is external to the worker (Marx 1978, Giddens 1971, Braverman 1974). Marx then makes an additional argument that the division of labor inside a capitalist enterprise requires the workers to increasingly focus on small tasks, further alienating them from the product of labor. Further, mechanization makes those tasks increasingly simple, making the worker "a living appendage of the machine" (Marx 1867/1977, p. 614).

Braverman (1974) extends Marx's theory, arguing that since the capitalist purchases from the worker abstract labor rather than specific products of labor, organizing this labor in order to maximize productivity becomes the responsibility of the capitalist. (Such organizing is achieved through capitalist' agents: the management.) Since the workers have no intrinsic interest in the work, the capitalist cannot rely on the workers to apply their labor in the most productive ways. The capitalist's solution, according to Braverman, is to capture the knowledge that was once in possession of the practitioners of different crafts, and then prescribe to each worker exactly what to do. (Braverman thus relies on a theory of knowledge that is akin to "the codification view" described earlier, though he focuses on the disempowerment of the worker, rather than the empowerment of "the society." Braverman's acceptance of the positive but not the normative side of the

59 According to Giddens (1971), Marx uses two different German words interchangeably, both of which are traditionally translated as "alienation": *Entfremdung* ("estrangement") and *Entäusserung* ("externalization").

codification view, points out the political assumptions—and class interests—inherent in the standard articulation of the codification view.)

Such organization of "the labor process" completes alienation of labor, as it removes the need to rely on workers' creativity, and allows the capitalist to simply buy workers' time. This "degradation of labor" depresses wages, since any worker can in theory do any work. Additionally, Braverman argues, the worker's "working time becomes sharply and antagonistically divided from nonworking time, and the worker places an extraordinary value upon this 'free' time, while on-the-job time is regarded as lost or wasted" (p. 278).

Braverman also attempts to update Marx by presenting an argument against "middle layers" of employment. Much of those workers have been absorbed into the labor market, he argues, and their work is also increasingly deskilled. While Braverman makes his argument most convincingly for clerical workers, he suggests that even "professionals" (Braverman uses this term in quotations) are not exempt from the alienating effects of capitalist labor process.[60] Braverman thus presents an image of work that stands in stark contrast with that painted by Hughes (1958). Hughes's understanding of work is most certainly influenced primarily by his understanding of skilled work. While he considers briefly unskilled work ("jobs," which require no more than "to present one's self at the proper time and place when manpower of a certain age, sex, and perhaps a certain grade of intelligence, is wanted," p. 34), he seems to treat such work as

60 For example, he writes: "By the end of the 1960s, however, the rising rates of unemployment among 'professionals' of various kinds once more brought home to them that they were not free agents they thought they were, who deigned to 'associate themselves' with one or another corporation, but truly part of a labor market, hired and fired like those beneath them" (p. 408).

atypical of the modern society, focusing instead on work that does require skill (which he calls "professions," "arts" or "trades"). Braverman, on the other hand, treats the professions as an exception, suggesting that for most workers work increasingly lacks any meaning.

Braverman's book greatly influenced sociology of work (see Simpson 1989, Ritzer 1989), leading to a focus on deskilling and on the importance of differences among the different workers. Much of the ensuing discussion questioned Braverman's argument.[61] Form (1987), for example, argues that Braverman overestimates the importance of skilled work in the 19th century and that Taylorism had less effect on work than Braverman assumes and was only implemented in industries that already relied primarily on unskilled labor.[62] After considering the difficulties involved in measuring or even defining "skill," Form concludes that there is no support for deskilling across the board, and that the effects of technology and mechanization vary greatly by industry. The argument about deskilling continues until today, without any unambiguous conclusions, but with overall stronger support for upskilling than for deskilling. Perhaps the most important lesson that we learn from this literature is that there are very different kinds of work.

Some authors have used this diversity of skills to argue for the emergence of a "dual labor force" (see Simpson 1989 for a review), divided into skilled and privileged "core" (or "primary") labor force and deskilled "peripheral" labor force. This argument is

---

61 Form (1987) argues that some of this discussion predates Braverman's book (1974), and Braverman himself reviews some of the literature. However, there is little question that Braverman's book brought this discussion to the front of the stage.

62 In other words, Form argues that Taylorism has been applied successfully to make unskilled workers work harder, but has not been useful for reducing the need for skill.

later restated by Castells (1996/2000), who identifies the core labor force with "information-based managers" (p. 295). As the rest of this dissertation suggests, the boundaries between the two cannot always be drawn clearly.

Braverman's (1974) argument assumes, following Marx, that workers engaged in repetitive manual labor cannot find meaning in such work. Some of the other Marxist authors have disagreed with this. Burawoy (1972) presents an interesting fusion of Marxism and non-Marxist sociology:[63] drawing on his ethnography at a plant he argues that workers *do* identify with their work and put substantial effort into it because they redefine it as a game. In other words, while Burawoy accepts the Marxist premise that hard work in the factory goes against the workers' class interests, he draws a distinction between the objective meaninglessness of work (as seen by the social theorist) and the workers' subjective experience of this work.

Willis's (1977) *Learning to Labor* presents a similar argument, with even more powerful ethnographic evidence. Willis looks at how working class culture is reproduced in British schools, pointing out how the students' counter-school culture prepares them for acceptance of manual work. Willis's "lads" see themselves in opposition to "ear'oles" —the students who buy into the school's agenda (and presumably later move on to clerical work). They see manual work as masculine, "rough" work that not everybody could do.[64] They also see manual work as allowing them freedom of mind: requiring them

---

63 Burawoy (1972) does not identify the non-marxist side of his analysis with Hughes explicitly (whom he cites only parenthetically). However, that side of his theory aligns well with what I consider a Hughesian tradition.

64 See Lamont's (2000) *The Dignity of Working Men* for a more recent discussion of the way blue-color workers perceive their work.

to perform certain activities but allowing them to see those activities as they like.[65]

This dissertation draws on ethnography of software developers—people typically classified as "professionals." The image of work that it paints is clearly more similar to that drawn by Hughes and even Becker than to the one presented by Braverman (1974). I do not accept Braverman's argument that professionals represent a disappearing breed and that technical expertise is concentrated in just about 3% of the labor force.[66] (Though, they clearly do not represent the typical case, as one might infer from Hughes's writing.) At the same time, I take from Braverman the key insight that even skilled work must be understood as a form of labor. To be more precise, I believe that it must be understood as often being a matter of both practice *and* labor.

The Marxist critique presented a challenge of reconciling the cultural understanding of work with the realities of work as a "labor process." This reconciliation is partly provided by Van Maanen & Barley (1984). Speaking primarily to the organizational studies audience, Van Maanen & Barley argue for looking at workers as

---

65 Willis argues that middle class professionals "are being paid for what they *are*" (p. 131, original emphasis), while the *weekly* wages of the factory workers symbolizes the recognizes the selling of labor power. This arrangement, in certain sense leaves manual workers free to *be* what they want to be. At the same time, Willis's ethnography shows clearly the ways in which the occupation is intertwined with who the workers think they are.

66 Braverman (1974) writes:

> [W]hat is remarkable is the concentration of the technical expertise of United States industries in a relatively small grouping. Taken together, the technical engineers, chemists, scientists, architects, draftsmen, designers, and technicians represented not much more than 3 percent of the population in 1970s. (p. 241.)

Considering that Castells puts the number of professionals and technicians at 14.2% of the labor force for 1970 (vs. 16.9% in 1991, Castells 1996/2000, citing US Labor Statistics), Braverman either is wrong about the numbers or is willing to grant monopoly on technical knowledge to just a small fraction of professionals and technicians. A number of scholars (Orr 1996, Barley & Bechky 1994) have since argued that technicians work involve more knowledge than is commonly assumed.

embedded in "occupational communities." They define an occupational community as:

> a group of people who consider themselves to be engaged in the same sort of work; whose identity is drawn from the work; who share with one another a set of values, norms and perspectives that apply to but extend beyond work related matters; and whose social relationships meld work and leisure.

Like Hughes, they stress the fact that work is often carried out in bounded occupational groups, the members of which share culture and identity. Their paper elaborates this aspect of Hughes's approach, mostly leaving aside (as does early Becker) the question of division of labor between the occupational groups.

Van Maanen & Barley (1984) for the most part do not engage in an explicit debate with the Marxist literature.[67] At the same time, they present an image of work that is largely aligned with that of Hughes and Becker, but reinforced against some of Marxist critique by explicit discussion of the organization. They also apply the concept of "occupational communities" to a many different types of work, including work that is typically classified as "unskilled" (e.g., janitors or the street sweepers of Benares)—thus going explicitly against Braverman's claims.

Van Maanen & Barley's analysis incorporates the idea that members of an occupational community stand in a particular relationship with organizations that employ them, essentially recognizing that this relationship involves in part the sale of labor

---

67 They do briefly argue against Braverman's notion of deskilling in a different article (Van Maanen & Barley 1985), using Barley's study of CT scan technicians as a case of upgrading of skills, though they attack specifically the idea that *all* types of work become deskilled due to technology. Introduction of CT scans, they write, allowed some of the former X-ray technicians to become "CT techs" and engage in work that required new skills and gave them more autonomy and responsibility. They argue that "while a radiologist or a referring physician would never ask an X-ray technologist to read a film, such requests are routinely made of sonographers and CT techs" (p. 44).

(though they do not use this term). To some extent (and only while employing a micro-sociological perspective), we can see parallels between what Braverman (1974) calls "the capitalist" and "the capitalist enterprise" and what Van Maanen & Barley (1984) call "the organization." However, while Braverman would see the enterprise and the workers as being fundamentally in conflict, Van Maanen & Barley stress the complexity of this relationship. They see individual workers as participating simultaneously in two groups: the occupational community and the organization. They see this dual membership as often being a source of conflicts, which arise from the misalignment between the interests of the workers as members of the occupation and the organization. At the same time, they recognize that goals of the two may align to a substantial extent and that for many occupations the organization is a necessary evil, without which the occupational tasks cannot be practiced: "certain kinds of engineers often discover that their collective aims and identities can be satisfied only within large, heteronomous organizations where sufficient resources to pursue occupationally-valued ends are to be found" (p. 310).[68]

Van Maanen & Barley draw on a theory of motivation that is rarely employed in Marxist thought. Like many other Marxists, Braverman (1974) seems to recognize two theories of motivation: people may engage in work for the sake of the products that their work creates, which they then use individually or collectively *or* they may sell their labor for money, which is then used to obtain products. Both theories are fundamentally *economic*. Van Maanen & Barley, on the other hand, recognize, following Hughes, Becker and others, an entirely different source of motivation. To quote Becker (1953) again, "the

68 For a flip side of this relationship, see Kunda's (1992) *Engineering Culture: Control and Commitment in a High-Tech Corporation*.

motivation or disposition to engage in the activity is built in the course of learning to engage in it and does not antedate this learning process" (p. 235). Engaging in work activities then cannot be understood without considering how workers see their work, and their perception of work cannot be reduced to economic interests, as sociologically-minded Marxists have come to recognize (Burawoy 1972, Willis 1977).[69]

Boundaries

Van Maanen & Barley (1984) stress the importance of boundaries, which they define phenomenologically: "the first attribute of an occupational community is that it is composed of people who consider themselves 'to be' members of the same occupation rather than 'are' members of the same occupation" (p. 295). This focus on member-defined boundaries leads them to say that the relevant groups are often smaller than we might think at first:

> Conventional labels typically represent the theoretical limit of an occupational community. Within this boundary, socially significant types... are sure to exist. [...] Commercial fishing provides a useful example because within its boundaries are found several rather distinct occupational communities. "Traditional fishermen" recognize differences

69 Van Maanen & Barley also elaborate substantially the concept of "culture," a secondary concept for Hughes which they make central to their discussion. Like Becker, they see culture as a set of solutions developed by people who face shared problems (1985, p. 33), when such people form a group by virtue of "differential interaction" (1985, p. 34). The culture necessarily involves "collective understandings," which in turn comprise "social world" (1985, p. 34). Understood this way, the culture is not just an attribute of a group: "Culture is not something a group possesses more or less of at any given time; it is something it is" (1984, p. 307). Following Goodenough, Van Maanen & Barley understand culture as "the things a person must know to be a member of a given group" (Goodenough, quoted in Van Maanen & Barley 1984, p. 308). Finally, they argue that cultures do not exists apart from the people who carry them and "endure only to the degree that their content is transmitted from one generation to the next" (p. 35), and are subject to negotiation between interacting individuals. Unfortunately, this latter aspect is not developed further.

> between themselves and "nontraditional fishermen" such as "educated fishermen," "part-timers," and "outlaw fishermen" [...]. Even more important are the distinction made within types. Thus, in the port of Gloucester, Massachusetts, traditional fishermen divide themselves into two groups, Guineas and Greasers. (p. 296)

Van Maanen & Barley's stress on the boundaries recognized by the members as defining the occupational communities, however, creates a problem, if we try to use the concept of "occupational communities" as an elaboration of Hughes's "occupational groups." A crucial feature of Hughes's occupational groups is that members of a group get *license* to perform certain tasks by virtue of their membership. Since Hughes sees this license as something recognized by outsiders, the boundaries of the group as perceived by *outsiders* cannot be ignored. To practice medicine, one must be recognized as a doctor not only by colleagues *but also by patients, nurses and pharmacists*. While recognition by fellow members may often be a prerequisite to being recognized by outsiders, it is the recognition by outsiders that ultimately allows the member to take part in Hughes's division of labor, engaging in tasks that have been allocated for the members of his occupation. We thus need a better theory of boundaries and legitimacy.

We can find such a theory in the Science and Technology Studies (or STS) literature, which comprises another tradition that is closely related to the discussion of occupation (and especially technologically-intensive occupations). This tradition becomes prominent in 1980s, and later discussion of technical work is impossible without considering it at least briefly. The origin of STS can be found in two disciplines: philosophy of science and sociology. Philosophy of science provides the question around which most early STS work has revolved and the shadow of which can be seen in much of

the later STS literature: what distinguishes science from pseudoscience. This is often known as "the demarcation problem" (see Gieryn 1983 for history of the concept.) Philosophy of science has typically attempted to provide a normative answer to this question: a method that could be used to distinguish science and pseudoscience. Popper's (1959) suggestion that scientific claims must be "falsifiable" is an example of this approach.[70] Kuhn (1962), however, critiqued Popper, drawing on detailed accounts of how science was actually done. By presenting a *social* critique of what up to then was a philosophical discussion, Kuhn opened the door to sociological investigation of demarcation practices, which focused on how demarcation decisions *are made*, drawing heavily on the notion of "social construction," originally introduced by Berger & Luckmann (1966).[71] Latour & Woolgar's (1979) *Laboratory Life*, perhaps the most influential book in this tradition written in 1970s, provided a detailed analysis of how scientific facts are "created" in the lab.

The STS literature differs substantially from studies of work conducted in science or engineering labs (such as, for example, Barley & Bechky 1994) in that STS is typically interested first and foremost in the *science* (or the technology), subjecting the work of scientists and engineers to careful study in order to achieve a better understanding of the *result* of their work. Work studies, on the other hand, often take work itself as the primary

70 See Lakatos (1970) for an elaborated version of Popper's argument and a defense against some of the criticism.

71 See, for example, Turner (2008), who also provides an account of social study of science before Kuhn. Turner argues that most of the earlier social studies of science followed an "extensive" view of science, aiming to extend the scientific methods to other areas of social life. While Turner points out that Weber's discussion of science contains certain elements of the later critique, such discussion is hardly systematic. The research on science conducted in 1960s and 1970s by Merton similarly argues for autonomy of science (Turner, p. 49). Turner notes, on the other hand, the influence of Michael Polanyi's and J. B. Conant's views on Kuhn.

object of investigation. Consequently, studies of work applied to scientific and engineering occupations typically compare such work (at least implicitly) with other types of work. Despite those differences, the two traditions have often looked at the same types of work and some of the STS methods are quire relevant to understanding work broadly.

STS has traditionally taken a critical approach to the people it studies: rather than practicing Weberian "sociology of understanding" towards scientists and technologists, STS has typically aimed to undermine their accounts of how science and technology are done.[72] Such critical approach is a double-edged sword. On the one hand, STS literature teaches us how to cut through members' ideology, especially in the difficult cases where the subject matter of their work is sufficiently complex that the researcher cannot fully understand it, and when some of the subjects may be of higher social status than the researcher, be generally seen as more knowledgeable and better qualified to speak for the rest of the members. I employ some of those tools in this dissertation, in particular when trying to look beyond the idea of software development as a "global" profession.[73] At the same time, such critical attitude can incidentally cut off too much of the members'

---

72 Needless to say, ethnographers must always take critically what their subjects tell them. They rarely start their work, however, with the goal of undermining the accounts constructed by their subjects. I also recognize that some STS scholars present compelling accounts of how scientists experience science—for example, Latour 1979 looks at scientists' careers and tries to show to the reader the world through their eyes. Such accounts, however, typically come across as unsympathetic. (Traweek, 1988, presents an particularly complicated case for this generalization. However, the book is arguably not written fully in the STS tradition, but instead may be best understood as a cultural-anthropological study of a western scientific community—something rarely done before that—linked to the STS literature only in the later phase.)

73 Traweek's (1988) analysis of "culture of no culture" among the physicists provides a direct inspiration. Traweek's discussion of "culture" reminds us that culture can be not only empowering, but also oppressive.

experience, leaving us with overly rationalized and economistic explanations of their actions (see for example Latour & Woolgar 1979, Latour 1988a).[74]

Gieryn's (1983) notion of "boundary-work" is particularly valuable to understanding practice.[75] Gieryn looks at the problem of demarcation as a practical problem for scientists. He writes:

> Construction of a boundary between science and varieties of non-science is useful for scientists' pursuit of professional goals: acquisition of intellectual authority and career opportunities; denial of these resources to "pseudoscientists"; and protection of the autonomy of scientific research from political interference. (p. 781.)

He then introduces the notion of "boundary-work" as "an ideological style found in scientists' attempts to create a public image for science by contrasting it favorably to non-scientific intellectual or technical activities" (p. 781) and "their attribution of selected characteristics to the institution of science (i.e., to its practitioners, methods, stock of knowledge, values and work organization) for purposes of constructing a social boundary that distinguishes some intellectual activities as "non-science" (p. 782). While Gieryn looks at specific rhetorical tools employed by scientists in their "boundary-work," the term has gained a broader circulation, denoting more generally the active (and often purposeful) construction of boundaries between groups. The term has obvious application

---

74 It is sometimes said that the critical approach is justified because unlike factory workers, scientists and engineers have had the opportunity to speak for themselves and have made their position known; the researchers job is then to debunk their stories. This explanation conflates, however, the members' experience of their work with those accounts of the work directed to outsiders that are sanctioned by the community. Between such official accounts and economistic rationalization of researcher's observations, STS often misses the actual meaning that individuals attribute to their work.

75 Abbott (1988) expresses a number of similar ideas, though I find the concept of "boundary-work" more useful for micro analysis than those used by Abbott.

to the discussion of occupational cultures: while Van Maanen & Barley (1984) stress Weberian "consciousness of difference," the concept of boundary-work suggests that such differences are not something that the members are merely conscious off, but are something they actively construct.[76]

Finally, some of the research in STS has addressed directly the issue of space. I leave this discussion for later, however, when discussing Latour in chapter 1.3.

Practice, Power and Peripherality

Lave & Wenger's (1991) *Situated Learning* is written in a somewhat different tradition from Hughes, Becker or Van Maanen & Barley, but has since been incorporated into the work studies canon and is often read through its prism. Like Becker (whose 1972 article on schools Lave & Wenger cite), Lave & Wenger look at a broader category than "work" or "occupation." The book looks at learning, re-conceptualized as progressive engagement with "communities of practitioners" or "communities of practice," which they define as follows:

> A community of practice is a set of relations among persons, activity, and world, over time and in relation with other tangential and overlapping communities of practice. A community of practice is an intrinsic condition for the existence of knowledge, not least because it provides the interpretive support necessary for making sense of its heritage. Thus, participation in the cultural practice in which any knowledge exists is an epistemological principle of learning. (p. 98)

---

76 The discussion of boundaries of course precedes Gieryn's work. Lamont & Molnár (2002) provide a useful review of the way the concept of "boundaries" has been used in sociological literature, looking both at the historical use of the concept as well as the more recent developments in the literature.

The concept of "a community of practice" shares important characteristics with Becker's "groups" and Van Maanen & Barley's "occupational communities." It similarly involves a notion of a bounded group that carries out certain activities, made possible by the groups' "knowledge" (Hughes's and Becker's "technique") and a shared interpretive system (Hughes's "culture," Becker's "perceptions and judgments," Van Maanen & Barley's "culture" understood as "a meaning system."). Like Hughes, Lave & Wenger explicitly define communities of practice "in relation with other tangential and overlapping communities of practice," though they do not explore this aspect of communities of practice in any substantial depth.

The concept of practice provides a useful alternative to the notions of "work" and "occupation." It first of all gives us a *broader* term, helping us see similarities between organized activities that we consider "occupations" and those that we do not. (Lave & Wenger's examples of practice include "Alcoholics Anonymous," which is clearly not an occupation in Hughes's sense, but likely an "activity" and perhaps a "deviant group" in Becker's terms.) While a similar shift is made by Becker, it is obscured by the focus on deviance. Lave & Wenger's notion of However, "practice" is not merely a generalization of an "occupation," but a multi-faceted concept with its own theoretical history. It would therefore help to pause to analyze it briefly.

Oxford English Dictionary breaks the usage of the English word "practice" into four semantic subfields (after omitting obsolete and rare senses):

(1) exercise of a profession ("began his law practice"),

(2) actual application of an idea ("integrates theory with practice"),

(3) customary or established conduct ("has become common practice"),

(4) repetition for the sake of improvement ("hours of daily severe practice").[77]

When used in social science, the term "practice" typically takes one of the first three meanings, which for the sake of simplification can be reduced to two dimensions:

A. a system of procedures established within some group (most of 1 and 3 above)

B. action contrasted with theory (subfield 2).[78]

Those two dimensions are related, but are at the same time distinct, with different history in social science, though sometimes used by the same authors. Lave & Wenger (1991), in

---

77 After removing examples, and obsolete or rare meanings, the four subfields are:

> **1.** The carrying out or exercise of a profession, esp. that of medicine or law. Also as a count noun: the business or premises of a doctor or lawyer.
> **2.** a. The actual application or use of an idea, belief, or method, as opposed to the theory or principles of it; performance, execution, achievement; working, operation; (Philos.) activity or action considered as being the realization of or in contrast to theory (cf. PRAXIS n. 1a). [...]
> d. In Marxist and neo-Marxist thought: the political and social action which should result from and complement the theory of Communism. Cf. PRAXIS n. 1b.
> **3.** a. The habitual doing or carrying on of something; usual, customary, or constant action or performance; conduct.
> b. A habitual action or pattern of behaviour; an established procedure or system. Usu. with negative connotations in early use. [...]
> c. Law. An established legal procedure, esp. that of a court of law; the law and custom on which such procedure is based; [...]
> **4.** Repeated exercise in or performance of an activity so as to acquire, improve, or maintain proficiency in it; activity undertaken to this end; [...]

78 Most of the examples that OED assigns to field 1 would fit in field 3, and others would fit in 2. The differences between fields 1 and 3 appear to have much to do with a focus on boundaries vs. custom. My dimension A corresponds closely to the second of the three "notions of practice" identified by Schatzki (1996): "the temporary unfolding and spatially dispersed nexus of doings and sayings" (p. 89). My dimension B corresponds (though less obviously) to Schatzki's third notion: "performing an action or carrying out a practice of the second sort" (p. 90). Schatzki's first notion ("learning how or improving one's ability to do something by repeatedly working at and carrying it out," p. 89) corresponds to the semantic field 4, which I discard in my analysis. (Schatzki similarly dismisses this first notion as irrelevant to his analysis.)

particular, invoke *both* dimensions and it is important to understand how they combine them.

A. "Practice" understood as a coherent set (or a system) of activities in which some group of people engages fits quite naturally with the cultural theory of occupation discussed so far. It corresponds quite well to Becker's "activity" and can be seen as a superset of Hughes's and Van Maanen & Barley's "occupation."[79] Specific practices in this sense are defined by *differences and boundaries.* A "practice" is always implicitly contrasted with some other way of doing things (or simply *not* doing something). Additionally, within the tradition reviewed so far (and also for Lave & Wenger), a practice is always linked (and mutually constitutive) with a group. When people who engage in a practice are in contact with those who do not, engagement in the practice marks social boundaries and serves as a source of identity for the practitioners. Lave & Wenger invoke this sense of practice when they talk of practices as having histories and involving sets of relationships, and also when they use "communities of practitioners" as an alternative to "communities of practice."

B. "Practice" understood as a form of engagement with the world that aims to transform rather than merely understand it, has no explicit parallel in the work of any of the authors discussed in earlier sections, but is common in Marxian thought.[80] Lave &

---

79 We could say that a "practice" is a type of "activity" and an "occupation" is a type of "practice." Use of marijuana is an "activity", and perhaps a "practice" but not an "occupation."

80 Marx's notion of practice (or "praxis") can itself be further broken down into two: (B1) the idea of practice as a transformative engagement with the world and (B2) the question of the relationship between practice and theory. See Rasmussen (1979) for a discussion of the relation between the two and their relation to the Aristotelian origin of the concept of "praxis." Lave & Wenger incorporate both sides of this second dimension.

Wenger (1991) invoke this sense of practice explicitly, and more generally as they argue for the primacy of action over theoretical knowledge. Communities of practice are communities of *action* rather than theory, defined by what people do rather than by what they know abstractly.[81] The distinction between practice and theory becomes particularly import with the rise of "informational" understanding of work.

Lave & Wenger's (1991) discussion of practice uses the second dimension to explain the first: a practice, understood as a system of activities, persists in time because new people come to engage in it, and this entry must be understood as a matter of joining the actual activities, rather than absorption of the abstract knowledge that may be said to underlie the practice.[82] Reliance on this second dimension of "practice" leads Lave & Wenger (1991) to look at how the practice relates to the material world and how a novice comes to engage in the activities that comprise it.

The attention to the material aspects of the practice highlights the importance of tools. The activities carried out by the practitioners, typically involve transformation of certain objects.[83] (Sometimes the objects are just the members themselves, as is the case of the Alcoholics Anonymous example, but this is the exception.) This transformation is made possible by interpreting the objects in a particular way, but it often also relies on artifacts, which "carry a substantial portion of that practice's history" (p. 101). In many

---

81 Of course, developing theory is itself a form of practice.

82 Bourdieu's work (1977), referenced by Lave & Wenger, similarly bridges the two senses: Bourdieu's "pratique" opposes creative practical activity to generative structures, but is closely related to his concept of "habitus"—a system of dispositions that distinguishes one group of people from another.

83 The focus on objects and tools is also shared by scholars working in the Activity Theory tradition (e.g., Vygotsky 1930/2002, Leontiev 1972/1981, Engeström 2001). Lave & Wenger (1991) draw on Vygotsky, and the two approaches have much in common.

cases it is the access to such artifacts that limits entry for new members.[84] On the other hand, since material tools are mobile in different ways than persons or culture, they may be important when we later discuss reproduction across space, as we will see later.[85]

To understand how a novice comes to partake in the activities, Lave & Wenger (1991) also introduce the theory of "legitimate peripheral participation," arguing that new members enter the practice by engaging in peripheral tasks, and moving to other tasks gradually. Such peripheral participation allows them to become *members* (at least tentatively) and to start absorbing the knowledge and culture without having the expertise that is expected of the full members and necessary for engagement in the more central tasks. The notion of legitimate peripheral participation is in some ways similar to Becker's theory of social learning (1953), but it adds a crucial component of *legitimacy*. Since practice often requires a "license" (from other practitioners, the larger society or both), we must pay attention to how the novice obtains the initial permission to enter. When looking at the role of the university programs that train software developers, we must consider not only what the future software developers might learn there, but also the way in which such practices may allow the students to become legitimate participants in certain practices.

---

84 In some cases, lack of access to the material tools leads to a creation of a parallel practice, which aims to replicate the original practice without the missing tools. For example, a world-wide flight simulator community aims to replicate in minute details some aspects of the pilot's practice using flight simulator software to compensate for lack of access to real airplanes. (To do this they work together with people who simulate the practice of air traffic controllers.)

85 A similar idea is expressed by Bruno Latour (1988a) who writes about the importance of "immutable combinable mobiles" for reproduction of technoscience. This idea is discussed in chapter 1.3. More generally, Latour's stress on inclusion of "non-human actors" in analysis fits with the Lave & Wenger's (1991) stress on materiality and tools.

The notion of legitimacy links the discussion of practice to the notions of *power* and *access*. Power relationships within the community as well as those of the larger society grant access to some participants and deny it to others. "Hegemony over resources for learning and alienation from full participation are inherent in the shaping of the legitimacy and peripherality of participation in its historic realizations," write Lave & Wenger (p. 42). Furthermore, those who are admitted do not necessarily become equal to the old-timers upon this admission. Legitimate peripheral participation therefore "can be a source of power or powerlessness" (p. 36). In particular, "as a place in which one is kept from participating more fully—often legitimately, from the broader perspective of the society at large—it [peripherality] is a disempowering position" (p. 36).

The theory of legitimacy, access and power, however, is underdeveloped in Lave & Wenger, and is critiqued by Contu & Willmott (2003, 2006), who label Lave & Wenger's theory of power "embryonic" and hold Lave & Wenger in part responsible for the later "conservative" misreading of their book. In particular, Contu & Willmott (2003) argue that Lave & Wenger often seem to take for granted the cohesiveness of communities:

> In *Situated Learning* and its subsequent refinement (Wenger 1998), the concept of community is ostensibly positioned on a conflictual terrain (Wenger 1998, Ch. 2). However, the condition of existence of such communities is located in harmonizing categories such as "a sense of joint enterprise relationship of mutuality shared repertoire of communal resources" (Wenger 1998, emphasis added). Community is conceptualized in a way that tends to assume, or imply, coherence and consensus in its practices. Such "unequal relations of power" (Lave and Wenger 1991), we suggest, glosses over a fractured, dynamic process p. 42) of formation and reproduction in which schisms and precarious alignments that are held together and papered over by unreflexive invocations of hegemonic notions including "community," "family," "team," and "partnership." By default, Lave and Wenger's usage of

> "community" is complicit in the reproduction and legitimation of this hegemonic process. (p. 287.)

Contu & Willmott underscore "the danger of assuming consensus in communities of practice," and urge their readers to focus their attention on the idea of "practice" rather than "community."[86] In my analysis of my fieldwork I specifically attempt to avoid this alleged shortcoming of Lave & Wenger, avoiding the term "community" except as a "native" term used by the participant, and treating it with caution then. Instead of assuming that software developers join a "global community" of software developers (as they often do describe it), I see them as engaging in a practice that requires negotiation of many power relationships. "Boundary-work" and other notions from STS (see above) become useful in uncovering the "construction" of the community.

Lave & Wenger (1991) use "legitimate peripheral participation" as a micro-sociological concept: something that applies to the activities and position of concrete individual people. However, this notion can also become useful at a higher level of analysis, when considering the position of certain groups within the larger world of practitioners. We can draw certain parallels between the role of an individual "newbie" software developer who have just started working at a software company and the collective position of Brazilian software developers who have achieved legitimate but *peripheral* role in the larger world of software, and whose progression to more central participation involves substantial difficulties. Such position can similarly be "a source of power or powerlessness" (see Lave & Wenger 1991, p. 36). Such macro notion of

86 See also Duguid (2005) for the discussion of "community" vs. "practice" in "communities of practice," including the suggestion that the wide appeal of Lave & Wenger (1991) "owes a good deal to the seductive character of *community,* aptly described as a 'warmly persuasive word'" (p. 109).

peripherality has been expressed in some of the political economy literature (e.g. Evans (1979). I discuss such links in chapter 1.3.

Work as Situated Practice

Julian Orr's ethnography of Xerox technicians (1996) is perhaps the best known of the books answering Van Maanen & Barley's (1984) call for ethnographic studies of occupational communities. At the same time, the book "puts the flesh of everyday life on Lave & Wenger's (1991) idea of a community of practice" (Barley, 1996, p. xiii). Orr's focus on "work as doing, as practice" (p. 10) is indicative of this fusion—and the source of the title of this chapter. (Note, though, that Orr appears to focus mostly on practice in its "common sense" meaning, as defined above, without indexing Marxian praxis.) It is important to note that this fusion is not undertaken by Orr (1996) explicitly. In fact, as Duguid (2008) points out, Orr's book "studiously avoids" any mention of Lave & Wenger. The first publication that uses Orr's ethnography to illustrate concepts from Lave & Wenger is Brown & Duguid (1991).[87] This union is later blessed by Barley (1996) in his foreword to Orr's book.

Orr's ethnography follows a small "community of practitioners" that he calls "the reps"—Xerox field technicians, who attend to photo copiers installed on client's premises—and looks at their relationship to a specific organization and the organization's

87 Since Orr's book (1996) was not yet published at the time, Brown & Duguid (1991) rely on Orr's earlier work, including his dissertation (Orr 1990). Brown & Duguid's paper presents provocative ideas that go far beyond just using Orr's work to illustrate the notion of "communities of practice," but I do not discuss those other ideas here, as they speak primarily to organizational studies concerns and lie outside the main axis of my dissertation.

clients. The book provides a rich account of work that helped not only illustrate some of the ideas discussed above (especially those of Van Maanen & Barley 1984 and Lave & Wenger 1991—see Brown & Duguid 1991), but also allowed for a *critique* of such ideas. For example, Contu & Willmott (2003, 2006) provide their own interpretation of Orr's observation, accusing Orr himself and also Brown & Duguid (1991) of accepting the management point of view. While I agree with Contu & Willmott's call for a more critical analysis of Lave & Wenger (1991) and Orr (1996), I see this re-interpretation as first of all a sign of the richness of Orr's ethnography and aim for similar richness in my own examples.

While greatly admiring Orr's ethnography, I also see danger in its richness. As Barley (1996) and Barley & Kunda (2001) argue, social science of work often relies on the available "images of work," often outdated. Orr's ethnography provided an image of work that helped replace the earlier images. The book greatly affected (both directly and through Brown & Duguid 1991) the discussion of work in the years that followed its publication.[88] Such a discussion has consequently ignored some of the issues not captured by Orr's image. It had a particularly strong effect on they way Lave & Wenger's "communities of practice" are understood.[89] Even the literature that extended the term "communities of practice" to virtual communities bears the mark of Orr's image, often talking about such virtual communities as tight-knit and inward-looking.

---

88 See, for example, the 2006 special issue of *Organizational Studies* dedicated to the tenth anniversary of Orr's book for the discussion of the influence that the book has had.

89 See Duguid (2008) for a discussion of what was gained and lost by making Orr's "reps" the standard example of a community of practice.

Orr's ethnography, however, is based in a particular and highly peculiar place, Silicon Valley. Even on a clear day, such as the one that Orr describes in his first vignette (p. 15), someone driving through the valley sees a world with remarkably narrow horizons, bounded by hills on both sides. (It is a *valley*, after all.) Orr's writing captures the imagery of that small world in great detail, but largely ignores anything that happens beyond the hills that bound it. The reps' world appears to be fully contained within this narrow space. (The corporate headquarters of Xerox happen to be on the other side of the country, but this fact is hardly apparent in Orr's book—and is perhaps simply not relevant. Orr mentions the fact that "San Francisco" service area stretches as far as Guam, but does not describe any practical consequences of this.)

Geography does come up in one small passage, which gives us a glimpse of what Orr might have observed had he chosen a different place:

> Frank began working with the corporation in Indiana, which may account for this reference to a Chicago Fix. Some districts get new machines before others, and some have accounts that will use the machines much more intensely than others. These districts encounter most of the problems before others will and must develop their own ways to fix problems until engineering and manufacturing catch up. Word of these innovations will spread, usually carrying acknowledgment of their origin. Such districts have a reputation among the surrounding districts as something like the big time, the cutting edge of the blunt instrument that is field service. Silicon Valley is such a district on the West Coast, Chicago is the same for the Midwest, and this may remind us that although service is intensely local, they do work as members of a multinational corporation, and what happens elsewhere in the corporate geography may matter to them too. (pp. 60–61.)

In other words, reps working in places such as Indiana do have to pay attention to practice occurring elsewhere—for instance in Chicago. Those working in Chicago or the Silicon

Valley, on the other hand, must rarely be reminded about the rest of the world. What happens elsewhere, *may* matter to them, but does not warrant a discussion beyond this one passage.

I aim to present a different image of work: one that follows Orr in attempting to show the importance of local work, yet also captures how, for many people, what happens elsewhere has tremendous importance on a day to day basis, and where "elsewhere" refers to places that are much further from them than Frank's office was from Chicago. Such an image will help us extend the insights of Lave & Wenger (1991) to the new, globalized work.

Both Lave & Wenger (1991) and Orr (1996) rely on the notion of "situated activity" to explain their projects. (Orr also uses the terms "situated practice" and "situated action.") They do not use them in the same way, however, and the difference between the two gives us a chance to get a better understanding of the ways in which action can be "situated." Orr, who explicitly attributes his concept of situated action to Suchman (1987), focuses on the way action is situated in its immediate physical context —hence the commitment to detailed descriptions of work and the places in which it occurs, as well as the stress on the importance of the reps' physical presence near specific machine and their ability to hear their characteristic sounds. Lave & Wenger (1991), on the other hand, reject a similar notion of situatedness, comparing it to "naive views of indexicality" (p. 32). While their understanding of situatedness is not defined as clearly as Suchman's and Orr's, it is clear that the book shows less interest in place and minute details of activities. Instead, it looks at learning as situated in the larger context of

practice—that is, situated in a social world of action and in history of the practice, rather than in physical space understood narrowly.[90]

My approach to situatedness integrates Lave & Wenger's and Orr's, looking at the work of Brazilian software developers as situated simultaneously in two different contexts: the local one, similar to the one described by Orr, and the more abstract context of a practice based far away—in fact, the very place within which Orr's ethnography is situated. I stress that we can only understand their practice if we look at those two contexts (and those two kinds of situatedness) side by side and examine the contradictions between them.

A Synthesis

This dissertation takes as a point of departure the notion of "work as practice," recognizing that work often involves simultaneously an economic relationship of employment and a membership in a community of people who engage in similar work and share certain technique, culture and identity. Work is thus subject to both cultural and politico-economic analysis, and the two must often be performed simultaneously.

At the most basic level, practice can be understood as a system of activities in which a group of people engages. Members see some sort of meaning in the activity and participation in the activity forms part of their identity. Those activities may be

90 All of Lave & Wenger's examples involve communities bound by place to some extent, and for some the place is mentioned explicitly ("Yucatec midwifes," "Liberian taylors"). However, the role of place is not discussed explicitly.

recognized as valid and meaningful by some people who do not engage in the practice, though such people's understanding of those activities is likely to differ from the member's own understanding. The activities that comprise the practice and the group of practitioners often mutually define each other: the group is typically understood (by both insiders and outsiders) as a group of people who engage in the activities, and at the same time the activities are considered a part of the practice to the extent that the group engages in them.

The meaning of the practice is typically defined in relation to other practices and the practice often has a place in a larger division of labor. Those relationships can be economic or non-economic. They can also be cooperative or antagonistic. A practice sometimes needs to be understood by its relationship to "client" practices; in other cases, it needs to be understood in its relationship to its "anti-practices." For example, the practice of investigating drug traffic cannot be understood in separation from the practice of trafficking drugs. Cooperative relationship with other practices must not be understood as necessarily equal and fair. They may well be exploitative, and the balance of power between practices may vary widely.

Practice typically involves transformation or manipulation of certain objects as well as transformation of people from outsiders into members. Successful manipulation of the objects (per group's and its clients' definition of successful) requires a particular perception of the objects and situations and application of certain techniques to them. This involves doing things outsiders usually do not do. The application of technique may presuppose access to tools, which embody certain aspects of the practices heritage.

The novice's entry into the practice requires acquisition of perceptions and technique, as well as access to tools. The perceptions and the technique can only be fully acquired by engaging in the practice together with the more experienced members. (Later development of perceptions and technique requires continuous engagement in the practice together with other members.) Such participation is often possible because the members recognize the need to bring new people and allow novices to engage in peripheral tasks that gradually lead them to fuller participation. It may also be possible because old-timer benefit from the work carried out by the novices. The relationship between old and new members can thus be unequal and exploitative, and the group of practitioners will likely more generally be characterized by a system of power relations.

The need to engage in the practice together with other practitioners requires that the practitioner is recognized as a member (or at least as a member-in-training) by other practitioners. Member-defined boundaries are important for this reason. However, certain groups of outsiders may recognize specific groups of practitioners as legitimate providers or consumers of certain objects or services (further recognizing their right to engage in the activities that are involved in the practice). Since such recognition is done at the level of a group, being recognized as a member of the group by *outsiders* may have consequences for the individual's ability to engage in the practice. While outsiders may often rely on the membership judgments made by the members themselves, we cannot assume the insiders' and outsiders' understanding of the group's boundaries to be the same. Members engage in "boundary work" to bring outsiders' understanding of group's boundaries in line with their own.

This synthesis provides a foundation of a discussion of work as practice, but suffers from an important flaw—failure to theorize the relations between practice in different place. The next chapter presents a slice of my own ethnography, which I hope will expose this problem. I then return to theory in chapter 1.3, presenting a theoretical account that tries to make up for this failure. Parts 2 and 3 of the dissertation then put more ethnographic flesh on the theoretical skeleton presented in chapter 1.3.

1.2 The Global Tongue

This chapter describes the use of English among the software professionals in Rio de Janeiro. I look at this topic not out of interest in the use of language *per se*, but rather because the interactions that surround the choice of language represent in the clearest way the contradictions inherent in the peripheral participation in a global technical practice. Following the use of English, we take a tour through the different social contexts that I explored during my fieldwork in Rio de Janeiro. This tour will show us some of the limitations of the theory of technical practice presented in the previous chapter when facing the situation of the peripheral participants. I then attempt to bridge this gap in chapter 1.3, presenting a theoretical account of how a technical practice is reproduced in new places. I then come back to ethnography in Parts 2 and 3 of the dissertation, looking in more detail at some of the contexts visited in this chapter and considering how other types of cultural codes are negotiated.

Linguists use the term "diglossia" to refer to the situation where a single social group routinely uses two different languages, with most speakers being relatively proficient in both. In a typical diglossic system, the two languages have different roles. One language, which linguists typically call "High" or "H," is used for formal communication as well as for high culture. Another, "Low" or "L," is reserved for informal communication, especially among close friends and family. Use of the High language connotes professionalism, education, culture, status hierarchy, and commitment

to the larger (national or international) institutions. Use of the Low language connotes intimacy, equality and a commitment to the local place.[91]

It would not be accurate to say that Brazilian society, or even more specifically the community of software developers of Rio de Janeiro, is diglossic in their use of English if the word "diglossia" is used in its narrow linguistic sense.[92] While some Brazilians learn to speak English fluently, it is still a foreign language in Brazil, and during my fieldwork I have never heard two Brazilians speaking English to each other, except for the sake of a foreigner or as a joke. All speech presented later in this chapter is my translation from the original Portuguese, except where stated otherwise. Written English, however, is omnipresent in the work of Brazilian software developers, as are short spoken phrases, which may or may not be altered to comply with Portuguese phonology.[93] While such co-

---

91 Ferguson (1959/1971) originally introduced the term "diglossia" to refer to the situation when two (or more) varieties of the *same* language are used within the same community under different conditions. For that reason, Ferguson described the languages as sharing much of the grammar and the vocabulary. (For the examples that Ferguson studied, he also suggested that the grammar of the "High" language tends to have a richer inflection system.) The concept of diglossia was soon extended to the case of two unrelated languages, such as the diglossia between Guarani and Spanish in Paraguay, as well to different relations between the two languages. Grosjean (1982), uses the term to refer to any two languages (related or unrelated), but requires an asymmetric relationship between them, stressing "that the H variety *is not used by any sector of the community for ordinary conversation,* that it is learned largely in school and is used for most kinds of writing" (p. 132, original emphasis). I follow Grosjean's use of the term.

92 Some scholars have made a claim that Brazil is diglossic between two varieties of Portuguese: the formal Brazilian Portuguese that closely approximates the Portuguese spoken in Portugal and the colloquial Brazilian Portuguese that differs from it substantially (e.g. Azevedo 1989, Bagno 2001). While my own experience with Brazilian Portuguese leds me to agree with this claim, I do not discuss this complex issue. Therefore, when I use the term "diglossia," here, I refer to the *diglossia-like* relationship between English and the different varieties of Portuguese.

93 Such use of spoken English usually does not rise to the kind of mixture of two languages that linguists call "code-switching," since only specific English words and phrases are used, and they are usually phonologically adapted to the Portuguese context. They are better described as "borrowings" (or to use Grosjean's terminology, "speech borrowings") from English into a particular variant of Portuguese. (See Grosjean 1982, p. 309, for a discussion of the difference between borrowing and code-switching.)

existence of English and Portuguese is not diglossia in its traditional sense, it retains some of its features: the High language (written English) can be used to communicate status and global links, while the Low language (Portuguese) communicates local connections.

While linguistic literature on diglossia often focuses on the mechanics of code-switching and second language acquisition, diglossia is nearly always a power-invested phenomenon and its social side of diglossia often cannot be understood without considering how the two languages tie together local power relations and external resources. In a typical case, proficiency in "H" marks the individual's status *vis-à-vis* those who are less proficient in it, becoming an instrument of exclusion. Proficiency in H is often predictive of the individual's social status because it requires access to educational resources only available to the privileged few. H also becomes important for local power relationships by connecting proficient speakers to a powerful external community interacting in H, allowing them to draw on the resources of this community. At the same time, however, such use of H often underscores its speakers' dependence on the external resources and their subordinate position *vis-à-vis* the group that defines the norms of H.

All of this applies as well to the quasi-diglossic relationship between English and Portuguese among Brazilian software professionals. Proficient command of English, and especially fluency in spoken English, often reflects the socio-economic origin of the developer. At the same time, it gives them access to crucial foreign resources, further elevating their status by helping them acquire cultural capital in the occupational world.

At the same time, however, the use of English highlights the developers'

peripheral status in a largely foreign practice of software development. As we will see in the examples below, Brazilian software professionals often downplay such tensions: good software professionals are expected to display a global perspective on the world and to accept the dominance of English as a matter of fact. At the same time, elements of resistance do surface occasionally, and the careful handling of misalignments between the local and the global nature of their practice requires daily attention.

This quasi-diglossic relationship between English and Portuguese is just one of the dimensions of the larger phenomenon that we can call "cultural diglossia": the situation where a particular social group simultaneously maintains orientation towards two cultures, which stand in an asymmetric relation to each other. In our case, it is the local culture of Rio de Janeiro (and more broadly Brazil) on the one hand, and on the other hand, the "global" technical culture of software development, based to a large extent on the academic culture of the United States. (This distinction corresponds to the relationship between the two types of situatedness discussed at the end of the previous chapter.)

I start my discussion with a few episodes from "Alta," a small software company in Rio de Janeiro, which serves as an example of what it means to be successful in Rio's software industry today. Those examples present the use of English and Portuguese in the relatively unproblematic situations. I then look at a more complicated case of Lua, a programming language developed in Rio de Janeiro, which until recently had no Portuguese documentation. I follow this story with a discussion of a small project based on Lua which truly straddles two different worlds, and gives us some of the most

complicated interactions. After that I look at how the developers acquire the knowledge of English, and discuss explicitly the social differences that English proficiency marks. Finally, I look at some of the ways in which the developers express resistance to *gringo* dominance.

The Language of Software Code

It was a cool day in June, one of the coolest months in Rio. I was in an office of "Alta," a company in downtown Rio, and, according to its promotional materials, focused on attending to the desires of its clients through the use of newest technologies. In 2007 this was understood by many to mean building web applications in Java,[94] which is what Alta did. The company was started a few years ago by three graduates of PUC-Rio, one of Rio's most prestigious universities, and the most expensive one.[95] The company, situated in a shiny office in the very center of Rio's commercial district (and walking distance from most of its clients), had been fairly successful, and the office was typically full of energy and laughter. Most employees were very young, all but one were under thirty.[96]

I had just recently started my fieldwork at the company. "Fabio," who had been designated to be my main contact in the company, had set me up with access to the company's intranet website, which I was now starting to explore. As I logged in, I saw a

---

94 See the glossary for technical terms. Some of them are marked in the text with underlining.

95 The cost of 21-24 units in "Ciclo Básico" (which includes computational engineering) costs R$1600 per month, according to PUC website (http://www.puc-rio.br/ensinopesq/ccg/anuidadesCCG.html) and this number was confirmed by PUC graduates. This would translate into roughly US$6000 a year.

96 The story of Alta is explored in more detailed in chapter 3.1.

menu, in Portuguese, offering options such as “Processo” (“process”), “Repositórios de Clientes” (“clients’ repositories”), “Suporte” (“Support”) and “Códigos” (“Code”). As I started browse the content of the site, I found that all of its content was written in Portuguese.

The situation changed sharply, however, when I arrived at the actual source code files. Below is a typical example of what I saw there:[97]

```
/**
 * @param weekday - Dia da semana em que o chronoEntry se encontra
 * @param start - Data de inicio do Cronograma
 * @param oldDate - Data do chronoEntry que esta sendo clonado
 */
private Date weekConvert(Integer weekday, Date start, Date oldDate){
 Calendar cal = GregorianCalendar.getInstance(new Locale("pt_BR"));
 cal.setTimeInMillis(0L);
 //Data do inicio do Cronograma, assumimos que seja sempre segunda.
 cal.setTime(start);
 //Ajustamos o dia da Semena
 cal.add(Calendar.DAY_OF_MONTH, weekday.intValue() - 1);


 //Copiamos Hora e Minuto
 cal.set(Calendar.HOUR_OF_DAY,oldDate.getHours());
 cal.set(Calendar.MINUTE,oldDate.getMinutes());


 cal.set(Calendar.SECOND,0);
 cal.set(Calendar.MILLISECOND,0);

 return cal.getTime();
}
```

There are two types of text in this example: the *comments,* included only to assist a human reader of the code (the text in “/* ... */” and on lines that start with “//,” shown in italics), and the *instructions*—the code actually interpreted by the Java compiler. All comments in the example are written in Portuguese, all instructions are written in

97 In this an other code samples, italics and bold were applied by myself to make the code easier to read. Software code is always stored without formatting (“plain text”). Most programming editors will apply *syntax highlighting*, coloring different types of commands in different colors. Syntax highlighting is also often applied automatically to code displayed on the web. The way code is highlighted, however, depends on the software one uses to view it.

English.

While the use of English could be related to the programmers' pride in having mastered the dominant language of the global software profession (and we will come back to this issue later), it is important to realize that, as is often the case, the programmer had limited choice over the language in which he wrote. (I use "he" here since all of Alta's two dozen programmers were men.) Three of the English words in the passage above ("new", "private" and "return") are *keywords* (or *reserved words*) of the Java programming language—three of the fifty nine words that have fixed meaning in Java. Any Java programmer must know and use them. Most of the remaining English words, while not part of the Java language *per se*, are defined by a set of software modules (or "libraries") that come with Java. They are avoidable in theory, but not in practice. Java programs create and manipulate data objects, each of which is associated with a bundle of instructions called a "class." The class of the object defines what operations can be done with it. Objects, classes and operations all have names, which are used to invoke them. While the programmers can and do define their own classes for their specific situations and can name those as they please, much of the standard functionality is supported by classes that are packaged together with Java.[98] Not surprisingly, those classes and their

98 More generally, most modern programming languages operate by defining data entities (general data structures, operations over them, and specific data items) and giving names to those entities so that those entities could be used later. The programmers are typically free to create their own entities and name them as they want (though, such names must typically be limited to the 26 letters of the unaccented Roman alphabet - "conexao" would be ok, but not "conexão"). Programmers' entities, however, typically build upon a pre-existing layer of entities (many of which are often "standard" to some degree) and inherit from them many of the names. The programmer often has choice of what entities to build his or her code on, but that choice is limited by what foundations are available. Starting from scratch is possible in theory but impractical due to the time it would take to provide an alternative to an existing foundation layer. Finally, a small number of names (typically under a hundred) have fixed meaning in the language. Those commands cannot be renamed, and their names cannot be used for other purposes. They are called "reserved words" or "keywords." (There is a subtle

operations are named in English, by programmers working for Sun Microsystems (now split mostly between Cupertino, CA and Bangalore, India).

In the example above, the programmer needs to perform date calculations, and can do this using a standard class "Calendar." He obtains an "instance" of this class by first creating a new object of class "Locale" and *passing* this new object to an operation ("getInstance") defined by a class called "GregorianCalendar"[99]:

```
Calendar cal = GregorianCalendar.getInstance(new Locale("pt_BR"));
```

The only point at which the programmer could choose which language to use is when giving a name for the new object that is being created (terms shown in bold in the code listing above). In this particular case he used a language-neutral abbreviation "cal": it could stand either for "calendar" or "calendário." The other four names that he introduced ("weekConvert," "weekday," "start," and "oldDate") are in English. The programmer *could* have chosen Portuguese names for them. However, doing so would turn the code into an odd mixture of two languages, the programmers often say to me.[100] It would also

---

difference between the two terms which is not relevant to this paper, and they will therefore be used as synonyms.)

99 Note that the name "GregorianCalendar" indicates that other calendars are also supported. As a global programming language, Java is designed with an assumption that the users of the software written in Java may use a variety of scripts, calendars, or sorting conventions. The *programmers*, however, are expected to use English.

100 The following passage of PHP code written by a Brazilian software developer "in Portuguese" demonstrates this problem:

```
session_start();
if (!isset($minhaConexao)) {
 $minhaConexao = new Conecta();
}
```

In this case the programmer used Portuguese names for all the variables that he could name, mixing two Portuguese terms with English "session_start", "if", "isset" and "new." He also had to leave out accents in the Portuguese names, settling for "conexao" instead of "conexão."

require the programmer to remember whether each particular object's name is English or Portuguese: is it "oldDate" or "dataVelha"?

"Boilerplate code" is another reason for extensive use of English. Many problems faced by the programmers have ready solutions, usually documented on the web (almost always in English) with extensive coding examples, which can be copied and pasted into the program. (While programmers, whether in Brazil or the US, nominally look down on such practice, copying small stretches of code is quite common in both countries.) The larger platforms—such as the ones used by Alta—typically come with complete sample applications. Programmers often find that it is easiest to start with such generic application and then modify it one step at a time until it fits their specific needs. For instance, rather then writing an e-commerce site from scratch, a programmer can download a simple "sample store" (a ready web application for a store called "My Store" selling two types of "widgets") and then modify it for the needs of a specific customer, which may include translating the interface into the customer's language.

It's Just More Natural

Looking at Alta's code some time later, while working on a small task that Fabio assigned to me, I did find cases of use of Portuguese variable and operation names. A few days later, as I was watching Fabio draw a diagram of Alta's next application, entering English field names like "price" and "quantity," I decided to ask him about the mix of languages that I had seen in the code. *It's supposed to be all in English,* said Fabio. He

then explained: *The Portuguese names were just someone's mistake. It all should be in English, except for the database tables.* (All speech presented in this and later chapters should be assumed to be translated from Portuguese, unless stated otherwise. Speech presented without quotes is reproduced approximately.[101])

"Except for the database tables?" I made a surprised face. *Yes, the Java code should be in English,* Fabio said again, *But the database tables should be in Portuguese.* Mauricio, sitting at the adjacent table turned around in his swiveling chair. "This is really stupid," he said. Mauricio, in his late twenties, was one of the developers on Fabio's team; he usually stayed quiet, so I knew that this had to be a topic he felt strongly about. *Having class names and database tables have different names makes no sense,* said Mauricio. He explained that there are many software applications that assume that the names of the tables and classes match, and a mismatch between the database table names and Java class names is often a source of endless trouble.

The two started talking about why one could possibly want class names to be in English and database tables in Portuguese. Mauricio's position was that it should all be in English, while Fabio explained that they simply did not have a say about the database tables. Those were administered by "Intermercado," Alta's largest client, and had to be in Portuguese. "Why? Because the database guys don't know English? What are they doing there then?" insisted Mauricio. (This was perhaps a jab at the "database guys" broadly, or more specifically at the clients' IT people, whose incompetence was a frequent topic of

101 Here and everywhere below, I use quotes to mark participants speech that is reproduced *verbatim*. Speech that was not captured verbatim, is sometimes phrased as direct speech, but is presented *without quotes*. I italicize such speech when doing so aids reading. (For details see appendix B, "Transcription and Quoting Method.")

discussion among some of Alta's engineers.) Yes, Fabio explained, database guys may or may not know English. In either case, this was not for him or Mauricio to decide.

"But why should the *code* be in English?" I asked. "Good question," said Fabio, "I ask this myself sometimes." *But it is just more natural this way,* he explained. *The programming languages themselves are in English.*

Listening to Fabio's comment about programming languages being "in English," I remembered a conversation I had a week earlier with Rodrigo Miranda, a coordinator of a Rio-based open source project whom we met in the Introduction. In the early 1990s Rodrigo worked in software translation. At some point a call came from Microsoft: they needed someone in Brazil to translate Excel's Visual Basic into Portuguese. Rodrigo could barely hold laughter when telling the story. *Yes,* he told me, *they literally wanted to translate all the keywords. They wanted to make it "se" instead of "if," for example.* Despite the prospect of making good money quickly, Rodrigo tried to dissuade Microsoft from doing this. When they decided to go ahead with the project, however, he agreed to do it—the money was too good to pass up. "I was telling them that this shouldn't be done, but that if they want to get it done, I should do it." *Of course* it failed miserably, he continues, and people were harassing him for a very long time. "Which idiot translated Visual Basic into Portuguese?" they would ask. "It was me," Rodrigo would reply. If the person continued to say what a stupid idea they thought it was, Rodrigo would typically explain: "I got a new computer from this. Is that a stupid idea?"

I told the story to Fabio and Mauricio. Too young to have witnessed the fiasco, they found it most entertaining, rolling their eyes. *This must be one of the stupidest ideas*

ever! they exclaimed at the same time. *How would you even do it?* asked Fabio. *How would you translate "DIM?" What does DIM stand for anyway? Dimension? So, perhaps it would be "Dimensão." This would be so strange and so verbose!* I pointed out that "Dimensão" could be abbreviated just like "Dimension" was—in fact to the same "DIM." *I suppose,* conceded Fabio. *But Portuguese just isn't a good language for programming languages. The grammar is too complex. What would you write in the end of the function? "Retorno? Retorne? Retornar?"* Fabio rattled off several forms of the Portuguese verb "to return." In English it all makes more sense, he concluded.[102]

Lua: If or Se?

The feasibility of designing a programming language with Portuguese keywords had also come up two months before my conversation with Fabio and Mauricio, when I conducted one of my interviews with Roberto Ierusalimschy, a professor at PUC-Rio. In

---

102 "Retorno" is the noun "return", "retorne" is the formal imperative of the verb "retornar" (to return). A formal word that could traditionally be used only intransitively (to return from somewhere), it is now also used transitively (to return *something*) by programmers—a usage clearly influenced by English. "Devolver" would be a better translation of English "return" in the transitive case, and "voltar" would perhaps be a better translation for the intransitive case. Interestingly, however, many programming languages exploit the ambiguity of English "return," allowing it to be used either by itself or with an object. Given that, the English-influenced "retornar" might be the only reasonable translation.

Rodrigo Miranda added the following comment when reviewing this chapter:

> My point about translating the VB [Visual Basic] language was more concerned about the fact that once you have N versions for VB (one for each locale) you can't copy and paste source code between different locale versions of Office. A thing that happens not only when you get code from the web, but also when your company uses more than one locale version of Excel. This is very common in Brazil, some people use the Brazilian version of Windows or Office, some refuse to use other than the English version.

In other words, Portuguese version of Visual Basic would introduce "language barriers" not only between Brazilian and American programmers, but also between those Brazilians who choose to use it, and those who will stick with the English version.

the early 1990s Roberto (following my interviewees, I will refer to him by the first name) and two of his colleagues[103] designed a programming language called "Lua," ostensibly for the needs of a specific project done for a particular large client of Tecgraf, a PUC research laboratory where Roberto worked at the time (Ierusalimschy et al., 2007). By the late 1990s Lua became fairly popular outside Brazil for certain types of software applications. As I show later (chapter 3.2, " Porting Lua"), to understand Lua's success we need to look at how it got successfully "disembedded" (Giddens 1990) from its local context, becoming "portable" to the unknown contexts of future use abroad. (This requirement is not universal—projects born at the center can typically leave it to the users to re-create the necessary context.[104]) In the process, Lua has become somewhat foreign in the land where it was born. Here I focus more narrowly on the use of English and Portuguese by Lua and the Lua community.

"Lua" means "moon" in Portuguese. "In our language it is a very beautiful word," Roberto wrote on the list in the late 1990s.[105] This name, however, is Lua's only connection with the Portuguese language, and a dubious one at that. "Lua" is also a pun on "SOL"—the name of Lua's predecessor; "SOL" means "sun" in Portuguese, but is at the same time an English abbreviation: "Simple Object Language." At the time of our

---

103 Lua was developed by a team of three people at PUC-Rio: Roberto Ierusalimschy, Luiz Henrique de Figueiredo and Waldemar Celes. To simplify the narrative, this my discussion of Lua in this chapter focuses just on Roberto Ierusalimschy. Lua's history is discussed in more detail in chapter 3.2.

104 The designers of Windows Vista, for example, did not have to worry about what hardware their future users might have and how to make the operating system work on them—instead, they *tell* the users what hardware ought to be purchased and *tell* the manufactures of peripherals what drivers ought to be written.

105 In a slight variation on this comment, another member of the team wrote in 2007: "'Lua' is a beautiful word in Portuguese and has a poetic meaning. But it means 'toilet' in Hawaiian and is also a Hawaiian martial art. It also means something in Vietnamese, but I don't know what. "

interview, Lua's documentation was available only in English. All of several books written about Lua were written in English. (The most popular one, by Roberto Ierusalimschy, had been translated into German and Korean but not into Portuguese.) Lua's manual became available in Portuguese only in September 2007, six months after our interview and ten days after a version in Russian was released.[106] It perhaps would not surprise the reader when I say that Lua's keywords are all based on English words. Or to put it differently, Lua uses "standard" keywords, such as "function," "if... then... else," or "return."[107]

When the first version of Lua was being developed in 1993 the choice of language was seriously discussed, if briefly:

> Roberto: I remember that we discussed a lot about both error messages and reserved words. There were people, even me, that talked about... that maybe instead of "if" we should use "se" and use "enquanto" instead of "while." And we just decided that this is not English—this is reserved words. Someone said that, I don't remember who: those aren't even quite English words, even for English people they couldn't... that they were picked by European people who didn't speak English properly. [Laughs.] But anyway, so we decided to stick with the usual reserved words. And I think that the error messages went together; they should be in English, it would be strange to write "while ..." and then get "Erro na linha..." So *maybe* comments were in English for the same reason. I really don't remember. I can maybe try to find some... but I think that I usually already wrote many things in English. [...]

---

106 Prior to that, the only documentation available in Portuguese was a brief summary at http://www.lua.org/portugues.html. The Portuguese translation of the manual was made by a Ph.D. student of Roberto, whose work was funded by the money obtained by Rodrigo Miranda (see "Kepler's Wiki" below). The Russian translation (http://www.lua.ru/doc/) was made independently from the Lua team and without their knowledge.

107 Each of Lua's twenty keywords has appeared earlier in some programming language, and nearly all of them represent a conventional choice of the options that had been earlier used for a particular function. Standard libraries that come with Lua, however, add a substantial number of names that are not borrowed verbatim from other programming languages, but are also all based on English words.

> Yuri: I am curious: at the time who did you see as the audience for those comments and error messages?
>
> Roberto: For the error messages for sure Tecgraf—that's why we discussed it. For the comments just me, Luiz Henrique and Waldemar. We didn't think about other people reading the comments. And we put them in English because... That's what I said: I tend to use English in comments because it is difficult to put accents.[108]

Roberto's mention of reserved keywords not being "true" English most likely refers to "else"—one of the most common keywords in modern programming language, used to introduce an instruction to be performed when a particular condition fails ("if *condition* then *do X* else *do Y*"). The "if... then... else" idiom was introduced by an international group of computer scientists working in Europe (with the US scientists present, though) and is arguably poor English.[109] The statement that the "usual" keywords are not actually "English," is of course immediately contradicted by the fact that their use necessitated English error messages.

I said earlier that Lua was *ostensibly* developed for a local project. I do not doubt that the needs of Tecgraf's clients were the *main* reason for Lua, and that Lua's later success in fact came as a major surprise to its authors (as they assert in publications and

---

108 All of my interviews with Roberto Ierusalimschy were conducted in English. I have corrected minor grammatical errors, but preserved his choice of vocabulary and manner of speaking.

109 Such use of "else" goes against English usage since it would be ungrammatical in English to say "If it is cold, then take a taxi, *else* walk." The *else* statement was originally introduced into ALGOL 60 in the early 1960s, after a meeting in Paris attended by some of the most prominent computer language designers of the time. While many of the attendants were Europeans and not native English speakers, there was also a substantial number of US-born computer scientists present, so the choice of *else* might have more to do with the computer scientists preference for a short and simple keyword over the more English "otherwise," which had been earlier introduced for the same purpose in COBOL and its predecessor FLOW-MATIC. (FLOW-MATIC and its successor COBOL were the first programming languages designed to look like English, rather than like mathematical formulas, though ALGOL and FORTRAN were developed at the same time. Most popular programming languages today show strong influence of ALGOL 60.)

interviews). We must keep in mind, however, the factor that I call "subvocal imagination" and that one of my interviewees calls "megalomania"—imagined futures that are treated as too unlikely to be publicly presented as a rationale for action, but which nonetheless can affect action profoundly (see chapter 1.3). In chapter 3.4 ("Glocal Dreams"), I talk about my discussion with "Pedro" (whom we will also meet later in this chapter), who at the time of my interview with him, was finishing his final project for an undergraduate degree. Much like Lua, the project was designed for the needs of a very specific client, but was written in English, top to bottom. When asked why, Pedro hesitated to answer, laughed, then said: "Out of megalomania." *What if*, the project would succeed and become big? Pedro was mocking his own global imagination, calling it "megalomania" (a word as derogatory in Portuguese as it is in English), yet he entertained the idea enough to write all of his code in English, on the off chance that he would one day be hiring programmers in India. I suspect that similar "subvocal imagination" was likely a factor in the choice of Lua's keywords. (Another important factor—Roberto's broader preference for English in public contexts and the lack of support for Portuguese accent marks—is discussed in the following section.)

The day after our interview, I received a message from Roberto, with the subject "Paleontology." Roberto told me that after our conversation he went looking through old files, finding that while Lua and SOL code were written "in English" from the very beginning, test files for them were in Portuguese until 2003, over a ten year period.

> On the Sol project, I found a file called "teste", which (like its name) is in Portuguese. [...]

> BTW, the official tests for Lua were in Portuguese (variable names, comments, etc.) until version 5.0! (They are not shipped with the distribution.)
>
> So, I guess I already had an habit of writing code in English and tests in Portuguese...

This distinction should make intuitive sense in light of the literature on diglossia. When two languages co-exist, they can do so by taking different functional roles. One language ("High" or "H") may take the function of a *public* language, while the other language ("Low" or "L") serves as the *private* language. L is used for things that are not seen as being serious—talking to friends, perhaps writing a diary. H is reserved for formal communication. L may be seen as unsuited for this later domain, either because it is seen as lacking in resources (remember Fabio's notion that Portuguese is unsuited for code because its grammar is too complicated), because it might not be understood by the larger audience of such communication, or because it would bring into doubt the author's command of H. (Since H is typically acquired in school rather than learned natively at home, preference for L can cast doubt on the author's education and hence social status.)

Lua's implementation code is public, not only in the sense of being available to the general public, but also in the broader sense of being visible to Roberto's peers. (Note that while Alta's code is not released to the general public it is also somewhat "public" in this latter sense of being open to programmer's peers and to Alta's clients, though it is not circulated as widely as Lua's code.) Public code must adhere to certain standards—it must be "clean," it must be well-organized, and it must be in English. Test files, on the other hand, are often treated by programmers as private ("not shipped with the

distribution," as Roberto says in his email). The standards for such files are thus more relaxed, and beauty can be sacrificed for getting the work done faster. Such files can, among other things, be written "in Portuguese," or, rather they can tolerate a mixture of the two languages which many developers consider inappropriate in the public code. The test files for SOL, which Roberto sent me later, as well as the later test files for Lua, did contain such a mixture of English required by the language itself and Portuguese used for names defined inside the test:

```
type @curva {c:bool, s:string = "profundidade"}
```

Other software developers also say that they often write "quick" test files in Portuguese.[110]

This distinction between public and private code corresponds to a distinction employed in choosing a language for prose: as we will see below, Roberto has preferred writing articles in English for the last fifteen years and in fact lacks the tools needed for writing Portuguese professionally. He writes his email in Portuguese, however.[111]

The situation has changed somewhat over the last decade, as Lua development has

---

110 While many programmers prefer to write "in English" and some "in Portuguese" there is also a third approach, where the two languages may be mixed in the same name. For instance, a function that returns a list of students may be named "getAlunos." This pattern of mixing makes sense if we consider the linguistic distinction between *function* and *content* words. Function words express basic grammatical relations and are typically few. Content words name entities, and are often counted in tens of thousands. In case of language contact, the mixing of the languages often proceeds in an asymmetric way, with one language (the *host* or *matrix* language) absorbing certain words from the other (the *embedded language*, also called *substratum*). (See chapter 6 in Muisken, 2000 for a discussion of the issue.) While "get" is a content word in English (that is, just another verb), in Java it conventionally represents one of a small number of basic relationships between a function and an entity. It can thus be seen as a *function* word. "Alunos" is a content word, denoting a part of programmer's local reality. In that sense, "getAlunos" is *English* (or, rather "English-Lua"), which is borrowing vocabulary from Portuguese—the embedded language. Note the opposite is the case in speech, where content words from English are inserted into the "matrix" provided by Portuguese.

111 The first two messages sent to the Lua list were similarly in Portuguese. The first one read: "Estou apenas testando a lista." ("Just testing the list.") The second - "acho que funciona :-)" ("I think it works"). Since then, the list has functioned in English.

been making small steps towards the open source development model. This transition demonstrates one of the new challenges that open source development brings to peripheral participants. Open source development blurs the line between public and private, as the users of the software often want to have access to the author's complete working environment rather than just the final product. Reliance on the local language becomes problematic even for such things as tests, since the open source logic demands that such code also should be rendered public. In 2001 some members of the Lua community started asking for Roberto's test scripts. In March 2002 Roberto finally responded to one of such requests with a URL for the zipped tests (still in Portuguese) on Roberto's website (rather than the Lua.org site)—in a place where they would be unlikely to be found. (The only way to find them would be to read the message that Roberto sent to the list.) After more requests for tests (from those users who quite literally "did not get the message") Roberto finally announced in 2006 that he had translated the test suite into English (despite some participants' assurance that this was not necessary) and offered to "move it to (some obscure place in) the Lua site." As of October 2007, the files still remained in a hard-to-find corner of Roberto's personal website.[112]

Lua's Documentation

Much like the comments in Lua's code, its original manual of eighteen pages was also written in English, as was the very first paper about the language, presented at a

112 The process of adapting to the foreign rules, including Lua's step towards being a "proper" open source project, is discussed in chapters 3.2 (" Porting Lua") and 3.4 ("Glocal Dreams").

Brazilian conference in 1993. The conference accepted papers in English, Portuguese or Spanish and the choice of language did not make any difference at the time as to how the paper counted towards researcher's productivity metrics. Nonetheless, there were several good reasons to choose English. The first one concerned the larger audience that could be reached by an English publication, or, rather the exceedingly small size of the Portuguese audience:

> Roberto: The problem I see is that when you go to a specific academic area, even in the world, the number of people interested in this is small. So if you write in Portuguese... In Brazil maybe there are four or five people who are going to read what I write. It's not a problem of Portuguese, it's a problem of any language. You must write in a language that everyone can read, unfortunately, or fortunately, because the number of people is so small. There is no point of writing a technical paper in Portuguese, or in Spanish, or in French, or in German or in whatever language.

Consequently, Roberto nearly always wrote his papers in English. "I usually prefer to write papers in English," he said. "Since that time I only wrote papers in Portuguese when it is very fast. Currently even if it's fast I prefer to write in English." Earlier in his career, he used Portuguese for the relatively unimportant papers ("when it was very fast"). Now he prefers English in all cases.

Roberto then mentioned a different factor, the technical challenges of writing papers in Portuguese:

> Roberto: And currently my tools—even when I want to do something in Portuguese, using LaTeX [typesetting software], I always have a lot of problems. I am not even well prepared to write Portuguese anymore.

As most computer scientists in the US, Roberto uses a software product called "LaTeX"

to write his papers.[113] LaTeX is essentially a programming language designed for typesetting documents, and, like most programming languages, it cannot easily handle non-English letters or accents. (This limitation also applies to Lua itself—a Lua programmer who wanted to give Portuguese names to their variables would have to forgo the accents.) "There are a lot of packages to solve that but I do not have them installed properly," said Roberto, "Or sometimes they *are* installed properly but I change the version and I only discover two months, three months later that they are not working. Apart from emails I almost never write anything in Portuguese."

English is thus intertwined with the practice of academic computer science in a very material way, being assumed by many of the tools on which the practice relies. Roberto started using LaTeX during his post doc in Canada in the early 1990s (he did all of his earlier training at PUC) and has been using it ever since, even though he says many of his colleagues "even today [...] prefer to use Word." (Roberto's post-doc abroad is not unusual for his department—all but three faculty members either did a Ph.D. or a post-doc abroad, typically in the US, Canada or France.[114])

Nearly all publications about Lua were thus written in English. Around 1996 one such publication, in a popular American computer programming magazine, attracted a substantial number of questions, which led Roberto and his collaborators to start a mailing list. (No mailing list was necessary before that since all users worked in the same room—see chapter 3.2.) From the beginning, most of the subscribers were foreign and

113 Many US computer science conferences until recently preferred papers submitted in LaTeX format and the connoisseurs can often tell that the paper was typeset in LaTeX just from its appearance.

114 See appendix D ("Doctoral Degrees of PUC-Rio Department of Informatics Faculty").

the discussion was conducted in English, though messages sent in other languages occasionally popped up and were typically met with friendly amusement or curiosity rather than disapproval. In 2002, when the community had grown substantially, a question was raised whether a separate Portuguese list was needed or whether the list should be declared officially "multi-lingual." In response to this discussion Roberto wrote:

> Currently, the list members from Brazil (.br domains) are only ~15% of the list, from a total of almost 30 countries. (Our second "minority" are 10% of German speakers.) Whether we like it or not, the only language we can all communicate is English.

The message then explained that poor English was welcome: "We can have a much wider audience with poor English than with good Portuguese :-("

In 1999, another member of the Lua team announced on the list that Roberto was working on a book on Lua. Seriously or not, an American list member asked: "Do us poor language-handicapped folks in the States have a chance of being able to read it?" "I hope so," responded Roberto, "I am writing it as close to English as I can :-)" The book was in fact written in English, and a draft was published on the list in 2000 and was met with much excitement. ("'Programming in Lua' is absolutely the most lucid introduction to a programming language I've ever read," wrote one of the list members.) Lua was changing so fast and so radically, however, that the book was obsolete before it was finished and had to be re-written for the new version of Lua. In early 2003 the new version of the book was finally ready, and Roberto asked the list for ideas on how to publish it. The list members responded with suggestions, offers to proof-read the book or to represent Roberto in the United States. (It was largely taken for granted that the book would be

published there.) O'Reilly, the most well known publisher of computer books in the US, was everyone's top suggestion.

Roberto did contact O'Reilly, but received a response that said Lua was too much of a niche language to support a book.[115] Some of the list members tried to convince Roberto to release the latest draft of the book electronically and ask the readers for donations, but he (and some of the other members) felt that this would jeopardize a future contract with a publisher. The book was eventually self-published via a print-on-demand service that also acted as a distributor and could thus satisfy the number one requirement: the book became available for purchase on Amazon.com.

Roberto's announcement of the book's availability on Amazon came with a note in parentheses: "For those in Rio de Janeiro: I am trying to print a small local batch to get a better price, so you may want to wait a little..." This plan, however, never materialized. Roberto then tried to get the book into some of the Brazilian stores, including those online, but again without success. The fact that the book was written in English presented a problem, however, since Brazilian stores as a general rule do not sell books in English.[116] The fact that it was a *self-published* English book created even bigger challenges. The stores did not want to buy the book from a foreign distributor, and insisted on buying it

115 O'Reilly later did agree to publish the second edition of the book and went as far as to pay an advance, but then canceled the book together with many others after a management change.

116 A few weeks earlier, in fact, while browsing through books in PUC's book store, I noticed books on some relatively obscure languages, such as Haskell, but not Roberto's Lua book. I then realized that all books that the book store was offering were in Portuguese—including the book on Haskell, written by a Brazilian in this case. There simply was no Portuguese book on Lua that they could sell. I am not quite sure why Brazilian bookstores do not sell English books. My guess is that they simply cannot compete with Amazon on prices of English books and thus stick with Portuguese books where they have more of a captive market.

directly from the publisher instead. Since the book was published by Roberto himself, he would need to set up a company to do proper accounting of books sales and expenses. This ordeal of starting a company is not taken lightly in Brazil: it is hard to start one, expensive to maintain it, and nearly impossible to close. (Roberto would therefore need to pay an accountant and possibly taxes for years to come whether or not the venture would end up making any money.) Foreign publishers avoid this problem by maintaining their tax home abroad. Roberto could avoid it by staying out of the Brazilian market.

Writing a book in English, thus gave Roberto not only access to a larger audience, but also to more efficient institutions for publication and distribution of books. While some of those problems might have been avoided had Roberto written a book in Portuguese and found a Brazilian publisher, many would have remained and many other would have surfaced. Writing a book in English, self-publishing it and making it available on Amazon.com might after all be the most efficient way to reach *Brazilian* readers. Many of the developers are quite used to buying books from Amazon, pointing out that international shipping costs less than the markup charged by Brazilian book stores.

While one cannot buy a copy of *Programming in Lua* in any of the Brazilian book stores, there are at least a few places in Rio de Janeiro where it can be obtained. One can buy it from Roberto in his office, where he maintains a small stack, as I did at the end of that interview. One can also borrow the book (as I had been doing earlier) or sometimes even buy a copy from Rodrigo Miranda, who also keeps a few extra copies in his office in Copacabana and buys more copies from Roberto when they meet at PUC. The situation exemplifies a more general pattern. While Lua maintains both local and global ties, the

local ties are *really* local (mostly within the PUC network), while there are few ties at the city or national level. One can buy the Lua book from Amazon (global), or from Roberto's office (really local). One cannot buy it in a local book store or in a Brazilian bookstore online.[117] The only way to pay for the book in Brazilian *reals* is to hand the *cash* to Roberto or Rodrigo. In much the same way, Lua benefits from contributions of many Brazilians, but those are almost always people who know Roberto *personally* (his former PUC classmates or students). The virtual contributors (who are also quite important), are almost exclusively abroad.

By the spring of 2007 the book had been translated into German and Korean (and later also into Mandarin), but no Portuguese translation is available to this day. Rodrigo Miranda has been trying to organize a translation but so far without success; a half-completed translation by his friend Renato died with Renato's hard drive. Rodrigo had not yet had the courage to tell Roberto about the loss, but Roberto was hardly counting on this effort to come to fruition:

> Roberto: But for many years, I think since the first edition, that Nas Nuvens, Rodrigo Miranda and his brother João,[118] they plan to translate the book to Portuguese because *they* have a lot of users of their tool that for sure prefer Portuguese documentation. But they are planning that for many years, but nothing happens till today. [Laughs.] And they are still planning this translation. And apart from that I've never got any contacts.

Roberto said that he had offered to revise a book if it ever got translated, but had little hope for the book, considering the difficulties in distribution described earlier. ("The

117 While the book is only translated into two languages (German and Korean), the English edition is available in online stores in Canada, France, UK, Germany, Korea and Japan.

118 The real names in the quote are replaced with the pseudonyms that I use elsewhere in this text.

biggest problem is not the translation," he explained. "The biggest problem is after that, what we are going to *do* with the translation."") Meanwhile, he is content with waiting to see if anything comes from Rodrigo's efforts.

> Roberto: There is nothing else I can do. I am not going to die to translate this book to Portuguese. I really would *love* to have it translated to Portuguese but I am not going to translate it and as I told you, it's really not in my mind to beg people for things. So I am not going to try to convince other publishers in Brazil whether they would like to translate my book.

After all, Roberto did not have to *ask* anyone to get the book translated into German or Korean.[119]

Kepler's Wiki

Lua is a niche language, which has gained substantial popularity in a particular set of applications, typically those requiring performance, simplicity and close interaction with software written in C. In California, Lua is primarily used in development of computer games or small devices. Both of those applications are too specialized for Rio's small market, which depends primarily on building web applications for local businesses. Rodrigo Miranda, a former Ph.D. student of Roberto Ierusalimschy, had dedicated a decade of his life to turning Lua into a platform for developing web applications. Doing so has been an uphill battle. As most of its users agree, Lua provides an excellent solution

119 As discussed in chapter 3.3, Rodrigo did eventually manage to organize a translation of *Programming in Lua,* which was completed in January 2009 and is being revised as of the writing of this dissertation.

for a particular set of problems; web development is not one of them.[120] In 2007, I spent four months of my fieldwork following Rodrigo's project (named "Kepler") from inside his office in Copacabana—a story that I tell in more details in chapter 3.4 ("Glocal Dreams"). Since then, I have followed Kepler through an involvement in a satellite project.

After a week of sharing an office with Rodrigo Miranda, I decided to get involved a bit more in the project. (Since the project is so highly "virtual," observing the work in the "real world" was so uneventful that I was starting to doubt if Kepler existed at all. Getting involved in coding and entangled in the network of relationships that comprise the project did cure those doubts and I discuss this issue in chapter 3.4.) I decided to start by helping with the documentation—a comfortably peripheral role, which reminded me of Traweek's (1988) work at SLAC.[121] Kepler's website indeed needed some work. Over the next two weeks Rodrigo and I had some long discussions about the structure of the new website, and I built a prototype. One of the things that Rodrigo asked for was that the website (or at least parts of it) would be driven by a wiki, allowing the users to edit the

120 As discussed in chapter 3.2 (" Porting Lua"), Lua's has been successful in applications that require a scripting language to be embedded into a body of C code, and especially in computer games. In theory, Lua could also work well for many other situations (including web development), if it were not for a lack of libraries—collections of reusable code written in Lua. As I learned first hand, programming in Lua may at times be like trying to bake a cake starting with an unplowed field, growing your own wheat on it, harvesting it, milling it, then figuring out how to make do without yeast. (I also learned the satisfaction that comes with doing things this way.) Lack of libraries is a common affliction for new programming languages, though Lua might be somewhat unique in that up until recently Lua's community did not seem to consider this as a problem, stressing that the specific applications where Lua shines do not require libraries. (Lua's situation with libraries is discussed in more detail in chapters 3.2 and 3.4.)

121 Traweek (1988) worked part time at SLAC's Public Information Office, where she "explained the activities of the lab to visitors, who ranged from junior high school to college students, from the general public to special interest groups like safety experts, electrical engineers, and chemists, from new employees to visiting dignitaries" (p. 11).

content.

A couple of weeks after our first discussion about the website, I realized that having designed most of it, we had not thought at all about where exactly the Portuguese version of the documentation would go. The existing Kepler website did have Portuguese documentation, though it was spotty and the visitor was often sent back to the English version. Running the website as a wiki creates additional challenges: we would now need to figure out how to synchronize the content between the two languages. I brought this issue up while talking to Rodrigo over Instant Messenger (IM)[122]:

(5:29:39) Yuri: what worries me is the howto. if we'll be relying on users to extend the howto, how do we make sure that Pt. version keeps up?
(5:30:40) RMiranda: we don't
(5:30:44) RMiranda: that's the whole point
(5:30:50) RMiranda: I plan only to offer the mechanism
(5:30:52) RMiranda: not the content
(5:31:00) RMiranda: and count on the community for that
(5:31:09) RMiranda: we have fernando for example
(5:31:19) RMiranda: he can monitor the site and update things
. . .
(5:33:38) Yuri: are you planning on adding more languages later?
(5:33:51) RMiranda: surely not
(5:33:56) RMiranda: i wouldnt add not even pt
(5:33:57) RMiranda: :)
(5:34:04) RMiranda: thats finep requirement
(5:34:04) Yuri: explain
(5:34:09) Yuri: oh, i see
(5:34:11) RMiranda: too much trouble
(5:34:24) RMiranda: I have nothing against supporting it on the wiki
(5:34:30) RMiranda: if people manage the content
. . .
(5:35:37) Yuri: what does Finep require?
(5:35:50) RMiranda: all of the product documentation in pt
(5:35:56) RMiranda: but this can be a static html or pdf

122 At this point, I happen to be at home while Rodrigo is likely in his office a few blocks away.

(5:36:07) Yuri: doesn't Fernando's document satisfy this?
(5:36:12) RMiranda: exactly

This conversation left me utterly confused. Did Rodrigo want Portuguese documentation or not? It was only a few months later that I managed to make sense of it. Kepler was sponsored by FINEP, a Brazilian agency that funds research projects in industry, and by "Nas Nuvens," a company that belonged to Rodrigo's brother João (see chapter 3.4).[123] Nas Nuvens hoped to eventually offer Kepler-based solutions to its local clients and needed documentation in Portuguese. For this reason, it had requested FINEP funds for writing Portuguese documentation as one of Kepler's sub-projects. There also was an extra benefit: subcontracting documentation work to "Fernando," created an opportunity to build a relationship with Fernando's organization, a research institute near Rio de Janeiro. As I learned later, Fernando had recently sent Rodrigo a draft, but Rodrigo has not found time to read it. In April 2007, Portuguese documentation was yet to become a high priority: Kepler had to be finished first and for that it needed attention of developers. This meant documentation in English.

A week after our discussion over IM, we found ourselves rethinking our initial choice of the Wiki software. During that deliberation, I decided to try writing a wiki in Kepler (a decision that will affect greatly the rest of my fieldwork). A few days later, with the initial version of the wiki ready, we revisited the question of Portuguese pages, talking about ways of interlinking Portuguese pages with their English equivalents.

The URLs of wiki pages are typically based on their titles, which makes it easier

123 "Nas Nuvens" literally means "in the clouds." See chapter 3.4 for the explanation of this pseudonym.

to create pages and to link to them. The URL of a page called “Introduction” would end in “Introduction.” If the same were true for Portuguese pages, then the corresponding Portuguese page would show up under “Introdução”. This creates a minor challenge for automatically linking corresponding pages in two languages. When I asked Rodrigo what he thought about the issue, he expressed shock at the suggestion of using Portuguese in URLs. *Absolutely not!* he says. *The pages names that appear in URLs should most definitely be in English!* He did not want to have any accents in URLs, he explained, and he wants the pages to be easily mappable. *And we are not going to have a page called “Introducao”!* He carefully enunciates the hard “c” and the “a” of “Introducao” to show the effect of the missing cedilla and tilde.[124] *This page should be called “Introduction.” Calling it “Introdução” would be just as ridiculous as using Portuguese words for Lua or Kepler keywords*. (Using Portuguese keywords in a programming language or a framework apparently defines “ridiculous.”) Sure, this meant that Portuguese pages will need to specify the English page name for every link, continued Rodrigo, but that is a small price to pay.[125]

Another week later, Rodrigo and I were about to leave the office when we got

---

124 The *-ção* of Portuguese “Introdução” is pronounced somewhat like English verb “sung.” Without the accents (“-cao”), the pronunciation approaches that of English “cow.” The difference would therefore be quite noticeable even without the extra stress that Rodrigo puts on “cao.”

125 Wiki engines simplify the creation of links between wiki pages by allowing the user to link to the pages by referring to it by the title. In Wikipedia, for example, putting “[[Introduction]]” in page, will generate a link to the page called “Introduction.” In this case the link will appear on the page as the world “Introduction” and will link to a page with that name. Occasionally, however, the author may want the link phrase to be different from the title of the destination page. For instance, she may want to make a link to a page called “Portuguese Language” but use only “Portuguese”as the link phrase. In this case, the title of the article and the link phrase are separated by “|” (“the pipe”), e.g.: “[[Portuguese Language|Portuguese]].” Rodrigo’s suggestion is to simply use the same approach for all pages in the Portuguese section of the wiki: [[Introduction|Introdução]].

chased by Rodrigo's brother João (a partner of Nas Nuvens). João asked Rodrigo about the documentation—he had been busy trying to "sell" Kepler to local companies (so that he could literally sell Nas Nuvens' services later) and Portuguese documentation would help. Plus he is worried about delivering the documentation to FINEP. Rodrigo told him that he has glanced at Fernando's document and that it was good enough for now. (The FINEP project is not due for another nine months.) "You are now talking like Roberto," said João in frustration. I perceived that this was not meant to be a compliment. Rodrigo in fact got defensive. *It's not quite the same,* he says. "I am not saying that there *shouldn't* be Portuguese documentation," he explains. There were just better things to worry about. The elevator arrives giving us a chance to escape, leaving João behind to worry about Portuguese documentation by himself.

Kepler's complicated relationship with Portuguese documentation reflects the complex way in which Kepler connects to two different worlds: the local world of Brazilian clients and funding agencies, and, on the other hand the foreign world of web development in which Kepler is trying to find a place. Roberto's situation is simpler. As a Brazilian academic researcher, Roberto is shielded from many local factors. For him, Portuguese documentation is simply a luxury (or perhaps something "cute"—the word that some of my interviewees use to dismiss Lua).[126] Rodrigo Miranda's approach to Portuguese documentation, on the other hand, puzzled me for many months, as he went back and forth between spending time to figure out exactly how it would work and

126 Rodrigo's accounts present Roberto as more dismissive of Portuguese documentation, compared to his position in our interviews. This difference may be due to a combination of him changing his opinion over time (Rodrigo has had a substantial influence on him in the last few years, it seems) and presenting himself differently to me (a foreign researcher with a recorder, as Rodrigo points out to me) and to Rodrigo (a former student).

seemingly questioning any need for it.[127] (As I show later, Rodrigo's position on creating a mailing list in Portuguese—in addition to the current one, which is in English—has similarly changed a few times.) For Rodrigo, Portuguese documentation was a balancing act between conflicting commitments. Kepler's destiny was tied to that of Nas Nuvens. As we will see below, some programmers at Nas Nuvens could read English only with difficulty and the company needed Portuguese documentation. Like many other software companies in Rio, Nas Nuvens also was dependent on local clients and had to build other local alliances. (The reasons for such dependence, which are not unique to Nas Nuvens, are discussed in the consequent chapters.) Yet the small funds that the clients and FINEP could provide, did not allow Rodrigo to obtain what he needed most for his project: expertise. He looked for such expertise in two places: abroad, by trying to recruit invisible programmers from the Lua community and at PUC, among people who considered it silly to spend time on Portuguese documentation, not to mention worrying about how to deal with accents in page names.

Learning English

The fact that Roberto Ierusalimschy's *Programming in Lua* was available only in English did inconvenience some of the developers working with the language. For Luciano, one of the programmers at Nas Nuvens, *Programming in Lua* was the first and

127 It changed once again in April 2008 when Rodrigo contacted me for assistance in setting up a Portuguese wiki using the method I originally suggested. Note that by that point, my wiki had acquired a life of its own and was drawing on the input of other contributers based outside Brazil, so Rodrigo was looking for the "standard" way to handle foreign language content.

the last (as of 2007) book he had read in English:

> Luciano: The only book in English that I've read was the Lua book. Just that.
>
> Yuri: You actually read it, beginning to end?
>
> Luciano: Uhu.
>
> Yuri: And understood?
>
> Luciano: [Laughs.]
>
> Yuri: Did it take long?
>
> Luciano: Very very long.
>
> Yuri: But now that you've read one, reading another book must be easier, right?
>
> Luciano: Yes, but I am not going to run the risk again. It takes forever.[128]

Luciano is twenty years old and is of lower middle class background.[129] He went to a public school, where he had English classes, which he says, taught him nothing. ("English for twelve year olds. Doesn't count.") What *did* teach him English, he says, were role-playing games (RPG) on a computer:

> Luciano: Games, RPG games, you know? I played a lot—the RPG games. The games gave me the minimum and then... For example, to buy things, you need to apply [...] the word "buy," and the word "sell," you know? [...] If you need to open this door, it's "open the door." You apply: "open the door." [Pause.] It's... it's *buy, right?*[130]

---

128 See appendix C, "Original Interview Quotes," (Luciano, July 2007, "O único livro em Inglês").

129 I use the terms "middle class," "lower middle class," and "upper middle class" as they are used by my interviewees. Roughly, lower middle class families are those families that can keep their kids in school through the end of high school but cannot pay for their college education or support them after high school. Middle class families can support their children through college, though cannot pay for expensive private schools like PUC.

130 See appendix C, "Original Interview Quotes," (Luciano, July 2007, "Jogos RPG"). Luciano's "buy" in the end of the quote appears to refer to the discussion of buying/selling a few sentences earlier, rather than confusing the English words "buy" and "open."

Luciano was not the first person to mention computer games as the introduction to English. Since computer games serve as a common entry point into the tech world for many Brazilians men, Luciano's learning of English was from the very beginning pulled by his growing involvement with the tech culture.[131]

Around the same time as he started playing computer RPGs, Luciano started fixing computers as a part time job, then got interested in the Internet, got himself a computer and started learning HTML. He got a job in tech support, and started learning Linux and PHP—a popular programming language for web applications.[132] His source of learning was translated books and online forums in Portuguese ("some forum mentioned PHP," he says). It did not take him long to discover, however, that most of the material he needed was in English and he started trying to make sense of the English on the web. He is now strategic about selecting resources in English or Portuguese, considering the time it would take against the likely utility of the documents:

> Luciano: For questions—anything. You know? If I have a question... Here: [unclear] monitor resolution [unclear], something like that. I would put this into Google in English. [...] But for learning... I want a tutorial about [unclear]. Then I will put it in Portuguese.[...] With English it is really easy for me to get lost, one word will totally change the meaning of everything. I can't be sure I got it right.[133]

Step by step, however, he seems to be growing more competent in understanding written English.

---

131 The link between English, games and software is discussed again in chapter 2.1.

132 PHP is often considered a "low end" web development language, since it is easier to learn, while Java is a "high class" language. (There is a somewhat natural transition from HTML to PHP, while understanding the basics of Java typically requires some computer science training.) PHP programmers typically earn substantially less than Java programmers.

133 See appendix C, "Original Interview Quotes," (Luciano, July 2007, "Fácil me perder").

The need to solve practical problems and the interest in engaging with the tech culture, however, are not the only motivations for learning English. The language also is widely seen as more generally providing access to the larger world. During one of my days in Alta, I started a conversation with a PUC student who had recently joined the company as an intern and was taking time to learn the new technology that Alta is using. Looking at the stack of books on his table, all in English, I took the opportunity to ask him how he learned English. *I learned it by myself,* he says. *Never took any course. I wanted to understand song lyrics for bands like Black Sabbath.* I asked him if he ever *speaks* English. *Not often, but sometimes,* he responded. Sometimes some of his friends end up going out with *gringos*, so he would then talk to them. *Sometimes you go out,* he explained, *and there is a hot woman ["mulher muito gata"], but she is German. So, you need English for communicating.* Another engineer, Flavio, who had been listening to our conversation jumped in. *And I learned English through RPGs*, he says. *Playing a game?* I asked, remembering my conversation with Luciano. *Not so much playing as reading the manual,* he responded. *"I learned by reading."* As I later realized, unlike Luciano, Flavio was not talking about computer games, but "real" role-playing games where the players meet face-to-face to re-enact a fantasy adventure, rolling dice to simulate chance. Flavio presumably spoke Portuguese to his fellow-players, but the game design had been imported from the United States and the manual was thus in English.[134]

Luciano's level of English represents the lower end of the spectrum for Brazilian

134 While not played on a computer, face-to-face RPGs are quite popular among computer "nerds." At the same time, RPGs are associated with a larger world of "nerds" of which "computer nerds" are only a part. (See chapter 2.1, "Nerds.")

software developers: not being able to read technical documents in English (something Luciano can do with some difficulty) is widely considered a serious impediment, and developers with less English than Luciano would have trouble finding serious work. (One young manager that I interviewed complained about a recent "mistake" of hiring a developer without testing their knowledge of English. "That's terrible for someone who is supposed to work in computer science," he explained. The company since introduced an English test for all new job candidates.) His limited English proficiency will also serve as a marker of lower middle class background, highlighting the distinctions between him and PUC-educated professionals like Rodrigo, limiting Luciano's opportunities for career advancement.

Due to the perceived importance of English in software work (and the lower class connotations of inability to speak it), developers are reluctant to admit their use of Portuguese and often overstate their use of English, as is demonstrated in this interview:

> Edmundo: I use Google... For example, I have a question about how to read a binary file. I forgot. How do I do this? [...] I do a *search* [says in English]: "*java binary file*" [says in English]. Then I see how people have done the code and do mine.
>
> Yuri: You always use English words?
>
> Edmundo: Always. English words. Ah... I beg your pardon: 80% in English, 20% in Portuguese. And I will tell you why. Why 20% in Portuguese? Because here in Brazil we have one of the largest Java communities in the world. [...] And 80% because in the whole world [...] there are Java programmers all over the world, so it's easier to find a word that you post in English than in Portuguese. [...]
>
> Yuri: But how do you choose [which language to use]?
>
> Edmundo: Good question. How do I choose... Do you really want to know? Do you really want to know? When I am too tired to write in

> English, then I enter it in Portuguese. When I am too tired to write in English. [Chuckle.] There isn't a pattern like "Ah, now I'll search in Portuguese." [...] When I am tired. [Laughs.] When I don't want to read English, when I don't want to read *anything*. [...]
>
> Yuri: So, reading English is more difficult for you?
>
> Edmundo: [Pause.] Of course. Because it's not my native language. It's not my native language. It's like you, even though you've lived in the United States already for five years, or three years, or was it five? [...]
>
> Yuri: Ten.
>
> Edmundo: Ten? Ten years, damn! And still reading in Russian is much easier than reading in English.[135]

What is significant about this passage is not the fact that Edmundo finds it more difficult to read in English than in his native language, but his reluctance in admitting that. Edmundo sees his use of Portuguese as a weakness—a weakness that he hopes I would understand given my own status as a non-native English speaker. As most interviewees he feels that a software developer is *supposed* to stay up-to-date (*se manter atualizado*) in the world of technology by going "directly to the source" and using the dominant language of that world rather than getting the information second hand.

Those who learn to read English well typically learn it the same way: through private English courses. While a handful of developers who went to some of the best private schools say that they learned English there, as a general rule, neither public nor private high schools are mentioned when developers are asked how they learned English. Neither are the universities, whether public or private. The better universities may expect their students to read English, especially for Master's courses, but they do not teach those skills. Instead, most developers mention one of a number of companies that offer private

135 See appendix C, "Original Interview Quotes," (Edmundo, October 2005, "80% em Inglês.").

English courses, typically at night. One of the developers at Nas Nuvens says:

> Pedro: Because what I did is I didn't do my undergraduate program in four years. I did it in four and a half years. I did it in four and a half because I decided to reduce the work load to study English. So, I reduced the load to study English, to study English for a year and a half, with a commercial English course. *If you want, we can talk.* [Says in English. Laughs.] My English is still, still... I hear better than I speak. [...]
>
> Yuri: And how did you learn it?
>
> Pedro: The English course. A semi-intensive course of a year and eight months, which I am doing now. Still doing. So, I reduced my load at the university during the sixth semester. There are eight semesters. I reduced it during the sixth, planning to stay an extra semester. Because during the sixth I started thinking: if I don't study English I won't be able to do a Master's. [...] Because with the Master's program here—they [expect] that a person would be fluent... not fluent, but would be capable of reading. [...] Must know to read and write English. [...]
>
> Yuri: And it went well?
>
> Pedro: For me it did, it worked.
>
> Yuri: Before that course you didn't know...
>
> Pedro: ...any English at all.
>
> Yuri: Didn't even know how to read?
>
> Pedro: Nothing. Not even read. We have here an English course in the primary school, but it is really bad. An English course based on translation. This way you never learn. I knew very little. In terms of reading—very little. I can say that today I can read any article in English.[136]

Pedro's English class also coincides with the time when he made a major step forward in his career as a software developer, starting to learn to program in Ruby using the Rails framework. The course cost Pedro R$200 per month (about US$100 in May of 2007) and

136 For this and the following quote see appendix C, "Original Interview Quotes," (Pedro, May 2007, "Curso de Inglês").

offered two 1.5 hour classes per week in groups of 8 people.[137] For well-off Rio families, enrolling their kids in such courses is an obvious decision, and such kids typically start their course well before college. (Some of my interviewees started private courses as early as ten.) Others, like Pedro, have to wait until they can afford the course, in terms of both time and money. Like Pedro many developers start by doing a number of semesters of a night-time university program, as this allows them to get a better job and increase their earnings. Having established somewhat of a financial base they can then start thinking about spending additional money on an English course and perhaps slowing down they progress in the undergraduate program.

Even for students of PUC, who typically come from expensive private high schools and had taken private English courses since early age, English often remains a language that is often read and heard (in movies and music), sometimes written, but rarely spoken.

The Speakers and the Non-Speakers

A few months before my interview with Luciano, I was sitting in the office of Nas Nuvens, in the room that I shared with Rodrigo Miranda, when Luciano knocked on the door. He stepped inside and said something to Rodrigo. Rodrigo responded by pointing to

137 For comparison, a four year undergraduate program in Information Systems at Estácio de Sá (one of a number of large for-profit private universities operating in Rio) costs R$440-880 (~US$250-500) per month as of September 2007. (The number R$440 is from Estácio's website. Pedro, a recent graduate of Estácio de Sá, says however, that it's a marketing ploy, since the cost is lower during the first semester but goes up to R$880 afterwards.) For comparison with PUC tuition, see footnote 95 on page 98.

me as the person to ask. As I understoond, the question had to do with running Kepler on Linux, which was presumably why Rodrigo forwarded Luciano to me—he had been trying to make use of me as a resident Linux expert, since I was running Linux on my laptop. (Or perhaps Rodrigo just wanted to put me to work. He had recently been making comments that seemed to hint that I was distracting him from work—which I most surely was—so perhaps this was a way to compensate.) Luciano explained to me the problem, which concerned Kepler's LuaZip module. As it turned out, the module needed "zzlib" library, which in Luciano's case was in a different directory from where Kepler's build script expected to find it. I opened the Makefile and pointed Luciano to a variable called "ZZIP." He said he had already changed it and that after that his code compiled, but it would not run—at the run time the executable seemed to again look for the library in the wrong place. I realized that the question is out of my league, and told this to Luciano. *You should ask Alan,* I said, directing him to an active Kepler developer who used to be at PUC but recently moved from Rio to Porto Alegre. *However,* I continued, *I know that Rodrigo would want you to ask this question on the Kepler list rather than emailing Alan directly.* About a week after I arrived, Rodrigo had told me that he wanted to start running Kepler as a "real" open source project and that this would involve routing more communication through the mailing list and relying less on face to face interactions or private email.[138]

Rodrigo nodded. Luciano looked at him in a bit of disbelief. *You are not going to make me do that, right?* said his face. *Yes,* Rodrigo responded to Luciano's silent

138 This topic is explored in more detail in chapter 3.4.

question, *Write to the list*. I did not catch Luciano's response, but the prospect of writing to the list in *English* did not seem to appeal to him.[139] I remembered at this point how a few days earlier I walked into Nas Nuvens to find everyone seated in a circle in the lobby, listening to one of Nas Nuvens' partners, Daniel. The moment I walked in, Daniel greeted me and launched into a lengthy monologue, in the most fluent English (Daniel had lived in the US for a number of years) telling me that they were talking about the wiki I had written in Kepler (see "Kepler's Wiki" above). As Daniel talked, I found it hard to focus on his words, as my attention was drawn to Luciano, who was rolling his eyes while simultaneously keeping them wide open. When Daniel finished—or perhaps paused to inhale—Luciano said something along the lines of "My god, I understood *nothing* of that." The comment drew laughter from nearly all developers.

I was about to volunteer to help Luciano to compose the email, when Rodrigo sighed and said: "Write it, email it to me, I will translate it for you." Luciano nodded and left. A bit later, a message from Luciano arrived via the mailing list. As I later learned, Luciano wrote most of this message by himself, and Rodrigo only corrected a few typos. It also was not Luciano's first message to the list—he had sent one before, that time having it written it all by himself. *But that's it,* he told me in our interview. Rodrigo had recently created a Portuguese list for Kepler, and Luciano would only write to the Portuguese list now.[140]

---

139 A few months later, when Rodrigo sets up a Portuguese mailing list for Kepler, I learn first hand the terror of writing in a foreign language to a mailing list that archives all messages and puts in public view, realizing that anyone who would even want to know exactly how strong my Portuguese was at the time of my fieldwork would only need to know how to use Google.

140 Luciano did participate in the discussion on the English Kepler list in the months since the interview, though he has been more active on the Portuguese list.

A few weeks after the incident with Luciano's email, I was at Nas Nuvens' office again, slouching in a bean bag (or *puff*, as it is called here), while Rodrigo was standing on the floor in front of me. I was trying to explain to him a problem I thought we had with a code example we were working on together. I was speaking Portuguese and I stumbled as I searched for a Portuguese equivalent of "smart quotes." Rodrigo was following my line of thought, however, and completed my sentence for me: "smart quotes." He used the English phrase, but pronounced it as in Carioca Portuguese: "ishmahchi quotish."[141] After we concluded this discussion, I switched to English and asked Rodrigo why he would say "ishmahchi quotish," if he knew how to pronounce this phrase in English.

He sat down on the *puff* next to me, leaned back, and took a breath. *Many people don't know enough English to know how to say it right,* he said. I understand this, I responded, but *you* know how to say it. Why do you say it this way?—*My English is not so good, actually,* he said. "This is ridiculous," I think to myself. Rodrigo's English was almost as good as mine; he must have been avoiding the answer.[142] Listen, I said, maybe your English is not perfect, but you know how to say "smart quotes." It is clear, I continued, that people who are fluent in English often say English words with a strong Portuguese accent when using them in a Portuguese sentence.

---

141 See appendix B ("Transcription and Quoting Method") for a discussion of phonetic aspects of Carioca Portuguese.

142 Rodrigo Miranda was about the most fluent English speaker among the people I met in Rio de Janeiro. Since I am myself a non-native speaker known to drop articles, Rodrigo's English might in some ways be more correct than mine. The only clear difference between Rodrigo's English and mine is his relative unfamiliarity with American slang and idioms, as well as occasional mispronunciation of words that are often read but not heard by Brazilian software developers.

Ok, said Rodrigo, *I'll tell you why. It's because Luciano was in the room. I tend to speak this way when there are "non-speakers" in the room,* he explained. There is a thing about using English in a politically correct way, he continued. *When you use English, you don't want to make it sound like you think you are better than other people and if you speak overly correct English people might think that. If I say "ishmahchi quotish," it makes me just one of the guys. It's a way of "making fun of English, making it less elitist,"* he concluded. *But you do this even when there are just the two of us talking,* I said. *Maybe, he agrees, at some point it just becomes a habit.* Somewhat in disbelief, I ask Rodrigo if he has just come up with this theory on the spot. *No,* he says, *I first thought about this twenty years ago, in high school.*

A linguist might disagree with Rodrigo. Saying "ishmahchi quotish," might well be simply a matter of adjusting the pronunciation of an English phrase to the phonological context of the Portuguese sentence into which it was inserted. Such adjustment is quite normal for foreign words that are treated as already accepted into the host language (see chapter 6 in Grosjean, 1982). What is poignant about this episode is that the question that I ask is something that Rodrigo has thought about. His response shows the way he feels that use of English separates him from others and that he occasionally needs to find a way to be "one of the guys" by using their language.

The World Language or the Gringo Language?

I was at Nas Nuvens, and an email arrived from Rodrigo, sent to me, his friend

Renato, and the mailing list that includes all Nas Nuvens' programmers. It was a link to a blog post entitled "No mundo da Lua." The Portuguese title was a pun—it could be translated as "In the World of Lua" or "With Heads in the Clouds" (literally: "in the world of the Moon"). The blog post was in Portuguese, but was published on IBM's "Developer Works" website. The blog post lamented the fact that Lua was unknown in Brazil and that *Programming in Lua* was in English, German and Korean but not Portuguese. It then directed the readers to an article about Lua in *English* Wikipedia.

The incident left me curious about the relative sizes of Wikipedia articles about Lua in different languages. I spent some time looking at them, compiling a table.[143] I was not particularly surprised to see Portuguese below Korean and Spanish, while seeing my native Russian above Portuguese even gives me a bitter-sweet feeling of satisfaction. Rodrigo walked into the office just when I was getting the word count for the Esperanto version. I asked him what his bet would be: would the Esperanto article on Lua be longer or shorter than the Portuguese one (visually they appeared quite similar). Rodrigo bet on Portuguese, without too much excitement. The Portuguese article did turn out to be longer, though barely. I announced the result to Rodrigo. He looked at the Esperanto article in disbelief and seemingly a bit irritated. *I just don't get it,* he said. As it turned out, he was not irked by the fact that Portuguese nearly "lost" to an invented language, but by the fact that people waste their time on Esperanto. "Anyone who can speak Esperanto can also speak English," he explained. *If they can speak English, why do they bother with Esperanto?*

143 See appendix E, "Length of the Wikipedia Article on Lua in Different Languages."

Perhaps people want a neutral language to communicate in, I said. We headed out to get dinner, continuing our discussion on the way. I asked Rodrigo if he ever had a pen-friend. He mentioned "Matti," a Lua programmer in Finland who had been somewhat involved with Kepler and had become a personal friend. They were exchanging messages about all sorts of things, most not related to the project. I asked him if he thought his interaction with Matti would be different if he knew Finnish or if Matti was fluent in Portuguese. *Yes, it would be,* he says, *as it is in your case.* (Paradoxically, I noted, we were speaking English at the moment, but perhaps it was the fact that we *could* be speaking Portuguese that made a difference.[144]) I said something about the power imbalance created when one has no choice but to speak the language of the other. But Matti and I are equals, countered Rodrigo. They were equals in terms of our standing in the Lua world, he explained, and neither was a native English speaker. But isn't there always an invisible third person, the American, present, when you speak English, I asked. *Yes,* says Rodrigo, *it makes sense. I never thought about it this way, though.*

I speak English because it is practical, continued Rodrigo. *A while back I figured out that I could only learn one foreign language. English was the best option since that was the language spoken by most people.* "Why didn't you learn Chinese then?" I asked. True, he responded, there might be more people speaking Chinese, but Miami was closer. He added something about McDonald's. There you go, I said, it was not about the number

144 My conversations with Rodrigo alternated between Portuguese and English, without a clear pattern. According to Rodrigo, this was a source of much discussion by the non-English speakers in the office, who struggled to understand why we would not just pick one language or the other, or, rather, why we would not just stick with Portuguese. (Rodrigo's English was far more correct than my Portuguese, though I seemed to be able to sometimes speak Portuguese faster than he could speak English—perhaps indicative of the difference between learning a language in school vs. learning it mostly by speaking.)

of speakers, but about McDonald's, Miami and Disney World. Or, to put it different, it was about cultural dominance. If Buddhist temples were more important to you at the time than McDonald's, I concluded, maybe you would have tried to study Chinese after all.

Sure, agreed Rodrigo. *But English is not the same as the United States.* In fifty years, he continued, the US will no longer play a dominant role on the world stage, but English will still be the main language. It has nothing to do with the US and with its culture. The United States of Canada will be mostly Spanish-speaking anyway, and nobody is going to care what people speak in Jesusland, he said, referring to a joke I told him earlier.[145] English will thus no longer be seen as the language of the United States, just as a means of international communication.

Rodrigo was speaking in a somewhat humorous tone, and his words seemed to be carefully picked to express neither hope nor disappointment at the eventual demise of the United States that he was foretelling. He talked as an indifferent, if curious and somewhat amused, observer. (Rodrigo always picks his words carefully, mindful of the possibility of offending one party or another. He sees his mediation skills as a key asset. And he is also always half-joking, to a point where it becomes hard to know when he is serious.) The scenario he painted, however, reminded me of another conversation, from just a few days before. I was at the apartment of another Brazilian programmer, also a fluent speaker of English (with a Ph.D. from Canada), and the conversation touched on the United States.

145 A joke popular after the 2004 elections presented a map of North America with the "blue states" attached to Canada (labeled "the United States of Canada") and the "red states" shown as a new country labeled "Jesusland." I had just told Rodrigo about this joke a week before. He liked it a lot and has referred to it on many occasions.

"Do you know what people's reaction here was when they heard of September 11th?" he asked me. "They were quite excited (*animados*)." *People were thinking,* he continued, *that this was going to be the end of the US dominance.* His eyes light up. *It didn't quite work out this way, but still in the long term the US will fall.*

About a month later, I was walking to lunch with a group of Alta programmers. We are going to *Rio Sabores*, which we had nicknamed "I don't know" ("Não Sei"), a place we would go to when nobody had a preference and when the first person who asked to suggest a place would says "I don't know." Two of the guys were discussing Linux music players. They were comparing the more established XMMS to the newer RhythmBox and some other, "new thing" that a third developer (not present) is now running on his computer. As is common, they were talking about technology, without actually talking about work. (They hardly ever talk about work at lunch. That would be boring. Linux *music players*, on the other hand, are a valid topic—after all, they are not talking about Linux Java compilers.)

When we sat down to eat, the discussion turned to a different topic—measuring temperature in Fahrenheit degrees. *How stupid is that?* said one programmer. Others nodded in agreement. *The whole world uses the metric system, except for the United States,* he continued. *Why can't they act like a normal country?* Others nodded again. *And then we end up using the stupid American measures too,* jumped in another developer. *Like measuring monitor sizes in inches!* (Brazilians measure TV and monitor sizes in "polegadas," a term which apart from that only comes up in translated books.) *But one day the US will decay, and perhaps the idiotic measurement system will facilitate*

this.

"I hope," said another developer, Marcos, "to live long enough to see three institutions go down. The first one is the United States. The second one is Rede Globo, which won't take that long. The third one is Microsoft." Marcos then moved on to stronger imagery, talking about how each of those needs to be "destroyed." (The picture of destruction was painted most vividly for Rede Globo—the country's main news network which middle class Brazilians love to hate. The despisal expressed for the United States seemed bleak in comparison.) Others developers made supporting comments regarding all three. Nobody seemed to question that this was about the right list. The conversation did not linger on this topic for very long, however. After a few cliché (and seemingly pro-forma) curses towards Microsoft, the discussion moved on the activities of the Gates Foundation, then quickly to investment, and the rest of the lunch was spent talking about personal investment in stocks. Marcos talked about how he invests money in stocks while others listen with curiosity—this seemed to be something they had heard of before but had not done.

On the way back I asked Marcos what he meant when he said "the United States" and whether this included all of the US companies. "Ok," he said, "It was partly a joke." *Though*, he continued, *there are lots of reasons to wish for the United States to be destroyed. They approach the world from the position of the strong, they do what they want everywhere.* "But what about American companies?" I asked. After all, Alta's engineers spent their days working with technology made by the US companies? *Yes,* he said, *I am aware of this; there is a contradiction. And yes, I think about it some time. And*

it's true that much of this technology is available to us because the United States has made a really big investment into research. But this doesn't justify the way they treat other countries. And also, if the US were destroyed, some other country could take the role of the technological leader. "And act the same way?" I asked. *Maybe,* he agrees. *To some extent, the US behaves in accordance with human nature. It's in human nature to abuse power. But still... The billions and billions of dollars that they spent destroying Iraq could have been spent improving life in Africa.* "Would this all affect Brazil, though?" I asked. *Probably not*, he says. *Brazil is not in a position to be the next technological leader, nor is it Africa. So, no, this wouldn't affect Brazil.* After a pause he added: "But it makes a difference for our consciousness."[146]

Open expressions of hostility to the United States, such as those presented in the few pages above, are not something a US-affiliated researcher hears every day. What I did hear regularly, were the more mild references to *gringos*, typically accompanied with the lightest touch of resentment, and immediately retracted upon any interrogation. From my first days in Brazil, I was continuously surprised by the extent to which the Brazilians I interacted with closely (software developers or not) highlighted my *Russian* origin when presenting me to others, bringing up my connection to the United States only when wanting a joke at my expense. (The situation was a bit different in the more formal interactions, where being "from the United States" seemed more valued, though even there "from Berkeley" or "from California" seemed to be preferred.) While part of the preference for seeing me as Russian rather than American no doubt had to do with the

146 See Pait (2008) for a discussion of the reactions of São Paulo residents to the conflict between Israel and Lebanon.

curiosity towards a distant and nearly mythical country, it seems clear to me that such identification was also meant to allow me to not be seen as "a gringo."[147] Of course I did get this label applied to me quite often *in jest*, with an expectation that it will annoy me. And though at first it did not, after some time I internalized their values to a point where being called "gringo" actually started to irk me. (I also started introducing myself as being from Russia without waiting for my Brazilian friends to point this out.)

The resentment towards "gringos," however, is never expressed as resentment towards *English* among the software professionals that I interviewed. (Brazilians working on certain other fields do that sometimes.) Or, to be more precise, they never express resentment towards English in the context of software, though sometimes lamenting inappropriate love of English among other Brazilians broadly:

> Roberto: And there is this inferiority complex and everything, I think, that comes too.
>
> Yuri: What do you mean by that?
>
> Roberto: I mean the people always consider that if it's from outside it's better.
>
> Yuri: Why?
>
> Roberto: Why... I think this is something very [unclear] in the Brazilian culture. I mean, I have some theories, but I don't think I should explain them. Because the thing in Brazil... Always since the Portuguese court

147 The denotation of the term "gringo" is a bit vague in Brazil. One the one hand, the word can be applied to refer to any foreigners, sometimes even those from Latin America, or at least from the richer countries of the continent. (People from the poorer countries have a different set of derogatory terms reserved for them.) At the same time, there is a clear image of a prototypical "gringo"—a white American tourist in a Hawaiian shirt (presumably because he assumes Brazil to be just like Hawaii), largely ignorant of the world around him, though at the same time smug and empowered to decide which country will be bombed next. Other foreigners can be then be "gringos" to an extent that they share in the attributes of the focal gringo.

> came here[148] there were... first there were the Portuguese, then the English, then the French, then the Americans. There is always this kind of... I think one of the reasons, actually, is that rich people here are immigrants, and so they try to keep this kind of "I am different from other people here, I deserve to be in the upper part of Brazil because I am not really Brazilian." I think there is a lot of cultural things to connect. People here love to use English phrases. In the beginning of the nineteenth century everyone loved to speak French. In the beginning of the twentieth century it was English but from Britain. Now it's English from America. But it's always this important part of the world that we want to be a part of, to be connected with the important part of the world. We are part of the important part of the world. Just by chance we are living in this... this undeveloped country. I think this is very deep in Brazil. [...] Music is a much stronger example, because in music we have a... unlike software, we have very very strong music creation here. So to be number one in music here is really important. But still people give much much more importance to... [...]
>
> Yuri: You think it applies to music as well?
>
> Roberto: Yes! The way people love... If we were to be fair here, I think most people would never listen to foreign music here, because Brazilian music is so good. [...] But if you go to radio stations here, most radio stations here in Rio play only foreign music. [...] It's unbelievable that this [happens] in Brazil. Eighty percent of the stations should play Brazilian music! Ok, there is twenty percent for other stuff. If you go to disk stores here, sometimes they are organized here, like here is "Pop," "Rock," Jazz," "Brazilian Music" - it's a kind of a subsection. [...] Some people even put it in "World Music." [Laughs.] I already saw one or two stores. But it's not common, this is very strange. But I already saw it. But it's very common that you see all that American music in the front and then there is one section—this is "Brazilian Music," with all different kinds of Brazilian music. Classified like they do in the United States.

Roberto links speaking English to the more general adoration for foreign things, which may go as far as preference for American music. Alternatively, software professionals may make fun of those Brazilians who prefer overpriced imported whiskey to Brazilian *cachaça* or have *cachaça* substituted with cheap local vodka in their *caipirinhas*. This

148 The Portuguese court moved to Brazil in the early nineteenth century, fleeing first Napoleon and then the republican movement in Portugal, bringing to Brazil the Portuguese nobility which was largely absent there before.

love for all things foreign, however, should not be confused with *their* use of English, they point out. The latter is an entirely practical affair. And in the near future their use of English might have nothing to do with the United States anyway.

The situation with language here represents in many ways a more general issue that is common place in Brazil. As I show in consequent chapters, Brazilian software community in many ways lacks cohesion, the actors all too often feel "stuck" in the wrong place, with the wrong people. Managers say they can get nothing done in Brazil because of the incompetence of the employees. The programmers say you cannot get anywhere because of clueless managers. Both blame the government and the clients. The clients and the government find their reasons to be dissatisfied with both the workers and the managers of the IT firms. And everyone accuses others of being in love with foreign things, while arguing that their connections to the outside are simply a matter of realism.

It is also true, however, that individuals like Roberto and Rodrigo (as well as many other of my interviewees) are sufficiently worldly and global (at least in comparison with the rest of Brazilians) that they do not need to show their worldliness to others (or they do this in more subtle ways). They read and write English so often that they do not need to *speak* it in front of others to make a point. (In fact, as we saw above, their concern might be how to fit in the world of non-speakers.) They do not *love* English—they just use it. It is true that while Brazilians with less education occasionally use English words when talking to me seemingly to show me that they know them, PUC graduates typically do so only when there is no obvious Portuguese equivalent or when they feel I did not understand the Portuguese word they used. In other words, the use of language as a

marker of boundaries does not need to be overstated.

Use of English can thus be sometimes a pragmatic choice, a matter of reaching the largest audience, and at other times a way of establishing social status and flaunting connections with the larger world. It can also mark local connections (between the members of the educated elite or between engineers sharing in the jargon), or draw a distinction between those with and without education. Use of Portuguese can similarly mark connections or boundaries. As we will see later, much the same can be said about many other types of cultural codes.

Work as (Foreign) Practice

The episodes presented in this chapter illustrate many of the elements of work as cultural practice. They show that being a software developer is very much a matter of participating in a culture, rather than just performing certain tasks in exchange for money. (Though, the importance of finding local jobs comes up in a few of the episodes, and is explored in more details in later chapters.) While looking at how the developers learn English, we also touched upon the question of how they come to engage in the professional culture of software—a topic later explored further in chapter 2.1. We saw some of the ways in which the language is intertwined with the tools used by the developers. We also saw how the language and culture of software can structure local social relationships, connecting to the discussion of power in Lave & Wenger (1991). The theoretical discussion provided in the previous chapter thus helps us understand many of

their experiences. It leaves us unprepared, however, for the most startling aspect of those episodes: the fact that the practice of software comes in a package with a foreign language, and (as we will see more clearly in later chapters) with a foreign culture. We quite clearly cannot understand the experience of Brazilian software developers without considering the fact that they engage in the practice of software in a place quite far from where this practice is strongest.

In the next chapter I extend the theory of work as practice and introduce the concept of "worlds of practice" to capture the larger collective towards which the developers orient themselves—an orientation that they stress should not be confused with the simple liking of all things American. I explore the several types of links that hold such occupational worlds together, placing language, the topic of this chapter, among other cultural ties, and contrasting those with the material relations which are equally important to understanding such "worlds." I return to Alta, Lua and Kepler in part 3 of the dissertation (chapters 3.1, 3.2 and 3.4 respectively), looking at each as a different potential configuration of local and global commitments.

1.3 Practice in Space

The synthesis of Lave & Wenger's (1991) "communities of practice" with the Chicago School writing about work and practice presented in chapter 1.1 provides a good model for thinking of how a practice is reproduced *in time.*[149] This model, however, sheds little light on reproduction of practice in space. In chapter 1.2 we looked at the work of software developers in Rio de Janeiro, noting complex interactions that cannot be understood without considering the fact that those developers are engaged in a practice based far away from where they live and work. Furthermore, considering their peripherality relative to the practice is not only important for understanding how their work fits in the larger world of software development. Rather, we cannot understand what happens locally, in Rio de Janeiro, without considering this peripherality.

Before turning to the way peripherality structures local interactions, however, I ask a few broader questions about reproduction of practice across space. How does a practice spread in space, sometimes extending its membership to people who live thousands of miles away? How does software development, a practice once confined to the east coast of the United States,[150] come to be accepted as something sufficiently universal that a

149 Schatzki (1996) defines one of the notions of practice as "the temporary unfolding and spatially dispersed nexus of doings and sayings" (p. 89). In this sense, the notions of time and space are central to the idea of practice.

150 While Ada Lovelace (1815–1852), who lived in London and published the first description of Charles Babbage's Analytical Engine, is often credited with the title of "the first programmer," the lineage of the modern practice of software development would be better traced from the team of engineers who developed the ENIAC at the University of Pennsylvania starting in 1946 (Ceruzzi 1998/2003). The ENIAC developers later formed Eckert-Mauchly Computer Corporation (EMCC), a company that built the UNIVAC. EMCC was later acquired by Remington Rand, which was later acquired by Sperry, which later merged with Burroughs to form Unisys (ibid), still headquartered in Pennsylvania.

youth in Rio de Janeiro can aspire to be a software developer without making people laugh?[151] I then look at the local politics involved in such reproduction.

Approaching the question of how practice spreads across space from the perspective of a Chicago School reading of Lave & Wenger (1991), we might think of practice as spread and synchronized by itinerant practitioners. Indeed, the role of such people has been described by some authors. Saxenian (2006) talks about the "new argonauts" who build links between the silicon chip industries of California and Taiwan, or between the software industries of California and India. Similar travel of practitioners is described in some of the studies of physicists (Traweek 1988/1992, Knorr Cetina 1999). While looking at traveling practitioners is increasingly important for understanding the sharing of certain kinds of practice, such people are still a fraction of world's population and their travels tend to link rather specific location (for instance, San Francisco and Bangalore). No matter how close San Francisco may be to Bangalore, most cities in the world, even large ones like Rio de Janeiro, are far from both in the experience of most people who live and work there. At least in the case of the software developers working in Rio de Janeiro, it is clear that most of the work of keeping the local practice "up to date" is done by people who rarely (if ever) leave Brazil—the case of most people whom we

---

Burroughs main competitor was IBM, headquartered in New York, only a few hours away by car.

151 While it is hard to find an example of an occupation that would be altogether unthinkable in Brazil today, until recently Brazilians would be a lot more likely to laugh at a youth aspiring to become an astronaut. In 2006, however, Brazil got its first astronaut. The mission was mostly handled by the Russian aerospace agency, with the Brazilian space agency acting as a sponsor. The person selected for the mission (Marcos Pontes) was a Brazilian then pursuing a Ph.D. in the United States, who appears to have been mostly living in the United States since 1996.

met in chapter 1.2.[152,153] And while the traveling practitioners may well be crucial for the initial introduction of the practice in a new place, I will try to show that we cannot easily separate this initial step from the later synchronization. Reproduction of a practice involves setting up a system of relationships between people and objects. It cannot be achieved by simple arrival of individual practitioners.[154]

The concept of "networks of practice" (Brown & Duguid 2000, Duguid 2005) aims to extend the notion of "communities of practice" to groups of practitioners scattered in space, and allows for electronic communication between the practitioners. As Duguid (2005) points out, however, successful communication in such networks *assumes* prior commonality of practice.

> The network of practice (NoP) designates the collective of all practitioners of a particular practice. For example, Knorr Cetina's (1999) "epistemic culture" of high-energy physicists constitutes a global NoP that has within it multiple local CoPs [communities of practice]. Though practice is not coordinated within a NoP as it is in a CoP, common practices and common tools allow distant members to exchange global *know that* and to re-embed it (Giddens, 1990) in effective, coherent ways through the mediation of their locally acquired *knowing how*. Consequently, where practice precedes it, explicit knowledge may appear to have global reach (or to be "leaky"). Where it does not, the same knowledge may appear remarkably parochial (or to be "sticky").
>
> The central distinction between the CoP and the NoP turns on the control

152 While two of Lua's authors have spent several years abroad, all other people mentioned in chapter have only been outside Brazil for short periods time.

153 The situation of workers who are localized but controlled directly from a remote location, such as the Irish software developers described by Ó Riain (2000), presents another interesting and increasingly common (though still relatively rare) configuration. It is hard to tell in those cases what role such remote control play in re-creating the practice locally. The experience of Brazilian developers suggests that this direct control is not necessary for such re-creation.

154 For instance, Saxenian (2006) shows the multi-stage process through which the Silicon Valley model got applied in Taiwan, India and China, pointing out, among other things, the argonauts role in working together with local policy makers to build the necessary institutions.

> and coordination of the reproduction of a group and its practice. Newcomers enter the network through a local community. You become an economist by entering an economics department in Chicago, or Berkeley, or Columbia—a route that may mark you for life, in part because the tacit knowledge of the local community profoundly shapes your identity and its trajectory. (p. 113.)

In other words, where shared practice exists, the practitioners can share knowledge and further synchronize their practice through a range of means, including electronic communication. Someone living in Brazil who is experienced in the practice of software development as done in California (that is, fluent in the culture and language of that practice and comfortable with its tools) will have no difficulty understanding documents authored by software developers in California. (Though, they may still have some trouble in applying this understanding to the local context.) But how do they become experienced in that practice in the first place and why do they even want to engage in it? Duguid's answer (that they enter the practice though a local community) points us in the right direction. It assumes, however, the prior existence in different places of local communities that can be said to engage in the same practice. We need to understand how this is possible and what it even means for the "same" practice to be carried out in a new place, looking both at the way those communities achieve substantial similarities of practice and also at how the claims to participation in the same practice are asserted and judged. Duguid (2005) writes, for example, that one becomes an economist by entering a department at Chicago, Berkeley or Columbia. We may want to ask, however, whether one can become "an economist" in the same sense by entering Instituto de Economia at the Federal University of Rio de Janeiro or the Faculty of Economics of Eduardo Mondlane University in Mozambique. In other words, the question of how one becomes

an economist cannot be treated separately from the question of what counts as "being an economist" and who gets to do the counting.

Abstract Groups of Practitioners

To answer such questions we first of all must recognize the crucial analytic move introduced by Duguid (2005): the quote above drives a wedge between the notions of "practice" and "local community." To push this wedge further we need to look more carefully at the concepts of "groups" and "communities" invoked in the previous chapter, and the different levels of analysis implied in the Chicago school view, in particular distinguishing between *concrete communities* (typically tightly knit and local) from *abstract groups (or "categories")* of people united by similarity of practice—not forgetting the all-important question of who acts as the judge of similarity. To draw on Becker's (1963) example, we can look at the individual dance musicians as members in a concrete group, such as "the X— Avenue Boys" (whom Becker describes as "a clique of extreme jazzmen"), or in the more *abstract* groups such as "the dance musicians." Once this distinction is drawn clearly, we can examine the relationship between the two. How does being one of "the X— Avenue Boys" make one "a jazzman"? Conversely, how does being "a jazzman" allow entry into specific local cliques?

To illustrate, the software developers I interviewed in Brazil, often asked me if I was a software developer myself. When they did this, they were not inquiring whether I was a member of some local "occupational community" or "community of practice"—

they usually knew that I was not.[155] Instead, they wanted to know whether I belonged to a larger, more abstract category of people who write software code and with which they themselves associated. As a foreign member of this group, I was not expected to understand *local* meanings and norms. For instance, the developers took time to explain to me the many difficulties of doing this kind of work in Brazil. They also pointed out specific people in specific organizations that I should talk to—again, correctly assuming that I would not know by myself who the important people were. At the same time, they expected me to understand their technical jargon (when used in English), as well as certain values and practices. For example, having identified myself as a "former software developer," I was expected to not just know what a "source control system" is, but also the technical and the social implications of the statement that a particular company lacked one. (At the technical level I would need to imagine the likely outcomes, while at the social level I would be expected to form the appropriate opinion of the people who run the company.) In fact, I often needed to make special effort to make my Brazilian interviewees *suspend* the assumption that I share their meanings and opinions and to explain everything to me, *as if I was not one of them.*

This willingness to assume that I would understand their terms and practices is not a matter of wishful thinking. The practices of Brazilian software developers do replicate those of software developers in other countries to a remarkable degree. As I will show in more detail later, replicated practice involves not just solving the same problems, but

155 During the first round of my fieldwork I was not involved with any local projects and was clearly outside any local communities. During the later phase of participant observation, my relation to the local communities became more complicated.

solving them in the same way, relying on the same set of concepts, calling relevant objects by the same names (either in English or by Portuguese terms borrowed from English), and making many of the same jokes in the process. In an important sense, Brazilian and American software developers are members of a large collective of people, spread around the world, who write software code, and we must recognize the similarity of practice within this group must be recognized. At the same time, we cannot take for granted either this similarity as observed by the ethnographer, or the fact that Brazilian developers' claims to similarity of practice are accepted (usually) by local outsiders and remote colleagues.

So far in this section I have used the terms "abstract groups" and "categories" interchangeably when referring to larger classes of people engaged in the same practice. I do so to highlight the dual nature of such collectives: they are "groups" that exist objectively, identified by similarity of practice and communication, but they are also "categories" in the sense that they are *named* groupings that are recognized by social actors. This distinction corresponds to the one that Lamont & Molnár (2002) draw between "social boundaries" and "symbolic boundaries." Symbolic boundaries, are "conceptual distinctions made by social actors to categorize objects, people, practices, and even time and space" (p. 168). Such boundaries correspond to what I call "categories." Social boundaries are "objectified forms of social differences," which are "revealed in stable behavioral patterns of association, as manifested in connubiality and commensality" (p. 168). Social boundaries define what I call "groups." Lamont & Molnár

argue that symbolic boundaries are crucial for continued existence of social boundaries.[156] (This argument is similar to Giddens's theory of structuration discussed below.) Both types of boundaries are "real." Similarly, classes of practitioners are simultaneously "groups" and "categories." This duality is part of the reason why I avoid Brown & Duguid's (2000) term "networks of practice"—the concept of "networks" in sociology typically focuses on the patterns of interaction ignoring the symbolic boundaries perceived by social actors.[157]

Abstract groups of practitioners are therefore real not only in the sense that they represent actual similarity of practice, but also in the sense that successful claims to membership in such groups (or categories) have rather tangible effects on one's interaction with other members as well as with outsiders—perhaps even more so with outsiders who are less likely to recognize the local cliques. This effect is distinct from that of membership in concrete local group. In some situations, perhaps when interacting with the closer associates, what matters is membership in the small group. (Chapter 3.1, "Aplicações Corporativas," starts with a vignette that shows how membership in a small clique, "the Herculoids," opens for me the doors of "Alta.") In other cases, for instance when looking for a job (or simply presenting oneself) outside one's immediate social group, it may be more important to be "a policeman," "a jazz musician," "a fisherman" or "a developer."

---

156 "Only when symbolic boundaries are widely agreed upon can they take on a constraining character and pattern social interaction in important ways. Moreover, only then can they become social boundaries, i.e., translate, for instance, into identifiable patterns of social exclusion or class and racial segregation ..." (p. 168–169).

157 [Removed.]

While some of the different authors that we considered in chapter 1.1 stressed the more concrete groups and other stressed the more abstract ones, they rarely draw clearly the distinction between those two types of groups. Hughes's discussion seemingly focuses on abstract groups ("the doctors"). Becker's *Outsiders* (1963) also mostly deals in abstract terms ("marijuana users," "jazz musicians") and while concrete "cliques" are mentioned and sometimes described in some details ("the X— Avenue Boys") the relationship between the clique and the larger abstract group is left unexamined. The early discussion of communities of practice (Lave & Wenger 1991) similarly leaves the level of analysis unclear. The book often refers to the larger groups, operating at a regional or national level ("Yucatan midwifes," "Liberian tailors", "butchers"), yet the mechanisms that it proposes often index individual interactions and would be more appropriate if the group were to be understood as small and local. Legitimate peripheral participation goes a long way towards explaining how novice comes to practice tailoring in a particular small community of tailors, but it leaves us wondering how each of them becomes "a Liberian tailor." Orr (1996) focuses explicitly on the smaller local communities but mostly avoids the discussion the any larger groups that "the reps" are parts of.[158]

Van Maanen & Barley (1984) come closest to recognizing the need for clarity on this issue. Dissatisfied with the discussions in terms of abstract categories, they call upon researchers to look at the smaller (and typically local) groups. Their examples include the

158 Later literature on "knowledge management" which embraced the term "communities of practice" (in its Brown & Duguid 1991 reading, since regretted by Duguid—see Duguid 2008), have looked at "virtual communities of practice" (e.g. Hildreth et al., 1998, Hildreth & Kimble 2004). While the understanding of "communities of practice" in such literature differs substantially from Lave & Wenger (the differences stem from the instrumental view of knowledge embraced by knowledge management but rejected by Lave & Wenger), "virtual" communities of practice are also small, tightly-knit groups.

different subgroups within the "traditional fishermen" of Gloucester, Massachusetts (e.g., "Guineas" and "Greasers," each with its own occupational culture), "sweepers who live together in closed communities in Benares" (p. 303) and "cab drivers in Boston" (p. 311) —groups often defined by place. While this attention to links between occupational groups and place is important, Van Maanen & Barley do not carry it through consistently. Lacking a clear articulation of the relationship of such smaller groups to the larger categories of which they are instances ("fishermen," "cab drivers," "sweepers"), Van Maanen & Barley inevitably fall into traditional Hughesian discussion, using abstract groups like "police detectives" and "janitors" as examples of "occupational communities."[159] While the article at times seems to distinguish between "occupations" (presumably understood in Hughes's sense) and "occupational communities," this distinction is not discussed explicitly and is not employed consistently.

While mostly relying on local occupational communities in their examples, Van Maanen & Barley (1984) add an important caveat: their use of the term "community" does not presume that such groups are localized. Instead, they argue:

> Propinquity is... an attribute along which occupational communities vary. Certainly, propinquity may hasten and otherwise contribute to the development and maintenance of an occupational community, but it is not itself a definitional matter. Whether a particular community is geographically dispersed or clustered is an empirical question to be answered as communities are identified an analyzed (p. 298)

I see this observation as valuable and important, but not sufficient for resolving the

159 For example: "Within some occupational communities, centrality may be attached to working in particular settings. Gold (1964) notes that janitors gain recognition from peers by becoming custodians in upper middle class apartment buildings..." (p. 326).

problem of expansion across space in general. Van Maanen & Barley's first definition of the occupational community seems to assume that the members actually know each other. They later clarify this when they contrast occupational communities to "external reference groups":

> Although external reference groups may exist for members of some occupations such as tradesmen, academics, or industrial scientists, people in many lines of work do not know people who do denotationally similar work in other settings. Police officers, teachers and fishermen know there are other police officers, teachers, and fishermen in other work settings, but they may not know them or interact with them on more than a sporadic or episodic basis. In many lines of work there are no annual meetings to attend, trade journals to read, or frequent opportunities available to meet colleagues outside the workspace who are not also members of one's employing organization. (p. 341.)

"Police officers" thus do not constitute an "occupational community," but rather a group defined by "denotative similarity" of work or "an extended work group" (a concept not developed further). Assuming, then that "occupational communities" are understood as small groups where the members personally know each other, Van Maanen & Barley are correct in observing that such communities *may be scattered geographically*. American "social theorists of... symbolic interactionist... bent" (p. 296) may be a good example of such a community, scattered by virtue of the fact that few cities have demand for more than a handful of social theorists. Unlike the policemen, they *do* reconvene for annual meetings, where the members have a chance to meet each other. Such communities, however, are an exception rather than the rule, and allowing for the possibility of the geographically scattered occupational communities does not help us understand how occupational communities (typically localized) relate to their "external reference groups."

The fact that American social theorists keep their community alive through annual meetings sheds little light on the relations between police work in Berkeley and New York or between software development in San Francisco Bay Area and Rio de Janeiro.

As we consider the distinction between the local groups based on mutual acquaintance and the abstract groups based on denotative similarity of work, we may find that many attributes that Van Maanen & Barley attach to occupational communities are actually linked to those "extended" groups. For example, Van Maanen & Barley point out that member of occupational communities share specialized jargon and give an example from police talk: "We apprehended that dirtbag on a stand-up just next to my duck pond on 3rd and Main" (p. 299). While this phrase may be meaningless to true outsiders, it would be readily understandable for policemen in much of the United States, not just in "Union City," where John Van Maanen presumably heard it. For example, we can see the term "duck pond" explained in an editorial on a website called "officer.com": "Depending on what part of the country you're in, the local cops will have a 'duck pond,' 'cherry patch,' or 'cash register' to visit when things get slow on the street" (Dees 2006). The editorial further talks about the so-called *ten codes* (for instance: "What am I supposed to say when I'm 10-8?"—meaning "in service"). Commenting on Virginia's recent attempt to ban ten codes (Sheridan 2006), Dees, who served as an officer in a different state, writes: "Radio codes are part of the jargon that cops use to set themselves apart from common citizens." Dees and Sheridan both note that ten codes vary from one place to another:

> To Arlington police, "10-13" means "officer in trouble." To Montgomery

> County police, the same code means "request wrecker." Even everyday police commands can get lost in translation: In Alexandria, "10-54" refers to an alcohol sensor. For Virginia State Police, it's livestock on the highway. (Sheridan 2006.)

At the same time, however, the *use* of ten codes seems to be associated with the larger group of policemen—something that "cops" do. The efforts to get rid of ten codes is of course an attempt to unify police practice, making it easier for policemen who enter the practice through different departments to work together.[160]

Van Maanen & Barley argue in several places (for instance p. 295 and 341) that occupational communities must be defined "phenomenologically," by finding out, through ethnographic work, where the members themselves draw the boundaries. This approach brings obvious benefits, but is problematic for two reasons. First, it ignores the fact that the members may draw different boundaries in different situations. While policemen may draw clear boundaries between specific departments or cliques in some cases, they are likely to identify as just "cops" in other situations. (See the quote from Dees above.) In other words, we cannot assume that member-recognized boundaries just exist—we need to look at boundary-drawing as an active process. Second, to the extent that some Van Maanen & Barley's observations apply to the larger, abstract groups, we must remember that boundaries recognized by outsiders matter, sometimes more then

160 To give Van Maanen & Barley due credit, police work may have changed in many ways since the time Van Maanen did his fieldwork. Among other things, a complete outsider to the police practice (such as myself) probably would not be able to learn the meaning of the term "duck pond" in less than five minutes of web search in 1970s or 1980s. People who practice police work probably also had a lot less opportunities to learn about police practice in other cities. Today, American policeman can read "Officer Blog" (http://officerplod.blogspot.com/), or "The Policeman's Blog" (http://coppersblog.blogspot.com/) if they prefer to learn about police work in the United Kingdom. The latter site will also inform them about the practices of the Russian police. American policeman who happens to read Russian can then learn a lot more from a Russian blog dedicated to police work in Russia (http://se-tr.livejournal.com/).

those recognized by the members. A police officer would not be able to carry out many of his activities as a policeman unless he or she is recognized as such by non-policemen. The same is true (to a lesser extent) of software developers. Importantly, this is a two way relationship. The group often has a substantial power over who the public recognize as members (e.g., police officers prove their membership by showing their badges). At the same time, the outsiders' perceptions of group boundaries most certainly affect the group's own definitions of who is a member.

To understand the reproduction of a practice such as software development we must look at the groups of people defined by participation in this practice, recognizing those groups as delineated simultaneously by what "social" boundaries ("objectified forms of social differences") and "symbolic" boundaries ("conceptual distinctions made by social actors") (Lamont & Molnár 2002, p. 168). We must also look at the symbolic boundaries as contextual and negotiated. I will now elaborate this idea using Giddens's theory of structuration (1979).

Structuration Across Space

To analyze the effects of abstract categories we need a theory of how they relate to practice on the ground. To develop such a theory I draw on Giddens's theory of structuration (1979), extending it to deal with the issue of space. Following his notion of "duality of structure" and the stress on "essential recursiveness of social life" (p. 5), I see the abstract categories as descriptive of structural properties of a relatively stable (but

often evolving) system of social interactions. Such categories are simultaneously *constituted* by what people do and are *structuring* their actions. The structuring effect of the categories is possible because they provide *resources* that actors can use to influence each other's actions. To give the simplest example, "police officers" have the power to stop cars because the power of "police officers" to stop cars is *recognized* by most people. The fact that the concept of "police officers" is understood and recognized by the outsiders allows the members to generally go about their job without resorting to violence, reserving the use of force only for those *isolated* individuals who do not accept their authority. When violence *is* applied, its use is aided by the fact that those to whom it is applied cannot count on support from others.

Software developers in Rio de Janeiro *in a certain sense* do the same work as software developers in San Jose. This similarity makes it possible for us *and for them* to talk about "software developers" as a group and about "software development" as something that they do. Once the ideas of "software developers" and "software development" become discursively available to both members and non-members, some people may seek services of "software developers" and other people may claim to *be* software developers. If someone's claim to be a software developer is accepted by the potential clients, then this individual can derive income from "software development." They can then engage in software development activities full time and gain access to better tools and more central projects. This would further strengthen their claims to membership in the category. Those who fail to convince clients that they can do "software development," would have to either write software as just a hobby or give it up altogether.

In a similar way, the individual needs to be accepted as "a software developer" by the local community of practitioners in order to draw on it for support. (To be more precise, they might need to be accepted as "a hacker" or "a coder," since the members might use different labels from the outsiders.)

On the other hand, the need to continuously assert membership in the abstract category creates reasons for keeping the practice synchronized with that of people whose identity as "software developers" is unquestioned—for example, the people who write software at Google. Synchronizing the practice in this case means not only synchronizing the "technique" that is applied to the relevant objects, but also displaying fluency in a system of cultural codes that would presumably make one acceptable as "a software developer" by those members. In other words, to pass as a software developer locally you need to show that you can act in such a way as to make it believable that *if* you were to go and talk to software developers at Google, *then* they would accept you as one of them. (Yet the software developers at Google are only *hypothetical* judges of membership.) Local representations of how the practice is carried out remotely thus become powerful structuring (and synchronizing) resources, as they give local members ability to censure lack of compliance with the "standard" practice. (As we will see later, the ability to *improve upon the standard practice* through innovation is in some sense the ultimate test of "central" membership in the category. However, the local innovator must be ready to defend their practice as an innovation with global significance—something that can realistically be adopted by the other practitioners and become a part of "standard" practice later—rather than a local divergence born from insufficient mastery of the

standard. This sets a very high bar for innovation.)

Internal and External View of the Categories

There are two ways to understand a category such as “software developers”, both of which can be applied recursively in Giddensian way. We can loosely call them the “internal” view and the “external” view. Both are important and I will look at the relationship between the two of them after first considering each of them individually.

Hughes’s (1958) suggestion that an occupation can only be understood when considering its place in the societal system of division of labor represents the “outsider” view of the occupations. This view is also acknowledged (but not elaborated) in Lave & Wenger (1991), who write that communities of practice must be understood in relation to other practices. The practice of policing traffic can be understood as the activities that are undertaken to prevent certain other practices such as speeding or running red lights. In addition to being inseparable from the practices that it aims to prevent (there would be no point to traffic policing in the absence of speeding and other forms of traffic violation), the practice of policing traffic is just one of several practices that share the work of curbing traffic violations. Traffic police officers can only fine drivers for running red lights if the red lights are installed and working, yet installation and maintenance of the red lights is not something they do. The work of traffic police can thus be understood as a *role* in the larger system of regulating traffic. This then gives us a clue as to how we could identify “traffic policemen” in a new context: if we follow a car in Rio de Janeiro and see

it stopped after running a red light, we would have good reasons to think that the person who stopped it is a traffic police officer. As this example shows, it is important to look at a range of relationships that define the practice, and not narrowly at the collaborative division of labor—the practices of running red lights or hijacking cars are as important for defining the work of traffic police as the practice of installing the lights. It also points out that we must consider the practice in relation to material objects recognizable to outsiders—such as the red lights and the vehicles.

An *internal* view of the categories involves instead looking at the similarities between the members of the group that the members themselves recognize as important or even defining and of which the outsiders may be mostly unaware. American police officers speak in ten codes (see above). Software developers world wide write instructions for computers using "text editors" with names that would be unrecognizable to most outsiders ("emacs," "vi," "Eclipse") and they may likely question any claims to being "a software developer" coming from someone who does not use a "proper" text editor to write code. To use Hughes's term, the members can recognize each other by the *technique* that they use. They also recognize each other by what Hughes and Becker call "culture." This culture consists of both particular ways of seeing and labeling objects, but also of particular fixed ways of talking about them—what I call "lore," a collection of stories that circulate between the members and illustrate certain elements of collective wisdom. They may pay attention to how a particular member came to engage in the practice, considering both where they entered it (see the earlier quote from Duguid 2005), and for what reason (see chapter 2.1, "Nerds"). This identification, however, is not

necessarily a matter of conscious evaluation. Brazilian software developers sometimes say that ultimately one can recognize a true developer because "you can see it in their eyes."

The internal and external views of the category do not always correspond cleanly to insiders' and outsiders' definitions. Outsiders who are looking for members of the category to perform services for them may be keen on finding people who are members of the category according to other members. (After all, *anyone* that they hire will end up satisfying the external definition by virtue of being hired to do the job.) Similarly, insiders may consider the members role in the external system of practices when evaluating their claims to membership.

This dual definition of membership demands a theory of a relation between the internal and the external understanding of the category. Becker's (1963) suggestion that culture arises from shared problems may at first glance seem to suggest the external definition as primary. Under this interpretation, we can see membership in a category like "software developers" as a matter of taking a defined role in the system of division of labor. Cultural affiliation can be then seen as simply arising from the fact that those who take this role face similar problems, both in their relations to the objects with which they work and in their relations to other practices. Such an approach, however, would see the practitioners as somehow thrown into the role all of a sudden, only then to work out some kind of shared culture. This interpretation would be contrary to Becker's theories of learning (e.g., 1953) as well as the theory of Legitimate Peripheral Participation in Lave & Wenger (1991).

Alternatively, we can see the categories as referring to groups of practitioners tied by shared technique and culture. Members of such "named groups" (by analogy with Hughes's "named occupations") collectively *create* for themselves roles in the social division of labor. Over time, such groups can establish for themselves *new roles*, which in retrospect may have little to do with the role that they played originally, and the community may eventually split. For example, electronic engineers (predominantly men) gradually took upon themselves the task of programming computers—the work that was originally assigned to human "computers"—a "subprofessional" category of people performing calculations, composed predominantly of women. Considering the fact that the first general-purpose electronic computer (ENIAC) was programmed by a group consisting *entirely* of women (Fritz 1996, Koss 2003), it may seem surprising that software development is today not only *de facto* dominated by men in almost all countries (Galpin 2003), but is also constructed internally in highly gendered terms, leaving little space for women. We must consider, however, that this change did not occur by individual men entering the communities of human "computers" one by one, gradually masculinizing the culture of human "computers." Rather, electronics engineers, already a cohesive and highly gendered group, recognized programming work as promising and laid a successful claim to it, pushing out the female "computers."[161] This latter view, however, cannot explain how people in an altogether different location come to enter the ranks of a category such as "software developers."

161 Of course, individual women continued to play a role in software development, but could only do this by becoming members in the male community of software professionals. Grace Hopper, "the mother of COBOL" and arguably one of the most important people in the history of computing, joined the US Navy at the outbreak of the World War II, later rising to the rank of Rear Admiral, and earning in 1969 the "Man of the Year" award from the Data Processing Management Association.

Definitions as Tools

We can reconcile the two views if we look at the external and internal definitions of membership not as naturally constitutive of the groups, but rather *as discursive tools* used by members and non-members to negotiate rights to engage in certain practices. Fulfilling the role *de facto* is one test that individuals can try to *use*. The ability to demonstrate similarity to other members is another test. When those tests yield different results, the individuals may engage in negotiation as to which one is more relevant in a given context. Members of the group can also engage in "boundary work" (Gieryn 1983), trying to educate the general public as to what categories should be considered important in specific circumstances.

It will help to look in more detail at the different types of "moves" that can be performed using such definitions as resources.[162] I illustrate those moves with examples from my own research, which I expand in the later chapters. For each of the moves I consider the potential sources of resistance.

1. Homesteading a new role. Members of an existing "named group" of

---

162 My notion of "moves" is similar to the discussion of jurisdiction-shifting "moves" provided by Abbott (1988), but differs from Abbott's in several ways. Some of the differences stem from the fact that Abbott looks narrowly on "professions"—a class of Hughesian occupations that lacks clear boundaries, in part because Abbott struggles to bridge the functionalist accounts with Hughes's emphasis on work, looking to broaden the concept of "profession" without making it entirely vacuous. In my later discussion of software development I draw on specific concepts from Abbott, finding, though, that taken as a package his concept of "professions" does not fit the work of software developers. (In particular, software development lacks the formal institutions that defines professions and formal credentials play a much smaller roles then they do in most of the cases that Abbott considers.) Additionally, while aiming to distinguish his approach from Parsonian functionalism, Abbott ultimately privileges systemic explanations over all others, and looks at the "moves" as something that *professions* perform in the public arena. Contrary to this, I look at the moves as something that *individual members* perform in myriad private negotiations, looking at groups as resources that such individuals use to back up their claims.

practitioners may decide to engage in a new line of work, seeing this work as fit for their technique and culture, and using this fit to justify their claims to this work. They may originally engage in this work *as members of their group of origin*. For instance, electronic engineers may argue that *as electronic engineers* and makers of things like computers they are uniquely qualified to program computers. Their claims may compete with those of other groups, who may have their own arguments for why *their* technique is more fit for the job. If one group's claims are accepted by those who control access to computers, the members of the group will be given a chance to work with the computers. Overtime, such people and the communities they form may develop specialized technique and an independent identity from electronic engineers.

2. Individuals joining the group by taking a replicated role. A role may be replicated in a new place as a part of a larger replication project (in which a system of roles is replicated), creating a need for local members of the abstract group. People who take those jobs will the become members in the category by virtue of taking on the role, despite likely lacking in both technique and the culture that other members of the category share. For instance, when the Brazilian statisticians purchased an American UNIVAC in 1960 in order to replicate American census-taking practices, they created a role for people who would program it. Some individuals were selected to take this role and had to then learn the technique and culture of American UNIVAC programmers.

Such learning presents a formidable task for the new practitioners who may be culturally isolated from their remote colleagues. Physical travel may help somewhat. (The first group of Brazilian UNIVAC programmers completed training in the United States,

and American programmers were brought to Brazil to work with them side by side.) Reliance on shared background (e.g., common knowledge of standard mathematics) helps too. Finally, access to *tools* and *objects* is crucial: the presence of the UNIVAC itself (the same machine as used in the American census earlier) gave the Brazilian programmers both a claim to membership and the ability to face many of the same problems as their remote colleagues, establishing some common ground. Such resources turned out to be hardly enough, however. As I show later, many things went wrong with the UNIVAC in Brazil. While some of the problems may be attributed to the lack of mastery of the technique, it is important to consider perhaps the hardest hurdle faced by the Brazilians tasked with taming the machine: the need to take a role in a *partially* reproduced system of division of labor. For example, while having a "real" computer, the programmers did not have access to spare parts and had to use Brazilian-made punch cards. Perhaps more importantly, they had to mediate between the American machine and the Brazilian census organization. As a result, the census of 1960 did not get tabulated until 1975.

3. Local individuals building ties with the group, then carving out a role locally. Around the same time, other individuals in Brazil claimed membership in a related occupation (electronic engineers) by first developing ties to remote colleagues. They did so at the time when there was no role for "electronic engineers" in the Brazilian division of labor. They traveled abroad to learn the technique and culture; some of them worked in research labs in the United States and got doctoral degrees in American universities. When they came back to Brazil, they worked to *transform* the local and global systems of division of labor in such a way as to *create* a role for "electronic

engineers." They did so by convincing other actors that a different system of division of labor was possible because Brazil had people capable of building computers, namely themselves. Doing so of course also meant overcoming many sources of resistance. The clients, such as the census bureau and the Navy, understood quite well that a handful "electronic engineers" would not be able to build computers by themselves. The engineers thus had to argue that the requisite system of economic relationships could be constructed. A number of them consequently had to put aside engineering and become policy administrators, working to create the necessary alliances between customers, investors, entrepreneurs and the state (Evans 1995). As I discuss below, such alliances worked for two decades, but fell apart eventually, largely closing the space for electronic engineers in Brazil. (See chapter 2.2.)

It may be instructive to compare this "move" as it happened in Brazil with the process described by Saxenian (1999, 2005, 2006). The Brazilian engineers who went to study in the US are similar to Taiwanese and Indian "argonauts" in that they went abroad to the places where the practice is strong, learned the practice there and then returned to apply at the periphery what they learned at the center, having to configure local resources so at to make it possible. Compared to the "argonauts," however, most of them spent relatively little time in the US, typically returning soon after completing their degrees. They thus likely had more limited understanding of the practice they were trying to replicate, understanding the technology without as much insight into the social workings of American industry. Perhaps more importantly, their ties to the American industry were weaker, and they largely detached themselves from it upon returning to Brazil. Their

approach was also different from that of the Taiwanese and Indian "argonauts" in that they focused on replicating in Brazil a miniature version of the American practice of computer design, using knowledge they brought from the US and additional knowledge that they hoped to *purchase* later. The strategy put less emphasis on participation in the *same* system of activities as their American counterparts.

4. Shifting to a different role when the current role disappears. Having created local space for engagement in a practice, the members can continue doing so for a while, but their ability depends on the continued existence of the role that they created, which in turn depends on *other* practices. If the role disappears, they will have to shift to another one that may be suitable for their technique and culture. In some cases, this new role may itself be an outcome of the same change that led to the disappearance of the earlier one.

For example, in the early 1990s Brazilian government opened its market to imported computers, which led to quick disappearance of Brazilian computer manufacturers, closing the role for Brazilian electronic engineers. This also dramatically expanded, however, the need for software developers who could help local organizations build custom applications to be run on the now available cheap imported computers. Most electronic engineers consequently shifted to software development.

Such a shift may involve many of the difficulties of Move 2. In Brazil, however, it was substantially easier than the ones described above. The foreign system of practices had been replicated more thoroughly by the 1990s. The clients' suspicions as to whether the local engineers were qualified as "software developers" were also limited by the

relative lack of alternatives: while the clients could buy generic computers from abroad and license or pirate some of the generic software, they often needed custom software adjusted for their specific needs, and such work had to be done locally. Bringing American software developers to Brazil was hardly a possibility, especially considering that their salaries were going up by the day. The clients had little choice but to hire local contenders.

As Brazilian software developers soon realized, however, there was a limit to what they could do. While the role for people who would write custom software for local organizations was quickly established, developers who tried to take upon themselves the role of writing generic software for sale or many types of generic services on the web, discovered that foreign competition was getting fierce as American firms were discovering localization, while many certain key inputs were missing, the most important of them being access to venture capital. Sales of generic software and many types of web services require a large up-front investment in development and marketing, which in many cases is never repaid. They thus depend crucially on access to organizations willing to invest in highly risky ventures. Venture capitalists often also provide strategy advice to the companies they fund. (This is called “smart money” in the parlance of California startups.) Software developers in California often start their companies in Palo Alto in order to be biking distance rather than driving distance away from Sand Hill Road, where many venture capital firms have their offices. Many Brazilian software-developers-turned-entrepreneurs found themselves a bit too far from Sand Hill Road and lacking comparable resources locally. (See the story of Nas Nuvens in chapter 3.4.)

5. Ongoing synchronization. Having established themselves as members (or tentative members) in the category, the local practitioners may have good reasons to maintain and expand ties to their remote colleagues. Such links may help them solve practical problems faced in the course of their activities, and they can often enhance their status as "software developers," both individually and collectively vis-à-vis the abstract group. Such synchronization is particularly important if the foreign practice is seen as continuously evolving and the practitioners are perceived by their clients as being at the risk of falling behind. In this case, their membership in the practice is always tentative, and they must work to stay up-to-date with the technique and culture of remote members. In addition to synchronizing the technique and culture, they may also try to synchronize their *role*, working to further transform the local division of labor in such a way as to ensure that they have access to the same inputs as their remote colleagues and get credit for the same outputs. Some aspects of such ongoing synchronization appear relatively easy. Brazilian developers who have learned to read English can have access to uncountable documents that describe the foreign software technique in minute detail. Such documents also often teach the culture of the foreign practitioners, as they both express their views of the relevant objects and present specific ways of talking about the world. The local developers can also have access to some of the same computer hardware. The combination of such documents, the developers' experience with the machines, and the fact that their clients are themselves interested in keeping up with *their* counterparts abroad make it possible for the developers to keep the basic aspects of their practice up to date with that of their foreign counterparts, despite the occasional *faux pas* such as

mangled pronunciation of English terms. Moving towards more central kinds of practice, such as developing software *platforms* rather then custom applications presents clear challenges. (See chapters 3.2, 3.3 and 3.4.)

As these examples show, the relationship between fulfilling a role, identifying with a group and acquiring a technique can go in different directions. Rather than logically entailing one another, roles, identity and technique each provide resources that may enable the actors to make certain moves, which in turn give them access to new resources. *De facto* performance of the role may provide a key to acquiring a technique and membership, while membership in the group is often crucial for entering the role. Each move requires active work and overcoming of the different sources of resistance. Those who gain access to the role may have to overcome their lack of mastery in the technique. Those who managed to pick up certain aspects of the technique may have to then overcome the challenge of finding a local role. Strengthening membership in the group may involve gradually using different resources in turn to strengthen the other.

The first four of the five "moves" presented above are crucial for understanding why software development, in the American sense of the term, is possible in Rio de Janeiro today—and more generally how a practice can enter new places. I explore the history of those moves in chapter 2.2. The last "move" (the ongoing synchronization of practice) is crucial for understanding what Rio software developers do today—and more generally how practice proceeds after establishing a foothold in a new place. The different aspects of such "synchronization work" are explored in the other chapters.

Steps 2, 3, 4 and 5 above all involve use of reflexive representations of a social

system to structure social interactions. Such use, however, differs from the traditional Giddensian structuration, which assumes that the actors use reflexive representations of *their* society when structuring their interactions. Instead, the actors here use reflexive representation of a *foreign* social system to transform local interactions.[163] While in the traditional structuration members' mutual knowledge that things are done in a certain way normally becomes a resource for reproducing the same pattern of interaction, in his case, the members' knowledge that things are done in a certain way *elsewhere* (e.g., "at Google") creates a resource of *changing* how things are done.

The importance of local practitioners' understanding of foreign practice and social structure for local structuration makes it imperative that we look at the ways such understanding is achieved, considering among other things the specific proactive uses of communication technologies to learn more about what is happening outside. We must also be mindful of the potential gaps between members' models and reality, and consider the ways in which the members may identify and mend such gaps.

Imagination

It is important to consider that while local members may know a good deal about foreign practices and social structure, they do not necessarily always draw on them as a

163 This type of structuration is discussed by Meyer et al. (1997), who look at the ways in which nation-states reproduce foreign models and argue that "the dependence of the modern nation-state on exogenous models, coupled with the fact that these models are organized as cultural principles and visions not strongly anchored in local circumstances, generates expansive structuration at the nation-state and organizational levels" (p. 156). (In my reading, the term "expansive" is used here to describe the resulting expansion of the state, rather than the expansion of the model.)

resource. The argument that a California company does things in a particular way may carry a lot of weight in some situations, but may be dismissed as irrelevant in others. This has much to do with local members' understanding of the ways in which the foreign context differs from their own, which in turn depends on their model of the world as a whole and their place in it. For example, their view of themselves as living in "a developing country" becomes crucial in negotiations of when the foreign model applies.

Appadurai's (1996) notions of "imagination" and "the imaginary" ("a constructed landscape of collective aspirations," p. 31) help us understand how reflected understanding of foreign structure is used in structuring local action. Individuals' actions are constrained by the space of outcomes that they can imagine individually and collectively. (Individual imagination may be enough for individual action, but collective action requires *collective* imagination.) Reflected foreign practices and structure provide elements for construction of imaginary worlds, which may potentially become blueprints for action. Such imagined worlds may be different from reflected foreign practice: reflection concerns with how the world *is* (locally and abroad), imagination concerns what is *possible*. As Appadurai argues, collective imagination can become fuel for action. Such collective imagination is often a *sine qua non* of collective action.[164]

I believe we must also recognize, however, the ways in which imagination may be inhibitive of action. A Brazilian proverb says that a dog once bit by a snake becomes afraid of a sausage. When looking at Brazilian software development today we must

164 A closely related argument is made by Adler in *The Power of Ideology* (1987), which stresses the importance of ideology in organizing collective action while looking at Brazil's and Argentina's attempts to develop computer and nuclear industry.

consider the ways in which one such past snake accident (the "failed" attempt to build a sustainable computer industry in 1970s and 1980s, discussed in chapter 2.2) affects today's actions.

Appadurai (1996) draws a distinction between "imagination" and "fantasy," pointing out that "the idea of fantasy carries with it the inescapable connotation of thought divorced from projects and actions, and... has a private, even individualistic sound" and contrasting it with the *projective* and *collective* aspects of "imagination" (p. 7). I argue for a more subtle treatment of the boundary between the two and for recognizing the importance of "subvocal" imagination: imagined worlds that are too unlikely to be publicly presented as a plan for action or as an explanation for actions taken in the past, but which none-the-less influence action in profound ways. When discussing the belief in witchcraft in France, Favret-Saada (1980) uses the phrase "I know, but still..." to refer to the gap between what the actors "know" to be true (in the sense of being ready to defend their beliefs as rational) and what they might nonetheless act upon. Such "subvocally" imagined futures are often presented as a joke to make it easy to retract the idea without losing face. (At the same time, such imagined worlds may be discussed at length as long as the label of "fantasy" is kept on them.) This presents the ethnographer (and presumably the members themselves) with a challenge of differentiating collective fantasies used for sublimation from those that represent unspoken plans for action. (See chapters 3.2 and 3.4.)

Centers and Periphery in Worlds of Practice

Our discussion of how a practice is replicated across space have assumed so far that we can identify places that serve as a *source* of practice and those that serve as its *destination*. Such distinction may not always be meaningful: we may find cases where practitioners in several places aim to synchronize each others practice in such a ways that we cannot easily identify the predominant direction of replication. In many cases, however, we can understand a practice geographically in terms of "center" (or "centers") and "periphery." In particular, the practice of software development has a clear set of centers with the most prominent one around San Francisco Bay Area (see chapter 2.2).

Before discussing the notion of peripherality, however, we need to look more broadly at the internal structure of the "abstract" groups of practitioners. To do so, I borrow certain elements from the literature on "social worlds." The idea of "social worlds" is associated most strongly with Anselm Strauss, and in particular his 1978 article "A Social World Perspective." Strauss himself offers much of the credit to Shibutani (1955) and we can understand the concept better by looking at Shibutani's discussion first. Shibutani writes that

> [...] even in common parlance there is an intuitive recognition of the diversity of perspectives, and we speak meaningfully of people *living in different social worlds*—the academic world, the world of children, the world of fashion.
>
> Modern mass societies, indeed are made up of a bewildering variety of social worlds. Each is *an organized outlook*, built up by people in their interaction with one another; hence each communication channel gives rise to a separate world. Probably the greatest sense of identification is to be found in the various communal structures—the underworld, ethnic

> minorities, the social elite. Such communities are frequently spatially segregated, which isolates them further from the outer world, while the "grapevine" and foreign-language presses provide international contacts. Another common type of social world consists of the associational structures—the world of medicine, of organized labor, of the theater, of café society. [...] Each of those worlds is a unity of order, *a universe of regulated mutual response*. Each is an area in which there is some structure which permits *reasonable anticipation of the behavior of others,* hence, an area in which one can act with a sense of security and confidence. Each social world, then, is *a culture area, the boundaries of which are set neither by territory nor by formal group membership but by the limits of effective communication.* (p. 566, my emphasis)

A few things should be noted about Shibutani's notion of "social worlds." First, it has clear resemblance to the discussion of "culture," "groups" and "outsiders" by Howard Becker (writing around the same time), but relies on a stronger and dangerously seductive metaphor of "worlds." The term connotes—and Shibutani seems to largely embrace its vernacular connotations—substantial differences in perspective (or "outlook") between those who are part of the world and those who are not. While looking at differences in outlook is often very important (and as our earlier discussion suggested), we will have to be careful to not let the metaphor of "worlds" lead us to thinking that of such difference as defining of practice, at the expense of other factors, such as the material conditions of the practice. Second, Shibutani's concept of "social worlds" is quite broad and can be applied to virtually any group. Furthermore, any two people, no matter how similar, can be said to be living in two different "worlds" relative to some division (p. 567). If they do the same work, then perhaps one lives in the "world of women" and another "in the world of men" or, perhaps one lives in the "world" of "Generation X," while another one is in "world" of "Generation Y." Finally, Shibutani suggests that the reach of the social worlds is limited by effective communication. While communication is clearly important for

establishing a shared outlook, we must never forget that it is not sufficient: one cannot gain the perspective of software developers (thus entering their "world") just by talking to software developers or reading what they write. One must actually *write software*, and doing so requires access to more than communication.

Strauss (1978) extends Shibutani's concept of "social worlds" in a way that makes it much more relevant to our discussion. First, without redefining the term explicitly, Strauss focuses his discussion on a particular type of social worlds, considering "opera, baseball, surfing, stamp collecting, country music, homosexuality, politics, medicine, law, mathematics, science, Catholicism" (p. 121). Those examples are still quite diverse, but they are constrained compared to Shibutani's examples (which include "the world of children" and "the social elite"). Strauss's main focus is quite clearly on the worlds that are similar to the abstract groups of practitioners that I have discussed earlier. Strauss then introduces three ideas that further constrain the meaning of "social worlds" to groups of practitioners: he suggests that each world has specific *activities* (one or more of which can be considered *primary*), that the activities occur in specific *sites,* and are aided by specific *technologies* and *organizations*. Note that it would be hard to think of "the world of children" (one of Shibutani's examples) in terms of its "primary activity" or "technology." If social worlds are understood as defined by activities and technologies, we can map them to some extent onto abstract groups of practitioners discussed above, with an important caveat that Strauss's social worlds are *broader*, as they include not only those who actually *engage* in the world's activities, but also those who are "associated" with those activities more loosely, and the activities themselves are understood more

broadly. Strauss himself does not offer clear examples of such extensive analysis of social worlds, but Becker's *Art Worlds* (1982), which Strauss cites in Strauss (1982), can be read as providing such an example. In this book Becker attempts to describe the different roles involved in production of different kinds of art. In case of film-making, Becker looks at the producers, camera operators, focus-pullers, janitors, film critics, and many other roles. Strauss's discussion (1978) seems to allow for such broad analysis.

Strauss (1978) also extends Shibutani's analysis with a more sophisticated analysis of boundaries, introducing a notion of "authenticity" and "authentication" (p. 123), pointing out that some members of the world are seen as "more authentically of that world, more representative of it" (p. 123) and raising the question of "who has the 'power' to authenticate? and how? and why?" (p. 123). Combined with the notions of legitimate "sites," those questions become essential for our analysis.

While Strauss himself does not analyze the role of space in much detail, other authors working in the same traditions do. Levine (1972) uses the term "Meccas" to refer to places that carry tremendous power in a social world, both in terms of practical ability to coordinate resources and in terms of their *symbolic* power, as sources of legitimation and arbiters of membership. Looking at the Chicago art world Levine points to New York as such a "Mecca," arguing that succeeding in New York is crucial for communicating status in Chicago. As one of Levine's interviewees says: "If you want to show in Chicago, you must move to New York" (p. 298). Local processes thus orient themselves in relation to the Meccas of the social world. As I show later, the software world similarly has easily identifiable "Meccas" on the West Coast of the United States (plus perhaps a few more

places such as Boston, Helsinki, and Bangalore). References to those areas come up frequently in conversations with Brazilian software developers. (Levine says that New York's predominance has much to do with economics: the city is home to a large number of wealthy buyers. The large number of buyers allows for many galleries, and as a result also a large number of knowledgeable and *discerning* buyers. The best Chicago artists consequently sell their work to New York and often move there. This means that ability to sell one's work in New York becomes crucial to gaining status in Chicago.)

Strauss's concept of "social worlds," however, suffers from a number of problems. First, the concept of "social worlds" still lacks clear boundaries, making it hard to know what is and what is not a social world, and potentially letting us see one in any group of people. Second, Strauss moves back and forth between his analog of what I called "internal" and "external" definitions when discussing abstract groups of practitioners. Shibutani's "worlds" are unambiguously a matter of individual perspective—a view that Strauss does not explicitly disclaim. Strauss's own discussion, however, focuses (1982) sometimes on sources of legitimacy perceived by other members and "a collective definition that certain activities are worth doing, and 'we' are doing them" (p. 174), and sometimes on the perceptions of outsiders. The relationship between those is left unexamined. Additionally, like Abbott (1988), Strauss often assigns agency to "worlds" and "subworlds" without explaining how such agency relates to the individual agency of its members.

Considering those difficulties with Strauss's analysis, I do not incorporate the notion of "social worlds" directly, but rather develop a similar notion of "worlds of

practice" defined within my theoretical framework as presented in this chapter and in chapter 1.1, which aims to be narrower than Strauss's social worlds, but sufficiently similar so as to allow me to graft some of the most important insights that Strauss adds to our discussion, in particular the importance of "sites" and "authentication."

We can say that a practice such as "software development" defines "a world of practice" (understood as a type of Strauss's "social world"), to the extent that people who engage in it share certain collective definition of that practice. Understood this way, "a world of practice" is a much narrower concept that "a social world." First, "worlds of practice" are understood as wide in their reach, operating at either global or national scale. Second, they refer to collectivities built around *practice* as defined earlier. I do not, for example, extend this notion to Shibutani's "world of children." Third, I do not aim to include everyone associated with in any way with a particular set of activities (which is the intention of Becker 1982). Just like every world has a set of "primary activities" (or we could say "primary practices"), every practice has its primary social worlds. In other words, while software developers participate in many Straussian social worlds, the practice of software development plays an auxiliary role in most of them. In my later discussion of the "software world" I focus specifically on those people who develop software, with less attention to those who use the software, market it, or make computers on which software is built.

Thinking of a practice as forming "a world of practice" allows us to ask important questions about the geography of that practice, considering what *sites* are central to that practice and how such sites are implicated in *authentication* of individuals as members.

We must be careful, however, to not assume that members who work in central sites necessarily act as arbiters of authenticity directly. Instead, authenticity is often judged separately in each site, and central sites are important mostly because they serve as models of the practice.

In the world of software, for example, there is currently broad consensus that the most important work is done today in two small areas on the West Coast of the United States: the Silicon Valley in California (the eighty mile long area between San Francisco and San Jose) and Seattle/Redmond area in the State of Washington State. (This co-exists with the ideology of irrelevance of geography.) At such "centers," the local community's authenticity is rarely questioned and it suffices for an individual to focus on finding his place in the local community, without needing to worry where this community fits in the larger world. We can contrast such centers with *peripheral sites*. At the periphery, the global status of the local community is questioned regularly. The individual involved in a social world at its geographic periphery thus has to decide whether to cast his lot with the local community or to seek ties with those parts of the social world that lie outside. (This is of course not a perfect dichotomy—as we will see in the case of Brazilian software developers, all participants have to maintain their ties to both the local community and the outside world. They only choose what ties to focus on.) To understand what is happening, we therefore must consider side by side at least three entities: the individual, the local community (with all of its factions) and the larger world with its central sites. By understanding what drives the individuals to seek direct links with external parts of the world we will come to understand how the local community synchronizes its practice

with the rest of the world.

Considering the peripheral members also brings into focus the interface between the world of a particular practice and the broader local society. Strong worlds can overtime transform the society around their centers, making it easier for the members to move back and forth between their world of practice and the mainstream society. For example, while the "nerd" identity associated with software developers was seen in United States as somewhat unmanly in the past, it has been partly incorporated into "hegemonic masculinity" around 1980s and this process continues today (Kendall 1999). Due to this historical work, a software developer working in the Silicon Valley today rarely experiences conflict between his identities as "a man" and "a developer." This integration is perhaps most visible in the late 1990s movies like "The Matrix." Software developers working in Rio de Janeiro, on the other hand, operate in a place where different forms of masculinity are the norm. Reconciling the identities of being a man and a nerd is thus substantially harder in Rio de Janeiro than it is in San Francisco. The conflict between the two identities thus might be easier to observe there.

A world of practice may also be at peace with the mainstream culture at the center because aspects of that mainstream culture are often incorporated into the culture of this world as basic assumptions. Peripheral members, on the other hand, again have to face the contradictions between the demands of the social worlds and those of their local mainstream society. This effect is perhaps easiest to see with language: software developers in California can perform all of their daily activities in one language, while their Brazilian counterparts must switch between the language of the software world

(English) and the language of the local society (Portuguese). The choice of language may become important as a marker of allegiance to one or another group, as we will saw in the previous chapter and will see again in chapter 2.1.

This observation has an important consequence for innovation that occurs at the center vs. at the periphery. Practices and knowledge generated at the center are mobile from birth. While such practices and knowledge may be inextricably tied to local culture and context, this fact does not provide insurmountable problems, since the rest of the world is typically ready to accept such practices on those terms. A book on software development written in California in English does not need to be translated to become successful world-wide: the author can count on the potential readers to either learn English or struggle through the book with a dictionary. Practices and knowledge generated at the periphery, on the other hand, have little change of success outside their local context unless they are actively disconnected from it. In other words, central actors can "disembed" their knowledge using the simplest strategy available, leaving others the hard work of re-embedding it at the periphery. Peripheral actors must perform the most thorough disembedding, to make re-embedding at the center a trivial task. In doing so, they might have to forgo the needs of local users, as we will see in the case of the Lua programming language (chapter 3.2, 3.3).

Reinforcing and Challenging the Centers from the Periphery

The actions of peripheral members might affect the global structure of the social

world in other ways as well, simultaneously re-enforcing and undermining it. Latour (1988a) points out the intellectual gains that arise from being at the center of a global network where knowledge is brought from around the world:

> The zoologists, in their Natural History Museums, without travelling more than a few hundred meters and opening more than a few dozen drawers, travel through all continents, climates and periods. They do not have to risk their lives in those new Noah's Arks, they only suffer from the dust and stains made by the plaster of Paris. How could one be surprised if they start to *dominate* the ethnozoology of all other people? It is the contrary that would indeed be surprising. Many common features that could not be visible between dangerous animals far away in space and time can easily appear between one case and the next! The zoologists see new things, since this is the first time that so many creatures are drawn together in front of someone's eyes; that's all there is in this mysterious beginning of a science. (p. 225.)

Latour points out here that the foundation of European science lies in the massive increase in basic knowledge of the world made possible by its central position in a colonial empire. (Latour refers to such places as "centers of calculation," pp. 215-257.) Such accumulation was not limited to the animals, but to cultural artifacts and the indigenous individuals themselves. While a few centuries ago colonial subjects were often brought to the center by force, today many go there of their own will, and it is usually the most talented of the peripheral members that gather at the center.

At the same time, peripheral recreation of the local practices can in the long term create alternative centers. While most parts of the world are unlikely to ever occupy this role, *some* of them can rise to rival the earlier centers.[165] Bangalore, for example, has risen

165 United States itself has of course been the underdog until after the World War II. History of computing starts in Britain and Germany. See, for example, Huskey & Huskey (1980) on the history of Babbage's Analytic Engine.

to be a minor Mecca of the software world, though it is far from becoming a rival of Silicon Valley, so far playing a clearly subordinate role. (The best work available to software developers in Bangalore today is provided by companies based in the US.) Brazilian developers and economists might wonder why India and not Brazil has risen to play second fiddle, but it is still California that rules their imagination.

Additionally, such success might require that the local members of the world of practice create an enclave that is separated from the local context. The success of Bangalore as a software hub, for example, might be attributed to the city's lack of commitment to any local language and willingness to adopt English as the working language.[166]

Such places may also be handicapped by lack of proximity to the centers of *other worlds*. San Francisco is a Mecca of a number of worlds, which reinforces the position of each of them. On the other hand, while Helsinki has become somewhat of a Mecca of the mobile world due to Nokia, this hardly means that mobile developers in Brazil would consider learning Finnish. This means that while considering the geography of individual

166 While many Indians learn English in secondary schools and colleges, Hindi is the *lingua franca* and the most commonly spoken language in the northern part of India. On the other hand, Southern India, where most of the population speaks non-Indo-European languages that are quite different from Hindi, has resisted the proliferation of Hindi, typically preferring English as the *lingua franca*. While generally more welcome in the south, however, English has been seen as somewhat in competition with the local languages in some states, in particular in Tamil Nadu—the most prosperous of the souther states. On the other hand, Bangalore, located in the state of Karnataka not far from the border with Tamil Nadu, has historically prided itself for being cosmopolitan and not taking Kannada (the main language of Karnataka) too seriously. (Note that Bangalore's political status in Karnataka was raised by the British, who preferred the city to Karnataka's older capital because of its cooler climate Bangalore is sometimes known in India as "an air-conditioned city".) What some call "cosmopolitanism," others call "submissiveness." "They are used to being dominated," says a Tamil software developer to whom I talked in Bangalore. However, willingness to forgo attachment to local language and culture—and to be "dominated" by the remote centers—is often a pre-requisite for participation in many of the global worlds of practice.

worlds of practice, we must keep in mind the politico-economic structure of the world as a whole. Rio's position in the software world in many ways corresponds to Brazil's "semi-peripheral" position in the world economy as a whole, as suggested by Cardoso (1972) and Evans (1979).

Parts 2 and 3

I presented in this chapter an abstract discussion of the process through which a technical practice that originated in one place is reproduced in others, and the asymmetric geography of practice that results from this process. It is in this context of a partially reproduced system of relationships and peripheral position in the larger world defined by the practice that we must look at the work done by software developers in Rio de Janeiro. In Parts 2 and 3 I try to put more flesh on this model. Part 2 explores in more detail the history of the individual and collective entry into the world of software. Part 3 looks at the different ways in which peripheral actors may organize the local and global resources available to them.

2. Histories and Maps

2.1 Nerds [167]

In April of 2007 I sat in an office in one of Rio's tallest buildings, overlooking Largo da Carioca, talking to "Renato," a manager in a relatively successful software company, which at the time was experiencing some turmoil. We talked about the company's challenges, one of which was hiring quickly a large number of "good developers." (The company's resources did not allow it to look for "great developers," Renato explained, so he had to settle for "good" ones.) I asked Renato what "good developers" were like. *I follow Joel Spolsky's advice,* he replied, referring to a popular US blogger who writes about software development and is widely read by Brazilian software professionals. *I look for people who love to program, who are smart and who get things done. First of all, for people who love to program.*[168] It is easy for him to recognize them,

167 The word "nerds" is used as a Portuguese word in the title of the chapter. This Portuguese word is pronounced ['nɛxʤɪʃ], which can be approximated as "NEH-jish," though this would omit the voiceless velar fricative [x], which has no equivalent in English.

168 Spolsky's essay on interviewing, originally published as a blog post and later included in a book (Spolsky 2000/2004), states two of the principles that Renato mentions: "smart" and "gets things done" (p. 157). While Spolsky discusses the importance of passion in the same essay (p. 160), he does not suggest looking for developers who started to code early or use the phrase "love to program." Renato is most likely attributing to Spolsky an oft-quoted passage from an essay entitled "Great Hackers," written by another software personality, Paul Graham:

explained Renato, since he loves to code himself. *I simply ask them,* he continued, *'How did you get into this profession?' If they say, 'Oh, I wanted to be an accountant but couldn't get a job and then I took a course, and...,' then this is not the right guy. When I hear 'When I was twelve years old I got my dad's Amiga...'—then it's worth continuing the interview.*

In this chapter I look at how Rio software developers make their early steps towards becoming software developers. I focus my attention on a small number of individuals and try to understand their paths to software development in some detail, while pointing out how their stories are similar or different from those of the other developers whom I interviewed. In the next chapter, I take a macro view of software development and the history of its arrival to Brazil, introducing a broader context for understanding the biographies presented here. In this chapter, however, I leave this larger world mostly undefined, focusing instead more closely on the subjective experiences of the young "nerds" who themselves have a limited understanding of the world they are entering. (We return to individual experiences in part 3, after a macro discussion, this time, however, looking at adult professionals, who often have a much better understanding of where they fit on the map of software.)

In addition to looking at the relations between biography and history, the task and the promise of "sociological imagination" according to C. Wright Mills (1959), the two

---

> I know a handful of super-hackers, so I sat down and thought about what they have in common. Their defining quality is probably that they really love to program. Ordinary programmers write code to pay the bills. Great hackers think of it as something they do for fun, and which they're delighted to find people will pay them for. (Graham 2004/2005, p. 98.)

chapters of this part examine the role of *place*. I look at the future software developers as entering the world of software development *in* a particular place and *from* a particular place, and as joining simultaneously a set of economic relations of production, largely localized, and a global world of practice. While I leave the most focused discussion of geography for the next chapter, I point out in my discussion of individual entry into software the many ways in which the experience of young Brazilian "nerds" reflects their peripheral position and foretells some of the issues that we will discuss in part 3.

In Love with Technology

As Renato's interviewing strategy indicates, many successful developers start early, and "fall in love" (*ficam apaixonados*) with software before they start practicing it professionally. For them, programming becomes a hobby before it becomes a profession. The practitioners often say that good developers do not *do* technology: they *live* it and *love* it. The choice of software development as a career becomes a matter of finding an arrangement that would allow those who "love" programming to make a living while doing what they enjoy and find interesting. Jobs, consequently, are often judged by the extent to which they support the practice of software development or interfere with it.[169]

---

169 In the same essay as quoted in footnote 168, Graham writes:

> Great programmers are sometimes said to be indifferent to money. This isn't quite true. It is true that all they really care about is doing interesting work. But if you make enough money, you get to work on whatever you want, and for that reason hackers *are* attracted by the idea of making really large amounts of money. But as long as they still have to show up for work every day, they care more about what they do there than how much they get paid for it. (http://www.paulgraham.com/gh.html)

While this appears to capture well the experience of many programmers working in the US market, especially the best of them, the Brazilian programmers I interviewed typically displayed more concern

While "living" the practice is implied in most of the "Chicago school" accounts reviewed in chapter 1.1, the expectation of *loving* it is far from universal. Most practices develop explanations for why the members engage in them, but they often draw on reasons other than "passion." For example, a professional practice can be understood by the members as a way of making a living without sacrificing freedom of thought (Willis 1977 on manual workers), as service done for the benefit of other people (Orr 1996 on Xerox technicians), or as disciplined and honest work (Lamont 2000 on white working class men).

Renato's use of passion as a way of identifying "good" developers also shows that even among software developers, not all enter the profession in pursuit of a teenage passion. And of those who do, few later choose jobs based *only* on whether the work aligns with what they love to do. Finding a job that pays, and pays reliably, is typically a major concern, especially for the older developers. Software careers are similar to academic careers in this way. In both cases, the new members are often first drawn to the community of practitioners and their esoteric knowledge. However, those who will continue their engagement with the practice must eventually learn to engage in it in a manner that would allow them to earn a living, freeing them from having to dedicate their time to other kinds of work. To move towards more central and more valued forms of participation often similarly requires learning to engage in the practice in very different forms from the ones that may have originally attracted the novice.

The role of passion may also be specific to particular places. Renato attributes the

about finding work that paid the bills reliably. See also the discussion of Van Maanen & Barley (1984) in chapter 1.1.

idea of hiring developers who "love to program" to an American blogger working in New York City, though it was most famously captured by an essayist programmer working in Silicon Valley. In my conversations with software developers in Bangalore, India (including those working for the world's most prestigious IT companies), I typically heard an explanation that one rarely hears in either Brazil or the United States: "passing for computer science." "Passing" means getting a high enough score on the national university entrance exams to get into a computer science program. Young people do not *choose* to do software work in India, they are *chosen* for it. Those who get the highest scores on university entrance exams study computer science. Those who score less do other things. The outsourcing economy guarantees computer science graduates such high salaries in comparison to everyone else that few seriously consider not doing computer science when offered a chance. While Indian developers often talk about their love for software as well, they learn to love it later, in college.

In this regard, the situation of Brazilian software developers is more similar to that of their American developers than the Indian ones. While software development provides good career opportunities, it is one of many upper middle class careers in Brazil, and not the best paying. For PUC-Rio's Department of Informatics, the most prestigious computer science department in Brazil, attracting good applicants for its day time program in "computational engineering" is a challenge at times, says one of the professors. Those who do well on the entrance exams and can afford an expensive day time university face many competing options.

PUC's cheaper program in "information systems," taught at night, is more

popular, however. It appeals to lower-middle class students who see it as a route to social mobility, though a difficult one. The Brazilian software industry primarily serves domestic clients, who often seek relatively simple systems at low prices. While this may contribute to making the software work less attractive to highly educated Brazilians (who sometimes see themselves as overqualified for the work they get to do), it also creates many opportunities for less sophisticated software work, at lower wages. Many software companies respond to this by hiring software developers with incomplete college degrees and occasionally with no college experience at all, then relying on a small number of highly educated individuals to manage and mentor them. The lower-tier developers are typically expected to be enrolled and to eventually complete a university program at some university, but can pursue their studies at night. They typically enroll in private universities (though ones much cheaper and far less prestigious than PUC) and often say that they attend them for the sake of the diploma ("a piece of paper"), without expecting to learn.[170]

One of my interviewees, Miguel, started his career at the age fourteen as an "office boy" in a software company—an assistant tasked with things like delivering documents to clients, and receiving the minimum salary for the State of Rio de Janeiro,

170 A small number of my interviewees saw less prestigious private universities as potentially a source of good education, stressing that such universities often higher qualified instructors (including those who teach in the more prestigious public universities during the day). Furthermore, the instructor's teaching is typically evaluated more stringently in such schools than in public universities, where professors cannot be dismissed. (Stories abound of public university professors who show up for the class just twice in a semester: in the beginning to hand out the syllabus and at the end to hand out the exam.) The main weakness of the private programs thus comes from student's poor preparation and the programs desire to keep the students enrolled and making progress towards the degree (and paying tuition) regardless of whether they are learning anything. According to one of my interviewees, who attended a private university and later proceeded to a prestigious Master's program at PUC, a motivated student can take advantage of the qualified personnel, but few students have such motivation. See also the story of Jason's attendance of a private university below.

around R$260 (~US$140) a month.[171] Between the deliveries, Miguel spent time learning to use the computer, and later the basics of web development, relying on conversations with the developers, books and practice ("reading and testing, reading and testing"). He was eventually allowed to take on simple web development tasks needed by the clients. Two years later Miguel joined his current company as an "intern" working on web development and earning R$300, while attending high school at night. Another six years later, at the time of our interview in 2007, Miguel was considerably more confident in his skills as a developer, was attending a university at night, and was earning between R$1000 and R$1500 a month. At age twenty two he was making substantially more than his father, who had not finished high school. Miguel was looking forward to yet higher earnings in the future. Having grown up in a family that he describes as "more towards poor than middle class," Miguel talked about software development in pragmatic terms—a way to make good living. Some of the other developers whom I interviewed had moved into software in similar ways.

Miguel's story shows that falling in love with dad's Amiga in early childhood is not the *only* way to enter the world of software development. (Note, though, that while Miguel enters the world of software through a work environment, he also does this at quite young age.) The path to software that starts as a childhood hobby is an important one, however. Developers who enter software as Miguel did typically stay at the lower rungs of the software industry. (Much of this is of course likely due to the larger class effect.) The more ambitious ones also often look for alternative careers, unless they

171 In 1999, the minimum salary for Rio de Janeiro was around R$260, with the exchange rate of around R$1.9 for one US dollar.

develop a "passion" for software along the way. At the end of my interview with Miguel I learned of his plans to apply for a government job unrelated to information technology.

As Renato's words indicate, falling in love with dad's Amiga is also how new members are *supposed* to enter the practice. Workers who have "passion" for software are prized by employers like Renato, who devise various strategies for identifying such developers, though for many the ultimate test is simply the feeling of passion that exudes from such programmers. "You can see it in their eyes," many say. And while some developers may display more passion for software than others, nearly all describe software as something they do out of interest. Those who may have come into it for purely economic reasons are careful to not advertise this fact.

While Renato's preference for programmers who love to program can be viewed as a matter of economic rationality (an attempt to hire people who will work hard without need for supervision and may accept lower pay), it is also a matter of a looking for a cultural fit and signs of central membership in the practice. Understanding software as something that one can enjoy and do "just for fun" is one of the "perceptions" (Becker 1953) that distinguishes members from non-members. People who are passionate about software may be skilled at it because they likely dedicate more of their time to it than those who do not enjoy writing code. However, this passion is often itself a sign of certain experience, of "getting" what software development is all about.

Hanging Around, Mapping Interrupts

"Zé Luis," who also goes by an English nickname "Jason,"[172] is now in his early thirties and, like the majority of my interviewees, has lived his whole life around Rio de Janeiro. As many other software developers with lower-middle class background, Zé Luis grew up in the suburbs of Rio, in his case in Nova Iguaçu, a large municipality fifty kilometers from Rio, a place he describes as "a peripheral city, in a third world country."

Like many others software developers of his age or younger, Jason's starts his software biography from fairly early childhood:

> Jason: I've been playing with computers since I was eight, eight years old. I started working with small computers using Sinclair logic, which in Brazil were commercialized by the name TK 85, TK 82.[173]Those were really small computers and my dad bought one of them for me, and I developed little games on it, and my cousins, who were the same age as I, played those games, suggested changes, and I would go ahead and implement them. I learned BASIC using the manual of the computer, which came with native support for BASIC. So I learned it there more or less by myself, and got really interested. But I didn't pursue this much further. Actually, I wanted to be a writer, to write fiction. I always had rather diverse interests, in different areas.
>
> So, it was only years later, when... In the 80s, the education system had a series of problems with the government at that time, for a few years. So there were many strikes and they created gaps of sometimes up to four months during the academic year. [...] During one of those my dad thought it was important to put me in some sort of course so that I wouldn't lose a year without studying. So he put me in a computer [*informática*] course. [...] There in this computer course I was introduced to other technologies: databases, those things. And then eventually got interested in this as a career.[174]

---

172 "Zé Luis" is a pseudonym, "Jason" is the actual nickname, used (with Jason's permission) for reasons that will become clear later in the chapter.

173 See the glossary for definitions and additional information on underlined terms.

174 See appendix C, "Original Interview Quotes" (Jason, June 2007, "TK 85").

Jason's story presents his involvement with computers as happening in two phases. At age eight, his dad got him a computer on which Jason learned to program in BASIC. He did not, however, pursue this interest further at the time. He then came back to programming much later, when he was fourteen. This two-step story is remarkably common, and I believe it reflects the developers' desire to establish the time of their *earliest* experience with computers, since engaging with software in early childhood is one of the ways of demonstrating credentials in a practice that expects passion. (Jason's credentials, however, are put to shame by Célio, quoted later, who starts his story at age *six*.)

Jason got the computer from his father, himself an engineer:

> Jason: My dad was an electronic engineer, but he never worked as an electronic engineer. He graduated in Electronic Engineering, but worked giving electronics classes. When I was a kid he worked fixing electronic equipment. He fixed TVs, radios, those kind of things. And my mom was a teacher in an elementary school. To tell the truth, my dad didn't have a fixed job, he worked doing sporadic gigs. And my mom was always a teacher and earned little. [...] We were always short on money, so I don't know how it happened that they could buy this little computer for me.[175]

The computer was a substantial expense for the family and was purchased specifically for Jason. While some developers talk about first encountering a computer as a tool actually used by their fathers, Jason's story is more common: a father who does not use a computer himself, buys one for his son, seeing it as something that would be worth learning.

Jason's father's later decision to put his fourteen year old son in an a computer course brought Jason in contact with a social environment where he found friends and

175 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Meu pai").

mentors, who would help him develop his interest in computers:

> Jason: Instead of going to a mall we were *hanging around* [says in English] at this computer course. The instructors of the course were experienced people, experienced professionals, they knew a lot, they were good, and so we would be there, picking up tricks and tips from them. People who programmed at a very low level. One guy knew assembler, another one knew C++, another knew C or I don't know.[176] [...] This group of people, we "traded cards" [*trocava figurinhas*], right? We would say: "But how did you manage to do this?" "Ah, I figured out that at such and such interrupt of DOS you can put this thingie and the cursor would then notify you every time that it's... you can intercept the pause at the clock and then you can get the key of the thingie and then you can call this program on top of that one..." Cool ways to do stuff.[177]

While access to mentors and peers was quite important as a means of learning about computers, Datacenter also gave Jason access to a milieu in which learning about computers would be understood as *cool*, and where exchange of findings could be integrated with simply "hanging around" with friends—an alternative to going to the mall, as Jason points out. "Trading cards" (roughly equivalent to English "comparing notes," but with a more playful connotation), provided Jason and his peers not only with an opportunity to learn from others, but also with a *reason* for learning new tricks.[178]

This social side of Jason's experience should not, however, distract us from the more mundane side of working with software, and the individual effort involved in

---

176 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Em vez de ir no shopping").

177 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Interceptar no relógio a parada").

178 Two months before our interview I experienced first hand this importance of immediate social milieu. After starting to work on Sputnik (a wiki based on Lua and Kepler, see chapter 3.4) in April 2007, I found this project consuming more and more of my attention, soon becoming quite a bit more interesting than my fieldwork itself. It took me some time to realize the socialization process that I was going through. Time spent working on Sputnik led to ideas that could be discussed over lunch with other Lua developers the following days. Time spent thinking about my fieldnotes, on the other hand, led to insights that had no immediate audience.

"pushing horizons" by trial-and-error:

> Jason: Those were difficult times, I remember, because finding information was difficult. To figure out how to do something you had to go by trial-and-error. The books were expensive and hard to find, you had to order imported books, so you would go to the bookstore, ask for the catalog, the guy would show you a book. "So there is a book about this?" "Yes, there is." "Then get me one." Then it would take months for the book to arrive, then you would go and buy, and it was crazy expensive. So, a solution normally was to get programs in whatever way possible, someone who had it would make a copy, and you would go and try checking it out and discovering how it worked. Then you would use its resources, and perhaps find someone who had already done something more advanced with this: "Hey, how did you do that?" Then the guy would explain it to you and you would apply it in your program.[179]
>
> [...] So it took a lot of time to push our horizons. In return, this was very *thorough* ["bem *thorough*, bem minucioso"]. We managed to do things that sometimes surprised the instructors: "Wow, how did you manage *that*?" "Yeah, I had to map all the interrupts there and find out that this one did this, the other one did that. I had to find some way to work around this thing that I couldn't do." That happened...[180]

Jason stresses the lack of books in those "difficult times" as the reason for having to do by hand the hard work of understanding system's low level behavior by systematically mapping it out ("mapping interrupts"). This specific problem is rarely mentioned by those who started learning programming later, in the age of Google. One important aspect of software work has remained unchanged, however: now as then, software development requires countless hours of individual work, much of which goes towards understanding why a technical system does what it does and how it could be made to behave differently.

The two sides of software work—the solitary investigation and the social "hanging around"—are inherently linked. Programmers usually understand software

179 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Uma época complicada").

180 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Bem thorough").

work as being, at its best, a process of making discoveries ("cool ways to do stuff") and sharing them. This sharing helps to expedite individual discovery work and creates an audience for "war stories" (Orr 1996) about the achieved results. To be able to share, however, one must first discover something. And as many programmers point out, the time that one has to spend alone in front of the computer for this turns away all but those who enjoy this process for its own sake. The effort of "mapping interrupts" requires dedication that is seen as obsessive by outsiders, and often by the programmers themselves, who often say that anyone who is not "obsessed" in this way and does not find joy in this painstaking pursuit of obscure knowledge is likely to find this work too frustrating. "In return, this was very *thorough*," says Jason. Being "thorough" (Jason uses an English word here), is its own reward—a *return* (*compensação,* also "compensation") for the hours spent with the machine. It is only in the right group of peers, however, that Jason comes to see the "intercepting the pause at the clock" as something "cool," a legitimate form of "hanging around," and a reasonable alternative to going to the mall.

After Datacenter, Zé Luis opted for a vocational secondary education (*2º grau técnico*), still in Nova Iguaçu, now thinking of pursuing a career in information technology (*informática*). There he soon found himself ahead of the class and out of touch with his classmates.[181]

> Jason: I finished the primary school in 89, I think. At that point I could program in BASIC, dBase, Clipper, Pascal, a whole bunch of languages. Then I decided to go to a vocational school in IT [*2º grau técnico de informática*], right? Secondary school was a *peace of cake* [says in

181 Typical graduates of vocational computer schools either end up working as system administrators, often a lower-status occupation, or leave computer industry altogether.

English]. I passed everything, since I already understood everything that they were offering there, since I already had contact with all of that.

My secondary school was divided into three years: the first was the basics, we studied all the disciplines: biology, physics, chemistry, and just an introduction to informatics. In the second year the actual technical disciplines started: programming, algorithms, those kind of things. During the first year I discovered a need for automation, for developing a system for the school, and started writing software to keep track of students, for their courses. So that was kind of funny: they had a first year student who wasn't attending technical disciplines yet but was developing a rather advanced system. Which ended up giving me a lot of access to people, the teachers, the labs. I had access to the labs to do this and this gave me the nickname that I use until today professionally.

They called me "Jason," since there was that film "Friday the 13th," about Jason with a mask, etcetera, who would never die. You could shoot him, and then... And I was someone really obsessed with programming, so I would go there and program, spending days there. I studied in the morning that year. [...] And in the afternoon the computer lab was open for me. The access to the lab was restricted, not everyone was allowed to use the computers, but I had access since I was writing this system for the school. So, sometimes I would arrive and start using the computers at noon or so, when the lab opened, and stayed there until ten at night, until they kicked me out. [...]

I remember that once I had an English exam in the morning. I was also good with English, so I did the exam quickly. I finished this English test and left the exam at 8 am, and the labs only opened at one 1 p.m. So I was like: "Man, what am I going to do now to kill time until I can start programming?" So, I went there as if I didn't want anything, to have clear conscience, went there to the lab and it just happened to be open, they were doing maintenance on the computers. So I was like: "Nice, I can go in."

I got there and sat there programming, and the groups were coming and going, coming and going, and I stayed there from eight in the morning until eleven at night without getting up from the chair. And since secondary school is a place where rumors spread naturally, the next day the whole school knew of the boy who had stayed at the computer from eight in the morning until ten at night. So during the next week they started calling me all sorts of names: "zombie," "vampire," "the living dead," "without signs of life." What stuck in the end was "Jason." So, everyone would be like: "Ah, Jason who wouldn't die, who is there at the

computer, as always."[182]

Jason's "obsession" with computers was not understood by his peers at the new school (despite the school's technical focus) and marked him as different from fellow students. Jason persevered, however, accepted the nickname, and now wears it with pride, using it as his "professional name" and incorporating it as a part of the name he uses when signing emails: "Zé Luis Jason da Costa." A year later, as Jason got hired as a teaching assistant in his school's computer course, his technical knowledge started bringing him certain rewards in terms of social status at school. Another year later, as Jason started an internship at the Naval Research Institute, he met more students like himself and eventually developed a group of friends equally "obsessed" with topics few high school students found interesting.

"In a Place So Far Away"

In addition to individual discovery and the tips picked up from peers and mentors, Jason mentions "imported books" as the source of knowledge about technology. While having to order books from abroad must have highlighted the foreign nature of the practice he was starting to engage with, the power of the remote centers over the local practice was not as apparent to Jason in the late 1980s as it is today.

> Jason: We wanted to make applications because at that time there were few applications. There were few things. So, since we understood a bit of programming, we thought that we had what we needed to build those applications and become rich and famous. And it was even more exciting

182 See appendix C, "Original Interview Quotes" (Jason, June 2007, "O Jason que não morre").

> to see that we could build things that were good.
>
> I had a friend at the time, his name was Rogerio, who... Everyone had their own interest and he was the guy fixated on text editors. He wanted to make a text editor because he was tired of the one that existed at the time, called WordStar. [...] So he stated to write a text editor that started to have functionality that was better than WordStar. A 16 year old kid, stuck in a place so far away! And that was cool, this joy...
>
> Yuri: Far away where?
>
> Jason: In Nova Iguaçu, far away from... Even far away from the closest metropolitan center, which was Rio de Janeiro, but also far from the place where commercial software was made, which is there in the United States, there in *Silicon Valley* [says in English], et cetera. So, in a peripheral city in a third world country, the guy managed to make a program that in comparison to the commercial software that was available... you could say: "This software is *good*!" This potential motivated us to study, to learn things.[183]

Jason refers to Nova Iguaçu as "a peripheral city in a third world country"—far even from Rio, not to mention Silicon Valley, the Mecca of the software world.[184] While the physical distance from Silicon Valley to Nova Iguaçu has remained the same since, the meaning of this distance has changed. In 1980s, Jason and his friends were entering the software world in a place that was a lot more isolated. Paradoxically, it was this isolation that helped them dream big. Jason's friend Rogerio wrote his text editor software as an alternative to WordStar. Ironically, by that time (1987–88), WordStar was dramatically losing market share in the US, suffering devastating competition from WordPerfect and

183 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Num lugar lá bem longe").

184 Jason's TK 85 used a CPU by Zilog, a company headquartered in San Jose. WordStar was developed by MicroPro, a company based in California. On the other hand, DOS, C and Sinclair all come from outside the Silicon Valley. Whether or not Jason knew such geographic details at the time, it is clear that "Silicon Valley" has iconic meaning, in many ways standing for all of the US software industry (the "et cetera").

Microsoft Word.[185]

Rogerio's ambition, however, was no match for Jason's own:

> Jason: I remember that at the time I was the guy who liked to do graphics. I wanted to do something that would allow you to run several programs at the same time in different windows. Now you see: I wanted to do this in graphic form, on the DOS screen, but it was very slow, not very good. I wanted to keep trying better solutions, to put smarter video drivers, to copy the data faster. So I arrived at the conclusion that to do this I would have to use the disk and that it would end up being very slow, so I decided that this would not work and gave up. And went to pursue other things. I was quite annoyed when Windows came out a few years later, using of course the disk—which was the idea that I had and discarded as undoable. I thought: "Damn, if I had pursued this, I would have become rich." [Laughs.] Or not, right? [Long pause.][186]

While Jason remembers Windows coming out "a few years later," the first version was actually released in 1985, when Jason was ten years old. Jason did not see Windows until 1990. Technical news took time to reach Nova Iguaçú in the 1980s.

This relative isolation made the 1980s a difficult time to do software work, Jason says. It also, however, made it a time of unchecked dreams. As a teenager, Jason thought he would have become rich had he managed to develop a good way of running programs in multiple windows (something that had made Bill Gates wealthy three years earlier). Now he seems to doubt that this would have helped. While this loss of optimism undoubtedly has much to do with growing up, younger developers rarely express the same

---

185 Already in 1984 WordPerfect was creating serious competition for WordStar. WordPerfect 4.2, released in 1986, dealt WordStar a blow from which it never recovered. By 1987, WordPerfect had 30% of the US word-processing market, reducing WordStar's share to 16%, with IBM's Displaywrite and Microsoft's Word at 13 and 11 percent respectively (Peterson 1994, p. 115). After an advertising campaign that aimed to deliver a "knockout punch" for WordStar in 1997, WordPerfect Corporation focused its efforts on fighting against Microsoft Word (ibid, p. 122), which was growing in popularity and eventually took over the word processing market.

186 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Tinha ficado rico").

sense of excitement as those who entered computing in the late 1980s. The Internet has made Brazilian developers simultaneously more and less isolated. While being more connected, Brazilian developers today appear to be more aware of how isolated they are. In 1989, the Silicon Valley was a rather vague idea. It was hard to imagine concretely what it would be like to be there. Today, the developers are lot more exposed to what is happening in the United States. They are thus more aware of being "stuck" (*enfurnado*) in Brazil.

Perhaps more importantly, the relative scarcity of foreign applications was seemingly creating opportunities for local developers. This situation started changing rapidly in 1990s, as American companies increasingly started to enter the Brazilian market. Local application developers found themselves judged by standards that they were unable to match.

People who start product companies are crazy, says Rodrigo Miranda as we sit down to discuss the history of Nas Nuvens, a company founded by his brother João (see chapter 3.4). *João is crazy in this sense,* he continues. *But he founded Nas Nuvens in 1997. Starting a product company now would be even crazier.* In 1990s the customers knew little of what was happening abroad, explains Rodrigo. Now they compare everything with foreign alternatives. *Then you could say: 'This is a search engine.' And they would say: 'OK.' Now they respond: 'This isn't a search engine. Google is. Is this as good as Google? Does it do the same things?'* As we will see in chapter 3.1, successful companies avoid such competition by building custom software for specific clients, where their location becomes a source of strength vis-à-vis foreign competitors.

The strength of connections to foreign centers varies between Brazilians, and it did so more in the late 1980s than it does today. Unlike Jason, Rodrigo Miranda does not remember any lack of applications. *Hardware was always hard to get in Brazil*, says Rodrigo, *but software was easy. We've even had people from the US ask us to send pirated software from here.* By 1990s, there was "blue box," he continues, which emitted sounds that tricked the phone network into letting you make free phone calls, even international.[187] Some people he knew, says Rodrigo, used those to connect to BBSs in Sweden for five days at a time. *Yes, a five-day international phone call. You could download a lot of stuff this way.*[188] *One could ask them to download this or that specific thing.* Once one friend had the software, they all had it. "We were used to having new software as soon as one week after it was released," Rodrigo adds later.

Later there was also a machine on Fundão, says Rodrigo, referring to an island near Rio where one of the universities is located. *It was connected to the Internet and collected larger downloads. Of course its location and its very existence were secret.* I ask him how he knew about it. Rodrigo smiles. *Obviously all the nerds knew each other,* he explains. Living in Rio's upscale Zona Sul and later attending PUC, Rodrigo knew the right people. Fifty kilometers away, in lower middle class Nova Iguaçu, Jason was attending the wrong high school.

---

187 In the US "blue box" was often a physical device. In hardware-starved Brazil, it was a program run on a computer, that emitted sound through the computer's speakers. One had to place the ear piece of the phone near the speaker to make the call.

188 "Even considering that connections were typically around 9.6-14.4K baud," Rodrigo adds in a later email exchange. "The thing is that 'stuff' at that time was measured in Kilobytes, not Mega or Gigabytes." (9.6-14.4K baud is roughly 1/2 the speed of today's "slow" dialup connections, 1/5 of speed of today's "fast" modems, and 1/50 of a typical home broadband connection.)

In either case, however, the newcomers quickly discovered that they were entering a world centered somewhere far away (even if they were not sure how far) and that success in this world would depend crucially on their ability to build links to those foreign centers. In the very least, they had to obtain access to foreign technology—the hardware and the programs. They had to find the books and learn to read them in English. They also had to learn to build local ties, to make construction of global links a collective project.

Being a Nerd

Early in my research I neglected to ask my interviewees about their *earliest* encounters with the world of computers. Luckily, some of them set me straight. At the end of my interview with Célio, a Senior Systems Analyst for a foreign company in Rio, I asked if there were questions he thought I should have asked him.[189] Speaking in short English sentences, separated by measured pauses, Célio responded:

> Célio: Maybe when or why I got interested in computer science. This happened when I was six years old and I got an Atari—a video game. And this was my first contact with some electronic device. I was able to compute on something. [...] It was a gift from my father and mother. [...] I saw it on television. I was a kid, just a small kid, so this was just for the fun.[190]

Célio's comment took me a bit by surprise, since we *had* talked about his first

189 Completing each interview with such a question is recommended by Weiss (1994). In my experience, the interviewees most typically respond to this question a "no" or with a summary of what they have said. Occasionally, however, they use the opportunity to bring up things that did not come up in the interview or to make methodological suggestions, sometimes quite useful.

190 This interview was conducted in English.

programming experience, and he told me he started programming in Basic on MSX at age eleven. I thought that he was perhaps referring to the same time:

> Yuri: And this was when you were eleven?
>
> Célio: No, I was six. At eleven I already knew that I wanted a computer, and not a video game.
>
> Yuri: So, Atari was just a video game, you couldn't program it?
>
> Célio: No. But in the end video games are programmed. So I decided I wanted to do that for my life. Though I didn't know what "that" was.

In later interviews, many software professionals similarly mentioned video games as one of the first things that got them interested in computers.

Another developer, Mauricio, now in his late twenties, describes the experience of being a "nerd" in high school:

> Mauricio: And since I was quite a *nerd*, I spent most of my time in the computer lab.
>
> Yuri: What does "quite a nerd" mean?
>
> Mauricio: *Geek*. [Pronounces as in English.] I liked that a lot. I wasn't a very social person.[191]

He later offers an example:

> Mauricio: He [the teacher] would come, give a class, and let people go and the class would go to play soccer. The whole class would leave and we would stay there in the lab. The thing is that Doom came out, so... The big thing to do was to get a mouse and destroy it to make a modem cable. To play Doom against [each other]...
>
> Yuri: Explain...
>
> Mauricio: A cable... for the modem. The serial cable actually, to connect

191 See appendix C, "Original Interview Quotes" (Mauricio, July 2007, "Como era bem nerd").

> computers. There was no network then. Or, rather, there was, but it didn't work there. [...] The mouse had the right connector—serial. [...] It was cheaper to get a mouse, break it and make a cable. It got to a point that we had so much practice with this... We would pull it out of the mouse [*picks up an imaginary mouse, rips off its cord and removes the imaginary isolation with his teeth*], connect the wires, attach... It took less than five minutes to make a cable.[192]

Like Jason, Mauricio spent much of his high school time in the computer lab. Unlike Jason, who tells an almost-too-good-to-be-true story of immediate obsession with "mapping the interrupts," Mauricio talks about a somewhat more prosaic (and seemingly more typical) road: computer games and overall being a nerd.

For many developers, computer games provide the early source of interest in computers. As they spend time playing, many develop an interest in understanding the inner working of the computer, sometimes for pragmatic reasons (finding ways to cheat the game), sometimes out of general fascination (as in Célio's case). They often learn peripheral skills, from installing software to making a serial cable from a pair of computer mice, as Mauricio's story indicates. As we saw in chapter 1.2, computer games often also provide future developers with the early motivation for learning English—a crucial skill for their later engagement with software development. Like the world of software development, the world of computer games is centered far away, requiring newcomers to learn how to deal with their peripherality. Perhaps most importantly, however, computer games help them learn the satisfaction that can be found in spending hours in front of the machine. To put it differently, computer games help them learn to enjoy being "nerds."

192 See appendix C, "Original Interview Quotes" (Mauricio, July 2007, "Destruir mouse pra fazer o cabo").

While computer games are the most commonly mentioned immediate path to interest in computers, "being a nerd" (*era bem nerd*, *sou um nerd*) is among the most common *explanations* that software developers offer for their their interest in both games and computers. The word "nerd" is written the same way in Portuguese as it is in English, though pronounced differently: "nehji"[193] in Carioca Portuguese. It has roughly the same meaning as in English, though with a heavier connotation of computer use and often a more derogatory sense. When I ask Mauricio to explain what he means by "being a nerd," he uses another English word: "geek," this time pronouncing it just as in English.[194] Most other Brazilian developers who use the word "nerd" similarly avoid defining it. When pushed, they explain it as a matter of interest in computers and not being "social" or offer a humorous definition—"someone who craves machinery instead of carbon-based lifeforms," jokes one of the "nerds."

While interest in computers over "carbon-based lifeforms" clearly defines the typical nerd for many nerds (in Brazil as in the US), the word has a broader denotation in both languages, and considering this broader sense will help us understand the role of "being a nerd" in the formation of a software practitioner. One of the versions of Wikipedia's article "Nerd" defined a "nerd" as "a person who passionately pursues

193 More precisely, the pronunciation could be represented in IPA as ['nɛxʤɪ]. Here [x] represents a voiceless velar fricative, a consonant pronounced as a "harder" version of English "h," similar to the pronunciation of "ch" in German.

194 The association with computers is somewhat stronger for the Portuguese "nerd" (pronounced *nehji*) than for its English cognate, putting it somewhere in between English "nerd" and "geek." It appears to be a relatively recent borrowing from English, and its denotation appears to be widening as it spreads. The "native" Portuguese term for "nerd" is "CDF," an abbreviation for "cu de ferro" ("iron ass"), which refers to people who study too much, with less connotation of computer use. "CDF" is clearly derogatory, while the connotation of "nerd" is somewhat ambiguous. English word "geek" is generally recognized by many developers as a term with a more positive connotation in English than "nerd," but is rarely used in Portuguese.

intellectual or esoteric knowledge or pastimes rather than engaging in social life, such as participating in organized sports or other mainstream social activities."[195] This positive spin on the term (most likely written by a "nerd") provides a key to understanding the similarities between different ways of "being a nerd" and the consequent transitions between different practices in which the nerds engage. Instead of engaging in "mainstream" activities (such as playing soccer, in Mauricio's case), nerds find satisfaction in pursuit of "esoteric knowledge." While they often specialize (becoming "computer nerds" or even "the guy who liked to do graphics"), this pursuit of esoteric knowledge often extends to several domains. Those who find satisfaction in developing software and "mapping interrupts" have often earlier pursued with passion computer games and role-playing games (RPGs). In many ways it is the interest in engaging with *some* system of esoteric knowledge, rather than the choice of a specific system, that defines a nerd.

Nerds are often described (and describe themselves) as "not very social" and are often seen (again by both themselves and the non-nerds) as representing a particular *type* of people: those who *naturally* find satisfaction in pursuits like programming or role-playing games, perhaps being in possession of particular kind of minds, suited for understanding machinery and formal systems, but less for navigating the social environments of the real world. Popular press has entertained the idea that many nerds suffer from a mild form of Asperger's syndrome, a psychological disorder that sometimes

195 http://en.wikipedia.org/w/index.php?title=Nerd&direction=prev&oldid=167702459 (authored on October 27, 2007, last accessed on September 15, 2008). The definition of "nerd" on this page has changed over time and I am using a particular version that I found very apt.

described as a milder form of autism.[196] While people with mild forms of Asperger's may in fact gravitate to some of the activities that are considered "nerdy," successful engagement in "nerdy" practices typically involves a lot more social interaction than is commonly assumed. As we have seen, Jason's and Rodrigo's pursuit of programming benefited greatly from the right group of peers, while Mauricio learned to make a serial cable from a pair of mice so that he and his friends could connect computers and play Doom together rather than individually.[197] Successful participation in the professional software development similarly requires an acute sense for the social dynamics of software teams and the culture of the software world. This is especially true for open source projects, where the participants must rely on a limited set of clues to communicate with people they barely know.

Becoming a nerd is also fundamentally a social process. "I started with this computer thing around 1981 [actually 1982], but I was doing Texas [Instruments] calculators before, so I already knew something about programming," says Rodrigo Miranda.[198] It was at Computique, Rio's first computer store. "There was one (really one) computer store in Rio at the time, so we used to skip classes at school and go there. [...] It

196 The popularity of this idea owes in part to Time Magazine's famously diagnosing Bill Gates (in some ways the quintessential "nerd") with Asperger's, and the later reports of off-the-charts autism rates among the children of Silicon Valley. The story appeared in Time (1994) article that quoted side by side two New Yorker articles, one about Bill Gates and another about autism, finding them "strangely and intriguingly similar." The story has since been repeated many times, including by Wired news (Silberman 2001), which also introduced the reports of high autism rates in the Silicon Valley.

197 See Anderegg (2007) for a more extended critique of the link between Asperger's and nerdity.

198 This conversation occurred over IM and I adjusted Rodrigo's IM-style spelling, punctuation and capitalization in this paragraph. See appendix C, "Original Interview Quotes" (Rodrigo, October 2007, "1981").

was a [Brazilian clone of] Timex Sinclair ZX-81."[199] The computer store was usually empty, so the staff did not seem to mind a small group of kids staying in front of the computer for hours each day, programming on foot. "They were amused by those kids," says Rodrigo, "Everybody had that IBM look, and they wanted to show to CEOs how that TRS-80 really would revolutionize their companies. So I guess pointing at us and saying, 'See? Even kids can use these new computers!'—that could even help them." Rodrigo got introduced to Z80-based computers by a friend, who recognized in him a fellow "nerd": "A guy in my classroom was into electronics and he was trying to learn how to program in assembler. When he heard that the other nerd used to program a TI58 [a programmable calculator], he assumed I would be able to learn assembler too. So he gave me the Z80 ref man [reference manual]. [...] I simply loved it." Learning to program for Z80 became the next step in Rodrigo's career as a computer nerd, a logical step from the programmable calculator, which Rodrigo had earlier received as a present from his cousin. Even earlier, however, Rodrigo was "into electronics": "[assembling] small devices like sirens, LED-based dices,[200] door locks, alarms, digital clocks, nothing too fancy, but at 12 this is amazing. [...] people around me were nerdy too, so everybody discovered those things around the same age." Rodrigo later says that his "LED-based

199 Timex Sinclair ZX-81 was a microcomputer released in 1981 and discontinued in 1983, with a 3MHz processor and 1KB of memory (less than a modern graphical calculator), using a television set for display. It sold in the US for $100. Rodrigo used TK 82C, a Brazilian clone of ZX-81 produced by Microdigital Eletrônica Ltda. (Jason's TK 85 was a later computer and was, according to Rodrigo, "way better" than TK 82C, "due its vast amount of memory (48K) and the presence of colors." (A typical modern computer has at least 10,000 more memory, of course.) The rapid progress of hardware in the 1980s led many to think, says Rodrigo, that "newer nerds were getting things too easy for them, that they hadn't seen the 'real hard days.')

200 A simple electronic device that shows a pattern of lights corresponding to different patterns that can be obtained with a throw of dice.

dice" phase happened when he was between eight and ten, adding that assembling electronic devices at such young age "gets you a lot of points with an engineer father."

As in Jason's case, two groups of "nerdy" people were important to Rodrigo's formation as a nerd and later as a software nerd. While it were his peers that helped him develop an interest in programming computers, Rodrigo describes himself as already "a nerd" by the time his friend invited him to the computer store to learn programming a computer. He learned to be a nerd under the influence of his father and other male family members.

Rodrigo's father, however, was a civil engineer with an interest in electronics, not a software developer and in fact at first disapproved of Rodrigo's interest in software, not recognizing it as "real" engineering. While being nerds and sons of nerds, most software developers are not sons of software developers, Doom players or RPG fans. Like Rodrigo, they often learn from their fathers and other male relatives some of the key "perceptions and judgments" (Becker 1953, see chapter 1.1), including an appreciation for pursuit of esoteric knowledge and some of the skills required for such pursuit (what they call "learning to learn"). They typically apply such perceptions and judgments to other knowledge systems, however.

This change of focus is often encouraged by the fathers themselves, who see new technological areas emerging and consider them more promising for their sons, while not having the time to pursue those areas themselves. Ultimately, however, while able to bring up their sons as "nerds," fathers have limited influence over the specific system of knowledge that their sons choose to pursue. They can pass to their sons the general

perception of technical knowledge as something that can be explored with satisfaction, but not their choice of the domain. The pursuit of esoteric knowledge must be done in a group of peers, and the specific pursuits of the young nerd are ultimately determined by this peer group. This peer group is needed for its own sake (as any other person, a nerd needs a collective in which he could "belong"), for the help that it can provide in the actual learning process, but also because the existence of a peer group is entailed by the "perceptions and judgments" that make the activity desirable in the first place, since only fellow experts can truly appreciate the advanced levels of esoteric knowledge that the nerd acquires.

Professional Nerds

Being a "nerd" is not a career choice, but a way of life often accepted in childhood. Some systems of knowledge, however, underlie what Giddens (1990) calls "expert systems"—socio-technical systems that are essential for the functioning of the modern society yet are opaque to most of its members. People who master "expert systems" (the "experts") face good opportunities for gainful employment. The system of knowledge related to getting computers to perform various tasks supports one of the Giddensian "expert systems" and appears to be particularly appealing to nerds.

As young nerds grow up, they come to realize (with some nudging by adults) that some systems of knowledge bring more financial opportunities than others and start focusing on them as their future profession, leaving computer games as a hobby. As they

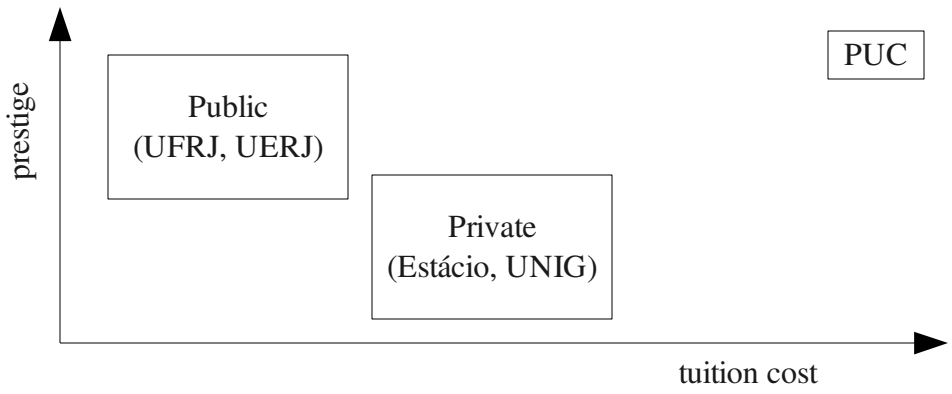
move into software development, they bring with them some of the skills they acquired in other "nerdy" pursuits—such as Mauricio's networking skills perfected for the sake of playing Doom. More importantly, they bring with them a set of Becker's (1953) "perceptions and judgments" that make it possible to see the software development as enjoyable because of the opportunity to "push horizons" that it offers.

This transition, however, involves more than a choice of one system of knowledge out of a few attractive alternatives. The future software developer must start to engage in an entirely new way with the social world surrounding computer knowledge. As I argued in part 1, a practice such as software development must be understood as simultaneously a culture and a system of economic relationships. A sixteen year old "nerd" who has acquired a good quantity of the culture of software development may nonetheless be quite new to its economic side, and in fact may have to re-learn some of the culture. This transformation typically requires a combination of two factors: experience working in an organization that produces software for commercial use and the acquisition of a broader theoretical base of software development. Most developers acquire this broader base at least in part through college education.

While the developers often stress their ability to learn by themselves and are frequently critical of the undergraduate experience, most of those who spend enough time in college recognize at least some value in the experience beyond the certification demanded by the employers. For those who attend college full-time, the early years often provide a crucial socialization experience, introducing the future software professionals to a much expanded circle of like-minded peers as well as to people who have engaged with

software for much longer. Many also stress the importance of the curriculum and the discipline demanded by some of the university programs. “When you go to the university you have a defined curriculum that you have to go through, whether you like it or not,” says Célio, who attended a public university. He illustrates this point with a story of how he learned Java in 1997, at the time when few people at his university even knew of the existence of this new programming language. Célio had to rely on books and the Internet to learn about Java, but credits the university for pushing him to learn by requiring him write a report about a less known programming language. Such learning of course could and does happen in the work place. Few employers, however, want to train their employees from scratch, and most usually require at least a year of college instruction even for those who have learned programming in high school.[201]

Rio residents usually group metropolitan area’s educational institutions into three classes, or rather, two large classes and one institution that stands apart by itself. The classes represent different combinations of prestige and cost, illustrated on Figure N.



201 Gary Becker’s (1962) notion of human capital and Spences’s (1973) discussion of job market signaling provide two useful approaches to thinking about this hiring strategy.

Public universities (*universidades públicas*) are owned and managed by the state, either at the federal level ("federal universities") or at the state level ("state universities"). Public universities do not charge tuition, but require a high score on an entrance exam (*vestibular*) for the more prestigious courses. (Federal universities, such as UFRJ, are typically more selective than state ones, such as UERJ.) Passing the *vestibular* for a popular course typically requires a prior level of preparation that is not offered by public high schools. To receive public higher education in a popular field one thus often needs to either attend a private high school or private preparation courses (*pre-vestibular*), though the situation has changed somewhat in the recent year with the introduction of quotas for students from public schools. Additionally, public universities typically offer classes during the day, making it more difficult for students to combine work and studies, making such universities most attractive to students who can rely on parents to pay for their living expenses.

Lower middle class students who cannot attend public universities are served by **private universities** (*universidades particulares*), typically run as for-profit institutions, such as Universidade Estácio de Sá. Some, like Universidade Iguaçu (UNIG), are set up as non-profit organizations. Such universities charge tuition, but accept students with less preparation and offer night classes, allowing the students to combine studies with full time work.

The third class of educational institutions is **catholic universities** (*universidades católicas*), represented in Rio de Janeiro by a single institution: Pontifical Catholic University of Rio de Janeiro (PUC-Rio or simply "PUC" with the city). Catholic

universities are private in the sense that they are not operated by the state, but are not run for profit. They are typically much more expensive than "private" universities (and are thus the most expensive educational options in Brazil) and are comparable in selectivity to the federal universities.

A year after finishing high school, Jason decided to start a university program. Like many of my other interviewees who attended public high schools, he did not see public universities or PUC as an option, since his high school had not prepared him for the entrance exams. Instead, Jason started a night-time program at UNIG. He abandoned the program after six months, concluding that he was not going to learn anything useful in it:

> Jason: I only spent three months in that university. [...] At the time I was a very technical guy, not very... I was very much a "bit twiddler" [*escovador de bit*] is what we called it, right. A really low-level guy. [...] And so I underestimated such knowledge as high-level analysis. I even considered databases trivial: "Meh, you just take the data, put it there, and later take out, put in again and take it out, put in, take out. Nothing special." So I somewhat underestimated the courses that they had. I though: "Whatever. They won't teach me anything new here, so I won't stay here." And so I went home to study other things. And I ended up doing a mix of things: studied this graphics thing, studied programming languages, learned C++ at that time, between 1991 and 1994, and other things. [...] At that time there already were decent books, right. It was already 91, 92. It was possible to go to a bookstore. They opened one, actually here, downtown, it was our *playground* [says in English]. "Livraria Ciência Moderna" ["Modern Science Book Store"], here in Edifício Avenida Central...[202]

Having up to that point studied largely by himself, Jason had appreciation for certain types of knowledge, which did not include the theoretical and "high level" knowledge that

202 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Subestimei as cadeiras").

the university program was offering him. While he later realized that such topics are important for his work, at the time a small local university lacked the prestige that might have convinced Jason to put aside his own judgment and take the courses more seriously. The seeming ease of learning everything at home, using foreign books that were becoming increasingly available, further contributed to his doubts about the university.

Jason's experience with the university illustrates a more general pattern that we will see many times later: peripheral actors often lack trust in each other, which leads them to focus on direct global links rather than on making use of what other local actors can offer. Students assume that local universities (especially the less prestigious ones) cannot teach them anything useful. The university instructors have equally little trust in the students' ability to learn.

Such quick abandonment of the university program is somewhat atypical, enabled in part by Jason's success in finding a programming job prior to attending university. In contrast, most software developers I interviewed in Rio de Janeiro had completed at least one or two years of a university program in computer science or *informática* before being able to get any serious software job. The percentage of those who complete their degrees, however, is much lower than it would be in the United States. Those who can get into (and afford) PUC or one of the public schools typically spend their first year or two just studying (and often remember that year fondly as a time of learning), but then look for an "internship" (*estágio*), which often means a relatively permanent job with somewhat reduced hours and additional flexibility, at a reduced salary. They typically finish their program (which typically leads to a raise), but often talk about the last few years of school

without enthusiasm. The most important effect of the university program, therefore, is that it helps the future developers obtain an internship.

Those who cannot attend public schools or PUC typically start working in a different (though often related) occupation, for example offering technical support, while simultaneously attending night classes. After a year or two of night classes they can often get an internship, though at a lower pay than PUC students. While they often doubt the quality of instruction in the universities they attend, they typically stay enrolled, knowing that such university enrollment will be seen favorably by future employers and that the eventual "piece of paper" will further expand their employment options. Their lack of trust in the program is thus primarily expressed in lack of attention to the course. After obtaining an internship and eventually more permanent employment, developers who study in private universities at night often slow down their progress substantially, sometimes taking many years to finish the program or giving up altogether.

After working briefly in a few small companies, Jason got a job in Petrobras—Brazil's semi-public oil company, widely seen in Brazil as one of the few sites of technical innovation. Working for Petrobras had been Jason's "professional dream," but turned out more complicated than he expected, as Jason discovered the limits of what he knew.

> Jason: It was at Petrobras that I kind of started feeling the crisis of my super-technicism. Because I got there and discovered that the world had changed a little, things were easier to do in the world of 1994, 1995, mid 90s. The technology had become easier, access to information was easier, and I had a lot of technical background in "bit twiddling" but this wasn't as valued as it was five or six years ago in the late 80s. So I saw a market

> that needed bank applications, basically information systems, and I didn't have much theoretical background in this, right. [...] Suddenly I ran into people who came from the information systems from the old days, from COBOL, from databases. And we kind of met in this market funnel, right. Actually, it wasn't exactly a funnel, the market was expanding. [...] In the beginning of 1990s, slowly, the first fruits of the end of the Market Reserve were started to appear, IT [*informática*] opened, bloomed, companies appeared, the market was growing, right. In 94 it had a different format, a different face. There was now demand for much larger systems, so it wasn't so much a funnel, it was an explosion.[203]

Brazilian IT market went through a major transformation in the early 1990s—a topic discussed in the following chapter. As the restrictions on import of foreign hardware were relaxed, the use of computers grew substantially, creating opportunities for people like Jason. This "explosion" brought Jason in contact with a different world, the one that existed before, but was hidden from him, the world of information systems running on mainframe computers, in many ways representing a different culture. This new world gave Jason some credit for his skills as a "bit twiddler" acquired from foreign books and long ours spent with the computer, at least enough to hire him. The world of mainframe computing ("the information systems from the old days") was itself in turmoil, facing the opportunity and the challenge of upgrading to the new microcomputer hardware that was suddenly becoming available. It also demanded, however, many skills that Jason did not have.

> Jason: I arrived there, migrating, and met those guys who were coming from COBOL, etcetera, right. And it was like: "Oh." I was at a disadvantage compared to those guys, because I didn't have theoretical background in data modeling, requirement analysis, things that nobody even talked about at the time. So I got in the mood for studying, etc. And

203 See appendix C, "Original Interview Quotes" (Jason, June 2007, "The Participants").

went to the University of Estácio de Sá, here downtown.[204]

This second attempt at university education, however, ended even quicker than the first. Jason was again disappointed by the quality of instruction and was busy with work.

Work inside Petrobras also presented other challenges. As is common in Brazil, Jason was employed in Petrobras as a contractor:

> Jason: So, it [Petrobras] had this cycle of public competitions for getting new employees, etc, there are people who are employed by Petrobras. But when it needs something done it gets a person from outside, who works on a temporary contract. However, the law doesn't let you do this for long. Labor laws in Brazil are very heavy. So, what does it do? It contracts an external company, a staffing agency, and tells them: "Hire this guy and sell him to me." So the company hires the guy, he does a contract. But then they have to do a tender, right? So they take bids, another company wins, so "Let this guy go. You—hire this guy over there." So the guy is contracted by three different companies, but really he is just working at Petrobras. They want *me* and they invite the companies to bid on contracting me. [Laughs.] I and thousands of people who worked for Petrobras, to make the process faster, to make things work. Because if they were to depend on opening the competition [for employees], that would take time, a year, two, and then there is corruption, what we call "fish soup" [*peixadas*] right—the politicians picking who will or won't get in, they would game the competition, all of this corruption.[205]

A year later, one such reshuffling ("Dismiss him here, contract him over there") resulted in a "bureaucratic accident" leaving Jason without pay for two month. While a few months of a gap in payment would shock few people in Brazil, Jason took it as a sign that Petrobras was not an appropriate work place for a software professional: "As an IT professional I am *naturally* averse to bureaucracy. And when it touches me, I get furious.

---

204 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Fiquei a fim de estudar").

205 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Vende ele pra mim").

I got ticked off and decided to leave. So I returned to working by myself."[206] Jason spent the next ten years working for his own company, finding, like many others, that this gave him more opportunity to practice software development in relative isolation from the Brazilian organizational context.

The contractual arrangement that Jason describes appears to be especially common in Petrobras, since its semi-public status means that hiring and firing employees is even more complicated than it is for private companies, but is often used by many other organizations. Many of my interviewees in fact often told me of multiple levels of contracting. Alta, the software company described in chapter 3.1, provides an example of this. Alta's clients, which include private and public organizations, outsource their IT needs to Alta, seeking to avoid the bureaucratic burden associated with having full-time employees. Alta, however, also hires its developers as contractors rather than full-time employees. The developers then form yet smaller companies, typically with just one or two programmers each, which make contracts with Alta.

While contracting appears to be a lot more prevalent in Rio de Janeiro than it is in Silicon Valley, it is hardly unique to Brazil and cannot be easily explained just by the peculiarities of Brazilian labor laws.[207] Around the same time as Jason endured his arrangement with Petrobras, "permatemp" workers of Microsoft, one of the most "central" sites of software work at the time, were fighting the company over a similar contractual arrangement.[208] Yet, the issue is often understood in Brazil as a uniquely

206 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Naturalmente averso a burocracia").

207 See Barley & Kunda (2006) for a discussion of IT contracting in the US.

208 Vizcaino v. Microsoft Corp., 120 F.3d 1006 (9th Cir. 1997). The case was brought in 1992 by

Brazilian problem. The US software working environments are often also idealized in other ways. For instance, they are often seen as places where ideas are judged purely on their technical merit, rather than on personal connections of their originators. Such idealization of the centers illustrates an additional challenge faced by the peripheral participants. When their understanding of how the practice *ought* to work exhibits a clear lack of fit with the local institutional realities, peripheral developers have no easy way of knowing whether the theory carried by the culture of the practice actually fits with the reality elsewhere or simply represents an idealization that should not be taken too seriously.

In part 3 I explore in more detail the many challenges the developers face in making the abstract software culture work in a particular place, looking at three cases that illustrate three of the possible ways of assembling local and global resources for practicing software in a peripheral place. Before looking at those cases, however, we need a better understanding of the larger world of software, its geography, the history that underlies this geography, and in particular how the software development practice has made its way to Brazil. This is the topic of the next chapter.

"freelancers" who worked for Microsoft from 1987 to 1990, who seek some of the benefits that Microsoft had offered to its permanent employees, arguing that they had in essence been employees of Microsoft (Harvard Law Review 1997).

2.2 Software Brasileiro

In the previous chapter we looked at the experiences of a few individuals entering the world of software development around Rio de Janeiro, Brazil, noting two aspects of this process: the global nature of the world entered by Jason and other developers, and their peripheral experience of this world. We followed the future software developers without a clear map of the world they were getting into, encountering local and global elements of this world one by one, much as they did themselves. To understand the developers' professional experiences presented in part 3 of this dissertation, however, we will need a better historical map of the software world, showing both its contemporary structure and its history, especially in Brazil.

This chapter starts with a quick look at the geography of the software world, considering both the spread of the software development profession and the degree of centralization. I then proceed to a discussion of how the global world of computing established itself in Brazil and in Rio, a particular city at the periphery of that world.

A Global Profession

Software developers are people who write software—compilations of instructions that tell computers how to process information. Such instructions are nearly always written as *text*, though such text must be organized according to very specific formats called "programming languages," designed so that the instructions could be understood

simultaneously by the human and the machine. (See chapter 1.2 for examples.)

Software represented as human-readable text is called “code” (or “source code”) by the people who write it. Such people call themselves “coders,” “programmers,” “software developers” (or simply “developers”), “software engineers” or “hackers.” Those are of course English words, but software developers in many countries understand them and typically use cognates or calques when speaking their own language.[209] For example, the English “(source) code” corresponds to Portuguese “código (fonte),” German “(Quell)code”, French “code (source),” Russian “(iskhodny) kod” (“исходный код”), Japanese “(sōsu)kōdo” (“ソースコード”), Finnish “(lähde)koodi.” Words for “coder,” “programmer” and “software developer” similarly have recognizable foreign equivalents, though their connotations can vary. I use the terms “developers” or “software developers” as relatively neutral in both English and Portuguese.[210]

---

209 *Calques* are similar to cognate borrowings in that they are based on words in a foreign language, but unlike cognates calques copy foreign terms and expressions by translating them part-by-part. For instance, Russian “kod” is a cognate of English “code,” but “iskhodnyi kod” is a calque.

210 The five terms listed above have different connotation in English when used by software developers in the US. “Programmer” is an *outsider* term, used by the general public but never by the *insiders* in the United States. “Coder” is an *insider* term that is not expected to be understood by the outsiders. “Software developer” and “software engineer” are professional terms that mark association with an industry. (Shorter “developer” is also somewhat of an insider term.) “Hacker” connotes passion for writing software and high achievement. Calling oneself a “hacker” may be thus be seen as a immodest. Each of those terms have a cognate in Brazilian Portuguese, but their use differs from English substantially, partly as a result of their use in Brazil’s rigid system of occupational classification. The terms “codificador” and “programador” were used officially for the bottom rungs of the software work (contrasted with “analistas de sistemas” - “systems analysts”). The terms therefore connote low occupational status and are avoided by Brazilian software developers. The same is true for the verb “programar.” To quote one of my interviewees: “In the US programming is related with the activity itself (making a computer understand how to do things) while in Brazil ‘programar’ is related with your status in a project. So in the US even a CTO could say that he is a programmer if that is the case, but in Brazil that would mean that he was still involved with the lower levels of work.” The terms “engenheiro de software” (“software engineer”) or “engenheiro de computação” (“computational engineer”) are also rarely used used colloqually. “Hacker” is used as in English, so it is rarely applied for oneself. “Desenvolvedor de software” or simply “desenvolvedor” thus become the most common terms. (Note that while English “developer” can also be used to refer to people of other professions, such as real estate developers, in Portuguese it only used for software developers.) For

Developers often stress that being a software developer is not *just* about writing code. As most other professionals, software developers spend substantial time meetings with clients and co-workers, coordinating their work. Even when actually working with code, software developers must engage in numerous activities that go beyond simply *writing* it. A large part of their work goes into testing and debugging—identifying, diagnosing and fixing errors in the programs. Another part goes into "research"—understanding available solutions, which can come either in the form of ready software or in the form of documentation. (The dual nature of software as both a text and a machine makes this line fuzzy at times. Software documentation often include snippets of code, while code often includes snippets of documentation.)

The existence of this diversity of tasks leads to a substantial variation in how much of their time individual people dedicate to writing software, and, consequently, makes it hard to draw the boundaries of the occupation. Some may focus their efforts to organizing work, spending less time working with the code. While such people occasionally abandon code work altogether to dedicate themselves completely to management, many combine a substantial focus on managerial tasks with work that involves reading code if not writing it. Others may focus on novel things that could be done with computers, inhabiting the boundary zone between software development and computer science research. A similarly fuzzy boundary exists between software developers and those occupations that are sometimes seen as requiring less skill. For example, software developers must often spend much time managing machines, while

---

those reason, I use the English term "software developers" as the neutral term.

people for whom keeping computers and networks running is the primary task ("system administrators") often write software to handle the tasks they face.

The difficulty of separating software developers from other "computer professionals" makes it hard to estimate precisely how many people do this kind of work. As a rough approximation, however, the number of software developers likely approaches about ten million people world-wide, comprising up to 2% of the employed population in the most developed countries.[211] Those people are distributed quite widely around the globe, though their density varies dramatically. Both of those aspects can be illustrated by figures 2.2.1 through 2.2.3, which show a mapping of IP addresses that have downloaded Lua libraries from LuaForge.org. (Nearly all the files downloaded from LuaForge.org are either libraries or tools for software developers.) We can therefore assume that the overwhelming majority of the IP addresses represent software developers.

The maps show substantial dispersion of software developers working with (or trying out) Lua. With the notable exception of Africa, most populated regions and most of the world's countries are represented, from Nepal and Bangladesh to Paraguay and Nicaragua. This spread is particularly notable, considering that the map does not represent the totality of software developers, but rather those interested in a rather specific programming language. Maps drawn for libraries in other programming languages look quite similar. Figure 2.2.4, for example, maps visitors to a website dedicated to a Python library written by myself, showing a comparable pattern of dispersion, though of lesser degree (explainable by the smaller total number of

211 This is a very rough estimate based on the numbers presented in appendices F and G.

observations). In other words, not only is the practice of software development dispersed as a whole, but so are the use of very specific libraries and tools.

Qualitative data, for example my own interviews with software developers in Brazil and India, confirm this impression of substantial homogeneity of practice. (This homogeneity is rarely noted, though, often being taken for granted.) As we saw in the preceding chapters, and will see illustrated again in the chapters in part 3, software developers in Brazil develop software using essentially the same tools and techniques. They also share jokes, adages and cultural references. (My interviews with software developers in India support the same observations.) They also share their identity as software developers, "nerds," people interested in information technology, often treating this identification with a global community of software developers as a natural part of who they are and as an explanation for their actions, as we saw in the previous chapter.

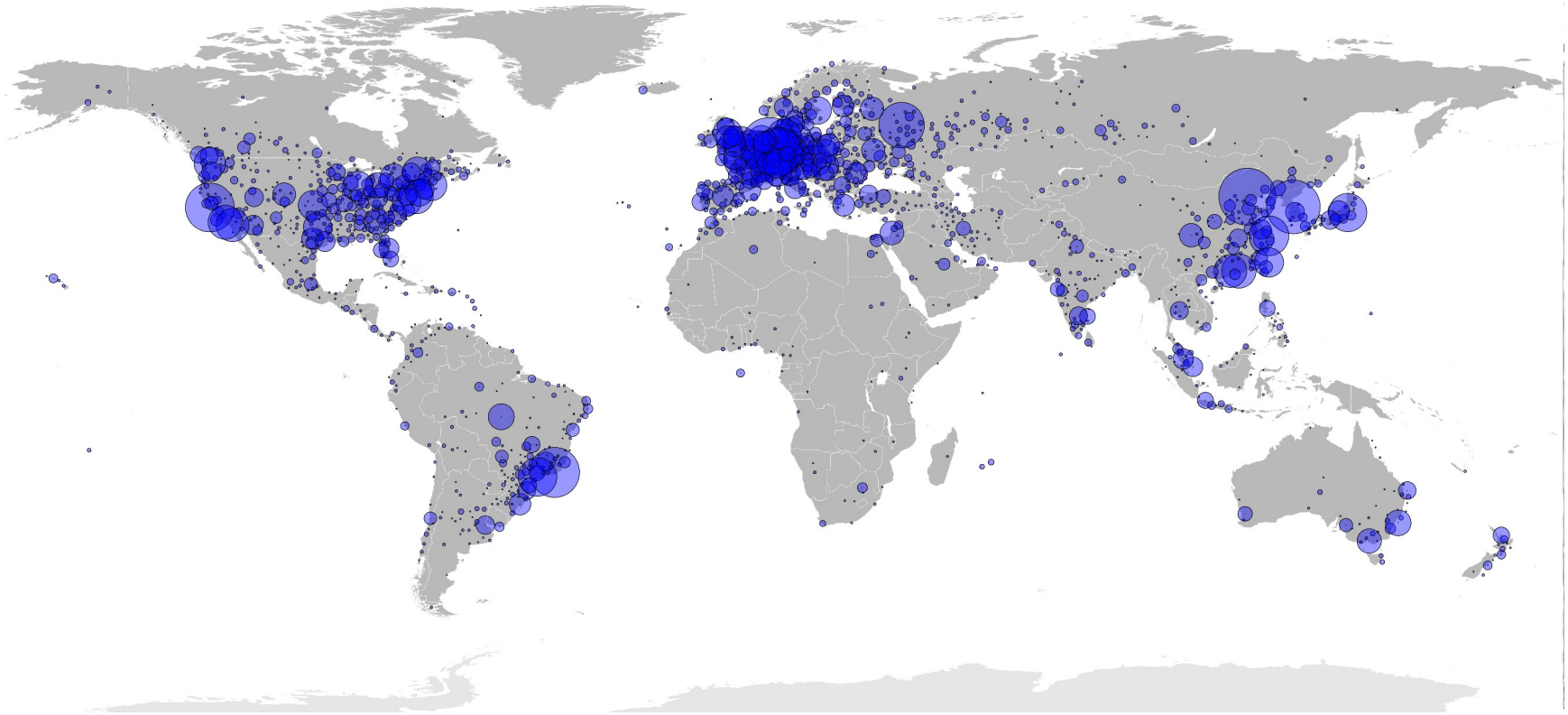


Figure 2.2.1: LuaForge.org downloads, January 2007 – April 2009.

The graph was constructed by taking unique IP addresses that have initiated downloads of Lua libraries from LuaForge.org between January 1, 2007 and April 28, 2009, a total of around 200,000 IP addresses. Those addresses were then mapped to latitude and longitude using GeoLite City database. The resulting observed locations were then grouped with others within 100km from them. The area of each circle is proportional to the number of IPs at the location, with the smallest circle representing one IP each and the largest ones representing around two thousand.

GeoLite City database is provided by MaxMind and is described at http://www.maxmind.com/app/geolitecity. The database maps IP addresses to correct countries in 99.5% of the cases. At the more fine level, IP addresses are usually mapped correctly to a location within 25 km from the true location (e.g., 79% of the cases for the United States, 54% for Brazil), but may be either mapped incorrectly (18% for US, 25% for Brazil) or not mapped at all (3% for US, 21% for Brazil). Unmapped locations are represented by a circle in the middle of each country.

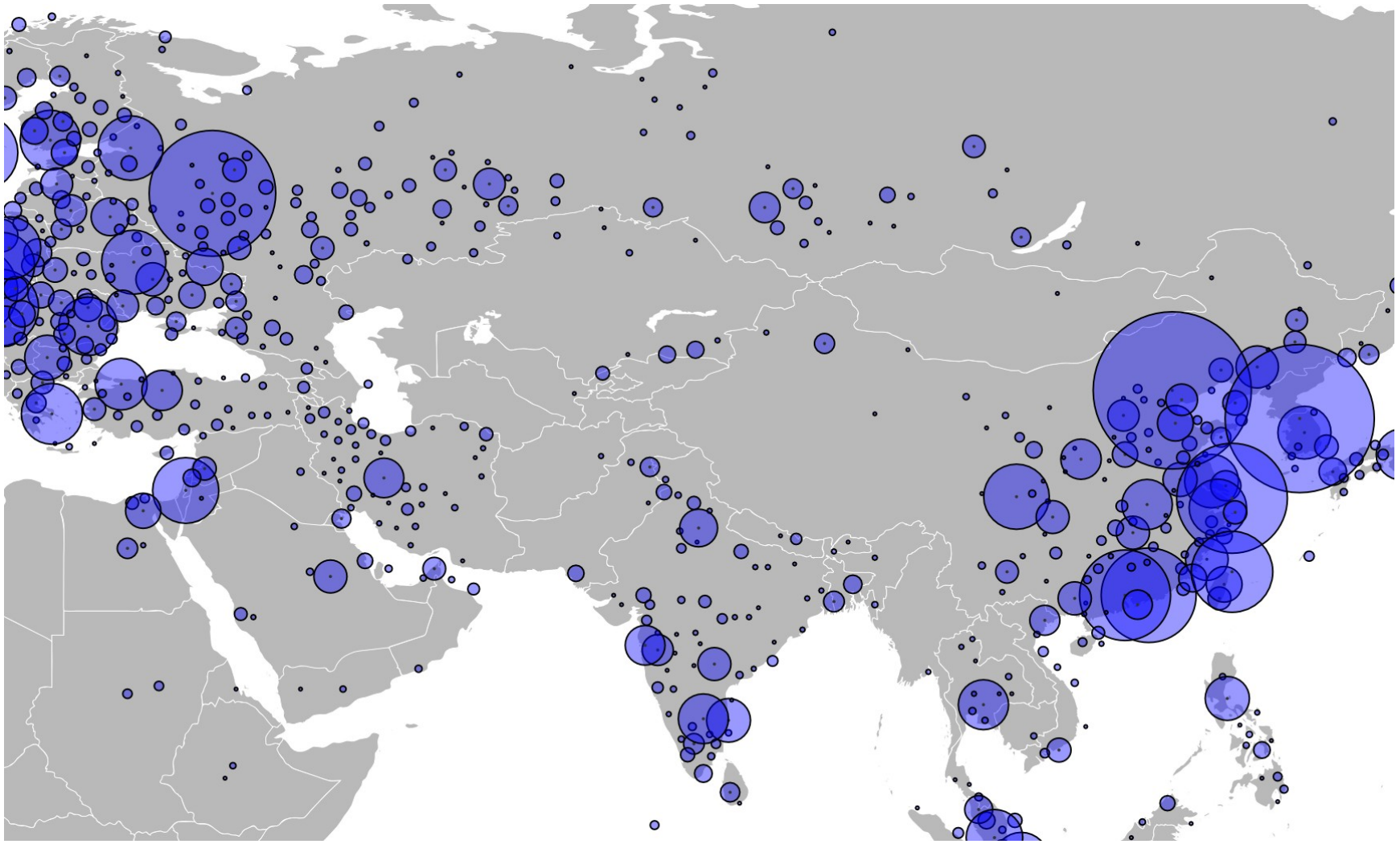


Figure 2.2.2: LuaForge.org downloads, January 2007 – April 2009, Asia.
(See notes for Figure 2.2.1.)

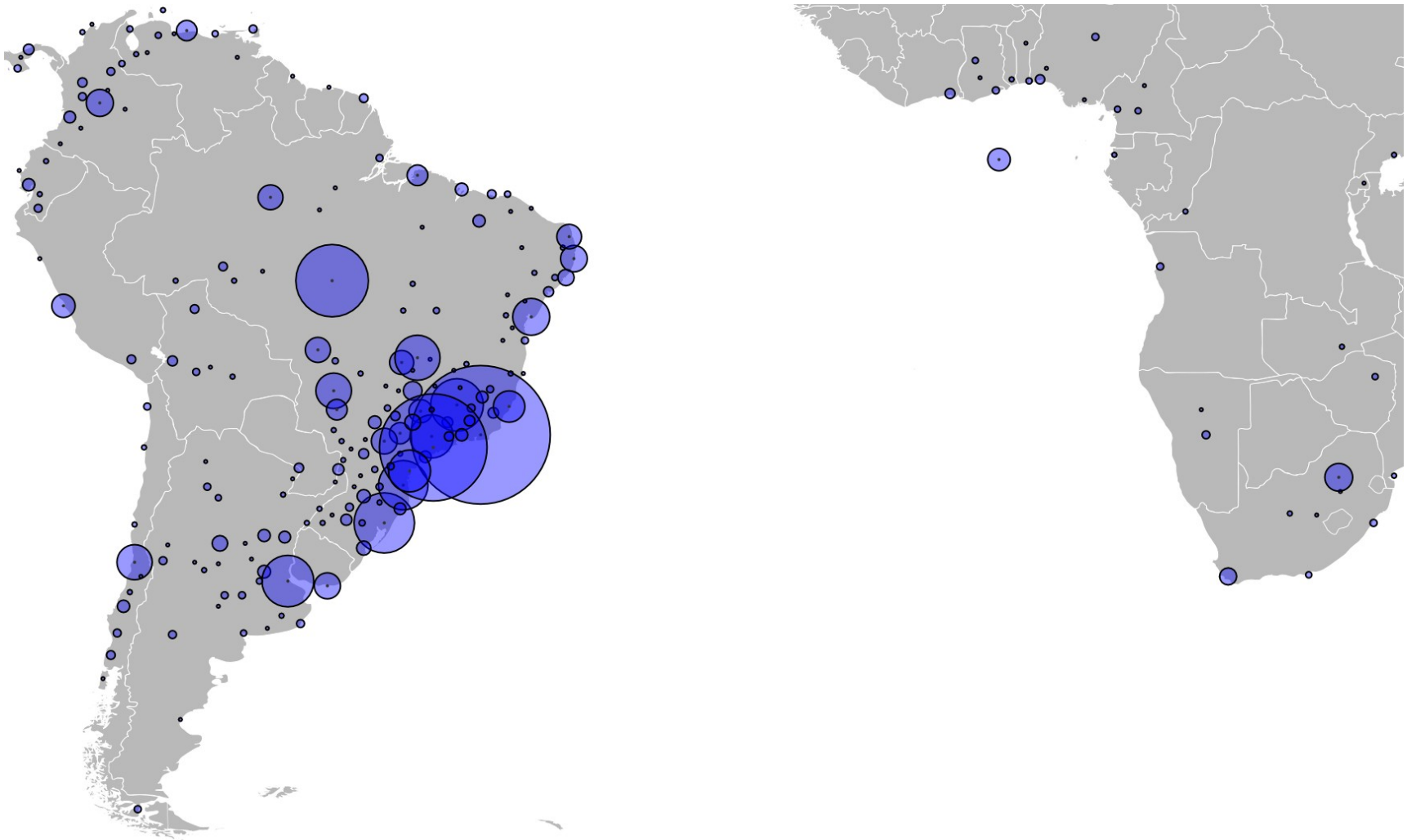


Figure 2.2.3: LuaForge.org downloads, January 2007 – April 2009, South America.
(See notes for Figure 2.2.1.)

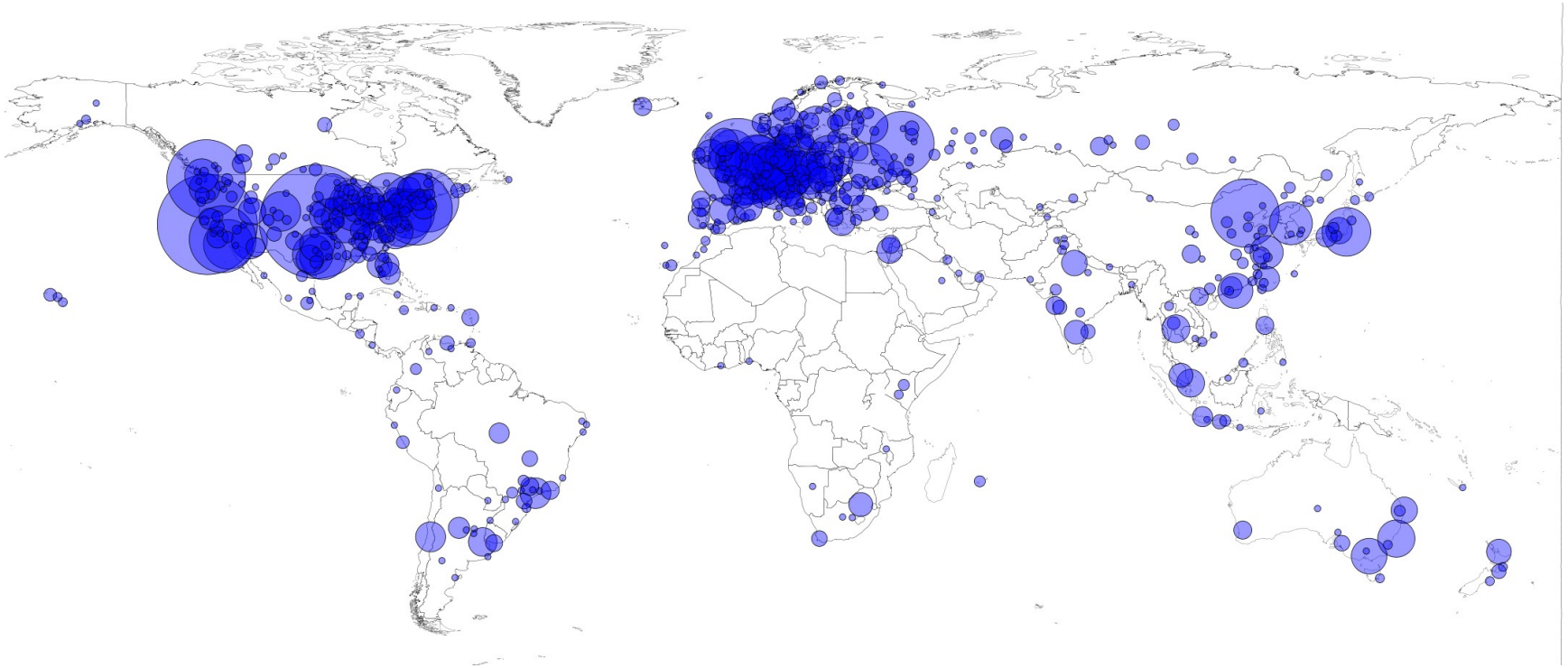


Figure 2.2.4: Python Markdown Visits, January – April 2009.

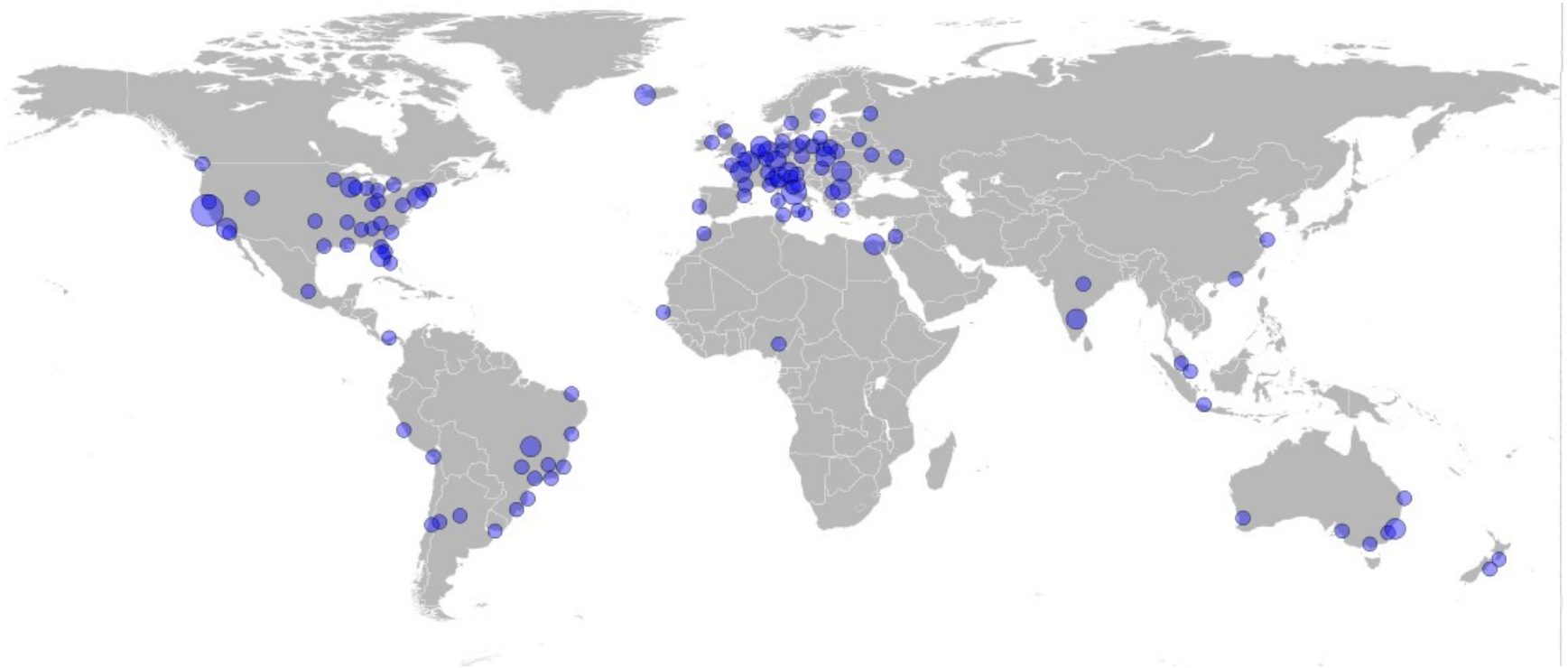


Figure 2.2.5: Java User Groups, April 2009.

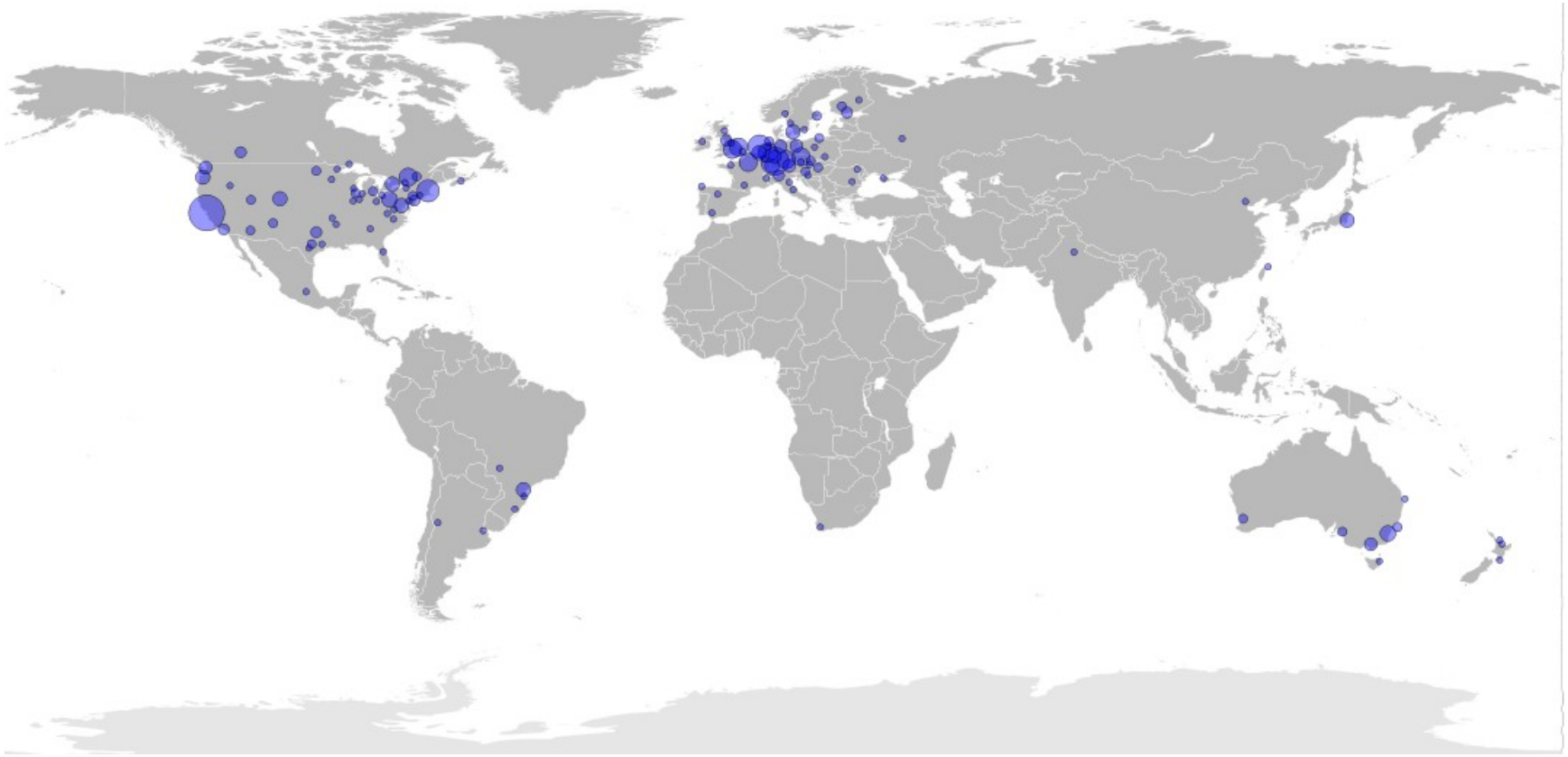


Figure 2.2.6: The location of the Linux contributors credited in the 2007 release.

The map shows the location of addresses included in the credits file for Linux 2.6.22.8. The addresses within 100km are merged and represented by a single larger circle (the area is proportional to the number of people included in the location).

"A server is a server," stress many of my Brazilian interviewees. Whether we think of software in purely technical or in cultural terms, one cannot talk about "Brazilian software" in the same way as about "Brazilian music," that is, as a distinct *kind* of software or a different kind of software practice that could be meaningfully differentiated from "American software" or "Indian software." The title of this chapter is thus an intentional misnomer. The term "Software Brasileiro" is rarely used outside presentations by the government agencies and industry associations. Most of Brazilian practitioners I interviewed had little interest in Brazilian software development, but were instead keen on expanding and improving in Brazil the practice of software development as understood globally.

Center and Periphery

While recognizing the global nature of the software practice, we must also note the extent to which the practice of software development is established in different places. The map discussed above illustrates this variation.[212] Some areas (the two coasts of North America and most of Europe) are covered completely. In other regions, the visitors appear either in smaller clusters or individually. Note that while the map represents traffic to a site hosted in Brazil, the visitors (represented by the points on the map) come predominantly from Europe and North America. Maps of people interested in other

212 This map, of course, represents the variation in the distribution of people working with a particular kind of software and must be taken interpreted with care, since the ratio between IP addresses and people may vary between countries depending on the extent to which such technologies as network address translation are used. (Use of network address translation may lead all the computers inside the same company to appear as one using one IP address.)

software technologies demonstrate similar patterns. Figure 2.2.5, for example, shows the pattern even more clearly (and perhaps more reliably) by plotting the locations of registered Java User Groups.

Official statistics confirm the impression that software developers are concentrated in specific places. Appendix F shows the numbers of computer professionals for the United States and Brazil, with a break up by metropolitan areas. Overall, the United States has around 3 million computer professionals (about 1% of the population and 2% of the employed). Brazil has about 150 thousand (0.1% of the population and about 0.25% of the employed)—about one tenth as much per capita. In Rio de Janeiro computer professionals account for about 20,000 residents of the city, or about 0.2% of its population of around 11 million people (0.5% of the employed population). In comparison, the San Francisco Bay Area in the United States employs around 200,000 computer professionals, who comprise around 5% of the population (7% of the employed)—clearly a much larger number.[213]

This already substantial difference is amplified tremendously if instead of simply counting people, we look at the nature of their work. There is a general impression among software developers in Brazil that the most important software employers are based in specific places, such as the Silicon Valley.[214] A look at the association between market

213 See appendix Appendix F: Counting Software Developers in the US and Brazil for exact numbers and explanations. Comparing Rio to Los Angeles (a city of roughly the same size) would give us about a four times difference in the number of computer professionals as a percentage of employed population.

214 It is important to note that software developers (an occupational category) are employed by organizations whose focus (industry category) may vary widely. For example, many software developers in Brazil are employed in "mineral extraction" companies. (See appendix F.) However, organizations that focus on production of software are generally seen as the best employers, as they

capitalization and location confirms this. In the spring of 2008[215] the value of publicly traded "software development" and "computer services" companies headquartered in the San Francisco Bay Area added up to nearly half a trillion dollars, over 37% of the total valuation of public companies in those categories traded on the US markets, corresponding to about $2.2 million per computer professional employed in the area. (See appendix H for details, including the question of whether this number represents fraction of world's or just the US industry capital markets.) With another 23% of the valuation attributable to the second metropolitan area (Seattle, $4.3 million per computer professional), the software and computer services companies in the rest of the world add up to less than 40% of the total. Companies headquartered in the San Francisco Bay Area and Seattle of course employ a substantial number of developers outside those regions. The work of such developer, however, is often focused on peripheral tasks, including what some of my interviewees call "tropicalization" — the adaptation of global software for the idiosyncrasies of the local context. To the extent that they focus on work central to the companies' products (as is sometimes the case for software developers working for major US companies in places like Bangalore), their work is directed from abroad.[216]

The software industry in a place like Rio de Janeiro is tiny in comparison with the larger centers of the software world. The only publicly traded company based in Rio and

---

make it possible for the employees to engage in the more central forms of the practice.

215 The numbers used in this section were collected in May of 2008. Due to the subsequent fluctuations in the stock market, I decided to not re-do the counts and use the numbers for 2008.

216 See Ó Riain (2000) for a description of remote control involved in the work of software developers working for a US company in Ireland.

engaged in this sector[217] had a market capitalization of about half a billion in 2007 — about one thousandth as much as San Francisco Bay Area's share and roughly the price of five "average" venture-backed companies in the US.[218] While market capitalization may be a biased metric (most of Brazil's largest software companies, such as CPM Braxis, Stefanini, DBA and Tivit were private in 2008, though with plans for IPOs[219]), we should note that there is no shortage of private companies in San Francisco Bay Area. (Palo Alto–based private Facebook, Inc. was typically valued in the range of US$10 billion.)

Perhaps even more importantly, software platforms used by software developers world wide also come from a small number of places. Software developers in Rio de Janeiro, for example, work primarily with two operating systems: Microsoft Windows and Linux. While the former is unambiguously associated with a specific place (so much so that the developers occasionally refer to Microsoft by the name of the city where it is headquartered — "Redmond"), Linux is often described as a globally distributed project, including, in fact a number of contributors from Brazil. A mapping of the addresses included with the names of people credited in a recent Linux release, however, again points to stark centralization, with the largest dot on the map again appearing on the west coast of the United States.[220] Other kinds of software platforms, e.g. databases and

---

217 IdeiasNet, a venture capital company with stakes in a number of smaller firms.

218 According to Fenwick (2008), $46.2 billion worth of VC-funded companies were sold in the United States in 2007, with the median price of $98 million. According to the same source, venture capital investments in 2007 added up to $29.9 billion, of which half was in IT ($14.8 billion) and a third was in San Francisco Bay Area ($9.9 billion).

219 While most planned to get listed on São Paulo BOVESPA rather than American exchanges, I expect that many will be available on US markets via ADR. None of those companies are based in Rio de Janeiro, however. Cobra Tecnologia, one of Brazil's oldest IT companies, currently based in Rio, is also private.

220 Other metrics offer somewhat more complex though not altogether different picture. Corbet (2007)

programming languages, whether proprietary or open source, are similarly associated primarily with a small set of places.

To explain this centralization, one could point to the concentration of venture capital and investors' reluctance to put their money in remote companies, especially those located far from the established centers (e.g., Zook 2002, Powell et al. 2002). We could wonder if places like Rio de Janeiro just lack sufficiently smart people, perhaps looking for the flaws in their education systems or at their losses of smart people to "brain drain."[221] We could investigate whether the governments of those places inhibit formation of new ventures through unnecessary regulation. Several of those factors are explored in part 3.

It would be wrong, however, to point to any one factor as responsible for the concentration of the software practice. Reproduction of practice involves re-creation of a

---

presents a list of companies and organizations that have contributed most changes to Linux 2.6.20. Corbet identifies 67% of the changes with eighteen companies and organizations. Of those, 12.9% correspond to companies headquartered in San Francisco Bay Area (Intel, Oracle, Google, HP, SGI, MIPS Technologies and MontaVista), 12.8% - to Raleigh, NC (Red Hat), 7.4% - to Greater New York (IBM), 5.9% - to Greater Boston (Novell), 5.2% - to the Linux Foundation (split between Beaverton, Oregon, San Francisco and Tokyo). Another 8.4 of the changes are split between seven organizations, each contributing between 0.9 and 1.6% of the changes and located in Scotland, Israel, Finland (Nokia), Germany, South Africa, Japan and Souther California. (The remaining 33% of the challenges are contributed by other organizations or by non-affiliated individuals.) It is important to note again that companies headquartered in the San Francisco Bay Area, Raleigh, NC and New York do hire developers in other places. In case of Linux in particular, the contributors include a number of developers working in Brazil.

221 It is important to note that while software platforms are predominantly developed in a small number of places, people who lead their development often come to those places from far away. In open source, the most notable examples would include Linus Torvalds (the author of Linux, from Finland, now in Oregon), Guido van Rossum (the author of Python, from the Netherlands, now working for Google in Silicon Valley), Rasmus Lerdorf (the author of PHP, from Greenland, now in Silicon Valley). Such migration towards the center have often been seen negatively, as aiding the central countries at the expense of the peripheral ones (e.g., Dedijer 1961, Johnson 1965). While the benefits that such migration has brought to the centeral sites seem clear, the peripheral locations may have gained from it as well in some cases. For example, Saxenian (1999, 2005, 2006) argued that migration of engineers from Taiwan and India has helped the development of high technology industries in such countries.

system of relationships between many elements of the practice, most of which are mobile only to a limited extent. People who are unable or unwilling to move to the places where the practice is well established, must recreate it locally piece-by-piece, importing some of the elements and seeking local substitutes for the other. The many challenges involved in this process of reassembly are explored throughout the dissertation. The difficulty of recreating a complex practice ensures that new central sites are established only infrequently. In most cases, it is the decline of formerly central sites and the rise of new ones that poses a puzzle rather than their stability.[222]

In the history of computing, we see only several major changes in what constitutes central sites. The first such change, the movement of the center of computing from Britain and Germany to the east coast of the United States, is best explained by World War II. (Note that this change was not complete, as Britain and Germany are both still home to major secondary centers of the computing world.) The second shift, from the east coast of the United States to the west coast, requires a more complicated explanation.[223] (This shift was again incomplete, with the east coast of the United States still including a large number of major sites and by some metrics still dominating the software world.[224])

Even establishing a peripheral site typically requires extended work by many actors, who must unite their resources. This process is often complicated by the fact that

---

222 Marshall's (1890) explanation of industry clustering applies remarkably well to the modern world. For a more up to date discussion, however, see the literature on industry clusters (e.g. Saxenian 1996, Powell et al. 2002), or the economic literature on spillover effects (e.g. Audretsch & Feldman 1996).

223 See Saxenian (1996) and Turner (2006) for a discussion of some of the reasons why San Francisco Bay Area provided a better home for the computer industry since the 1970s.

224 For example, the North Easter Corridor remains unrivaled in the United States in terms of the total number of computing professionals employed, outnumbering Northern California by a factor of five.

the local actors must often make a difficult choice between trusting other local practitioners to deliver some of the elements of the practice and attempting to import those elements directly from the centers. The rest of this chapter explores the history of such efforts in Brazil and in particular place — Rio de Janeiro.

Brazilian Informatics

The establishment of the practice of software development in Brazil cannot be understood in isolation from the larger system of practices related to computing, including production of hardware and the many uses of computers. It is also important to consider the processes of synchronization that preceded the arrival of the first computers to Brazil, which created in Brazil the context that information technology takes for granted, from the existence of basic research institutions to the availability of electricity. To describe all the different processes of "enrollment" that went into turning Brazil into a "software laboratory," the story would have to start at least as far back at the beginning of colonization of Brazil in mid-16th century, if not to the earlier story of the beginning of Portuguese expansion.[225] To simplify my narrative, I start it around 1930, a crucial year in the twentieth century history of Brazil.

The story proceeds in several somewhat distinct threads, all with their origin in Rio de Janeiro, Brazil's capital at the time. One story concerns the establishment of ITA (Technological Institute of Aeronautics), an educational institution conceived to be "the

225 Such a story would then of course link with one told by John Law (1986), which provides one of the earliest applications of Actor-Network Theory.

Brazilian MIT." Opened in 1950, ITA later becomes the place where the first Brazilian computer "Zezinho" is built (in 1961). The creation of ITA was therefore key to the development of Brazilian community of *makers* of computers. ITA graduates later staffed the government organs responsible for the Informatics Policy of 1970s and 1980s aimed at the creation of a Brazilian software industry. While such policies have failed to attain the result they seek, they affected the future of Brazilian hardware and software practice in important ways.

Another thread, which traces the practice of computer *use* in Brazil, starts with a look at the Brazilian Institute of Geography and Statistics (IBGE), the institution responsible for conducting the Brazilian census and collecting other government statistics. From 1920s through 1950s IBGE relied on mechanical tabulators supplied by IBM, tracking closely the American census practices. IBGE thus became a competent user of "computing" decades prior to the invention of the electronic computers in the mid 1940s. In 1960 IBGE purchased a computer for the upcoming census. This computer was not the first one imported into Brazil,[226] but the step brought electronic computing to Brazil's federal institutions. While the attempt to process the census of 1960s on a computer was a complete fiasco, the Brazilian government appears to have learned the lessons and quickly emerged as a competent user of information technology. (The different levels of the Brazilian government are to this day among the most important clients of the Brazilian software industry and their practices affect today's industry in profound ways.) Several other units of the Brazilian government, including the Navy and

226 The state of São Paulo bought a much simpler computer (UNIVAC 120) in 1957, and a private company (Anderson Clayton) bought a computer in 1959 (V. Dantas 1988).

Petrobras (the Brazilian oil company, state-owned at the time), started using computers soon after IBGE and played an important role in the development of the practice of use. (Petrobras in particular becomes important for the story of Lua, the subject of chapter 3.2.) I focus on IBGE to simplify the narrative, choosing it as the *first* user of computers in the federal government, as well as because its history is more readily accessible than those of Petrobras or the Navy.

IBGE's Univac was one of the three computers approved for purchase in 1960. Another one went to PUC-Rio, to set up Data Processing Center (CPD) there, with the goal of training human resources. In 1965, by which time IBGE's first attempt at a computerized census had been widely recognized as a fiasco, and the lack of trained personnel was among the obvious reasons for the failure (see the discussion below), the government purchased additional computers, giving them to a number of Brazilian universities. While IBGE had attempted to train people for its needs, it needed a more robust and globally connected community of practitioners. Giving computers to Brazilian universities helped solve this problem, as it introduced a new type of actor: academic researchers in computer science, focused more on software than computer hardware, experts in computer *use* with strong global ties. While several computer science departments arose from this purchase, I focus on the Department of Informatics at PUC-Rio—often recognized as the site of the most important computer science work in Brazil and directly relevant to the narrative presented in part 3 of this dissertation. In many ways, the arrival of the US-made computers to PUC-Rio, which I discuss below, provides a more direct starting point for the story of today's community of software developers

than either the arrival of the Univac to IBGE in 1960 or the creation of Zezinho at ITA in 1961.

In 1972 the Brazilian government established an organ called "CAPRE" to tasked with coordinating purchases of foreign computers. Although this was originally conceived as a measure to facilitate computerization while controlling spending (thus helping computer *users*), CAPRE was staffed by graduates of ITA in possession of foreign post-graduate degrees and quickly became the center of a technology policy aimed at stimulating computer *production* in Brazil through a policy of "Market Reserve." The policy continued for twenty years (with some changes), ending in 1992 largely in response to pressure from the US (orchestrated by American makers of computers and software), but also due to the growing frustration by Brazilian users, who felt that their own attempts to follow foreign practices were endangered by the lack of access to proper computers.

The abandonment of the Market Reserve policy decimated Brazilian hardware industry. At the same time, it introduced cheap computers, stimulating use of computing in Brazil and creating demand for software. Many former electronics engineers converted their skills and became software developers. They were soon joined by many younger people, brought to software through availability of affordable computers.

The story presented below is based primarily on existing literature, augmented with a small number of interviews with people who have worked with computers in Brazil in the 1960s and the 1970s. It relies on the best available sources, including several works by journalists (such as V. Dantas 1988, M. Dantas 1989, and Morais 2006),

memoirs (Staa 2003), and work of institutional historians (Freire 1993, Senra 2007). I do not aim, in this chapter, to establish a primary record of the history of informatics in Brazil, but rather to present elements of the known history to illustrate the interplay between the different actors' efforts to establish in Brazil different global practices and the gradual transformation of the local context resulting from this process.

The Brazilian MIT

In 1930 Getúlio Vargas, a governor of Brazil's southern state of Rio Grande do Sul, led a march on Rio de Janeiro, ending Brazil's "Old Republic" and starting a new era in Brazilian history. Vargas brought together a coalition of forces, known as the Liberal Alliance (*Aliança Liberal*), which included as one of its key constituencies "the Lieutenants" (*os tenentes, tenentismo*), a movement of lower-ranked army officers seeking a wide range of progressive reforms. Among the Lieutenants was Casimiro Montenegro, an army pilot and an aviation enthusiast. Once the Revolution was won, Montenegro turned his efforts to the creating an air mail network and otherwise popularizing aviation.

A decade later, as World War II was escalating in Europe, the United States was seeking new allies. In March 1941, National Research Council sent a group of executives, industrialists and researchers to Brazil, to investigate "the opportunities for cooperative industrial developments between the United States and other American republics through the application of North American technical capacities and production methods to Latin

American primary materials" (NRC, 1941, p.4, quoted in Botelho 1999, p. 140, my translation). According to Botelho (1999), the final report of the mission privileged Brazil as the primary future industrial partner of the United States. The report suggested the need to improve the quality of Brazilian education and research institutions and to bring them closer to the American model.

After some wavering between Allies and the Axis,[227] Vargas entered a military alliance with the United States, using it to strengthen both Brazil's industrial position and its military power by drawing on American technology. In 1943, Montenegro went to United States to negotiate a purchase of airplanes for Brazil. While there, he visited Boston and was taken on a tour of MIT by a Brazilian aviator who used to work under his command and who was at that point studying aeronautic engineering at MIT. The visit inspired Montenegro to start planning a "Brazilian MIT"—a higher education institution focused on aeronautics, modeled closely on its American counterpart, and built with the support of the latter (Botelho 1999, Morais 2006).[228]

Two years later, Richard Smith, a professor of aeronautic engineering at MIT, visited Rio de Janeiro, presenting what became known as the "Smith Plan." The plan involved creation of a larger research center (CTA, Centro Técnico de Aeronáutica), as well as a semi-civilian educational institution associated with it, which became known as ITA (Technological Institute of Aeronautics). Smith insisted that ITA followed closely the

227 Vargas's own fascist regime in Brazil ("the New State" established in 1937) made the Axis powers his natural allies, but the proximity of the United States eventually led Vargas to side with the Allies. Brazil then sent a small contingent of troops to Europe, becoming the only fascist state to fight on the side of the Allies.

228 Morais (2006) dates the original visit with 1943. Botelho (1999) says that Montenegro may have visited MIT in 1943 but must have returned for a more serious discussion in 1945.

American model, from guarantees of academic freedom to the faculty and the students, to the use of a *campus* as a model of spatial organization of the institute. Montenegro, who was appointed as the head of ITA, embraced Smith's position enthusiastically (Morais 2006). The first group of ITA students started classes in the early 1950, first in Rio de Janeiro, but soon moving to the new "*campus*" (a foreign word in Brazil) at São José dos Campos, located three hundred kilometers away from Rio and hundred kilometers away from São Paulo,[229] designed by Oscar Niemeyer, Brazil's modernist architect who ten years later designed Brazil's new capital Brasilia. Locating the campus away from Rio, Montenegro intended to isolate ITA from Brazilian politics, which he feared would interfere with the reproduction of the American system of education. (President Juscelino Kubitschek later followed the same strategy with Brasilia, though placing the new capital even further from Rio.) Most of the faculty—"the Wallauscheks, the Theodorensens and the Schrenks" as Morais (2006) calls them—were contracted from MIT.

In 1961, a group of ITA students, led by professor Richard Wallauschek (brought from MIT), assembled "Zezinho"—the first computer ever built in Brazil. Prior to building Zezinho, the students spent three months touring Europe together with Wallauschek, visiting, among other destinations, Bull, the French computer maker, then at its peak. While earning a place in the annals of Brazilian history and indicative of ITA's position in Brazil's education system, Zezinho was disassembled soon after being constructed, its parts reused for other electronics projects. The team that created Zezinho was disassembled in much the same way. One of its creators left Brazil the same year for

229 While ITA was inspired mostly by MIT, locating the *campus* at some distance from major cities made ITA more like some of the other US universities, such as Stanford.

Master's and then a Ph.D. at MIT (V. Dantas 1988), returning to Brazil only in 1972. Many of his classmates similarly headed north for graduate studies, finding few opportunities to apply their skills in Brazil.

For Brazilian engineers to be able to practice the *making* of computers in Brazil, another set of practices had to be established first: those involving *using* computers. I explore one of the origins of such practices in the following section.

Governing with an "Electronic Brain"

Starting with 1920, Brazilian census used Hollerith mechanical tabulating machines supplied by IBM. In preparation for the 1960s census, the American suppliers suggested that IBGE consider using an electronic computer for processing census data—as the US Census Bureau has done since 1951. IBGE originally planned to postpone the transition until the 1970 census, but the election of Juscelino Kubitschek in 1956 changed this plan. Running on a modernizing platform, Kubitschek promised to achieve "50 years in 5" in terms of economic and social development. Performing a computerized census became a matter of national pride (Senra 2007).

The new head of IBGE approved a purchase of "an electronic brain" ("cérebro eletrônico")—a Univac 1105, delivered in the early 1960 by Remington Rand.[230] The machine was installed in a computing center at Praia Vermelha, in Rio de Janeiro. In buying the Univac 1105, a machine installed at the US Census Bureau in Washington

230 Though it was IBM that suggested purchasing a computer, the company lost the bid because it could not promise to deliver a machine as quickly.

D.C. in the early 1959, IBGE statisticians were following the practice of their American counterparts. In fact, according to one of the people I interviewed in Brazil who worked at IBGE at the time, the actual Univac 1105 purchased by IBGE was earlier used by the US Census Bureau, which was returning it to Rand.[231]

The electronic census became the biggest failure in the history of Brazilian statistics. In 1964 the computer had to be turned off altogether for several months (Freire 1993, p. 27). The results of the 1960 census were not fully tabulated until *fifteen years later*, in 1975 (Freire 1993). (The data from 1960 census was processed at the new center set up for the 1970 census, after the 1970 census was tabulated.) While IBGE have never reached agreement on the causes for the disaster, Freire (1993) and the people I interviewed who worked with IT at IBGE in 1960s or 1970s, typically point to four reason, which I group into two classes: the problems inherent in the Univac itself (described as a "fragile" machine) and the local problems specific to the Brazilian context (the lack of parts, the lack of trained personnel and various organizational problems).

Unlike the later computers that relied solely on transistors, Univac 1105 belonged to the generation of computers that relied on thousands of vacuum tubes for data processing.[232] As other computers in that generation, Univac was massive: 35 to 65 tons depending on the configuration (BRL 1961). It required quite a bit of energy and powerful air-conditioning.[233] Installing such computers and getting them to work was quite

231 This should not suggest that Brazil was getting an outdated machine: the first installation of Univac 1105 at the Census Bureau only happened about a year prior (BRL 1961).

232 It also used transistors (BRL 1961).

233 "Air conditioning unit for cooling input water should be at least 35 tons capacity" (BRL 1961).

complicated even in the United States. The Univac had a long way to go in terms of disembedding.[234]

This "fragile" computer was brought by IBGE to a context that was particularly unfriendly towards it. One problem concerned the timely supply of parts. According to one of my interviewees, IBGE made a disastrous attempt to use punch cards made in Brazil. The low quality paper used for the punch cards, however, left paper fibers in the punch card reader, which then had to be deactivated and cleaned. Parts ordered from the United States were often slow to arrive.[235]

Even more serious was the problem with staffing. Lacking trained computer operators, IBGE selected a group of Brazilians and sent them to United States for training. Unfortunately, subsequent gaps in funding (common in Brazil then as they are today), led many of the trained operators to look for other jobs, leaving those who remain to pick up the pieces. As the government seems to recognize soon, taking new people unfamiliar with computers and training them abroad each time would not work as a long

234 While the first UNIVAC was "turned over" to the U.S. Census Bureau in March 1951, it remained at the plant where it was assembled until the end of 1952, according to Ceruzzi (1998/2003), one of the reasons being that "having gone through the heroic efforts to complete and debug the machine, they [Eckert and Mauchly] were apprehensive about dismantling it, moving it, and setting it up again" (p. 27). While by 1960s, UNIVACs were installed in many places in the United States, their assembly still required substantial effort. The description from an installation of a UNIVAC 1105 in the United States in 1959 specifies the exact parameters of the electric current that was used, observations that "the 100 hp motor, which drives a 75 KVA alternator, is started with no load and has a reduced voltage starter control" (BRL 1961, "UNIVAC-1105").

235 Vera Dantas (1988) writes:

> When the stock of imported valves ended, it was God-help-us! Without a rental and maintenance contract for the Univac, thanks to the inexperience of those who negotiated its purchase, IBGE did not have the benefit of the company's import channel. There were only two options: improvise, adapting national components and, when it was no longer possible, get the office of Ministry of Aeronautics, in Washington, to send the cursed valves, using non-conventional means. (p. 57, my translation.)

The problem with the supply of spare parts is also noted by Freire (1993).

term solution: IBGE needed a *broader* local market of people trained to operate and program computers. This solution lay in increased cooperation with Brazilian universities, an in particular with PUC-Rio. In 1965, PUC-Rio received another computer, in addition to the one it was given in 1960.[236]

Finally, Univac did not fit well into the turbulent organizational climate of IBGE at the time. While some of the problems were rectified (the 1970s census was conducted under they eyes of the military dictatorship), what emerged in the long term is a solution that put some distance between Brazilian government and its computers: unable to replicate the necessary organizational climate internally, the Brazilian government routinely outsources many of its IT needs, to companies that are often a lot more similar to their American counterparts than Brazilian government agencies are to theirs.[237] Alta, the company described in chapter 3.1, counts a federal agency among its major clients.

The most important result of the 1960 census for Brazilian informatics was the introduction of Brazil's federal agency to the *use* of computers. After the original disaster, the government appears to have learned important lessons, eventually becoming a competent user of information technology.[238]

The census also created programmers, who at the time were coming from among people engaged in other occupations. Such people learned programming on the job,

---

236 The next census did rely heavily on PUC participation. It should also be noted, though, its success is often also attributed in part to the more substantial assistance of USAID, which sent consultants who stayed at IBGE for months helping install and program the new equipment.

237 As a first step in this direction, a state data processing company (SERPRO) was created in 1964.

238 See Evans (1995) on Brazil's government's historical record of competent use of information technology.

sometimes with the assistance of American employees of IBM and other computer vendors who were sent to Brazil to assist with programming. Such early programmers, however, bear little connection to Brazilian software developers of today. Those of them who I interviewed are now scattered at the periphery of Brazil's software development or have administrative positions. Another community, which formed around the computers given to PUC-Rio in 1960s, had a much more direct role in creation the modern cadre of software developers.

Informática at PUC-Rio

The same year that Univac 1105 was purchased for IBGE, PUC-Rio received a computer B205, made by Burroughs. B205 was tiny in comparison with the Univac (weighing just about one ton), cost half as much (around US$ 1,500,000 vs. US$ 2,976,250 for the UNIVAC), consumed half as much energy while running (only 70kVA), and had half the memory (around 16K vs roughly 30K).[239] The machine was administered by the newly setup Data Processing Center (CPD), staffed almost exclusively by PUC students (Staa 2003).

Arndt von Staa, now a professor at PUC-Rio, joined PUC in 1961 as an undergraduate in mechanical engineering (Lattes 2009a) and soon started working at the CPD. In 1963, he met there Carlos Lucena, the person most often mentioned by many of my interviewees as the pioneer of Brazilian computer science. Lucena had started an

239 B205 numbers are quoted from Staa (2003). The UNIVAC price and memory are quoted from Senra (2007), while weight and energy consumption are from BRL (1961). 70kVA is roughly the power consumed by one thousand incandescent lamps. (The UNIVAC 1105 consumed 170kVA.)

undergraduate degree in Mathematical Economics the year before (Lattes 2009b). Many of the senior faculty members of PUC Department of Informatics started their undergraduate degrees at PUC around the same year in fields such as mathematics, economics or engineering.

In 1965 PUC received another computer. Introduced just that year, IBM 1130 was the least expensive computer to date and is credited by its many fans as the machine that "gave many people their first feel of 'personal computing'" (IBM1130.org n.d.). The machine was "desk-sized" and sold for just $32,280 in the United States (IBM 1965/2008). Computers were quickly becoming mobile. The arrival of IBM 1130 allowed PUC to offer the first computing course, based in the recently created Department of Mathematics (Staa 2003), coordinated by Antonio Olinto, a physicist.

In 1967 yet another computer was bought (IBM 7044) and several of the students, including Carlos Lucena, spent three month at the University of Waterloo in Canada (Staa 2003), following a visit to South America by Wes Graham, the head of Waterloo's computing center, paid for by IBM (Alkema & McLaughlin 2007). The visit laid the foundation for a link between the yet-to-be established Department of Informatics and Waterloo's Computer Systems Group which has lasted to this day.

The same year PUC opened its own Master's program, in which many of the classes were taught by the students themselves. Staa (2003) describes the strange "bootstrap" phenomena involved in starting a program without certified personnel:

> The most curious things happened, such as for, example, a student defending his master's thesis having as his advisor a "professor" who had

> not yet defended his. “Bootstrap” phenomena. Without such phenomena, nothing could have been accomplished. (p. 25, my translation.)

Staa uses the term “bootstrap” to describe the establishment of the Master’s program. While this term is nowadays increasingly used colloquially, to refer to achieving something without outside help, Staa invokes the technical sense of this term, which originated in computer science in 1950s, referring to the different solutions to the “chicken and egg” problems involved in starting (or “booting”) a computer.

One of the decisions made at PUC in 1967 was the name of the program, which soon became the name of the field in Portuguese. Staa (2003) describes the decision as follows:

> The name of the program came after a long discussion, to decide whether we should brazilianize the term *Computer Science*” of the Americans or the word *Informatique* of the French. *Informatique* won, as we considered it a more inclusive term. The first neologism of the field was thus born. It was a master’s program in which everyone one gave classes to another, and everyone was trying to learn together everything that was new. (p. 25, my translation)

The term “informática” has since established itself as a standard term in Portuguese, extending the language to make it appropriate for the discussion of the new practice. Most of my interviewees today use this term as a natural part of their language, applying it also to themselves, as in Jason’s description of himself as “informática person” (*pessoa da informática*). At the time, however, the choice of term was a decision to be made, one of the many decisions that helped shape the local context.

National Informatics Policy

By 1970s the growing demand for computers made the Brazilian government worried about the growing cost of imported computers, many of which were underutilized, having been acquired for status they brought to the agencies (V. Dantas 1988). An agency called CAPRE was set up in 1972 to rationalize the purchase of computing equipment. CAPRE was staffed by representatives of a group that did not exist until a few years prior, called by some authors the "frustrated nationalist *técnicos*" (Evans 1995) or "anti-dependency guerrillas" (Adler 1986, 1987). Those were Brazilian engineers, many of whom graduated from ITA (joined by a lesser number from PUC-Rio) and then received graduate degrees abroad. While some of them stayed in academia (as, for example, did Carlos Lucena and Arndt von Staa), those who looked for jobs outside the universities saw few options that they deemed worthy of their skills. In the intellectual climate strongly influenced by Marxist thought and Dependency Theory (consider Frank 1966, Dos Santos 1970), some of them perceived this dearth of interesting technical jobs as indicative of Brazil's broader dependence on the United States and internal social problems.[240]

Having entered the high ranks of CAPRE on the basis of their technical expertise, the frustrated *técnicos* realized that CAPRE's mandate could be used as a tool of industrial policy aimed at the creation of a genuinely Brazilian technology, by introducing

240 Evans (1995) writes: "They saw the computer industry as part of a broader problem. The number of technically educated Brazilians was growing rapidly in the 1970s. Unless technically challenging jobs expanded just as fast, education would only increase the "brain drain" from Brazil to developing countries. Industrial organization implied social structure. Brazil's current place in the international division of labor fit all too well with its polarized social structure." (p. 107.)

restrictions on computer imports and thus "reserving" some of the Brazilian computer market for the local manufacturers. (This protectionist policy is thus commonly known as "the Market Reserve.") The creation of CAPRE coincided with the growing concern by the Brazilian Navy about its increasing reliance on foreign computer in its naval vessels — a fact that did not fit well with Brazil's increasingly independent foreign policy. The Navy thus threw its weight behind CAPRE's project.

According to the new CAPRE policy, foreign companies would only be allowed to produce and sell minicomputers in Brazil if they made generous "technology transfer" agreements with Brazilian partners. The largest companies, such as IBM, chose to withdraw (though still providing mainframes), but some of the smaller international companies accepted the deal as a way to enter what would otherwise be IBM's domain. Several national companies arose in the process, creating a strong lobby for continuation of the policy (Evans 1995). The existence of such companies made possible (or, perhaps, created a reason for) further computerization projects, strengthening Brazilian government's position as one of the most competent users of information technology among world governments.[241]

In the late 1970s, forces close to Brazil's new military government entered the game, allegedly concerned with security of communications used by the Brazilian foreign service and finding CAPRE's work towards creating a national software industry too slow (V. Dantas 1988). CAPRE was replaced by a new agency, now run by the military,

241 For example, Brazil was one of the first countries to implement electronic tax filing. According to Tigre (2003), by 2001 90% of the tax declarations were submitted online. (Note, though, that Brazil does not require all citizens to submit tax returns so the total number of tax declaration is smaller than in the US — 11.5 million in 2001 according to Tigre.)

sending CAPRE's now somewhat less frustrated *técnicos* to work for the national computer industry that they had helped create. This industry had a number of substantial successes. One of such companies successfully cloned American computers. (Jason's first computer, which he describes in chapter 2.1, was produced by Microdigital Eletrônica, based in São Paulo.) They also successfully developed software, including a complete implementation of Unix, completed by Cobra in 1988 (M. Dantas 1989, Evans 1995).

The Liberalization

The new agency proposed a more aggressive policy, expressed in the Informatics Law passed in 1984. However, 1984 was also the year when Brazil started a transition towards democracy. The coalition of forces that had led to the Market Reserve policy, already damaged by the military takeover (Marques 2000, 2003), started to fall apart. As the industrial policy become accountable to Congress, industries that depended on computers and whose frustration with the inability to buy cheaper foreign technology had grown, found more opportunities to express their opposition. In 1985 the US threatened Brazil with trade sanctions, responding to the increasing losses that the restrictions brought to American companies (Luzio 1996, p. 60). This threat further increased the number of Brazilian industries that stood to loose from the continued policy.[242] As part of

242 For example, Bastos (1994) writes: "In Brazil, domestic managers of IT user companies, were natural allies of the managers of foreign IT subsidiaries who opposed the policy. Larger industrial producers in Brazil complained that the policy's restrictions on importation of IT goods and the alternative of less developed 'domestic similar' hindered their modernization programmes. The US/Brazil conflict added powerful new critics to this group of industrialists. Producers of industrial goods such as orange juice, shoes, textiles, plywood, steel products, and aircraft, whose interests and activities had never before been disturbed by the policy, were led to oppose it because of the American threats of economic sanctions against their exports." (p. 119-120.)

its negotiation with the United States, Brazil made a commitment to phase out the Market Reserve by 1992 (Bastos 1994).

The end of the Market Reserve is sometimes seen as a tragic collapse of an enlightened national policy under the pressure of neoliberal globalization (e.g., Schoonmaker 2002). It is important to remember, however, that the Market Reserve was itself an alliance in pursuit of globalization and its end signified, above all, a desire on the part of many members of this alliance to seek globalization by other means. As we saw in the history presented above, many of the actors that have shaped the policy since the 1940s were to a large extent driven by the same goal: finding a way to engage in Brazil in the global practice of their choosing. Brazilian aviators like Casimiro Montenegro were seeking to establish aviation, but found it hard to acquire airplanes and needed local engineers. Brazilian engineers, created through the efforts of people like Montenegro, were looking for a way to try their hand at the most exciting engineering projects of the twentieth century, such as building computers. The Brazilian government was seeking modern ways of measuring and governing its population, acquiring an interest in using computers and needing programmers to program them. As each group pursued their own globalization projects and required the elements that had to be provided by members of different worlds of practice, they had to decide when to rely on local practitioners and when to import the original elements of the practice. The alliances between the local practitioners of different trades were thus always marriages of convenience. In 1990, as Brazil was looking for change after two decades of oppressive military rule, many were willing to reconsider their alliances. For many of my interviewees, the end of Market

Reserve was a moment of awakening that they only wish had come earlier.

The opening of the Brazilian market to foreign computers decimated the Brazilian computer industry, but also led to a dramatic expansion of computer use in Brazil. (The causes of this expansion were many and included, among other things, the end of hyperinflation after the success of Plano Real in 1994.) The end of the Market Reserve also left Brazil with a substantial number of people who were trained as electronics engineers but now had few opportunities to work on design of hardware. Many of those engineers found that they could transfer their skills to developing software to run on imported hardware.[243] Additionally, some took refuge in local universities, where they started teaching. One of my interviews, once an electronics engineer, explains:

> Jorge: So when the Market Reserve opened [unclear], they [companies] had way to compete with the foreigners. [...] And we, the electronics engineers, we realized that our space was closing. There was no way for electronics to advance in Brazil. So, there were many centers of microelectronics in Brazil, and now there is only *one* — the only guy that was persistent, they continued. They are a kind of intellectual reserve in this area. [...] Just in Rio Grande do Sul. [...] They are still making chips. They make a Java chip now. [...] But we here moved to software.[244]

As Jorge saw it, developing software was an easier task than many of the ones he had faced as an electronics engineer. In a similar way, many of former computer companies have transformed themselves into software factories.[245]

---

243 This was originally pointed out to me by Sidney de Castro Oliveira who was, at the time, planning to write his doctoral dissertation around this idea.

244 See appendix C, "Original Interview Quotes" (Jorge, October 2005, "Migrou pra software").

245 According to Schoonmaker (2002), one of her interviewees, a former president of a major Brazilian computer company, described the skilled labor resources as "'eggs' left behind by 'the serpent of the market reserve'" (p. 128).

Around the same time (1988–1990), as a result of complex negotiations, several Brazilian research centers were allowed to establish digital links with BITNET hosts in the United States, thus becoming BITNET gateways for Brazil (Carvalho 2006, p.84). While this was not the first time Brazilians had accessed foreign computer networks, it was an important milestone. (Prior to that one could connect to foreign computers by dialing into them, as some of Rodrigo Miranda's friends could do with the help of the "blue box" — see chapter 2.1.[246]) In 1992, Rio and Brazil became connected to the Internet, a new computer network that was rapidly growing in popularity around the world.[247] Access to the Internet enabled real-time access to the World-Wide Web, transforming the practice of software development. "Then [in the 1980s] if you knew that the person knew about it, you would spend more time trying to talk to him," explains one of my interviewees contrasting his experience before and after the arrival of the Internet, "It's not necessary anymore. You don't need to, actually... And again, this is primarily due to the Internet. You can get any kind of information you want on the Internet."[248] The students studying in Brazilian universities could increasingly complement the knowledge of their professors with direct use of foreign technical documentation.

It is worth repeating — as this fact too often appears to be lost on many of my

246 See also chapter 5 in Carvalho (2006) for a discussion of BBS use in Brazil in the 1980s.

247 According to Carvalho (2006, p. 96), Internet access was delayed in Brazil due to a strong commitment to OSI — a networking protocol that had been accepted by the International Standards Organization but was being supplanted by TCP/IP, the protocol used by the Internet without any official standardization process.

248 The interview was conducted in English. In the rest of the interview, the interviewee stresses that talking to other local practitioners continues to be important but now takes a different form — that of sharing "hints" about what to look up on the Internet. See Takhteyev (2005) for a longer discussion of such "hints" or "pointers."

younger interviewees, who are often quick to make unfavorable comparisons between the limited knowledge of their university professors and the wealth of information accessible through the Internet — that the Internet did not come in Brazil by itself. So easily taken for granted as the basic infrastructure of the modern software practice, access to the Internet is a complex artifact that required both technical and political negotiations. It became possible in Brazil because of the accumulation of technical expertise in Brazilian universities and the Brazilian government who had over time learned to coordinate their globalization projects.

The late 1990s were a turbulent period of Brazilian informatics, a time of change and much uncertainty about what was possible in the future. Such uncertainty led to both fear and wild dreams. By 2005, when I had started my interviews in Rio de Janeiro, the dust had largely settled and many of my interviewees were ready to share with me what they thought was possible in Brazil and what was not. Access to knowledge was easy — through the Internet. There was also no shortage of local customers willing to pay people who could convert knowledge found on the Internet into running code that could solve their problems. On the other hand, access to capital and foreign markets was hard. The bureaucratic hurdles were there to stay. The pass to success appeared to lie in finding local clients, building strong relationships with them, then gradually expanding a service business. Part 3 considers this and other strategies for pursuing the practice of software development in Rio de Janeiro.

3. The Roads Ahead

3.1 Aplicações Corporativas [249]

It was late March of 2007 and I was in a *kombi,* speeding in the direction of "Centro," Rio's commercial district. Taking elevated highways from the campus of Rio's Federal University on Isla de Fundão, the minivan flew over many of Rio's favelas, finally landing on Avenida Getúlio Vargas, a block-wide avenue, cleared in mid 20th century to modernize the city. I got off at Rua Uruguaiana, a pedestrian street that serves as an entry point to a few remaining blocks of old windy streets, lined with lunch restaurants and office fashion stores, and filled with vendors selling pirated CDs and counterfeit watches. After two blocks, I arrived at Largo da Carioca, a wide square at the heart of Rio's business district. There I waited for Rodrigo Miranda who was going to take me to "Alta," a successful Java company was hoping might become one of the sites of my ethnography. Each minute of waiting felt like an hour in Rio's heat, but I knew it was worth the wait. I had spent the previous few weeks trying unsuccessfully to get myself allowed to come and spend a month inside a Java company. Rodrigo's introduction could make all the difference.

249 "Enterprise applications" (Portuguese).

When Rodrigo arrived a few minutes later, we headed south, crossing Avenida Rio Branco, and entering a tall building that was all too familiar to me. In the previous year I had interviewed people from no less than three companies there. As typical in such office buildings in Rio, the lobby had a system of "optimized" elevators, each going only to a range of ten floors, some of them with long lines. Joining the longest line we ran into several guys whom Rodrigo knew; it turned out all worked for Alta. We followed them to the office, which Rodrigo entered without introducing himself at the door, as if his presence there was perfectly natural. We only paused briefly in the lobby to appreciate the fancy engraved logo on the glass panel that separated the lobby from a large room.

As we entered the large room, I saw three dozen tables, organized into bays and separated by short dividers. Everyone had exactly the same table—including the owners of the company. All but one person looked under thirty. It was like the idea of a Silicon Valley startup from the late 1990s, complete with a bean bag in the corner, perhaps even taken just a little too far. I followed Rodrigo as he shook hands with people, nodding, making our way towards "Felipe" and "Luis," seated at desk in the corner. It turned out that they were two of the three co-founders of Alta.

The four of us went to a conference room where Rodrigo introduced me as a Berkeley doctoral student and an *Herculoid.* "Herculoids" was the name of Rodrigo' private mailing list, which he used mostly to forward technology news. When introducing two subscribers who have not met before, Rodrigo almost always introduced them to each other as "Herculoids." ("It's like saying that you are a friend," he explained to me after we left Alta.)

Rodrigo summarized my research, and talked about what an "opportunity" it had been to get interviewed by me, and how much he recommended that Felipe and Luis agreed to be interviewed as well. The two seemed unsure of what to make of me, but started talking about the company. They were two of the three co-founders, all recent graduates of PUC-Rio's Computational Engineering program. Felipe was now increasingly doing "the commercial part." Luis was working on a new company inside Alta. The third co-founder, "Eduardo," was Alta's main "technical guy." Earlier they all used to work with any programming language that a client would ask for, but now they were trying to focus on Java. Ninety nine percent of their work was now in Java, said Felipe. "99.9%," Luis corrected him.

I start part 3 of this dissertation with a look at Alta, presenting it as the most typical of the contexts that I explored in depth in my fieldwork, providing a background for the later discussion of Lua and Kepler. Calling Alta "typical" would not be entirely accurate, however. Already in 2007, Alta was a highly successful company that in many ways seemed to play all of its cards right. When I returned to Alta in December 2008, I discovered that Alta had grown several times, employing close to a hundred people and occupying three floors of a downtown building. It was now responsible for the main user-facing website of a major Brazilian brand. While Alta was in some way trying to do what many almost all other Rio software companies were trying to do, it was clearly pursuing this strategy more successfully than many of my 2005 interviewees. I will try in this chapter to point out some of the reasons for Alta's success, but this will not be my focus. Rather, I look at Alta as a successful implementation of a particular approach to

peripheral participation in a global occupational world: bringing global technology to local clients. While operating primarily with "standard" foreign technology, this approach involves arms-length relationship with foreign centers of the software world, combined with intense engagement with local organizations. Other chapters in this part of the dissertation introduce two other configurations. One of those involves production of technology with global significance locally, but in relative isolation from local needs. Another, a more complex case, involves an attempt to bring into alignment a wide range of resources, both local and global, in production of a global project aligned with local needs.

The Beginnings of Alta

As I later learned from my interviews with Felipe, Luis and others, the company started in 2003, four years ago before my arrival there, when the three co-founders decided to leave "Kubix," another company where they had all worked during most of they university years. Unsatisfied with the limited growth opportunities offered by Kubix, the three decided to start their own company, taking advantage of PUC's startup incubator, which provided cheap office space on PUC campus, free phone and Internet, as well as help with tasks like marketing.[250] While most software companies in Rio work in software consulting, building custom software for specific clients, entry into the incubator required a business plan that would involve marketing a software *product.* Felipe, Luis and Eduardo wrote a business plan around commercializing Eduardo's recently completed

250 For a brief discussion of the Genesis incubator, please see Didier et al. (2005).

Master's thesis, which proposed a method for integrating enterprise information system. Two months later Alta opened its doors on the premises of the incubator with an initial investment of R$18,000,[251] to which each of the co-founders contributed equally from their Kubix savings.

The product envisioned by Alta's business plan—"InterJX"—never fully materialized, and from my conversations with Alta's founders, it is hard to tell exactly how seriously they took it. The conventional wisdom in Rio de Janeiro is that a software *product* company cannot survive in Rio de Janeiro. ("It's not California," many developers explain.) The incubator, however, wanted to see a product plan and Eduardo's thesis made it possible to tell a believable story. The founders themselves seemed to half-believe the plan, putting an intern to work on polishing off Eduardo's code, while debating among themselves whether to sell licenses for the product or to open source it and hope to make some money on related services. They decided on the latter option, but were nor ready to bet on the product's success, if only for the lack of anything to bet: the starting capital of R$18,000 hardly provided a financial foundation for a product launch.[252]

When an opportunity came to take a contract with Petrobras, the fledgling entrepreneurs decided to take it.

---

251 Between US$7,000 and US$9,000.

252 Note that releasing InterJX as open source would imply a strategy half-way between launching a proprietary product (as described in the business plan) and doing services based on third-party software. It would potentially involve less development an work and less marketing costs than launching a product and the revenue would primarily come from consulting fees. At the same time, the company would potentially be able to use the product to attract customers—assuming that local customers would have any interest in InterJX over "standard" solutions.

> Felipe: This started right in the beginning, because it was one of Eduardo's contacts. A friend of his who worked at Petrobras needed a company that could do maintenance on this contract, with intranet in this case, and so he asked if we were available. And we were: "Well, clearly, let's start billing (*faturar*) and get into Petrobras, which is a large company..." And so we started.[253]

The company proceeded to accept a range of service contracts, which involved software development in different languages, training courses and other tasks.

One of their earliest projects involved Lua. "Rogerio," a student at PUC who had worked with Felipe and Luis at Kubix, wrote a library for linking Lua with Java—as an assignment for a class—and made a presentation about it at PUC. Rodrigo Miranda attended the presentation and offered to pay for some additional features from Kepler's grant. As Rodrigo could not pay Rogerio directly, he made a contract with Alta, which then used the money to hire Rogerio to work on the desired features. Luis and Eduardo then joined too, contributing to other Kepler modules as well. As the company was just starting off, any income was welcome. But it was not so much about the money, stressed all the founders. Above all, they wanted to have a good relationship with Rodrigo, who was ten years their senior and knew a lot of people. The investment in personal relations paid off: Rodrigo later matched Alta with its first large project, which the company could then use as a success case when making proposals to other clients. The founders remembered the favor. "I think that if we hadn't gotten this project Alta wouldn't even exist today," said Felipe.[254] (The fact that I was allowed to spend time at Alta and write about it perhaps had a lot to do with this as well.)

---

253 See appendix C, "Original Interview Quotes," (Felipe, April 2007, "Vamo começar a faturar").

254 See appendix C, "Original Interview Quotes," (Felipe, April 2007, "Nem existiria").

As the consulting business grew, InterJX was increasingly put aside.

> Felipe: So, from the start we moved our focus a little away from investing in InterJX, investing in integration, and from there it just grew. And as time went by, we were working more and more with development of web applications, development of solutions on demand for clients, right. And InterJX was something we were putting more and more to the side.[255]

When some time later Alta became an official "partner" of a large US-based company offering a Java web development platform, the fate of InterJX was sealed:

> Felipe: And a moment came also when we managed to get a partnership with a large, multinational company, which is EIT.[256] We got a partnership with them, and they are one of the main companies that sell integration software. So, we decided to give priority to learning their platform, and to try to offer services on top of it. And this closed the coffin on InterJX. There was a moment when we decided to give priority to working with the technology that is the market leader. Which was EIT. [...] Over there, they have developers, in the United States, Indians, all working to evolve this system. And we here have nobody. [EIT] was light years ahead.[257]

Alta's founders' stories about InterJX have a touch of nostalgia, though seemingly not so much the painful nostalgia of squashed dreams as the sentimental memories of lost naiveté. The decisions paid off, however. Early consulting revenues allowed Alta's founders to hire their most talented friends. The partnership with "EIT," a household name in Java circles, became a source of larger and larger contracts. Alta's fortunes were improving steadily and over the years it had attracted many of the former employees of Nas Nuvens. (Rodrigo lamented the loss of good engineers, but saw their departure to

---

255 See appendix C, "Original Interview Quotes," (Felipe, April 2007, "Deixado cada vez mais de lado").

256 "EIT" is my pseudonym for Alta's foreign partner, a large American company based in the San Francisco Bay Area. I refer to their product as "eiWeb."

257 See appendix C, "Original Interview Quotes," (Felipe, April 2007, "Fechou a caixão").

Alta as the lesser evil, since Alta allowed some of them to work on Kepler on the side.) Alta was invariably considered a success by those who knew it. When I started frequenting Alta's office in June 2007, the company's only problem seemed to be finding place for all the new developers it was hiring (a factor that delayed the starting time of my fieldwork there). This problem got resolved two months later when Alta got a chance to rent an additional, even shinier, office in the same building. Around the same time, Nas Nuvens was asking its employees to come to the office on Saturday to repaint the walls to give the office a slightly more presentable look.

While more successful than some of its competitors, Alta was hardly original in its strategy, using global technology to solve local problems, providing customized IT solutions based on Java web technology to local clients. (As we will see in some of the examples below, though, Alta also to some extent gets to instead adapt local problems to fit the available global technology.) According to the PowerPoint presentation given to the new employees, the company's focus was

> Consulting, integration and development of applications based on
> **new technologies.**

—with the last words shown in big, bold letters. Alta's founders and employees to whom I talked were equally excited about this company's commitment to use of the most advanced world-class technology, technology "at the tip."[258] As later slides explained, in

258 Two Portuguese words are commonly used to describe such technology. One is *tecnologia padrão.* While *padrão* literally means "standard," it could perhaps be better translated as "world standard," as it typically connotes the quality rarely found in Brazil, rather than mediocrity that is sometimes suggested by the word "standard" when used in the US. The second term, *tecnologia de ponta*, literally "technology at the tip," corresponds more easily to the English "bleeding edge."

2007 this meant building web applications in Java, using J2EE. Alta was "A 100% Java company," declared yet another slide. In reality, of course, conservative clients and the need to maintain legacy systems often required Alta's engineers to work on somewhat outdated technology. What the company advertised, however, was its *ability* and *willingness* to use the latest solutions. Alta's developers, or, rather "technology professionals," as they were called in the same presentation, followed the latest technology news, often in English, and peppered their speech with portuguesified English phrases, from technical terms, to business terms ("essa *delivery*," "o *deploy*"), or even general phrases ("é *feeling* mesmo").

The company's mission statement, stated in the same PowerPoint presentation, however, presented Alta's other, *local* side:

> to transform your desires into reality
> through IT services that match exactly your needs.

Rather than selling a uniform product, Alta cultivated durable relationships with local clients, most of which were quite literally within walking distance from Alta's office. Understanding the clients and their needs was a skill in which Alta's more senior personnel took pride. Alta's relationship with its some of its clients was in fact often so strong and durable, that the company almost acted as a part the IT departments of some of its clients. "Intermercado," a large Brazilian retail conglomerate and Alta's largest client, retained nearly half of Alta's employees on *per month* basis, paying a monthly rate regardless of whether there is any work. Such retained employees thus experienced the typical work cycle of in-house developers: periods of intense round-the-clock work when

a project is about to be delivered, followed by quiet time when the details of the next project were being worked out.

If anything distinguished Alta from its competitors clearly, it was the founders' commitment to attracting good developers (mostly from among PUC students) and retaining them with generous compensation. Perhaps taught by their own departure from Kubix, the founders offered free shares in the company for some of the key personnel and paid the salary of others out of their own pockets when necessary. "We had an opportunity to hire Fabio," says one of the co-founders, "but didn't have any money. But he was good, and we couldn't miss this opportunity, so we paid him out of our own pockets." When later talking about Nas Nuvens, he comments: "I knew they weren't doing well when they let Zé go. Zé is really good, you can't let people like him go. They must have been *really* short on money if they couldn't offer him a raise.") A similar, if less radical, approach is extended to all other employees. While the company employs all of its developers as contractors to avoid the complications of Brazilian labor legislation (a nearly universal practice among software companies), it makes a policy of offering them all the benefits that would be due to them had they been hired full time.

Skol with Fabio

A few weeks after my first visit to Alta, I went back to its office to meet "Fabio," one of Alta's young managers whom I had met briefly on my first visit. Fabio came down and met me outside as soon as I called him, apologizing for being a minute late. (*A*

professional indeed, I thought to myself.) He was dressed informally, much like a typical engineer, though with more attention to style. I noticed this mostly because of the stark contrast with the stylish dress shirt that I saw on him the previous time. As I later learned, wearing clothes that strike a careful balance between dressy and hip was the rule for Fabio, and today's t-shirt was an exception. Perhaps noticing me looking at his t-shirt, Fabio explained that he had a busy day, because they had to deliver a project to a client. He would in fact have to go back later to finish some work, he explained.

Fabio asked me if I wanted to sit down right there and get a beer. The space in front of the building was covered with yellow plastic chairs, all decorated with the logo of Skol, a popular beer brand. The number of those chairs, almost all occupied, made me wonder if all of downtown's employees have decided to come out for a beer at the same time. I soon realized that this was precisely the case: it was a Thursday, but Friday was a holiday. "Happy hour," said Fabio in English. We sat down; Fabio leaned back in the plastic chair, lit a cigarette and asked a waiter for a large bottle of Skol, which got brought to us with two plastic cups.

The conversation turned back to his dress. He normally dresses up, explained Fabio, because he is a minority partner in the firm and ends up interacting with clients a bit. They need to maintain a good impression. Also, he was in mid twenties, and a few years earlier he was even younger. The clients tend to dismiss younger people, he explained. So, he had to compensate for that, to look more serious. "More like an adult," he said. Otherwise, he explained, they would think: "What does he know?" Also, he sat in the management section of the room, Fabio continued. They are all together in one room,

but there is a management corner. They talk to clients in the conference room, but the clients often want to see the office. *Now, imagine,* Fabio said, *that the client walks into the office and sees a guy in torn up clothes, with huge hair, a giant beard.* "What does the client thinks? He thinks, *Awesome, this guy is a nerd, he probably knows how to program!"* But when they look at the management corner, they want to see people who are well dressed, people who look like they would understand their business.

Fabio came to Alta as an intern in 2003. Before that he worked together with the founders at Kubix. As is typical of "interns" in Rio software companies, Fabio worked full time for most of his university years, though with some flexibility in hours, so that he could schedule his work around classes and homework. (Though, he only really needed to study right before the exams, he said.) Fabio earned a little under R$400 (US$200) a month when he started as an intern but saw his salary rise dramatically as he learned more. His experience was typical: interns may earn even less than that (the intern status exempts them from minimum wage requirements, see chapter 2.1), while experienced programmers, even without a completed college degree, often ask for R$2000-3000 or more. There are limits on engineering salaries, however, say most developers, and to increase their income further the most talented engineers typically move to management. Fabio was starting this transition this year. About a year and a half ago, just before he graduated from the university, Fabio had become a minority partner at Alta.

Some time later, we started talking about Lua. Despite Alta's claims to be "a 100% Java company," a number of Alta's engineers had contributed to Kepler and two were involved with it at the time. Fabio was one of the two. He had started working with

Lua back at PUC, Fabio explained. He took a class from Roberto Ierusalimschy, one of Lua's authors, and another one from Roberto's spouse. Both had programming assignments in Lua, and in case of Roberto's class the task was to contribute something of value to the Lua community. While some other PUC graduates that I interviewed resented being required to use a "home-grown" programming language (see chapter 3.3), Fabio took the assignments more positively, since some of his friends at Kubix, including Luis, one of Alta's co-founders, were fans of Lua. (Luis in turn attributes his interest in using Lua to seeing it used extensively in Tecgraf, a research lab at PUC.) When Fabio later needed a final project for his engineering degree, Rodrigo Miranda, whom he met through Luis, suggested writing a new version of a Lua programming editor, originally written by Luis and based on Eclipse, a popular Java platform. Fabio wrote it together with Rogerio, a close friend who was now also working for Alta.

After finishing college Fabio continued working on the tool, getting paid a small amount from one of Rodrigo's grants. (As Rogerio said later, the money paid by Rodrigo was hardly much, but it helped establish a commitment.) It was hard to find time for it, Fabio said, but he wanted to continue participating. He gave me two reasons. When you work with something all the time you get tired of it, he said. He was working with the EIT's "eiWeb" platform all day, which he also found to be "very commercial." Lua work was a diversion. It was not serious commercial technology, not something he would suggest to a client, but it was fun (*divertido*), The idea of Lua as diversion from more "serious" technology reminded me or a slightly less sympathetic term I often heard to describe it—"bonitinho" ("cute"). In this case, however, Fabio seemed to use the word

positively.

It was also about being a part of the Lua community, he said then. The word community suggested to me the idea of local ties. I immediately remembered Felipe's story about Alta's involvement with Rodrigo's project. Perhaps he was referring to the fact the fact that he worked on his projects together with his friends at Alta, I thought. (As Rogerio told me in a later interview, working on the project was one of the many things he and Fabio did together on the weekends.) In this case, however, it turned out that Fabio had a different community in mind: Lua's mailing list. When he first started using Lua, he explained, he started reading the Lua list and seeing the announcements of people releasing their code. This helped him realize the meaning of "open source." "It's an exchange community," explained Fabio, hence the need to contribute. But it was also a matter of personal satisfaction. In any community, said Fabio, be it your samba group or your church, you want to be known, be a member ("*integrante da comunidade*"), be some who has done something for the community. From this, you get satisfaction. When he released his editor, people replied, they said "Great," some guy wrote back and helped correct a mistake in Fabio's English used in the code, so there was feedback, and people were contributing back.[259]

"Ok," I said, "I understand the attractiveness of an open source community, but why Lua?" Fabio's answer surprised me. The Lua community offers him a space to

259 As many others, Fabio stresses the contributions by the Lua's *international community.* While the foreign members of the list contribute most of the discussion (the value of which is not to be underestimated), the majority of code is actually contributed by people in Rio de Janeiro who are almost always associated with PUC. The latter fact is routinely downplayed by the Brazilian members in their attempt to present Lua as an *international community.*

participate, he explained. Not so with Java. Consider Java, he continues. There are Java User Groups ("JUGs"), like RioJUG. RioJUG has about a thousand people on the list, but the level of discussion is very low. There are probably three or four people who actually understand the technology and who have good questions. The problem with Rio's Java list, continued Fabio, is that the number of people who understand Java is low, but the number of people who *think* they understand it is high. Why? Because Java is fashionable. The market seeks people who know Java. So the guy gets a book, copies an example, runs it—now he "knows Java." Statistically, continued Fabio, there are may be more "Java programmers" in Brazil than in the US. But the difference is in quality. "Those people here, they don't even know how to use Google!" he exclaimed. "You have to use Google. Google is *pai de todos* ['everyone's daddy']. Before asking a question you have to think: someone probably had this problem before, so go look in Google!" That's something people on the Lua list know how to do. And yes, of course you need to know English to make use of Google, otherwise those thousands of results would be of little use. And half of RioJUG people do not even know English. Fabio was on a roll, and his rant continued with increasing passion, moving on to Alta's difficulties in finding good Java programmers and their need to hire "raw guys" (*cara cru*) and teach them from scratch.

Neither Fabio's frustration with the local Java community, nor his fascination for the Lua community are unique to him. The complaint that most Brazilian "programmers" do not know what they are doing is something one can hear every day, both from the graduates of elite programs like PUC's Department of Informatics, and from those who

themselves might be the source of frustration for PUC graduates. This un-ending source of complains links to the more general frustration that the Brazilian elite feels with its *povo.* The Portuguese word *povo* is often translated into English as "the people," but this translation cannot capture the separation typically implied between the *povo* and speaker when the word is used by middle class Brazilians. While English "the people" readily brings to mind the opening line of the US Constitution, "*We*, the people," the word *povo* inevitably connotes the undereducated Brazilian masses with which the Brazil's elite is "stuck," and whose crudeness makes it hard for middle class Brazilians to present a respectable image to the rest of the world.[260]

Fabio's praise for the Lua community is also repeated by everyone who is familiar with the group. On the Brazilian side, most list members are former graduates of PUC-Rio Department of Informatics—Brazil's best computer science department. Even they, however, are eclipsed by the list's foreign members who often can quickly answer the most complicated software questions. The caliber of the foreign subscribers can probably be best explained through self-selection: in order to discover Lua, still a somewhat obscure language today and virtually unknown until a few years ago, most of them went through dozens of programming languages, driven by curiosity and a search for perfection. The active Lua users whom I interviewed in California were among the most

260 The following popular Brazilian joke expresses this sentiment. As God was creating the world, an angel was watching his work and after some time asked: "Lord, you made different countries, some, like Russia, are very cold, others, like Arabia, are too hot, you made Japan with earthquakes and China with flooding rivers, the Caribean Sea suffers from hurricanes and North America from tornadoes. Every place has its own disasters, except for Brazil, a country that seems to be perfect in all respects. Why?" To this the Almighty responded: "Just wait till you see the people that I will put in that land." The joke is so well known in Brazil that it is often just shortened to its punch-line: *espera para ver o povinho*.

technically knowledgeable people I had met in California, and my conversations with them were a fascinating tour of the history of programming languages and computing platforms.

For Brazilian engineers like Fabio, Lua can thus be a ticket into a highly exclusive international (or, one could say, *foreign*) community of developers, and an escape from the mediocrity of local groups like RioJUG. Links to PUC and an early start on Lua gives them relatively easy entry into this community. Gaining comparable standing in a different group, e.g., the Linux kernel developers, would be a lot harder. Eric Raymond (1999) talks about the "ergosphere"—the space of work—as one of the key resources in the open source community. According to Raymond, open source communities are gift cultures, where one gains status by offering gifts to the rest of the community. Open source gifts are solutions to technical problems. Much like in academic research, good problems—the ones that would result in valued gifts that are not too costly to produce—are scarce. Ties to Lua's author and Rodrigo gave PUC graduates like Fabio an opportunity to identify good problems within a small but growing community, or jump into an existing project. The results of their work thus enjoyed ample downloads (one of the key metric of success in open source) and recognition within the community.

This success in the open source community, however, is hard to translate into income in the Brazilian market. If Kepler developed more, it could potentially allow them to make a realistic proposal to their clients, said Fabio. He was confident that this will happen eventually. He talked about Rodrigo's recent idea to "open" Kepler and invite more participation from people outside (see chapter 3.4), saying that this would give

people more confidence in Kepler. It won't be a "Kepler team" anymore, but "Kepler community" behind the code, explained Fabio, saying both phrases in English. But for now, he contrasted, there was no "case" of Kepler, no large company using it. And Brazilian clients were not into experimenting. When a client asks: "Why use Java?" The answer is "Because giant companies use it." (Fabio rattles off names of a few large US companies.) Compare this with Kepler, said Fabio. "Why use Kepler? Because it's good." This just does not mean much to the clients. They would ask: "Who is Rodrigo Miranda?" I know him, says Fabio, you know him, but they don't. "Who is Márcio? Who is Tiago?" They do not know. They know IBM and SAP.

The choice between using "good" technology vs. the technology wanted by the clients is not limited to Lua. During my time in Alta in 2007, most of company's work was based on "eiWeb," a proprietary platform provided by EIT, Alta's international partner. There was a strong sense among Alta engineers that eiWeb was no longer the best option, or at least not in all cases. The developers' attention was increasingly turned to the many open source alternatives, which offered a number of advantages: they were free (leaving the customer to pay a larger portion of their budget to Alta for customization), they made it easier for the developers to fix the bugs (because the code was open and there was more free documentation online), and because open source was new and *cool*. For those reasons, Alta tried to move its clients to open source solutions when possible. With the largest clients, however, Alta did not have this luxury. Such clients typically had a prior relationship with EIT, which sold them eiWeb licenses and then offered to recommend a "solution provider"—a local software company that would customize

eiWeb for the customer's needs. Alta was one of such providers, not the only one. When a client such as "Intermercado" called Alta saying that EIT suggested them as a company that could implement an eiWeb-based system, Alta's managers had to keep to themselves their opinions about advantages and disadvantages of eiWeb. EIT then typically assigned an software architect to supervise Alta's work and to make sure that Alta was not introducing any open source solutions that would serve as alternatives to upgrading eiWeb to a newer version, at a charge. As the client contemplated whether the expensive upgrade was worth it, Alta's engineers worked with outdated technology.

Some weeks later, when I had already started visiting Alta's office daily and looking at Alta's code, I found myself wondering what version of Java Alta was using. I had not worked with Java for four years myself, but knew that Java 5 came out soon after I started my doctoral program four years earlier and that Java 6 came out in early 2006. I walked over to Fabio's desk to ask him about this. "Java FOUR," he responded emphatically, making sure to enunciate the word "four" very carefully and to load the phrase with contempt to the brim. "Why?" I asked. A guy sitting diagonally from Fabio, who I later learned was Alta's third co-founder Eduardo, swiveled in his chair, turned around and jumped into the conversation: "This is a very good question. A question we ask ourselves all the time." But the answer was simple, he explained: Intermercado was using eiWeb 8.2, which used Java 4. They had been talking about upgrading, but this would take a while. Brazilian companies have a tendency to follow the rule of "n-1," he continued. They always use next-to-last version. When they were deciding on the version of eiWeb, eiWeb 9 was in Beta. Obviously they would not use beta. So, they used eiWeb

8. At some point they will upgrade. Until then Alta was stuck working with Java 4, something it would rather not talk about in the presentation to the new employees.

Inability to use existing solutions meant having to invent workarounds. One of Fabio's projects required a method for scheduling events. Fabio knew of a good open source solution called Quartz, but its use would not be permitted by EIT's architect. A scheduling mechanism had already been implemented in eiWeb 10, and the "proper" solution involved paying for an upgrade to the new version. Fabio thus had to find a way to implement his own scheduling method by hand—a job in which he seemed to find little joy. He did not expect his solution to be as good as Quartz. The work therefore involved pointless duplication of effort, a clear violation of cultural norms of software development.

Building an Online Store

A few weeks later I met with Fabio again, at a sushi restaurant near Alta's office (other Alta developers later described it as "very expensive"). We talked more about Alta, Java, Lua and open source—and the possibility of me spending some time inside Alta. It turned out that Alta's managers had agreed to let me spend time there (being an *Herculoid* had its benefits), and Fabio was to be my contact person. One problem, said Fabio: there was no place for me to sit. The lack of space got resolved a few weeks later, however, and I finally started spending time at Alta's office.

It was an early afternoon, and I headed to Alta's kitchenette to get some coffee. I

ran there into Mauricio, one of the developers working with Fabio. “Follow me,” he said and led me to the corridor, where Fabio was smoking with a cup of coffee. We exchanged some small talk, I told an anecdote about my trip to Finland. “Now let’s talk seriously,” said Fabio then. Intermercado people got back to him about the scheduling program that Alta just delivered. They said there needs to be a way to set events that start on one day and go to the other. He already thought about it and it wasn’t so bad. It would require some changes to a database, some to the interface. He started explaining a solution. Mauricio interrupted him: “Why not just have the user enter the beginning and the duration of the event?” “Ah, good idea,” Fabio replied. He started thinking out loud, following the suggestion made by Mauricio. This will make it much easier. So, we just let them do this, and then have JavaScript to show the end time. Again, the database will need to be changed, as well as some of the user interface. Fabio listed the specific things that will need to be changed, walking through the steps that the user will have to go through. He then turned to how long the change would take. “Three days?” he asked Mauricio. “A couple of days to write, a few days to test,” Mauricio responded. “Let’s ask for five,” Fabio summed it up, “three to write, two to test.” Let’s ask for five, he explained, so that we could then agree to four, if they insist. Finishing early is ok. Better than asking for four days and then having to accept three. He asked if Mauricio could start tonight. Mauricio explained that he had an exam. (It was about 5:30 p.m., and Mauricio was doing a college program at night.) Ok, said Fabio, let’s start tomorrow.

As we walked back to the kitchen, I started thinking of asking Fabio if this was something I could help with. I had been at Alta for a week at this point and had found it

hard to get to the details of what people do without getting involved in a project. Fabio anticipated my question. “How much memory does your laptop have?” he asked. “1 gigabyte,” I said. “That’s not enough to run eiWeb,” he sighed. We agreed that I would help with the user interface, which would not require running eiWeb. We went to Fabio’s desk and he showed me the application, then sent me the URL and the password. I spent a few hours playing with JavaScript for the new form. The next day, however Fabio informed me that the issue had turned out to be much more complicated, and in fact the requirements for the project were being re-thought altogether. Instead, he said, I could join him on another project that they were just starting: an e-commerce website for a different client.

Early next week I was sitting next to Fabio at his desk, watching him draw a diagram, which represented the relations between the “business objects” of client’s online store, and was to serve as a foundation for a database. (At this point we had agreed that I would later help with the database design.) As Fabio explained, they had done design “by hand” in the past, but he now wanted to try doing it with a proper tool. As I soon realized, Fabio was using not just a diagramming tool but a UML editor—a tool specifically designed for expressing relationships between software objects, which could later be used to generate code and database design.

Fabio was working without too much haste, explaining to me along the way what he was doing. I was surprised that he appeared to be coming up with the various properties of the objects without having to look them up anywhere or even to pause for thinking. I watched as he created a box for “Product” and then typed in the fields that a

"Product" is supposed to have. He then created another box, labeling it "SKU"—the English abbreviation for "Stock Keeping Unit" as I later learned. I thought about the "requirements document" that Fabio had sent me the day before, realizing that it did not have nearly enough detail about what the customer wanted. I asked Fabio how he knew what "business objects" the store needs? *Those are the same for all stores!* he laughed. They had been working with online stores for quite some time, he explained. This particular client did not even have anything specific in mind. They had only a vague idea of what a web store would be like. All they had asked for, explained Fabio, was for Alta to build them "an e-commerce website," and that this website would be no worse than their competitor's, plus a few extra things. As I realized later, the request for "a few extra things" did not refer to anything specific either—rather, the client just wanted their website to have *something* that their competitor's site did not have, leaving it to Fabio to figure out what that something could be. Fabio approached the task by looking at their competitor's site to see what it did and thinking what they would need to do to allow for the same features and a little more. He then had a long meeting with the client, telling them what his team could and could not offer. Fabio continued adding boxes and attributes as he spoke—the task seemed to consume little attention. At one point he paused, to think about what to do with a particular property. "Let's see what we did for Intermercado," he said. He opened the source code for Intermercado's store, looked at it, then entered the same property name into the new diagram.

Fabio's work on the diagram seemed to require little creativity, but it did require work, much of which went into aligning boxes. I asked him why it was worth doing.

“Because I will only do this once,” he responded. “Won’t it change?” I asked. “It will, but it won’t. You know?” As I understood, the design was going to change as the project develops, but the diagrams would not have to be updated. The diagrams are pretty pictures to be turned in to the client, rather than something to really guide the development. They could be used to generate the first version of the code, which would be the starting point for the implementation, but once generated, the code will then evolve on its own.

A few minutes later the Internet connection suddenly went down. The realization that the connection was gone spread quickly through the room and more and more people joined in humorous “Eeeeee!,” which then gave way to laugher and jokes. It turned out that the power would need to be turned on and off for the whole office in order to restore the connection; everyone started shutting down their computers. People started moving around, talking, joking. Many went to the common area, taking seats on the couch and bean bags. We chatted about random things. The system administrator had already flipped the power switches and the Internet was back, but the people were still talking. “The Internet is back up, you know,” said Felipe, the founder. His tone sounded half-jokingly apologetic: he seemed to understand that acting as an authority figure and calling on the employees to get back to work would sound funny in the midst of this free-spirited moment that seemed to show Alta’s startup culture at its high point. Indeed, the comment just drew laughter. Nobody hurried to get up.

I did not return to Alta next day, having scheduled some interviews about Lua. When I arrived the day after, Fabio told me he had a new idea about what I could do.

There was a new open source thing called “Spring Web Flow,” he explained, built on top of Spring, a Java framework that seemed to be on everyone’s lips at the time. Fabio wanted me to try using Spring Web Flow to implement a shopping cart. I asked him if he had used it before. No, said Fabio. He hadn’t. He wanted someone to try it and to build a proof of concept—a demo that showed that this was possible. And he thought that I could take this task.

I was flattered at getting recruited into Alta’s research-and-development efforts, but soon realized that Fabio likely picked the task because it was most appropriate for an unreliable worker such as myself. (A few days ago, while discussing my potential involvement with Alta’s projects, Fabio told me he would need to know on what days and what hours I would be in the office. I told him frankly about my time constraints and the need to do some more interviews outside Alta, which seemed to leave him uneasy.) The new task was something I could do at my own pace. And if I were to give up on the task without completing it (as I eventually did), this would not impact the schedules for Alta’s projects.

I returned to my laptop to try to get started with Spring Web Flow. I soon realized, however, that before I could get any where, I would need to do a lot of basic installation. I went back to Fabio, to ask him what Alta used as an implementation of J2EE. “Use Jetty,” Fabio told me, explaining that this was a new implementation that was much faster than the alternatives. I went back to my seat, googled for Jetty and installed it following instructions I found on the web. I then installed Eclipse, a Java development environment that I knew everyone at Alta was using. I realized that there had to be a way of starting

Jetty from Eclipse, I returned to Fabio's desk for further instructions. He told me to install Jetty Launcher from inside Eclipse. Seeing that I looked lost, another developer offered to show me. Here, he said, traversing Eclipse menus: you install it, then you go here, you put your Jetty Path, then click here, then it runs.

I returned to my desk trying to reproduce what I saw. I did not get very far: my Jetty Launcher and my Eclipse did not seem to like each other. As I understood, Jetty Launcher would not work with the most recent version of Jetty. I returned to Fabio several times with questions. At one point, Leonardo jumped in. Yes, he said, Jetty Launcher requires Jetty 5 and it would not work with Jetty 6. I mentioned that I saw a discussion of this on the web and that someone has offered a patch: a set of changes to Jetty Launcher that made it work with Jetty. I was hoping that Leonardo would tell me if this method worked, but his response was disappointing. He had read about the patch but had not tried it; he was still using Jetty 5. He encouraged me to keep trying with Jetty 6, however, and to tell him if I managed to get it working. Someone had to be the first to use it, so it could as well be me.

I returned to my desk, spending more time reading documentation and forum posts, eventually deciding to give up on Jetty 6 and to install the same version as everyone else was using. I could finally start Jetty from Eclipse, but I now needed to build a trivial application. I tried sample applications from Jetty, Spring, and other projects, but none worked. There were too many new applications involved, and it was impossible to tell which was causing problems. As I headed out for the day, I stopped by Fabio's desk to discuss the matter briefly. Leonardo overheard us again. "Use struts-blank," he said.

I returned to my task the next morning. Using struts-blank resolved a round of problems, but introduced me to the next set. I spent more time reading what I could find on the web and approaching the issue from different angles. I was finding lots of relevant documents, though few were useful. Spring documentation assumed too much, without making clear what one needed to run it. Jetty had documentation for version 6, but not for version 5. Jetty Launcher was missing documentation all together. I returned to Fabio with questions several times, at one point asking him if he had actually ever gotten all of those pieces working together. "No," he responded. "The whole point is to get it working."

After another hands-on demonstration, and another failed attempt at replication, I realized that I was using a different version of Eclipse from everyone else and downloaded another version. Everything was almost working—there seemed to be just one file missing. I returned to Fabio: "What Java are you using?" He gave me a funny look. I insisted on him telling me exactly where he downloaded Java from. "I just do 'sudo aptitude install java'" said Fabio. I returned to my laptop, and typed the four words at the Linux command prompt. I suddenly had the right version of Java and my simple project was finally working. I declared my work for the day done and went to the sofa to work on my field notes.[261]

As this episode illustrates, software work requires a peculiar fusion of globalized and localized activities. Much of that work involves interaction with software developed

261 This was far from the task Fabio originally proposed (implementing a shopping cart using Spring Web Flow), which was never completed, as I realized that it was going to take a lot more time than I expected and that my (and Fabio's!) time would be better spent if I limited myself to watching others work.

quite far away and documented in bits and pieces around the Internet.[262] The software and the documentation are quite mobile. In theory, anyone with an Internet connection can download and use them. Furthermore, downloading software never used before in the local context and getting it to work by following documentation on the Internet is part and parcel of software development. In my case, getting the components to work together, was "the whole point," as Fabio pointed out. A software developer who cannot use the Internet to find out how to solve a problem that is new to his colleagues is of little use to a company like Alta. A developer who always approaches others with questions without first making a *bona fide* attempt to find the answers online would be testing the patience of his colleagues. (My position in the company seemed to allow me to get away with a lot more questions than other developers could afford to ask.) On the other hand, the task of getting downloaded software to work to practice benefits dramatically from proximity to people who have worked with it before. A single phrase uttered by a colleague can substitute for hours of Internet search and trial and error, stress many of my interviewees. Software development consequently becomes an intensely local affair.

As the anecdote shows, the power of local advice and the Internet documents both have much to do with the shared context of work. Fabio's four words "sudo aptitude install java" brought my laptop in sync with all the Alta's computers, making sure that the steps that had worked for Fabio and Leonardo would work for me as well. This

262 Ubuntu, Java, Spring, and Jetty were developed by companies headquartered in London, San Jose, San Mateo, and Los Angeles respectively. (Jetty was originally developed by an Australian company, but is now maintained by a California-based company.) Eclipse is a widely distributed effort, based on a software that originated in an IBM lab in North Caroline. It is now coordinated by a foundation with offices in Ottawa and Portland. Jetty Launcher was developed by a programmer in Ottawa.

synchronization was possible, however, because Alta's machines, configured in Brazil, and my laptop, configured in San Francisco, were already running essentially the same software, Ubuntu Linux 6.10. Continuing synchronization of practice was much simplified by the extent to which the context had been synchronized through earlier work, a long process the beginning of which was described in chapter 2.2.

Being Local

Around 6 p.m., Eduardo started gathering people. They would be having a cake, he explained to me. It was a company tradition: once a month they got a cake and congratulated all the people who had birthdays that month. This time it was just Leonardo, a recent PUC graduate who had been transitioning to management and was the most recent minority partner. Everyone gathered in the conference room. There were two trays of snacks, a cake and several large bottles of coke. Eduardo suggested that we should do introductions for new people. The first of the new people mentioned that he lived in Niteroi—a city across the bay from Rio, from where a large number of Alta's employees commuted. After that each of the new people was asked to say whether they thought Niteroi was a better city than Rio. When my turn came, I introduced myself but dodged the Rio-Niteroi question. "And what about Niteroi?" several people demanded. I gave a vague response, hinting at a preference for Rio. Fabio, a native of Niteroi, aimed a bottle cap at me, as if ready to throw it I were to side with Rio. Amid loud demands for me to take a clear stand on this important issue, I ended my presentation vaguely,

unwilling to step on the neighborhood sensibilities of Alta's global IT professionals, skipping the opportunity to act as a foreign judge of this local rivalry. "Vaseline," snickered Fabio as he put down the bottle cap.

Leonardo started cutting the cake. "Who are you going to give the first piece to?" asked several people. There was some suspense. "I will give the first two pieces at the same time," said Leonardo. He cut two pieces. "Those are for my team," he said, giving them to the two developers who worked under his supervision. He then cut a piece for Eduardo, who was slouching in a chair, looking over the team as a patriarch. He owed everything to Eduardo, explained Leonardo, exaggerating the tone and making a joke out of his public acceptance of Eduardo's authority. The next piece went to Felipe, another co-founder. Luis, the third of the original partners was not there, so there was again suspense as to who is getting the next piece. It went to Fabio. The move caused a murmur. Eduardo and Felipe were unambiguously the bosses of the company. Fabio and Leonardo, on the other hand, were both recent minority partners. Leonardo's move thus appeared acknowledge Fabio's status, while also highlighting the difference between minority partners and everyone else. Leonardo laughed as he gave Fabio his piece, then put down the knife: others could cut their pieces by themselves. Startup spirit aside, Alta did have founders, minority partners and general employees. Fabio and Leonardo had to learn to manage their new status *vis-à-vis* others.

Culture and Ties

During one of my last weeks in Rio in 2007 I met with Rodrigo Miranda at a café in Copacabana. He had agreed to read a paper I had written for the American Sociological Association and to give me some comments. One of his comments concerned my discussion of the developers' ties to the foreign centers of the practice. Rodrigo suggested that "building ties" was perhaps too strong of a phrase. Most people adopted foreign technology and got quite good at it, explained Rodrigo. They learned to "walk the walk." Many also learned to "talk the talk," adopting the foreign software culture. The latter was harder, Rodrigo argued, because you could not read about it in just one place. It is also more important, since once you had it, people could see it in your eyes that you lived technology. But all this has little to do with building actual social ties to foreign communities, he continued. He had been trying to do it with Kepler and finding it extremely difficult. Most people never try. Look at Alta, he said. They adopted the culture, but without the social ties. Their clients are and may always be local.

Rodrigo was right to some extent. Alta was and continues to be an intensely local affair. The company uses foreign technology and much of foreign culture, without much direct contact with the foreign centers. In contrast to Kepler and Fabio's Lua projects, carried out on the side for "fun," Alta's main line of work is local, both in its location (involving little interaction with people outside Brazil), and in its significance. Such local work is often boring and brings the developers limited cultural dividends in the larger world of software practice. In many ways it comes down to sales engineering: doing what

must be done locally to allow a Brazilian company like Intermercado to use the software supplied by EIT.

The local focus of Alta's work, however, is also a source of strength. In addition to making a profit, Alta builds IT solutions that are actually used by many people. After my departure in 2007, the company proceeded to strengthen its relationship with Intermercado, eventually winning the bid to write software that now controls the front page of one the most popular online stores in Brazil. The months that followed were a period of much work and much learning, Fabio told me when I returned in 2008. He and other Alta's engineers had to learn to build a web application that could handle traffic never faced by any of their earlier applications—or by any software based on Kepler, I can add. (Another local contractor of Intermercado, obliged by Intermercado to work together with Fabio's team, was instrumental in this learning.) Local focus was bringing Alta to projects the scale of which made them exciting and a great source of war stories. In the same months, Fabio also finally stopped participating in Lua-related projects. There just was no time for such games.

3.2 Porting Lua

In 1993 a group of computer scientists working in a university in Rio de Janeiro developed a simple programming language called "Lua" to serve the needs of a Brazilian company based in the same city. Fifteen years later Lua is often ranked among twenty of world's most popular programming languages[263] (out of thousands) and has a user community spanning all five continents. While Lua has brought its authors rather modest financial rewards (it is distributed for free and brings little consulting revenue) its use in popular software such as Adobe Lightroom and World of Warcraft has made it in some ways one of the most successful software products ever developed in Latin America.

American users of Lua often credit it with being highly *portable*—Lua can run on many different computing platforms. While increased portability in this narrow technical sense is an important part of Lua's story, I focus here on a different kind of "portability": Lua's gradual transformation from a highly local project to an international programming language that betrays little connection to the city where it was developed and where it is still based. I organize my discussion around Giddens's (1990) notion of "disembedding"—the "lifting out" of social (or in our case socio-technical) relations from their local context, which then makes them mobile across time and space. Following Lua's transition from a highly embedded project, developed as a solution for a specific set of problems, entangled in a web of local relations, goals and commitments and reliant on what we may

263 For instance, Lua was ranked between fifteenth and twentieth in TIOBE TPCI index for most of 2007 and 2008, dipping to the twenty-second position in December of 2008.

call “tacit knowledge,” to an international programming language we observe the different mechanisms that enabled and facilitated this disembedding.

As we will see, Lua’s international success was not planned in advance. To a large extent the disembedding of Lua simply “happened,” in many ways without a conscious intention by its authors. It happened due to numerous decisions that most participants saw as quite natural. In some of the cases, acting otherwise, for example using Portuguese words as Lua’s keywords, would be nothing short of “ridiculous” according to some of my interviewees. It is important, however, to look closely at such “obvious” decisions. It is by understanding how such decisions come to be obvious, and why they are obvious to some and not others, that we can come to see the geographic logic woven into the professional culture of software development.

The story of disembedding told in this chapter complements the investigation of local re-assembly of a foreign practice presented in chapters 2.2 (the larger history of informatics in Brazil) and 3.1 (a look at a particular company dedicated to bringing foreign technology to local clients). After decades of work that helped establish the foundation of software practice in Brazil, the context was created that made it possible for some of the practitioners to engage in one of the most central roles in the world of software: authoring a programming language. This replicated context, however, is not identical to the original and is characterized by a distinct pattern of connections. Brazilian academic computer science has strong connections to foreign computer science, which gave Lua’s authors a good start in cultural capital. At the same time, much unlike American computer science, it is relatively isolated from both local and foreign computer

industry and instead exists in a somewhat of an enclave, which makes the experience of Lua's authors quite different from those working for Alta (chapter 3.1).

I turn to the limitations of this process of disembedding in the next chapter, looking at Lua's changing relationship with the university, city and country where it was born. Fifteen years into its history, Lua appears to be less known in Brazil than in the United States. Few Brazilian companies use Lua in their products. Lua's lively online community interacts exclusively in English. *Programming in Lua*, written by Lua's author Roberto Ierusalimschy in English, is yet to be translated into Portuguese. (German, Korean and Mandarin translations are available as of late 2008. With the exception of one book written in Japanese, all other books dedicated to Lua or covering it in substantial depth are available only in English.[264]) In addition to having to learn Lua from a book written in a foreign language, a potential Brazilian Lua programmer would likely have to order the book online through Amazon.com and have it shipped from the United States—*Programming in Lua* is not sold in Brazil, except in the author's office. In the end of that chapter, I turn to the difficult question of whether Lua's success can be of any benefit to the city and country where it was developed.

The two chapters are based on around forty interviews with people involved with Lua or related projects, complemented by my analysis of the archives of the Lua mailing list (Lua-L), Lua publications and the historic Lua code. Eight of the interviewees were

264 Jung & Brown (2007) is another introductory text in English, while Ueno (2007) provides an introduction to Lua in Japanese. Schuytema & Manyen (2005) focuses on use of Lua for computer game development. Gilbert & Whitehead (2007) and Whitehead et al. (2008) discuss using Lua to customize World of Warcraft. Varanese (2002), Gutschmidt (2003), Buckland (2004), and Dickheiser (2006) discuss computer game development more broadly, but dedicate a substantial attention to Lua.

Lua users residing outside Brazil, five of whom were interviewed in person in California. Most of the remaining participants were interviewed in person in Rio de Janeiro. (One was interviewed by phone.) As in other chapters, I try to minimize the number of people who I quote to make it easier for the reader to keep track of the characters.

Those two chapters present a largely self-contained account of Lua's history, while avoiding repeating material presented in earlier chapters (in particular in chapters 1.2 and 3.1). Readers interested in additional historical details can find them in chapter 1.2, while those interested in additional perspectives on Lua's place in Rio de Janeiro today should look at chapter 3.1 (as well as chapter 3.4).

Choosing Lua

"Craig" is an engineer at a small startup in California developing an online computer game. Like most other users of Lua whom I interviewed in California, Craig encountered Lua online, while searching for a scripting language to embed[265] in his application, coming across Lua "through [online] forums, googling and searching for ideas." None of the people he knew personally had heard of Lua before. Craig cites Lua's small size and simplicity as the reasons for choosing it over a "more mature" language such as Python. He was particular concerned with security of his application and felt that "a sort of more mature language like Python" was too complex for him to be sure that his application would be safe from malicious hackers. Craig notes to an important weakness

265 Underlined terms and abbreviations are defined in the glossary, which also includes some terms not marked in the text.

of Lua—the relative scarcity of libraries, a hurdle faced by all new languages. This factor, however, was of little relevance to him. “We were not building an application, like a web-server or something else that would need a whole bunch of specialized libraries,” says Craig. “We needed a language to primarily reference the objects inside our own system and to be able to script them.” (In chapter 3.4 we will look at Kepler—a project aiming to turn Lua into a platform for web development, exactly the thing Craig thinks Lua would be least suitable for.)

Craig’s use of Lua is quite typical: most Lua users in the United States employ Lua for scripting applications written in C, a programming language developed in the early 1970s, which had come to dominate software development by the early 1990s. Over the last decade, many developers have moved to newer languages such as Java, Python or more recently Ruby, which make it easier to develop software quickly. C and its close relative C++ remain popular, however, as they often make it possible to write software that runs faster. While most of the new languages can in theory be combined with C in a single application, potentially allowing the developer to get the best of both worlds, such usage is often complicated and frowned upon. For example, the authors of Java, a highly popular programming language designed by Sun Microsystems, pursued a targeted campaign to eradicate such mixed applications, encouraging the programmers to write their code in “100% pure Java.” Lua on the other hand, presents itself as the language primarily designed for hybrid use.

Lua has become particularly popular in the development of computer games, where efficient use of computer hardware is often crucial and the developers frequently

work closely with C modules for handling graphics. Lua allows such developers to use "easy" Lua for parts of the code that are likely to change often, while relying on the more efficient C for the tasks that are most likely to put strain on computer's resources. Lua's thus strives in a relatively small niche, where it has positioned itself as a complement to a well-accepted technology, offering certain unique features that shield it from devastating competition with the "more mature" languages such as Python. Developers like Craig note and appreciate Lua's close fit for the needs of the computer graphics developments.

Like other interviewees, Craig found it quite easy to get started with Lua. My question about how he learned to use Lua and what kind of resources he used takes him by surprise. "I am sure I downloaded everything, ran the command line, found out how that works," he says after a pause. Another interviewee "Steve," a lead engineer for a team of software developers working for a large software company in California, reports similar ease, which he then contrasts with JavaScript, a programming language developed by Netscape that my interviewees often consider as an alternative to Lua. He has not needed to look for people who understand Lua, says Steve. Had he decided to use JavaScript instead, Steve would have gone and talked to people in his company who had a JavaScript implementation. "But that's because JavaScript is messy," he explains.[266] "The great thing about Lua is that you don't need any of that." Lua's elegant simplicity makes its foreign origin irrelevant—Steve and Craig can use Lua even if nobody else in

266 JavaScript is similarly designed for embedding and is widely used due to the fact that its implementation is included with all modern web-browsers. Unlike, Lua, however, JavaScript does not have a standard implementation. For a long time no free implementation of JavaScript was available at all. At the moment, several such implementations are available but the most popular of them (Mozilla's Spidermonkey) is typically found by my interviewees to be overly complex. (Spidermonkey source code includes around 140,000 lines—eight times more than Lua's 17,000. Lua's source includes the code for an interactive shell and builds with a single command in just a few seconds.)

California does so.

Like Steve, Craig did not feel constrained by his lack of contact with other Lua users. Just as he was starting working with Lua in the summer of 2005, however, a Lua Workshop was organized at the Adobe office in San Jose, about twenty miles south of where Craig's startup is located. (Adobe itself was extensively using Lua in one of its projects—Adobe Photoshop Lightroom—which was released two years later.) The talk included presentations by two of Lua's authors and Craig attended a part of it to see if Lua was "serious":

> Craig: Well, my reason to go was to get some sense of how serious this is. To ask a few questions. To talk with some people about it. Curiosity. More and more now I am very confident about our choice but [unclear]. Who are the other people who use it and why do they use it? [...] I enjoyed it. I think I only came for half a day. I think it was a multi-day event. It was very nice.

Seeing live users of Lua helped Craig feel more confident about his choice. When I ask him whether we was looking to make contact with local users of Lua, however, Craig tells me this was not his objective:

> Craig: I don't know if I met any *local* users. I met *people*, I didn't... My interest was not in establishing long-term relations with people there, so I didn't find out where they lived or if they were local.
>
> Yuri: Why?
>
> Craig: Because I am busy. I am getting plenty of information and valuable help through the mailing list. So... That's not as important.

Craig finds that he can easily use Lua in the absence of any local community.[267]

267 One might suspect that Craig relies on the community of Lua users inside his own startup. This is only true to an extent—while a small number of engineers in his company write Lua code, he is the only person involved with the more complicated task of linking Lua with C.

A Global Perspective

All American users of Lua to whom I talked say that Lua's Brazilian origin was nearly irrelevant to them. "I take a global perspective on those things," says Rich, a software contractor who uses Lua in his projects when he has a chance. Taking a global perspective on Lua luckily takes little effort.

Like most interviewed users, Craig cannot read Portuguese (though he says he speaks German fluently and could maybe "make out" text in French). This of course does not present a problem for his use of Lua, since the Lua community interacts in English:

> Craig: I guess English language is the *lingua franca* for Lua as well. From what I can tell. I haven't seen any Brazilian, or Portuguese, emails coming up. So... So, I never had any... Or, I should say I've never had any concerns...

Emails in Portuguese *do* occasionally come up on the Lua mailing list, and are typically treated politely, usually receiving a reply in English (sometimes quoting the original question run through an online translator) and occasionally even in Portuguese. Displaying a global perspective simply requires treating such infrequent occurrences with humor. The discussion on Lua list show that some of the list members have certain curiosity about Lua's unusual origin and sometimes allow themselves friendly questions about Brazilian practices that they find surprising—for example the use of a pair of brackets to close email messages.[268] Those who want, however, can simply ignore such

268 On one occasion, a Brazilian member of the list was asked to explain his use of "[]s," in the end of the message: "Does this mean something? Some people sign their mails 'thanks' or 'regards' or 'good day'. You're the first I've seen sign 'boxes'." ("[]" is used in Brazil as a shorthand for "abraços" —"hugs.")

cultural differences. Most of the translation work is already done by the Brazilian members of the list.

Perhaps the only way that Lua's foreign base entered into Craig's calculation, is because of the increased difficulty of understanding how "academic" Lua was. Craig knew that Lua was produced by university researchers and was worried that it was "an academic exercise":

> Craig: My reaction was more... I don't know if it had to do with Brazil or not, my question was "Hmm..." Actually, I think at the same moment I realized that it sounded like it started out as an academic exercise, I think I read this there as well. And of course the question is: What are the primary interests of the caretakers of the language? Is it to satisfy their academic ambitions? And I don't mean that in... academic ambitions not as in a career but as in academia. Is it a language to highlight their work on the programming language theory? Or is it something for practical use?

Making this evaluation would have been somewhat easier in case of an American university, Craig explains, since he could meet the authors or would perhaps "know somebody who knows somebody who knows them":

> Yuri: This question of whether it's an academic exercise, how does it relate to it being in Brazil?
>
> Craig: Perhaps it is harder, it might be harder for me to determine than if it were for example at Stanford, then I could... A) It would be easier to meet the people or I would know somebody who knows somebody who knows them or something like that. So, in this context it is certainly more opaque.

While understanding whether Lua is "academic" takes Craig somewhat more effort, he ultimately finds other ways to understand the intentions of Lua's authors: reading about the language, using it, later attending the workshop, learning that Lua was used in World

or Warcraft, one of the most successful games at the time.

Other interviewees similarly express the desire to understand the people behind the language as a way of learning where the language may be going in future. Some mention that not being able to see the authors speak (at least on video) makes it harder to make this judgment. For many, however, Roberto Ierusalimschy's book provides enough answers. For instance, Rich, who started using Lua several years before having a chance to see Roberto live (at the same workshop as Craig) says:

> Rich: Like Perl, the book was written by the architect of the language. *Programming in Perl* is written by Larry Wall. And the Lua book was written by Roberto. So, by reading the book you get the sense of both the designer's personality and the language itself. And so just reading the book it was pretty clear that the guy who made this language was a really smart guy, and he valued principles that I valued: simplicity, elegance. And at that point it doesn't really matter where he lives or what nationality he is. He is just a smart guy who made a good language, that's all that matters to me. If he wasn't able to speak English it probably would have been a problem. But obviously there wasn't any communication barrier. So the fact that it was made in Brazil wasn't anything to me.

Lua authors' ability to successfully communicate (in English) their commitment to the principles that the potential users share makes Lua's geographic origin irrelevant as far as most of the foreign users are concerned. American users of Lua also find that they can understand Lua by placing it in the larger genealogy of programming languages, most of which were developed either in the United States or with strong involvement of American computer scientists. "Lua's syntax is based on Algol and Algol's syntax is based on algebra" says Rich. Anyone who is familiar with algebra has already made many steps towards learning Lua.

Steve draws a somewhat different genealogy, describing Lua as "Lisp-like." Some aspects of Lua's design closely resemble Lisp, an influential programming language developed at MIT in 1960s, which is rarely used today but is often hailed as an example of good programming language design.[269] Describing Lua as "Lisp-like" presents Lua as a descendant of a language that all programmers are expected to respect, giving Lua substantial gravitas. Others describe Lua by producing a list of Lua's features. "Suffice it to say," says a recent article on Lua, "that Lua is an elegant, easy-to-learn language with a mostly procedural syntax, featuring automatic memory management, full lexical scoping, closures, iterators, coroutines, proper tail calls, and extremely practical data-handling using associative arrays" (Hirschi 2007). Such a description again helps make Lua's geographic origin irrelevant by placing the language within a classification system developed primarily by American computer scientists.

The classification system that makes it possible to describe Lua using a list of eight concepts, as Hirschi does in the quote above, and the larger system of meaning to which it is linked provide an important disembedding mechanism. A programming language that is designed in such a way that it can be explained in the terms of this shared system is relatively free to travel. Such academic sophistication is not expected of all programming languages—some of them are notorious for being "messy" and "ugly" and become wide-spread primarily through a strong association with a powerful actor. For

269 Lisp occupies a paradoxical position in the world of programming languages: a language often admired but rarely used. Lisp is typically seen by programmers as elegant but hard to learn, requiring a highly abstract and somewhat unintuitive approach to programming. Fans of Lua often point out that Lua implements some of Lisp's ideas while also allowing a more intuitive style of programming used by other popular programming languages. (This more popular style is what Rich calls "Algol syntax.")

Lua, however, the academic credentials become crucial. (It should be noted that not all programmers would be expected to understand what all of those terms mean. "Coroutines," in particular, are a relatively rare concept outside of academic computer science. A description of Lua in terms of coroutines and tail calls may thus do more to legitimate the language than to explain it to a typical programmer.)

Users of Lua often also point to another important disembedding mechanism: Lua's internationalized credentials, typically expressed in the form "Lua is used by X," where X is a major company such as Microsoft or Adobe, or "Y was written in Lua," where Y is an internationally known software product, such as World of Warcraft or Adobe Lightroom. Steve, for example, says that his choice of Lua of course attracted questions within the company. "What is this? Where does this come from?" asked his colleagues. Steve replied that Microsoft had already shipped products with Lua, so they would not be the first big company using it.[270] "Used by Microsoft" can be understood as an instance of what Giddens (1990) calls "symbolic tokens." Symbolic tokens represent known units of value recognized across space and act as an important disembedding mechanism, by removing the need for situated trust. "Used by Microsoft" deflects the original question about Lua's origin ("Where does this come from?") and whether its authors can be trusted. If major companies use Lua, where it comes from and who wrote it is a lot less relevant. (Of course, what is important is not the actual use of Lua by such companies, but rather public *knowledge* of such use. We will return to this distinction

270 This was somewhat of an exaggeration, he tells me, since it was actually Microsoft Games that used Lua (in "Baldur's Gate"). Still, Steve explains, Microsoft shipped something with Lua, so, there were "respectable, known companies" using the language.

later.)

Legacy Stuff

Over its history, Lua has undergone some substantial modifications, breaking backwards compatibility many times. Craig remembers this issue being discussed at the workshop that he attended. As a new user, he had little interest in the topic:

> Craig: During the meeting a lot of people were worrying about legacy stuff, and Roberto was saying "Here is the new..."—Roberto, right? —"Ok, here are the new things in Lua 5" or whatever, 5.1. And then there were all of those people, "Oh, will this break compatibility? Will this break compatibility?" And in my case, I was just like: "Oh, I don't care, please break compatibility, make it good for me!"

Existence of old code ("legacy stuff") creates a serious problem in software development. When programming languages and software libraries are put to use, their limitations eventually become clear. While those limitations can often be overcome by additional code, such incremental additions lead to increasingly "ugly" and "ad hoc" design. The authors face the temptation of rethinking their design and "cleaning it up." To be truly effective, such "cleaning up" often requires radical changes, which would make the new version incompatible with code written earlier. Changing old code to work with the new version of the programming language or a library often requires a lot of work and introduces new opportunity for bugs. It is therefore not taken lightly. (Many companies still use software written in 1960s in COBOL.) Avoiding the need for such changes is called "backwards compatibility." The need for backwards compatibility typically impedes the evolution of a language or a library and leads to increase in size of the code

("bloat") and unnecessary complexity ("cruft").

While other Lua users in California express somewhat more interest in backwards compatibility than Craig, most consider Lua's willingness to break with the past as a major source of strength, comparing Lua explicitly with JavaScript, a language embedded into most modern web-browsers.[271] Lua devotees sometimes describe JavaScript as "nasty," comparing it to Lua's "elegance." Some point out, however, that this "nastiness" is very much connected with JavaScript's wide-spread adoption:

> Rich: JavaScript suffered from premature standardization. There is this web-browser, the web blew up, so everybody is using JavaScript. And then they thought: "Oh, we have to standardize this, make it interoperable." And the language hadn't really stabilized at that point. So there is a lot of cruft and nasty things. Both in JavaScript, that is the language itself, such as the "with" operator, and especially in the object libraries. [...] *Because* it was standardized so early, *because* it had a huge community. Whereas Lua didn't really have those forces. It had a small community and less momentum. So the designers could, when they realized they made a mistake, throw it out, unify the concepts under a different way of thinking. The different abstractions that went from Lua 3 to 4 to 5. So, JavaScript *could* have been as good of a language as Lua, I think, if it hadn't this clutter on, by the huge community, a huge user base.

For Rich, Lua is simple and elegant because it is not weighed down by commitments to any existing body of code. Not having such commitments is, however, a choice that Lua's designers made, as they broke compatibility with Lua's original applications. Comparing Lua and JavaScript, we should consider that while popular web browsers prevent JavaScript from achieving Lua's elegance, it is those browsers that make JavaScript relevant and cutting the links with those browsers would undermine JavaScript's

271 For additional information, see the glossary entry for "JavaScript."

popularity. Lua's early commitments were to custom software written for a company located in Rio de Janeiro. Most Lua's users agree that leaving such "legacy" behind was crucial for Lua's international success. As we will see, some users in Rio de Janeiro had to pay for this luxury.

DEL, SOL, LUA

If we went back in time to the early 1990s, when the first version of Lua was created, we would find a highly embedded project, tied to local goals, relationships and commitments. Had Craig tried to use an early version of Lua, he would have likely found it a daunting task, facing a piece of code written for a specific purpose (quite unlike his), offering him little in terms of social support and no justification if his choice of the language were to be questioned. At the same time, as we will see, from the earliest days of Lua, the authors had made a number of choices that made future disembedding easier.

In 1992, Roberto Ierusalimschy returned to Rio de Janeiro from a one year post-doc at the University of Waterloo, in Canada, and started working as an assistant professor at PUC-Rio, a private catholic university that is home to the most prestigious computer science (*informática*) program in Brazil. A native of Rio, Roberto had earlier completed Bachelor's, Master's and Ph.D. at PUC. As we saw in chapter 2.2, PUC had become a key center of computer science in Brazil after a long and curious history of "bootstrapping," when pioneers of Brazilian computer science worked to assemble the practice of computer science research out of whatever elements could be found. By the

late 1980s, however, the department was well established and had been granting doctoral degrees for a decade. Roberto followed up his PUC studies with a post-doc at University of Waterloo—the department that had long served as the North American counterpart for PUC's Department of Informatics and had been instrumental in helping develop computer science research in Rio de Janeiro.

In addition to his job as a professor, and his academic research in programming languages (an experimental language that "completely failed," according to him), Roberto started doing consulting work at a PUC-based consulting venture called Tecgraf.[272] In the early 1990s, Tecgraf was a fairly small group of PUC students and professors offering IT consulting services to a number of organizations, including Petrobras—Brazil's main oil company and one the country's largest corporations. Tecgraf was founded in 1987 in partnership with Petrobras, at the time when Brazil was pursuing the policy of self-sufficiency in computer technology and software (see chapter 2.2) and Petrobras had to rely on local consultants for its computing needs.[273] Petrobras was an unusual client—a semi-public company responsible for reducing Brazil's dependence on foreign oil by developing capacity to extract deep sea oil off the coast of Brazil (and especially the areas surrounding Rio de Janeiro). Petrobras thus faced substantial technological challenges and was an important consumer of scientific expertise.

---

272 At the time and throughout the 1990s Tecgraf spelled its name as "TeCGraf," highlighting the fact that the name was an acronym for "Tecnologia em Computação Gráfica" ("Computer graphics technology"). The transition from "TeCGraf" to "Tecgraf" happened some time between 2001 and 2003. In this chapter, I use the new spelling throughout.

273 "Tecgraf's clients could not afford, either politically or financially, to buy customized software from abroad: by the market reserve rules, they would have to go through a complicated bureaucratic process to prove that their needs could not be met by Brazilian companies" (Ierusalimschy et al., 2007).

Also at Tecgraf was Luiz Henrique de Figueiredo who had also just recently received his Ph.D. in Rio de Janeiro at the Institute for Applied and Pure Math (IMPA) and was employed by Tecgraf full time. Also a native of Rio and a graduate of PUC-Rio, Luiz Henrique had earlier spent three years pursuing a Ph.D. in England and another year working at the University of Waterloo. Luiz Henrique was trained as a mathematician, thus benefiting from a different and considerably longer history of efforts to transplant a scientific practice in Brazil. From his early days as an undergraduate at PUC, however, Luiz Henrique had been interested in computing. (PUC did not offer an undergraduate computer science course at the time.) After returning from England, Luiz Henrique later spent a year in Canada, working at University of Waterloo's Computer Systems Group. He focused his doctoral research on computer graphics, while working as a software developer at Tecgraf.

In 1992, Luiz Henrique turned to the problem of providing a unified way of configuring graphic interfaces for large number of software applications that Petrobras used for simulations related to oil extraction. "These were huge programs, that were very old and very refined and they didn't want to give them up," says Luiz Henrique, "But at the same time, because they were very old, the interface was very clunky." Tecgraf was asked to provide a better interface to this old simulation software, something that would allow the users to simply click on a diagram, enter a value and request a simulation. Realizing that Tecgraf would need to provide such an interface for a wide number of simulators, Luiz Henrique started thinking about developing a language for expressing the configurations:

> Luiz Henrique: So, we talked to them and came up with an idea: why not design a language so that we can write a single program that would capture this data. This was kind of a typical problem. You would write a simple text file, that would say: I want this diagram and in this diagram when I click this entity you should show this kind of a menu and do this kind data validation, things like that. And then when I am done I want this data to be output in this format. So we wrote this language to spec this kind of task.[274]

The language, written in 1992,[275] was described in a Tecgraf report (Figueiredo 1992) and the larger system of which it was a part was then described in a paper presented at a conference in Brazil (Figueiredo et al. 1992). The language was called "DEL," short for "data-entry language." It was what would today be known as "a domain-specific language" and was then called "a little language" (Ierusalimschy et al. 2007), designed to be used with a specific application. While DEL was a success in Tecgraf and among its users in Petrobras (Ierusalimschy et al. 2007), it soon became clear that it was too limited to build all the applications that Petrobras wanted.

At the same time, Roberto Ierusalimschy and Waldemar Celes (then a Ph.D. student at PUC), developed another domain-specific language called "SOL" ("Simple Object Language") for another of Petrobras's many specific problems. SOL was finished, but never delivered to Petrobras, as it became clear that, just like DEL, it lacked the expressiveness wanted by some of the Petrobras projects. In mid 1993 Roberto, Waldemar and Luiz Henrique met to discuss the possibility of replacing both DEL and SOL with a new language, which was soon implemented by Waldemar as a course project. They

---

274 All my interviews with Luiz Henrique de Figueiredo were conducted in English. See appendix B for the treatment of English interviews with non-native English speakers.

275 According to Ierusalimschy et al. (2007) and the comments in DEL code, the language was developed jointly by Luiz Henrique de Figueiredo and Luiz Cristovão Gomes Coelho.

called the language "LUA," meaning "moon" in Portuguese. This was as a pun on "SOL" (Portuguese for "sun"), but also, as somewhat of a joke, an abbreviation for Portuguese "Linguagem para Usuarios de Aplicação"—"Language for Application Users." LUA was a success and was quickly picked up by other projects at Tecgraf.

Comments in English

DEL, SOL and LUA (soon renamed "Lua") were all written in the C programming language, one of the most popular programming languages in the 1990s. In addition to being the most popular (the distinction that it is only starting to lose today), C was and still is the *lingua franca* of programming languages.[276] Most programming languages provide mechanisms to develop "bindings" to libraries written in C, giving the programmer an option of calling C code. Some (but not all) provide ways of being called from a C program. Like other "scripting languages" that emerged around the same time,[277] Lua aimed to be easier to work with than C. Unlike most of those languages—with the exception of Tcl—Lua was primarily designed for use inside a C application, making it possible to develop parts of the system in C and other parts in Lua. Rather than trying to serve as an alternative to the *lingua franca*, Lua was designed to work *with* it. The ease with which Lua and C could be mixed, is in fact often considered one of Lua's greatest strengths vis-à-vis similar programming languages.

276 TIOBE declared C "the programming language" for 2008, acknowledging the fact that C has grown in popularity in 2008, despite being one of the oldest languages in TIOBE's top twenty.

277 Perl in 1987, a variety of "structured" dialects of BASIC in the late 1980s, Tcl in 1988, Python in 1991, Ruby in 1992, to name some of the more successful ones.

Similarly, from early on, Lua displayed a commitment to another *lingua franca*: English. Lua and SOL were both coded "in English," in the sense that all entities had English names—a relatively common practice among Brazilian software professionals.[278] (See chapter 1.2.) Somewhat more unusually, and contrary to Tecgraf's general practice, Lua's error messages and the comments in the code were written in English as well. One of Tecgraf developers who worked with Lua in the early 1990s ("when it was still called SOL"), talks about the differences between Lua and other Tecgraf projects:

> Yuri: I know that in the first version of Lua, and even the code of SOL was written in English, even with comments in English...
>
> Antônio: Yes, even today we do this. Here in our group too. Even apart from Lua we try... No, sorry, the comments are in Portuguese... obviously, right? But variable names, we usually try to do all in English. But our comments are in Portuguese. Lua is different.[279]

Antônio starts by saying that Lua's use of English is no different from the general practice at Tecgraf, but immediately corrects himself, pointing out that while using English for variable names is the standard, Tecgraf *obviously* uses Portuguese for comments.

When I ask Antônio whether usage of Portuguese for comments is really so "obvious," considering that Lua's comments are written in English, he explains, contrasting Lua's case with that of most Tecgraf applications:

---

278 SOL implementation had minimal comments, a total of fifty five words. All of those comments were in English, however. The earliest available implementation of Lua (from July 28, 1993) contains a small number of comments in Portuguese, but is overall consistently written in English. DEL implementation consists of twenty three files, twenty of which are strictly in English, including all eighteen files attributed to Luiz Henrique de Figueiredo. The other three files use a mixture of English and Portuguese for both variable names and comments.

279 See appendix C, "Original Interview Quotes," (Antônio, May 2007, "Os comentários nossos são em português").

> Yuri: You said the comments are "obviously" in Portuguese. [...] It seems that sometimes people *do* write comments in English.
>
> Antônio: No, no. We... in Portuguese. Because... In Lua's case I think it even makes sense for everything to be in English. It was born to fly, for other... It has a much more globalized use than our applications. We, frankly, don't even encourage... we don't even want people here to try to write comments in English, because most of them are not fluent in English, so it would end up being broken English (*inglês capenga*)... Our products are not... Our source code is not for export, it's not open. What we do is not open. The code we write is for Petrobras and is their property, and they don't want to have to also... to know English to read our code. So, for several reasons, in applications we don't write comments in English. Variable names—yes, in the code. You read it more or less in English. But comments...
>
> Yuri: But then why are variable names in English?
>
> Antônio: This is a question we've been debating, but we feel that it ends up... For example, the code syntax would, even when you just read it, would be weird, no? We find it strange to mix things that way. But comments—no. And now that you said it... For me this is obvious, but looking back at it, maybe not, huh? Why is it natural for comments to be in Portuguese but not for other things, right?[280]

Choosing English variable names is standard practice at Tecgraf and many other Brazilian organizations. This practice already gives software a certain degree of mobility. Lua authors, however, take an extra step, using English not only for naming variables but also for comments in their software.

Antônio alludes to two differences between the actual code and the comments that may explain why Brazilian developers often use English for variable names and Portuguese for the comments. Code already typically includes English words, whether or not a Brazilian programmer wants it to (see chapter 1.2). Additionally, code uses English in a constrained way, with heavy reliance on abbreviations (e.g., "apl_unsel_group"),

280 See appendix C, "Original Interview Quotes," (Antônio, May 2007, "Uso muito mais globalizado").

making the programmer's lack of English fluency less obvious. Comments on the other hand, are written in full sentences, and offer little opportunity to hide the weakness of programmer's English.

Lua is different, however, says Antônio, and he does not remember anyone ever finding strange the use of English in Lua's comments and error messages:

> Antônio: No, no. It's like this. I don't think this is questionable. I mean, everyone kind of understands that Lua... The idea is that Lua would be used really in other contexts, in other places. That it would have... coverage. Would reach beyond Brazil. Rio is [just] it's base. I think it's natural for Lua to be that way.[281]

In retrospect, using English appears quite natural. The language "was born to fly," says Antônio.

World-wide success, however, was hardly a part of the Lua team's plan, according to Roberto Ierusalimschy, who instead explains his use of English as a matter of habit and convenience.[282] Compared to the Tecgraf developers who, as Antônio fears, might write comments in broken English, Luiz Henrique and Roberto were both comfortable with English, as well with the English-speaking academic culture more broadly, having spent some time abroad. Lua's authors could thus write their comments in English without the risk of embarrassing themselves with simple mistakes. This competence (and confidence) with English gave Lua an additional early start on disembedding.

When I point out to Antônio that Lua was clearly *not* meant to "fly," but was

---

281 See appendix C, "Original Interview Quotes," (Antônio, May 2007, "Em outros contextos, em outros lugares").

282 See chapter 1.2 for Roberto's own explanation for this decision.

ostensibly written to solve a specific problem faced by Tecgraf, he offers a different explanation of Lua team's possible rationale by referring to his own project, which he sees as *potentially* open.

> Yuri: But at the time it wasn't [intended for use abroad], right? At the time it was just a project...
>
> Antônio: Yes, nobody could think about... the explosion, the success that Lua would have. The acceptance in the *games* [said in English] industry. But maybe they, Roberto and Luiz Henrique, had this idea, I don't know. For example, this [hobby project] that I wrote, that's all in English. Comments are in English. Because in my case I was thinking, I don't know: one day I'll put it on Lua Forge, someone will want to download. I like this idea of *open source* [English], of "many eyes" [English]. Everyone will be looking. I think that when you do an open source application, you have to speak the most wide-spread language, the most common language, most easily understood. In case of our applications, no, us no. Not at Tecgraf. We actually don't want this to happen.[283]

While Antônio says that he has never written comments in English when working on Tecgraf products, he uses English comments when he works on a hobby project. He makes it clear that he neither expect this project to become anything big, and in fact he has not gotten around to releasing the code to anyone. "One day," however, he may put it on LuaForge, a website where users of Lua put code that they want to share. And who knows, it may become popular. Using English keeps open this possibility.

Antônio's project and Lua are both different from the products that Tecgraf built for Petrobras by their relative isolation from the local power relations. Neither was written to immediately become property of Petrobras. While both projects may technically belong to PUC, as an academic institution PUC appears to be content to let the developers to

283 See appendix C, "Original Interview Quotes," (Antônio, May 2007, "Vou botar no LuaForge").

treat their code as free. (See the next chapter on the question of Lua's ownership by PUC.) In addition to simply giving them the freedom to write such projects as they want, this makes the possibility of a global success somewhat more imaginable, since the projects' destinies are not as obviously in the hands of the bureaucracy of a Brazilian corporation

To understand Lua's early use of English we must thus consider the *potentiality* of Lua's later success, the inherent ambivalence of such projects, and the notion of "subvocal imagination" discussed in chapter 3.3. Created for specific needs (of those who pay the bills), projects such as Lua may from the beginning carry the imaginable possibility of global success. While such global future is rarely *planned* for explicitly, it can be *imagined*, and this may be sufficient reason for developing the project in a way that would not altogether preclude the possibility of a global success. (See a later discussion of this issue in chapter 3.4). This means, among other things, writing comments in English.

When explaining his rationale for writing his hobby project in English, Antônio draws explicitly on *open source* terminology. Today, the open source paradigm provides developers with a ready vocabulary and an accepted framework for explaining such decisions. While Lua eventually becomes "free software," it did not *start* this way nor does it appear that the free software / open source vocabulary was available to Lua's authors in 1993. We will come back to Lua's relationship with the world of "open source" software later in this chapter and again in the next one.

Wait, How Did You Do This?

Like code and comments, Lua's manual was similarly written in English, though none of the early users whom I interviewed remember using it. (Some seem to believe it was written in Portuguese.) As Tecgraf was quite small and all the developers were located in the same room, simply talking to other members of the lab was the easiest way to learn Lua. As another early user of Lua says:

> Bruno: I learned from the code.
>
> Yuri: Reading code that was already written?
>
> Bruno: Already written. By Luiz Henrique, by Roberto, and by others who started before me. I probably saw Rodrigo's [pseudonym] code, probably saw other people's code. [...] Since Tecgraf in the beginning was small, the people all stayed in the same lab. So for example, when I did Ph.D. here, I sat at a computer, next to me sat Luiz Henrique who worked here at that point. He just finished his Ph.D. but was still here. I started when he finished his Ph.D. Rodrigo was finishing his undergraduate; he sat at my other side. You see? So, we were all in contact. My dissertation, for example, was all done in C. So when I decided to move, to get into Lua, it was because I was following what other people were doing. I would write, I don't know, two pages of code and then look to my side and see that the guy sitting there did the same in, I don't know, a quarter of a page. "Wait, how did you do this... in such a simple way?" So, it was through this contact inside the lab was that people learned. You see? Today we've grown. There is no more lab. There are now groups and each has their own project. So, if you are with a project that doesn't use Lua, it won't be as easy to stay in contact and learn the same way. But back in the day it was almost natural.[284]

Others who started Lua at the time similarly say that they learned Lua by simply talking to its authors and other early users, rather than having to rely on the early version of the manual. Lua similarly did not have a mailing list—things that could not be resolved in

284 See appendix C, "Original Interview Quotes," (Bruno, April 2007, "Aprendi pelo código").

person could be discussed on the general Tecgraf list.

While Bruno stresses that much has changed at Tecgraf due to its growth (Tecgraf now employs more than 250 people) he still prefers to have lunch with Luiz Henrique once a month than to follow the discussion on the Lua mailing list. In fact, as Bruno and I leave Tecgraf's office and head to get lunch with Rodrigo Miranda (who sat on Bruno's other side in 1993), we run into Luiz Henrique. The days of face-to-face contact at Tecgraf are not quite gone.

Let's See What Happens

Lua's authors stress that the language was developed to solve specific problems Tecgraf faced in their work for Petrobras, and that Lua's later international success came as a major surprise to them:

> Yuri: Have your personal goals for Lua changed over time? I mean...
>
> Roberto: Completely, completely. Completely! This is so huge, I can't... It changes everything. When we started Lua... This is one of the things that people do not... When we say in our paper, that paper about history of Lua,[285] that it went beyond our most optimistic expectations, this is not very true. Because we didn't have *any* expectations. For instance, when we started, we never created a language for ta-ta-ti-ta-ta-ta. We really created a language to solve this specific problem we had at the time. That's why we joke, but it's true: There was no "Lua 1.0."[286] There was "Lua." We did code, and that worked and "oh great, it solved our problems here." We didn't even have an RCS project. We didn't have

285 Ierusalimschy et al. (2007)

286 The manual distributed with Lua 1.1 contains a link to ftp.icad.puc-rio.br:/pub/lua/lua_1.0.tar.Z, which presumably represented a distribution of "Lua 1.0." The link is currently dead and it appears that it's inclusion in the manual for Lua 1.1 was a mistake. A snapshot of a pre-1.1 version of Lua was later released in 2004 as "Lua 1.0" to commemorate Lua's ten year anniversary.

anything.[287]

Lua represented a specific solution for a specific problem. Lua's authors also did not attempt to be innovative for the sake of innovation. Lua represented a pragmatic combination of features, informed by academic research on programming language design, but not aiming to contribute to it.[288]

The pressure to publish, however, combined with Lua's early success at Tecgraf, soon led the authors to present the language outside Tecgraf,[289] starting with a short presentation at a conference in Brazil in 1993:

> Roberto: Yeah, I think it's exactly that: we are academics. [...] There was a symposium explicitly asking for a kind of industry... "real stuff," as they said. I think they still have this in software engineering. At least at that time they were very concerned about this division of academic and "real" applications. I remember sometime later them asking us to review papers for instance, and there was one or two grades related to how does these apply to real situations or how much this is really used, or things like that. So I thought, "Oh we're going to show Lua that way," I mean, it's a real tool, there are people, even... That was very funny, this I remember until today that even that very, very small base of users at that time, of still Lua 1.0, was considered big inside the conference and we said: "Oh we have some eight, ten people really programming in that language." [Laughs.] And that's much bigger than a lot of papers where, I don't know, only the writer [author] ever programmed, used that tool for anything. But that was not, I mean, it was academic because we always have this pressure to publish so if we can generate a publication, we always try to do that. But so this was just kind of: "Oh, they are asking for that, we have Lua ready here, let's try to stage and let's prepare Lua" and we gave a presentation there about Lua.

287 All interviews with Roberto Ierusalimschy were conducted in English.

288 Ten years later, Lua 5.0 (Ierusalimschy et al. 2005) became the first scripting language to use a register-based virtual machine, which brought Lua substantial academic interest. Additionally, in the more recent years, Roberto Ierusalimschy and his students have used Lua as a base for experimental work in programming language research.

289 DEL was similarly described in a Tecgraf technical report (Figueiredo 1992) and a papers presented at a conference in Brazil (Figueiredo et al. 1992).

According to Roberto, the presentation was extremely well received.

As Roberto explains, Lua was not written to advance computer science research, but its design did benefit from Roberto's academic background in programming language theory. Roberto thus borrowed design ideas from a wide range of programming languages with which he was familiar, including some that few software developers know or use. More importantly, Roberto's work as a researcher allowed him to *explain* Lua in terms of current computer science research, helping Lua travel away from Rio using academic papers as a vehicle. (A few years later the same ability to explain Lua in proper terms became important in giving Lua credibility in the eyes of American software developers working outside academia.)

In 1994, the team wrote a longer paper for another Brazilian conference. This time, the paper was written in English and referred to the language as "Lua" rather than "LUA,"[290] avoiding the hassle of explaining a Portuguese acronym in an English paper. The paper also included a link to download Lua ("Lua 1.1"). Roberto says that a colleague at Tecgraf "pushed" them to put Lua up on a website and to include a link in the paper "to show that this wasn't just vaporware." ("Vaporware" is developers' term for software that is described but does not actually exist or does not work as described.) Encouraged by Lua's success so far, the team was also curious to see what might happen next, and Luiz Henrique announced Lua on a number of newsgroups.[291]

---

290 The 1993 presentation was entitled "LUA: uma linguagem para customização de de aplicações" ("LUA: a language for customizing applications").

291 http://compilers.iecc.com/comparch/article/94-07-051

> Roberto: I don't remember, but someone kind of pushed us to put it on the Internet. We didn't even think about that. They started: "Put it on the Internet, that's good, people are going to use it." "Ok, let's put it on FTP." And then I have no idea of how some people heard about it, we didn't have a website or anything like that. I think... I think Luiz sent mail to [long pause] to some groups. But I don't know when that was. 94? 95? Just announcing that there was Lua. And then some people started using it. But it was kind of, "Let's see what happens."

The paper and the announcement started a slow trickle of questions, some of them from abroad.

Lua 1.1 was packaged with an informal license that allowed free academic use, but reserved the rights for commercial use:

> Roberto: Something we wanted, that I remember... Again someone gave us this idea to try to sell Lua. In the beginning we put it on the Internet with a free academic license, and "Please contact us for commercial use." So there was this idea "Let's try to sell Lua." And then it stayed this way one year and we got one contact. [Laughs.] Without success. Just a contact for "Maybe we could use..." for commercial use. So, we decided we were not going to sell it. But after that we noticed that there were people using it and people were liking it, and we were liking that idea of other people using Lua.

Lua 2.5, released in the early 1995, included a license written in proper English "legalese" and allowing almost unrestricted use of Lua for both academic and commercial purposes.[292]

The change of license not only gave additional freedom to Lua's potential users, but also signified the authors' changing perspective on what they could and could not achieve with Lua. Having started with a rather vague idea of what Lua could lead to and

292 "Naively, we wrote our own license text as a slight collage and rewording of existing licenses," later wrote Lua's authors (Ierusalimschy et al. 2007). While the authors say that they do not remember from what sources they borrowed the text of the license, the first part of the Lua 2.1 license is identical to the license of Tcl 7.3 (released in 1993), while the rest generally corresponds to the X11 license.

originally holding open the possibility of selling it for money, they move to "liking the idea of other people using it." Roberto then continues:

> Roberto: That kind of... touched... satisfying, a kind of gratification for us, gratifying, whatever. And so we started to feel good about that. [...] And then we published the article: "Let see, let's try to get more users, to promote Lua." So we put up... And then the reaction was very strong, and then it started to be really important—the outside users.

Consistent with Becker's theory of motivation (see chapters 1.1 and 2.1), while Lua's authors do not start off with an intention of distributing free software, they develop the appropriate perceptions and judgments as they engage in the activity. As members of an academic community, however, and in particular being fairly fluent in the culture of Anglo-American computer science, from which the free software movement was born, they were of course well prepared for developing such perceptions and judgments.

In 1996, the team wrote an article for *Software: Practice & Experience* (Ierusalimschy et al. 1996) and another one for *Dr. Dobb's Journal* (Figueiredo et al. 1996). Articles in those widely-read publications (the latter targeting software professionals rather than academics), introduced Lua to a larger audience and led to a stream of questions from abroad, and the decision to set up a mailing list.

> Luiz Henrique: Around that time, I remember now, we wrote this article in *Dr. Dobb's Journal* and from then on we started to get messages from abroad, people asking questions about Lua. So, we thought, well, maybe we are going to get too many questions and won't have time to answer them all. So we created the mailing list for that, so that other people could answer our questions. [...] Maybe Lua is going to get some interest, and how about creating a community? [...] If we were going to get a community, maybe we should have a mailing list so that they could talk among themselves? To not have to answer everyone individually.

The list (Lua-L) was setup in February 1997.[293]

Prior to the creation of Lua-L, the project had no dedicated mailing list. (The need for the list was largely obviated by the close proximity of the people using it—see previous section.) The English Lua-L thus became the central forum for the Lua community, attracting many Lua users from PUC, as well as a smaller number form other Brazilian research institutions. In 1997 the Brazilians (people with .br email addresses) comprised a little under a quarter of the list's participants, constituting the list's largest "minority." As the list grew, however, the percentage of the Brazilian participants started to decline, eventually getting overtaken by Germans in 2007.[294]

More Exciting Users

A month before the mailing list was set up, the Lua authors received a message from a programmer working for Lucas Arts, which read:

> To: "lua@icad.puc-rio.br" <lua@icad.puc-rio.br>
> Subject: LUA rocks! Question, too.
> Date: Thu, 9 Jan 1997 13:21:41 -0800
>
> Hi there...

293 Each of the three members of the team spent some time abroad (in different places) between 1995 and 1997, though this fact did not come up in any of my interviews. While their separation had roughly preceded the setup of the mailing list, the list did not become the locus of Lua development, even to the limited extent as happened with Kepler's list after Alan's departure (see chapter 3.4).

294 In 2007, the list received messages from 38 .de addresses (6.1% of the total) and 26 .br addresses (4.2%). This most likely undercounts the actual number of Germans and Brazilians because of the wide use of gmail.com addresses (28.2 % of the addresses in 2007). This most likely undercounts the actual number of Germans and Brazilians because of the wide use of gmail.com addresses (28.2 % of the addresses in 2007). Note that .br addresses account for a much larger proportion of messages (12.7% vs. 3.7% for .de), because of a small number of very active Brazilian participants. See appendix J for details.

> After reading the Dr. Dobb's article on Lua I was very eager to check it out, and so far it has exceeded my expectations in every way! It's elegance and simplicity astound me. Congratulations on developing such a well-thought out language.
>
> Some background: I am working on an adventure game for the LucasArts Entertainment Co., and I want to try replacing our older adventure game scripting language, SCUMM, with Lua. (Ierusalimschy et al. 2001, p. 8).

Once the list was created, he became an active participant and in April revealed to the list that he was working on a "scripted adventure game engine." Soon, other users started to discuss their use of Lua for scripting video games. A year later, Lucas Arts released *Grim Fandango*, which became Lua's first international success case—not quite yet "used by Microsoft" but a major step towards it.

Grim Fandango also gave it a new "place." Lua now had a new, international "origin" being associated with a community that was not narrowly localized. *People ask where Lua is from*, says Steve, the interviewee introduced earlier. *But they do not usually mean location, but rather which industry or context. For example, JavaScript comes out of the Web. So, I give two answers: "PUC-Rio" and "the games industry."* Steve himself understands quite well that Lua does not really "come out of" the game industry. The game industry, however, provides a context in which Lua can be understood. *People don't care who wrote it,* he says. *They want to know how it fits into the world.* The game industry thus provides Lua with a pedigree that makes sense in a foreign context.

Lua's success at LucasArts also gave Lua new advocates, located in about the best place for promoting Lua. As the LucasArts programmer explained on the Lua mailing list in 2001, face to face interactions in California were instrumental in helping Lua gain

popularity in the games industry. In 1998, some of LucasArts programmers attended the Game Developer's Conference, the largest trade event for computer games developers, held annually around San Francisco Bay Area. One of them made a presentation about implementing scripting languages for computer games.

> It was quite heavily attended, probably 200 or 300 people in the room. Rob talked at length about the benefits but also all the complexity of writing your own language from scratch, and went through lexing, parsing, analysis, compiling, etc. etc. in good detail. [...] Near the end of his talk many game programmers looked quite discouraged, realizing that a decent scripting language was not something you could whip up overnight, especially your first time out. As the questions started, he pointed out that you can take pre-existing language interpreters and get much quicker results, specifically mentioning Lua. He talked about Grim and pointed me out in the audience, and I stood up and gave a brief blurb about Lua. Between us we said that it only took a day to embed, code base was clear and easy to modify, was small and fast, extensible, easy to pick up for designers, etc. People lit up and furiously started scribbling notes and looked really excited. I got a few inquiries afterwards, but game developers being who they are, most of them just went out and checked it out on their own. Soon enough the list was overflowing with game programmer inquiries...

This "brief blurb" about Lua delivered at a conference in the very center of the software world, brought Lua to the attention of the larger gaming industry. This attention eventually led to the use of Lua in a game released by Microsoft Games, giving Lua the coveted "used by Microsoft" status.

The growing list increasingly became an important source of influence on Lua. "Tecgraf was kind of stable and was not demanding that much," says Roberto. While having a non-demanding employer may be a blessing in many lines of work, this is not necessarily the case in software development. (See chapter 2.1.) Finding that Tecgraf was

no longer presenting serious changes, the Lua team increasingly turned their attention to the outside. “So I think it was also kind of like: ‘Let’s try to find more... more exciting users,’” says Roberto. The list members supplied the desired challenge, by applying Lua to new domains and running it on new platforms:

> Roberto: Specific things that emerged, more than people asked. Problems that emerged, in different environments. One instance, one example is embedded devices. Tecgraf never used Lua in embedded devices. Now they use, but I am not sure if they would try to use it, if they would push Lua in this direction by themselves. That was something that from... people outside started to use Lua on those very different devices, on very small computers.

Lua’s gradual “porting” to foreign contexts went hand in hand with porting in the more narrow technical sense: Lua was increasingly used on computing platforms that were never used by Tecgraf:

> Roberto: That put more pressure to make Lua really portable. In the beginning our goal of portability was Tecgraf’s set of computers. So, that was our goal, must run on that. It was a very large variety of computers that Tecgraf had, so from the beginning it was very portable. So it must run on DEC, on VAX, on ta ta ta. And then later when people... I remember in 98 someone wrote and said they ported Lua to Cray, the supercomputer Cray 1. That was very exciting: “Wow, Lua is running on Cray.” And then these things started for instance to show us that we must really think about ANSI C and about real standards. And not about “It runs on those machines and that’s good enough.” For instance this was something that came from outside.

Paradoxically, Lua’s origin in Brazil offered it an early start on portability. Due to the restrictions imposed by the Market Reserve, Petrobras had limited choice of what computers it used, having to buy from different manufactures depending on who had was allowed to bring computers into Brazil in a particular year. Over the years, it had

accumulated a rather diverse collection of computers, requiring that Lua could be run on all of them (Ierusalimschy et al. 2007). This collection did not include things like the Cray super-computer, however.

Demanding and attentive users are often considered in open source communities to be a valuable resource *per se*, as they help to "push" the project forward, and provide feedback and a source of gratification to the authors. In Lua's case, however, foreign list members were not merely asking for features. Many became active members who helped answer questions and contributed ideas and resources. In 2000, one of the list members organized a wiki, which has become an important resource for Lua users. In 2001, one member's "misunderstanding" of Roberto's explanation of a future feature contributed a major improvement (Ierusalimschy 2007, page 2-15). The list members also offered substantial help with the first edition of *Programming in Lua* (see chapter 1.2).

Also in 2001, an email was sent to the Lua list asking: "Just out of curiosity, why aren't you guys using Lua.org for your main URL?" Other members quickly pointed out that lua.org, lua.com and lua.net domains were all taken, perhaps by domain speculators. (Short domain names tend to be quite popular.) The discussion moved to comparing the advantages of "lualanguage.org" versus "lua.org.br." The .br option was described by Roberto as complicated for bureaucratic reasons, while other users expressed concerns about the hassle of country-specific domains.[295] In the end, however, the difficult decision between the "clumsy" lualanguage.org and the "country-specific" lua.org.br could be

295 "I agree.. though lua.org.br would be shorter, I for one never think of adding country specific things to the end of a .org domain (well, I guess that isn't a .org, but still). I think one of the hallmarks of a radical domain name is that you can think of the name of the project and add .org to the end and you go there.. that's just me though."

avoided: the original poster approached the owner of "lua.org," purchased the domain and donated it to the Lua authors. (From 2004, another user has volunteered to host lua.org website on his company's servers in the United Kingdom.)

Breaking from Tecgraf

During the first decade of its existence, Lua was a "Tecgraf project" and Tecgraf served as "a good home" for it, according to Roberto. This continued through the late 1990s, even as Lua was increasingly looking outwards. Around 2003, however, Tecgraf stopped paying for Lua development.

While this separation was to some extent expected, some of my interviewees attribute the ultimate break to the transition from Lua 3.2 to Lua 4. Released in November 2000, Lua 4 introduced substantial changes in the way Lua connected to code written in C. The change was originally motivated by the desire to allow a program written in C to run multiple Lua programs at the same time.[296] This ability was requested by some of the users as early as 1998 and also turned out to be necessary for one of Tecgraf's own projects, called "CGILua," which aimed to make it possible to use Lua for developing web applications. (CGILua later became the basis of Kepler—the project described in chapter 3.4.) The team originally introduced the feature by making the smallest possible modifications to Lua, trying to make sure that the new version ("Lua 3.3") would require

296 When a program written in C loads and executes chunks of Lua code, this code modifies a set of data that represents the "state" of the Lua interpreter. Up until Lua 4.0, all such code would modify the same state. Lua 4.0 introduced the ability to maintain multiple Lua states within the same C program (a "reentrant API").

almost no changes to the existing software. However, the limitations of this approach soon became clear, and the team proceeded to make more and more serious changes to how Lua connects to C code. Eventually the interface between Lua and C was changed to a point where many existing programs would have to undergo serious modifications. The team then decided to use the opportunity to completely redesign the interface, thus changing it even further. When the new version was released in November 2000, it was deemed sufficiently different to be called “Lua 4.0” rather than “Lua 3.3.”

While offering substantial improvements, the new API made obsolete all old C code interfacing with Lua. Roberto offered suggestions on how to fix the old code to make it work with Lua 4, but few Tecgraf projects undertook such migration.

> Roberto: Then there was this big problem of compatibility. I think maybe this was the main breaking point. [...] The change from 3.2 to 4.0. That was a big change in the API so for people that only used Lua as a language, it was not that big, but for people that integrated Lua into other tools, the C API changed a lot and all applications in Tecgraf were in that kind of API stuff.
>
> Yuri: Was that something you foresaw?
>
> Roberto: The break or their reaction?
>
> Yuri: Well, either.
>
> Roberto: The break [in compatibility] for sure we foresaw, but their reaction I think... We wrote some compatibility code and some things to help, but people mainly didn’t use it, at all. [...] They never changed to Lua 4. So they started to drift apart from the Lua community. I mean, because everything was written in new manuals, and new discussions and new tricks and everything was evolving around Lua 4.0 and they were…

Even CGILua, the project that motivated the changes that eventually led to Lua 4.0, never released a version that worked with Lua 4.0. (This was to a large extent caused by the fact

that two of the key people working on CGILua left Tecgraf before Lua 4.0 was released. A version of CGILua for Lua 4 was done by one of its users, but was never released—see chapter 3.4.) Unwilling to make the transition, Tecgraf's projects got "stuck" with Lua 3.2, a version that soon started to lose the interest for the Lua community. Lua's and Tecgraf's paths started to diverge.

In addition to the practical problem of backwards compatibility, Lua 4 set the precedent for introducing features that brought only cost and little benefit to Tecgraf's projects. Some users of Lua did not take this well:[297]

> Roberto: I think that they got... with some reason I think they got a little offended with the change to 4.0. I think that's why it was kind of a break point. I think this was the first change that we saw that it could hurt Tecgraf but we are going to do it anyway. We thought that it was not going to hurt *that* much, we tried and thought... Not something like "Oh, we are going to do it *because* it's going to hurt Tecgraf." We tried to minimize that, as I say, we did a lot of stuff to try to do compatibility layers and things like that. But we knew that it was going to have some problems, was going to be a big incompatibility. And so I think. That's why I say it was a kind of break point.

Lua 4 was therefore not only introducing a technical break with existing Tecgraf software, but also demonstrating the new priorities of the Lua team.

The new version of Lua was an improvement—for the new users.

> Yuri: Why was this change made?
>
> Roberto: Because it was *really* much better. [Laughs.]
>
> Yuri: But better for who, let's put it this way?

297 Ironically, but not surprisingly, Roberto Ierusalimschy was one the few people willing to talk extensively and on record about this conflict. Most people who were there at the time either asked to not be quoted or (more often) downplayed the complaints.

> Roberto: For any new user of Lua.

It also worked quite well for the existing foreign community. Many of the list members we interested in Lua as a hobby and found the quick pace of change engaging intellectually. Others used it in games: software that is typically abandoned soon after it is released. (For example, two "versions" of Grim Fandango were released in 1998: 1.0 and 1.01. No other versions of the game were ever produced. Of course, some of the users may continue playing the game for years.) As far as foreign users of Lua are concerned Lua 4 was a clear step forward and not only because of the improved API. Lua 4 demonstrated the authors' commitment to building a good language and their willingness to leave behind earlier mistakes. As earlier quotes from Rich and Craig show, foreign users took note of this commitment.[298]

While some of Tecgraf's users of Lua complained about this transition, most of those to whom I talked in 2007 seemed to consider this sacrifice worthwhile. One of the early users of Lua says:

> Silvio: I think we skipped 4.0. I am not sure. I can send you this later, if it's important for you.
>
> Yuri: Ok. I am asking because I've heard that people say that there were some difficulties between around Lua 3.2 and Lua 4 and that some people at that point gave up.
>
> Silvio: True. Basically, because the communication API between Lua and C changed drastically. It was in 4, I think, that the stack was introduced in the communication between Lua and C. And so whoever had much C

298 The discussion on the Lua list in the months leading to Lua 4.0 shows that the authors and the users were not entirely indifferent to backwards compatibility, and in fact saw it as quite important. Some members also disagreed with the specific changes introduced by the new version. In the end, however, the desire to make improvement won over the concerns about backwards compatibility and Lua 4.0 was received by the list with much enthusiasm.

> code calling Lua had to make lots of changes. And there were people who really complained: "Oh, darn, must change..." But I don't see it this way. I think we must keep moving ahead. Lua has to evolve. We are not going to stop and make Lua stagnate, or stay with an interface that we know is worse, just because there are people who use it who are lazy to change their applications. It doesn't make sense. To stagnate for the sake of stagnation... Those who don't want to evolve can stick with version 3.2 and use it for the rest of their lives. It'll keep working, thank you very much.[299]

In 2001, Lua was clearly showing global potential, and limiting it for the sake of Tecgraf's older projects did not necessarily make sense, even from the perspective of some of the Tecgraf engineers already invested in earlier versions of Lua. Lua's success abroad was starting to bring certain dividends to PUC and Tecgraf (in terms of prestige if not money), as well as individual people at Tecgraf who, like Silvio, were incorporating Lua into their academic research. Constraining Lua's growth was not necessarily in Tecgraf's best interest.

At the same time, Tecgraf itself was increasingly looking at other technologies. The rapid software innovation in 1990s meant that by 2001 Tecgraf was getting requests for new types of applications and could make use of new tools for implementing them. The most important of those was Java—a programming language released by Sun Microsystems in 1995 that had become the new standard by 2001. Even such committed supporters of Lua as Silvio saw those new technologies as better for some of their projects.

> Silvio: It was about six years ago [in 2001] that we started increasing significantly the number of projects in Java. [...] In 2001 we had a request from the client, like, "Oh, we want a system with such, such and such

299 See appendix C, "Original Interview Quotes," (Silvio, May 2007, "Lua 3.2 e Lua 4").

> characteristics." And I thought it would be more interesting to use Java than... Because there is this thing... There is this saying: "For someone who has a hammer, everything looks like a nail." We have to have a tool box and to know when to use each tool, right? Lua is a great tool, but it's not the right tool for everything. Nobody would expect it to be. For the job that we had in front of us, the ideal solution was a mixture. That's what we did. We used Java to build the client, which we wanted to run over the intranet, et cetera, in the browser. At the time there was no JavaWebStart, but there already was an Applet plugin. So we managed, for example, to run over the intranet an application with a cool graphic interface, without having to worry about the platform of the client, the user. So, we chose to do the client and the server in Java, but the part that was distributed on the machines, which needed more of the operating system things we implemented in Lua. We did this shortcut. The only thing is that what happens is that the server and the client part requires a lot more implementation than the other, so today the project is mostly in Java.[300]

Ironically, despite Lua's oft-cited portability, Silvio's team found that using Java made it easier for them to run the application on their clients' computers without having to worry about the operating system that the clients were using. Additionally, while Lua still worked best for certain Petrobras-specific functionality, Java offered simpler solutions for the more generic problems such as the construction of user interface. Using Java also offered Tecgraf engineers an opportunity to gain experience with a new technology that was growing in popularity. For many, this dramatically broadened their options for employment if they were to ever leave academia and move into software industry.

While Silvio's project started as a mixture of Lua and Java, it has gravitated towards Java over time. Other projects started at the time were done in pure Java from the beginning. For those users of Lua less committed to the language than Silvio, the break introduced by Lua 4 served as a good opportunity to switch to Java.

300 See appendix C, "Original Interview Quotes," (Silvio, May 2007, "Sem se incomodar com a plataforma").

Disembedding

This chapter has looked at the history of Lua, seeking to show step-by-step Lua's transition from a specific solution for a particular need of a Brazilian organization to an international programming language that in retrospect may appear to have been "born to fly." As we saw, several factors were important for this transition. Lua was designed by people who were competent in the "global" software practices and could explain their actions in the right language—both in the sense of literal fluency in English, but also in the sense of being fluent in the conceptual system of academic computer science. They had acquired this competence in part through physical travel to foreign centers of computer science research (such as Waterloo and Cambridge), as well as through their access to a local island of computer science research in Brazil (PUC's Department of Informatics), which had been constructed in Brazil through a complicated history (chapter 2.2).

Lua was developed by a group with simultaneous ties to Brazilian industry and Brazilian academia. This mixed origin was crucial for Lua's later success. The language was developed for a practical purpose, which from the beginning set it aside from programming languages designed purely in pursuit of programming language research. At the same time, Lua's origin in an academic institution made it portable. The Brazilian academic computer science community has developed bidirectional links with North American computer science of the kind that have no equivalent in industry. If Lua had been developed by a Brazilian software company without strong academic connections,

its chances for success in the foreign industry would have been much reduced.

Understanding Lua's evolution requires paying attention to the gradual changes in the authors' goals for the language. Lua started as a practical project undertaken in an academic environment. This mixed origin introduced a crucial ambiguity to what the project could become. Had Lua been built specifically for the purpose of advancing computer science research, it would have suffered the fate of School—the language Roberto Ierusalimschy was developing as a part of his research, which has been all but forgotten. At the same time, stronger contacts with the local industry would have likely entangled the language in local relationship so closely that as to make its international success quite difficult. Unwilling to seriously pursue the option of commercializing Lua for the local market, the authors made early steps towards globalizing Lua and then made it a topic of academic papers, published both in Brazil and abroad. Foreign publications brought foreign users, helping the authors discover the satisfaction of interacting with a large number of users of software they wrote, who not only expressed gratitude but also had the sophistication to truly understand the virtues of the language—and to push its boundaries. Such satisfaction is of course similar to the one academics often seek when they publish their ideas with the hope that they would be valued by peers. The authors could thus acquire the "perceptions and judgments" necessary for development of free software by starting with a rather similar set of academic "perceptions and judgments," and then developing such perceptions and judgments gradually while interacting with Lua users.

This chapter has focused on Lua's gradual disembedding from the local context

and the increased success abroad. As we saw in the end, this success required breaking some of the local relationships. In the next chapter we turn more closely to this issue, looking at Lua's relation with the university, the city and the country where it remains to this day. We will also look at some of the complexities involved in managing a global project from a university in Rio de Janeiro.

3.3 Fast and Patriotic

Fifteen years after its first version was developed, Lua is a fairly popular language, used in a number of well-known software products, both commercial and open source. The work on spillover effects in innovation may lead us to think that Lua's success would present an important opportunity for local economic development: local companies could take advantage of their proximity to PUC-Rio to gain better understanding of the language and its future directions, finding better use for Lua in their products and engaging in related innovation. This is not the case. At the time when I was doing my fieldwork in 2007, Lua was largely unused in Brazil. Apart from Tecgraf, Nas Nuvens and two other small companies incubated at PUC, Roberto Ierusalimschy knew of no Brazilian companies using Lua. If local companies were using Lua, they were not advertising this fact. Roberto remembered only a few occasions when local companies had entered into contact with the Lua team, none of which led to any extended collaboration. In my five months in Rio that year I managed to find just one more company using Lua in Rio de Janeiro, bringing the total to five. By the end of 2008, three of those five companies were either moving away from Lua or had abandoned it altogether. The situation was promising to change due to the inclusion of Lua in the Brazilian standard for digital television (a story related in the end of this chapter), but Lua was yet to gain wide use in Brazil.

There are several reasons for this lack of local adoption. Some of my interviewees

point, sometimes with much frustration, to the Lua team's seeming lack of interest in expanding Lua's use in Brazil. In fact, while Lua's authors lamented the lack in local use of Lua in print (Ierusalimschy et al. 2007), I could see few signs of efforts towards helping local adoption. At the time of my fieldwork in 2007, for example, Lua had no Portuguese documentation—an issue that did not seem to cause much concern for Roberto Ierusalimschy. (See chapter 1.2.) A closely related reason is the seeming lack of fit between what Lua offers and what the typical needs of the local industry. Lua provides clear value for two kinds of software projects: desktop software with high performance requirements (for instance games) and small devices that cannot run the more popular programming languages. Both projects involve making *products.* Rio's software industry, however, focuses almost entirely on services to local organizations, which typically involves building web-based systems. Lua offers few obvious advantages in this domain. This lack of fit, however, can be seen as a symptom rather than a cause of the disconnect between Lua and the local industry. Lua's authors gradually adapted the language for the needs of foreign industry largely *because* of their lack of strong ties to the local industry. (In the next chapter we will look at a project that has aimed to make Lua useful for the sort of projects that the local companies do and the difficulties involved in this task.)

Lua's disconnection from the local industry exemplifies a more general pattern of lack of ties between industry and academic research in Brazil. There may be several reason for such lack of ties. The main proximate reason is the government policy. Brazil's government funds academic research in accordance with the perceived academic success of each department and university. Such success is evaluated quantitatively and involves

as an important component a metric of "intellectual production," measured by the number of publications. The publications are weighed by the rating that CAPES assigns to each journal. (The weighing system is known as "QUALIS.") In 2007, the list of accepted journals for computer science included 241 items, rated as A, B, or C based on the number of citations in two citation databases: ISI/JCR and CiteSeer.[301] Both databases index almost exclusively English publications. Consequently, for computer science, only those Brazilian journals that publish articles in English are given rating above C, and none of them are rated "A" (Qualis 2009). Computer science researchers thus have good reasons to publish abroad. This requires choosing problems that are of interest to their foreign colleagues.

A system of government funding that measures success by local use of research could shift this balance. The policy of giving incentives for publishing in foreign journals and conferences is not without merit, however, and its rationale aligns with the other, distal, reason for the disconnect. Brazilian computer science researchers are located at the periphery of academic computer science. Brazilian software developers working in industry are similarly located at the periphery of their professional world. Both groups must focus on building ties to the centers of their practice, both in order to keep their practice synchronized with the foreign models and in order to act as legitimate representatives of the practice locally. Successful publication in foreign journals is the best way of demonstrating that the research is up to "world standards." Focusing on the needs of the local industry may lead the researchers away from the central problems of

301 http://scientific.thomsonreuters.com/training/jcr/ and http://citeseer.ist.psu.edu/impact.html

the global practice. Moreover, it may potentially undermine their credibility in the eyes of the *local* industry.

Local firms similarly find it safer to stick with "standard" technology, choosing Sun's Java or Microsoft's .Net over PUC-Rio's Lua. Use of locally produced research is risky since the quality of such work cannot be easily assessed. Even in cases where the developers may believe in the technical superiority of the local product, choosing them may be unwise, as it may scare the clients. (See Fabio's discussion of Kepler in chapter 3.1.) Companies thus often fall back on the safer assumption that local technology can be ignored as irrelevant.

This creates difficult questions about what Lua could do for the country and the city where it was developed. To understand the challenges of putting Lua to use and of creating the kind of linkages that could help Lua bring about economic development, we would also have to look at the challenges of running a global project from the periphery. As we saw in the previous chapter, Lua's success was not planned by its authors and in many ways "just happened." This does not mean, however, that the future success of Lua poses no challenges. In fact, it only makes the situation more complex for the authors, who do not fully understand the factors that led Lua to its current position and hence have a limited idea of what awaits it in future. While such problems may be faced by developers located at the centers of the software world, we will see that some of the challenges have much to do with the peripherality of Lua's authors.

A Little Bit of Actual Patriotism

Even as the use of Lua declined at Tecgraf in the recent years, many of the original users of Lua, such as Antônio and Silvio, have continued to read the Lua list and maintained an interest in the language. Some of them have contributed code libraries to Lua. In addition to saying that they follow Lua because they are currently using it in some projects, Lua users at PUC typically mention two factors contributing to their interest in the language: Lua's origin at PUC, discussed as a matter of sentimental or personal attachment, and the desire to see Lua attain the success that it deserves:

> Silvio: I continue to use Lua in Tecgraf projects, so there is a practical reason [for following the list] as things happen in Lua, that's of interest to me. [...] So I want to be a part of anything that happens [on the list] because it may affect me in a project. The other reason is that I love this, I am an enthusiast of this language, I love Lua, I think it's awesome [*um barato*] and I *like* to read the discussions, I actually do it because I *enjoy* it. So there are those two aspects.

When I ask Silvio what he means by "loving" Lua, he explains:

> Silvio: Yes, because I've accompanied Lua since the beginning, right? Roberto was my Master's advisor. I mean: undergraduate, masters and Ph.D. I have a friendship tie with him. Every now and then we go and have lunch together, he tells me what's happening, I don't know what. So I very much live in this world of this language and I do enjoy it, being a part, seeing what happens, talking to Roberto, exchanging ideas. So it's like this. It's something I like, really like. Because of the people involved. It's a *fantastic* piece of work.[302]

Other members of the local Lua community at PUC often stress the same reasons: Lua is an amazing piece of software demonstrating the genius of Roberto Ierusalimschy, a

302 See appendix C, "Original Interview Quotes," (Silvio, May 2007, "O outro motivo").

person who they consider a friend or a mentor; they feel a connection to Lua having seen it from the days when it was completely unknown. Many add that Lua also brings prestige to PUC—a university with which they are themselves affiliated. The fact that Lua was developed in Brazil or in Rio de Janeiro is rarely mentioned.

In early March 2007, as I was reading through archives of Python-Brasil—a Brazilian mailing list dedicated to the Python programming language, I came across a thread entitled "Python—Lua." The thread started with a request for comments about Lua and a question about its advantages and disadvantages compared to Python. While such questions about "competing" programming languages sometimes invite hostile responses, most of the replies were quite positive. One of them read:

> Python has a more elegant syntax, a larger community, a more diversified field of use, better interoperability.
>
> Lua is faster, leaner, more patriotic, more adequate for "embedded systems" [in English] and has a VM for Palm [...].[303]

The next day, I mentioned the thread to Rodrigo Miranda, saying that "Python people" were discussing Lua on Python-Brasil. *What did they say?* asked Rodrigo. "They said it was fast and patriotic," I summarized the discussion. "It *is* fast," agreed Rodrigo. Noticing that he confirmed only half of the statement, I asked him explicitly: "Is it patriotic?" "I am not into this kind of stuff," Rodrigo responded.

While stressing their personal connection to Lua's authors and Lua's connection to the university, PUC users of Lua only infrequently bring up the fact that Lua was developed in Brazil as a reason for supporting it. When asked about this explicitly, some,

303 A message to python-brasil, February 2007, my translation.

like Rodrigo Miranda, explicitly deny any interest in "Brazilian" software, sometimes referring to such "nationalistic" sentiments as demonstrating narrow-mindness or even lack of education.[304] Others admit such feelings after some hesitation:

> Yuri: But it's not a matter of Lua being a Brazilian language, I don't know...
>
> Antônio: [Pause]. There is a little bit of this too. Because I know people who made it. I wanted to help promote it in some way. [Lists several reasons for promoting Lua.] But there is a little bit of this too, definitely. Of pride, of knowing where it came from, of promoting national software. Definitely.[305]

Saying that he supports Lua as a case of "national software" clearly makes Antônio uncomfortable. He says this only when asked directly, and only after first making clear that he has many other reasons for supporting the language. As in nearly all other interviews, the answer to my question about Lua as "Brazilian software" comes only after a long pause.

For many educated Brazilians, "patriotic" is a bad word, especially when used in the context of their area of professional expertise. Being "patriotic" or asking what is good for Brazil is hardly "cool." "Brazil" is a package of problems that one must live with, not something to be excited about. Rodrigo and others often say "we" when talking about the Brazilians, but the solidarity that is connoted by this "we" is the solidarity of prison-mates. When used by engineers in the context of technology, however, the word becomes particularly dangerous, as it may suggests parochialism and inability to grasp the

304 Though many of the people I have talked to alluded to the association between nationalism and lack of education, the author of "fast and patriotic" comment is currently pursuing a Ph.D. in one of Brazil's best computer science departments. (The comment of course may be sarcastic.)

305 See appendix C, "Original Interview Quotes," (Antônio, May 2007, "Um software nacional").

values a larger, global technical culture. A good engineer should not get their judgment swayed by nationalistic feelings, say the developers. For those in their thirties and younger, who were consumers but not producers of technology during the Market Reserve years, Brazil's technological "backwardness" (*atraso*) in 1970s and 1980s provides ample proof.

The perception of parochialism that casts its shadow over any local projects (and which can only be disarmed by personal acquaintance or international success), can be illustrated by a quote from Ricardo, who was introduced to Lua in 1998 as a student at PUC.

> Ricardo: I remember that we had to do a [class] project in C and she [Roberto's spouse, also a PUC professor] taught a new language, which just existed for a few years, invented at PUC and called "Lua." I looked at that and was like: "Eew! A language invented here at PUC? How stupid! I am not going to learn this. I'll never use it in my professional life! What will I do with it? And I remember there being two parts to the assignment that she sent us. One part was in C, another part in Lua. And a girl who was doing a part of the assignment with me... [I told her] "Here, do the part in Lua, because I am not going to learn this stuff, I don't want to know about Lua. I'll do the part in C, which is more interesting, since I'll use it."[306]

Despite the fact that the class assignment introduced Lua in the exact context where it was strongest (as a part of a Lua / C combination), Ricardo avoids any contact with a local language, referring to the very idea of a programming language developed at PUC as "stupid" (and relegating the task to a woman).

Responding to such clear expressions of prejudice is about the only context in

306 See appendix C, "Original Interview Quotes," (Ricardo, July 2007, "Que bobagem!").

which Lua users introduce—with hesitation—the topic of Lua's Brazilian origin.

> Silvio: I think it's great that Roberto managed this... There is also this other thing... It's like... I already saw lots of prejudice against this language. It's impressive how this happens. Like this: I saw one of our clients, a person inside a company, say the following: "Oh no, I am here trying to decide whether to use Lua or this Microsoft application. But I think I will go with Microsoft, because if I use Lua and run into difficulties, my boss will think I am crazy. And if I use Microsoft and run into issues, that's not a problem, because this happens every day." It's *ridiculous* to think like this. I feel ashamed to see someone talk about it this way. I mean... Instead of, being able to use... Instead of *incentivizing* work that was done completely in this country... [For] the guy not to incentivize it out of *pure prejudice*... I think it's totally ridiculous... So... Those kind of things motivate me to follow Lua, to try to use it. Because I see that the list, most of the people on the list are not from Brazil, most are foreigners. I saw that Lua has to succeed abroad to gain acceptance at home. [Pause.] In other words, it's a project that I think is fantastic, which I really like. I see Roberto's struggle to make Lua work, I see the work he has to do. Lua reflects, deep down, his genius. And I think Roberto is very good. [Pause.] For those things, I really like the language and have a pleasure to follow its growth. For this reason I don't leave the list, continue reading it. Even if I don't say anything, I stay on the list seeing what happens.[307]

Silvio mentions in this quote—only in passing—the possibility of "incentivizing" the work done in Brazil. This course of action, however, is presented only in contrast to the prejudice that Lua has faced in Brazil over most of its history.

As I mentioned in an the previous chapter, American users often say that they "take a global perspective" on things like Lua, concerning themselves relatively little with where such software comes from, as long as it is presented in fluent English in a way that shows technical competence of the authors. As indicated by the earlier quote from Ricardo and the story related by Silvio, Brazilian software developers often find that they

307 See appendix C, "Original Interview Quotes," (Silvio, May 2007, "Por puro preconceito").

do not have the luxury of such "global perspective." Often working in contexts where their competence is questioned on a regular basis, they avoid diligently any associations that may bring accusations of parochialism. Unlike Rich, who becomes convinced of Lua's bright future by reading Roberto Ierusalimschy's book and noting the demonstrated global competence, potential users of Lua in Brazil often themselves lack the cultural skills needed for deciding whether a programming language book written in English by a Brazilian author demonstrates the command of the global software culture or is a failed attempt to fake it. They thus find it safer to stick with tools the global status of which cannot be questioned.

While Silvio is disappointed by the "prejudice" with which Brazilians approach anything developed in Brazil, he realizes that he has to accept some of the basic principles underlying this prejudice. To argue for Lua as a case of national software would be to invite yet stronger prejudice, feeding the suspicion that those who support Lua do so for reasons of narrow-minded nationalism. The best way to counteract the prejudice toward a home-grown language is to downplay Lua's Brazilian connections and to look for global credentials that are valued locally. "I saw that Lua has to succeed abroad to gain acceptance at home," says Silvio.

Personal ties beat prejudice. A year later after dismissing Lua as "stupid" in college, Ricardo joined Nas Nuvens, a start up working with Lua. At the time of our interview he was again employed as a Lua programmer for a PUC project, building software on top of Kepler. He talked about Kepler with enthusiasm, mentioning among other things "the idea of domestic technology":

> Ricardo: I've always been following this Kepler thing and finding it interesting. Rodrigo would always pass by, like this, at PUC, and we would chat and he would tell me what was happening.
>
> Yuri: But interesting in what sense?
>
> Ricardo: [Long pause]. You see how things change? The idea of it being domestic technology [*tecnologia nacional*]... [pause], well structured [pause]. A proposal for an actual framework for Internet development. To compete—perhaps Rodrigo would say that it's not to compete, but let's say that for now—with technologies that exist out there... I found it interesting.[308]

The phrase "tecnologia nacional" is remarkably ambiguous. While it can be literally translated as "national technology," such a translation would connote a lot more patriotic pride than the term typically implies. Products that are described with the adjective "nacional" are often understood to be local substitutes for foreign products that are either not available or more expensive. I thus use the more neutral term "domestic" to translate it.

After mentioning the idea of "domestic technology," Ricardo quickly moves to the technical virtues of the project. I have to ask him to come back to this idea a few minutes later:

> Yuri: What do you mean by the idea of domestic technology [*tecnologia nacional*]?
>
> Ricardo: Huh?
>
> Yuri: You said that a part of what made it interesting was this idea of domestic technology.
>
> Ricardo: Oh, right. [Pause.] Why I find this interesting? [Pause.] It's hard to say. It's... [Pause.] I don't know really. Maybe a little bit of actual patriotism too. To believe that we [*a gente*] can develop really good,

308 See appendix C, "Original Interview Quotes," (Ricardo, July 2007, "As coisas mudam").

> world-class technology. To be used by people from all corners [of the world], and which works.[309]

Ricardo's "we" appears to refer to the Brazilians. When I ask him who he means by "we" he confirms my guess with hesitation:

> Yuri: "We" means who?
>
> Ricardo: I can say "we the Brazilians", or I can say, "we... PUC", "Nas Nuvens," "the open source community." I don't know, I don't know really. It's like this.
>
> Yuri: But who did you have in mind?
>
> Ricardo: I think that... [Pause.] This idea of domestic technology [*tecnologia nacional*], or let's say developed—if only initially—by people here, in Brazil. I think this excited me. Even if—"Ok, there are people from all over the world participating." Even better! Do you understand? There are people from all over the world offering recognition [*prestigiando*] to something that started here. [Long pause.][310]

Ricardo struggles to bring together two seemingly contradictory ideas: Kepler and Lua as examples of "domestic" or even "national" technology, something Brazilians can be proud of, and the "global" nature of those projects. He arrives at a formulation that is similar to Silvio's prescription for Lua's success: technology produced in Brazil can be a cause of pride when validated by acceptance around the world.

---

309 See appendix C, "Original Interview Quotes," (Ricardo, July 2007, "Prestigiando uma coisa que começou aqui").

310 See appendix C, "Original Interview Quotes," (Ricardo, July 2007, "Prestigiando uma coisa que começou aqui").

For Everyone's Benefit

On my second day in Rio in 2007, I went to meet Rodrigo Miranda, to catch up face-to-face on what had been happening in the year that I had spent in California. One piece of news that Rodrigo related to me concerned the recent interest that PUC had developed in Lua. In fact, Rodrigo said, there was now a project, to which I will here refer as "Iris," aimed to promote the use of Lua by setting up a number of publicly visible Lua projects (patterned after Rodrigo's own Kepler, described in the next chapter) and looking for funding from foreign companies and local agencies. Rodrigo talked about Iris with excitement, even as he mentioned some of the internal politics around the project and the fact that the talk had been running somewhat ahead of the action.

When we returned to the topic a few days later, Rodrigo told me that a little over a year ago he presented the idea to "Chico," a friend of his who at the time was starting to exercise a certain amount of influence at PUC—at least more so than Rodrigo himself. Chico liked the idea and went to talk to his boss "Carlos." Carlos had heard of Lua by then, but did not yet have a plan of how to use it to PUC's advantage. He liked Rodrigo's idea, however, and took it to *his* boss. According to Rodrigo, Iris eventually made it to the highest levels at PUC, and became "a big thing." As the project grew in status, PUC managed to line up some money from "Softnet," a large American IT company. Now that there was money involved, naturally even more people were excited.

Rodrigo said that he had discussed the plan with Roberto early in the process, but Roberto perhaps had not taken it seriously, thinking it was another of Rodrigo's "crazy"

plans, unlikely to materialize. After the project grew big, says Rodrigo, Roberto heard of it and complained about not being in the loop. While this had caused some tensions around Iris, Rodrigo was confident that this was a matter miscommunication, and was hopeful that the issue would get resolved, as Roberto would come to see Iris for the opportunity that it was.

A few days later, Rodrigo introduced me to Chico, who greeted me with much excitement. After a brief tour of his lab, Chico talked about his efforts to "evangelize" Lua. *PUC had not been paying attention to Lua,* he said. *But it's changing now.* At some point, he turned to Rodrigo and remarked: "I am going to go to the Dean and say: Look, there is a guy from *the United States* doing research on Lua! Why are people *here* not paying as much attention?"

Later that month, with some help from Chico, I met Carlos, who told me of his reasons supporting Lua. As he explained, he had know Roberto for some time, as a friend of a friend, and had heard of Lua before, but had not thought seriously about its potential until recently, when he started noticing its popularity abroad. "When I took this planning role, and started looking for the potential of the university, what it had," says Carlos, "I noticed that there was a fairly big movement actually using Lua."[311] Chico's return to PUC brought Carlos in contact with Rodrigo Miranda and his ideas about strengthening Lua's local position. Carlos embraced those ideas and took them to the administration of the university, making it his goal to reinforce the links between Lua and PUC, a tie that he thinks would be beneficial for both the language and the university.

311 See appendix C, "Original Interview Quotes," (Carlos, April 2007, "Percebi que existia um movimento").

> Carlos: The University wasn't really seeing the importance that Lua was having in the international context. And the members of the administration agreed with this. And I mentioned that we need to try and see what ways there could be of strengthening this association—between Lua and PUC.[312]

Carlos refers to his efforts as "evangelization." "You have to win people's hearts, people's minds," he explains.

Stronger Lua would benefit not only PUC, but also the city and the country, stressed Carlos. It could help curb the prejudice often applied to local technology—the kind of prejudice that Silvio discusses in an earlier quote. Unlike the younger engineers and scientists Carlos was not afraid to come across as someone whose technical judgment was clouded by his concerns about the future of Brazil, speaking enthusiastically about the opportunity of local economic development that Lua created:

> Carlos: The perception that if we can unlock the value [*valorizar*] of this language, it will serve for everyone's benefit, for Rio de Janeiro and for Brazil. Because a peripheral country always has this... Always: "Oh, no, this is just made by a Brazilian." You buy a product in Brazil, if it breaks you say: "Ah, it's Brazilian." You buy foreign equipment, a car, you say: "Ah, look how wonderful." If it breaks, the person almost doesn't even say anything. But if you bought a Brazilian product [and it breaks], you say: "That's because it's Brazilian." And this applies to software too. Despite our great position in banking software, in the financial sector, etc... But Brazil is not very aggressive in this *off-shore* field.[313]

According to Carlos, Lua could change this perception, leading to new economic opportunities for the region.

This change, however, would require support from PUC's administration, policy

---

312 See appendix C, "Original Interview Quotes," (Carlos, April 2007, "Evangelização").

313 See appendix C, "Original Interview Quotes," (Carlos, April 2007, "Em benefício de todos").

makers and funding agencies, explained Carlos. Local companies were unaware of the opportunity that Lua offers them, quite likely due to the same prejudice. The few that understood this value found themselves unable to find Lua programmers. (Carlos seemed to suggest that the fact that local companies did not use Lua then in turn means that there are little incentives for the programmers to learn it.) This could be changed with some support. "And what I also presented is that maybe we need to make an effort to look for resources," said Carlos. "To bring financing agencies here, tell them: 'Let's train Lua programmers.'"[314] Such training programs, combined with other forms of support to companies interested in offering Lua services to foreign clients could jump-start a new sector in Rio IT market.

The initiative was to be presented a month later in a meeting with the Secretary for Economic Development of the State of Rio de Janeiro:

> Carlos: ...where one of the proposals that we brought for him for the development agenda would be the issue of information technology, but with a focus on Lua, as Brazil's differential for *off-shore*. Java could be in China, in India... India has this ease with language, which here in Brazil—perhaps we do not have that. But without any question, Lua is a differential where today we have the conditions to quickly form a critical mass, if we can articulate this. [...] We'll invite financing organs, in the area of science and technology, the National Bank of Economic and Social Development, so that they come here, and bring the companies that are already in the process of religious conversion [*catequese*]... [Laughs.] For them to also talk about their interests. In other words, you would in a way be underlining and certifying that this is a Brazilian product, open source, and even [showing] this creativity here in Rio de Janeiro.[315]

314 See appendix C, "Original Interview Quotes," (Carlos, April 2007, "Programadores de Lua").

315 See appendix C, "Original Interview Quotes," (Carlos, April 2007, "Uma proposta").

With some support, Rio software companies could start offering outsourcing services in Lua, benefiting from their proximity to Lua's base. As Lua would grow further (in part thanks to this local business activity) and as long as its national origin is properly highlighted, Lua's success could help change the perception of Brazil and Rio by displaying the technical creativity that can be found in Rio de Janeiro.

While wanting to position Lua as "a Brazilian product," Carlos was also global in his thinking, feeling that Lua and PUC needed global partners to make Lua a true success.

> Carlos: And also to start looking for large partners, right? Because if Lua has a great reach at the international level, if it has a series of qualities that the community recognizes, why not attract heavy-weight partners, big international *players* [English], like Softnet, IBM, to talk about this topic.[316]

As a step along this plan, Carlos had recently negotiated, on remarkably good terms, research and development funding from Softnet, which was to go towards projects using a combination of Lua and Softnet's technology.

When I discussed the Iris project with Roberto Ierusalimschy, I learned that he was less then excited about the idea. While this had partly to do with his fear that Iris would end up competing with his own research group for "Lua" funding,[317] he also feared the potential costs to Lua itself. "I'm not sure whether PUC has a clear notion of what it wants to do with Lua," he explained. Even with the best intentions, the project could harm Lua.

316 See appendix C, "Original Interview Quotes," (Carlos, April 2007, "Parceiros de peso").

317 Roberto: For instance, Iris submits a project to somebody. And for the agency, this is "Lua at PUC." So [for them it's]: "I gave money for Lua at PUC." And then there is another project, our project for Lua at PUC: "Oh, we already gave money for that, we are not going to give money again for the same stuff." Outside, they are not going to see the difference.

> Roberto: For instance, because legally, they... PUC is the copyright owner of Lua. So for instance, if they write any contract binding Lua—the name "Lua"—with something... For instance, I don't know, but for instance IBM[318] presents: "Oh, I'm very excited with Lua! Let's do an exclusive contract for something, something about Lua." And then: "Oh, it happens that there is a contract now between PUC who is the legal owner of Lua and IBM that such and such development around Lua only can be done inside this."

PUC had absolutely no experience with free / open source software, explained Roberto, and hardly much experience with software licensing in general. In other words, lacking understanding of how free software worked and how Lua fit in the world of free software (an issue that Roberto himself felt he barely comprehended, as we will soon see) PUC's administration risked creating the wrong kind of ties between Lua and the local context which it was finally starting to separate itself.

Among other things, such contracts would highlight the image of Lua as software from "a South American country"—a product potentially enmeshed in the bureaucratic complications that one could only successfully navigate by having strong local connections.

> Roberto: For instance, that could be very bad for the image of Lua. Suddenly a lot of people in the world are using Lua and there is a risk that IBM could accuse them: "Oh, you cannot do that because it's under a contract between me and PUC..." This is a very, very big risk...
>
> Because one of the main problems Lua had... I think now finally it doesn't have it anymore, was that people were very unsure about using a software from... as you say, from a South American country, they don't know how we think and things like that. So they are afraid, "Can we use Lua? Is it really free? What they are going to do?" Things like that. *You* told me, you know that.[319] And then after all these years we kind of

318 Roberto appears to use IBM as just a hypothetical large company. In particular, IBM is not "Softnet."

319 Roberto is referring to our discussion of my interview with Rich, quoted earlier in this chapter.

> conquered some kind of credibility. “Oh, ok, Lua seems to be something stable. A lot of people use it, so I don’t think... there is not going to be any problem.” So more and more people are starting to use Lua. And more important, more and more people are starting to *admit* that they use Lua. And then suddenly it transpires that there is a hidden contract done in South America between IBM or Softnet or Microsoft or whatever and PUC, which is the legal owner of Lua, that... whatever it is in the contract [laughs], it doesn’t matter very much...

Since Lua had been released under an open source license (with the tacit agreement of the PUC administration), the question of who owns Lua’s copyright could turn out to be quite irrelevant.[320] The *uncertainty* created by badly written contracts (or simply bad publicity around contracts written well) could be enough to scare off Lua’s users. “Nobody wants to go to court in Brazil to try to use Lua,” said Roberto. “Even if they know ‘Oh, I’m almost sure I’m going to win that.’”

Concerned with unlocking Lua’s value and using it for everyone’s benefit, Carlos wanted to “reinforce the connection” between Lua and the local context in which it was born. This association, however, was precisely what had to be undone for Lua’s global success. As Silvio said, to succeed in Brazil, Lua needs to first succeed abroad. To succeed abroad, however, it had to avoid any associations that could go with “software from South America.” Such successful detachment, however, reduced Lua’s ability to change the perception of Rio and Brazil. “What I would want to happen would be that PUC got a better notion of the outside value of Lua before a notion of the inside value,” Roberto said. PUC could do little to enhance Lua’s global position, but could do much to hurt it.

---

320 As with other open source projects, the only real property involved is the trademark for the name “Lua.” Others can release a modified version of Lua, but they would have to call it something else. (The Lua trademark belongs to PUC.)

When I returned to Brazil in the end of 2008, the conflict around Iris seemed to have been largely worked out. Carlos appeared to have understood some of the challenges faced by Roberto—the challenges to which I turn in the next section. He started consulting Roberto more closely on PUC's approach towards Lua and the ambitions of the Iris project were largely scaled down.

Reading Smoke Signals

Despite Lua's permissive license and the active interaction between the authors and the Lua community, Lua never fully moved to the open source development mode: all changes to the Lua code itself have always been made by the three members of the Lua team, mostly by Roberto Ierusalimschy. Roberto explains this decision in part by the need to keep the language small:

> Roberto: Yeah because I think most other programming languages—open source—they are much more open than Lua. So they are... For instance, in Python or Perl, you have a lot of people that actually vote for changes and there are those kind of open decision, open-source decision-making strategies and things like that. You can enter as a committer and you are promoted as a developer and then you have the right to go and there is all this hierarchy. And Lua is just the three of us... [Laughs.]
>
> Yuri: Have you ever thought of doing that?
>
> Roberto: No. I mean, in what sense? We thought about *not* doing that. [Laughs.]
>
> Yuri: Ok, but you're saying this is a possibility that was discussed and you decided not to?
>
> Roberto: [...] Oh yeah, yeah. We decided not to… [Laughs.] [...] Because exactly... for several different reasons. One is exactly that I don't think that this is necessarily good for a language. I think that always it tends to

> grow. It's very difficult for people to agree on what to remove from the language so I think this is one of the main points.

As Roberto explains (and many users agree), open source development works well for *adding* features but makes it harder to control the growth of the language, not to mention *removing* features. Since Lua is generally recognized for its minimalism and small size, it may in fact benefit from the "cathedral" rather then the "bazaar" development model (see Raymond 1999).

Roberto then adds another reason, however: his desire to maintain control over Lua's future.

> Roberto: The other main point is that we really like—I mean, someone once said that in kind of a very aggressive way, not that aggressive, but... It's our language, I mean, we like doing it and it's…
>
> Yuri: Someone said what? That it was your language?
>
> Roberto: Yes, but it was kind of "Nobody has nothing to do with that, it is his language, he does whatever he wants and he doesn't care what people think about it" [says imitating aggressive voice] and I mean, I care a lot about what people think about it, but I really want to keep this privilege of—this is the language I developed, I want to have the language the way I want it.

While Roberto's desire to develop Lua as he wants it could be understood as a case of an artists' concern for the purity of the work, it connects to a concern that comes up frequently in the interviews: the risk of losing control over Lua.

There are two sources of this concern. One is the inherent uneasiness of Lua's position as an international programming language with a base in the wrong place—"software from South America" trying to make it globally. The other is the Lua team's

limited knowledge or understanding of Lua's use abroad, and especially of the social dynamics of that use.

At the beginning of our first 2007 interview, Roberto suggested starting with a different question from the one that I first asked him:

> Roberto: I think maybe start a little earlier? Because this is something that I was thinking today and something that I am always thinking. The main point is that we have a very, very rough idea of the growth of Lua and how Lua is being used and things like that. We are always kind of... I don't know if this is because we are in Brazil or if it would be the same if you were living in Silicon Valley, but my impression is that I always kind of try to read smoke signals to try to realize that there is a real growth, there is no real growth.

While Roberto says this issue is always on his mind, it turns out that a specific event has led him to spend time thinking about the topic earlier that day.

As Roberto proceeded to tell me, over the course of recent months (from November 2006 to March 2007), Lua had enjoyed a dramatic change in its position in TIOBE TPCI—a popular ranking of programming languages based on Google queries.[321] After spending a long time in the group of "next fifty" (languages that TIOBE rates as among "the top one hundred" but not to "the top fifty," without assigning them individual rankings), Lua entered the top fifty in December 2006, and then started a slow ascent within this group. The March 2007 ranking came out the day before our interview. Lua had made a dramatic jump: to the twenty fifth position. Since most of the software development is done in just a small number of programming languages (see chapter 2.2), being in twenty fifth position did not imply huge market share and did not yet qualify Lua

321 http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

for TIOBE's "A level" designation. However, it most certainly put Lua on the map, leaving it steps away from the doors of the most exclusive group in software: "major" programming languages.

This new success was so substantial, that Roberto laughed in disbelieve as he talked about it:

> Roberto: [Laughs.] I am not sure if the index is wrong, I mean, it's a very, very big jump. It jumped over twenty other languages in one month. Very strange. […] It's very strange, I'm not sure if they are right. But the main point is I have no idea how we are climbing up, what happened in the world that put us that much, I mean... We have a thin idea that there is a new book about Lua that is *Beginning Lua Programming*. Actually, this is the first, how can I say, the first outside book that is really, really about Lua.

The reasons for Lua's recent success were largely a mystery to Roberto, who had to read "smoke signals" from a distance to learn whether Lua was growing and what factors might have been contributing to its growth. He similarly had little idea if Lua's use will continue to grow, how quickly, and if so, what this growth would bring.

Apart from the discussion on the list, publications and blogs are the main source of information. "You're always trying to understand what is going on," he says laughing. Roberto relates a story about Cameron Laird, a well-known columnist writing about programming languages development, mentioning Lua in his column in 1998.

> Roberto: And then in '98 in one of his columns he wrote about Lua: "It's a very nice language, still has a very, very small base of some tens of thousands of users."[322] At that time for us—in '98—to learn that we had

322 "Lua is slimware; all the source code totals well under 10,000 lines. Its user base is also small; there might be only a few tens of thousands of Lua programmers in the world. They're very fond of this language, though, and the imminent explosion of ubiquitous embedded processing (computers in your

> tens of thousands of users! We were discussing it: if we have one thousand users, maybe… "Let me see, I think we have a few hundred users, maybe one thousand." And then this guy says: "They have very few users, just tens of thousands." We were: "How?.." We are always kind of... [...] I try to *acompanhar*, to follow blogs, and I have a link in my bookmarks. There is a search in Google blogs for "Lua" and "programming" or "games" and every day I go to see news about Lua in the blogs to try to have a feeling if there is something new happening or things like that. And sometimes I see: "Oh, this is interesting." But sometimes I see something that I get surprised: "Oh, there is..." It's difficult to...

Following Lua news from an office in Rio de Janeiro takes some work.

While Roberto wants Lua's popularity to grow further, this difficulty of understanding the factors underlying this growth makes this growth itself a source of uncertainty. Lua community members often mention the many occasions when Roberto had said that he would perhaps prefer Lua's popularity to grow *slower*. For example, after Lua's entrance into "top fifty" in October 2006, one of the list members asked: "How long before Lua gets into the top 20, I wonder?" Roberto replied to the list: "Veeeeeeeeeery long, I hope ;)"[323]

While such uncertainty is inherent in development of free software (since its use cannot be easily tracked through sales), especially when it is done on a small budget without the possibility of expensive market research, Roberto's situation is made more serious by his relative isolation from the place where his programming language is used most actively. Apart from the interactions on the list and the occasional Lua workshops

---

car, in your plumbing, and in your kitchen appliances) can only work in favor of Lua." (Laird & Soraiz 1998.)

323 In the months that followed this interview, Lua inched its way up the rankings, entering "top 20" in July, taking the 18th position. It has since maintained a position at the bottom of top 20, moving only slightly up or down every month.

(which happened in 2002, 2005, 2006 and 2008, three times in U.S. and once in the Netherlands[324]), Roberto has little contact with the programmers that use the language that he designed.

This lack of face-to-face contact (and the mystery that results from it) is illustrated most starkly when Roberto talks about one of the list members, whom he mentions as a source on influence on Lua's design:

> Roberto: Yes, in the list there are some people that... At least Joe Simon [pseudonym]. Joe is someone I really respect. This guy—it's unbelievable again. Luiz Henrique jokes[325] that this is really a group of ten people, because it's unbelievable! He knows everything, he answers everything in fifteen minutes... The most exotic questions about operating systems, about... A few days ago, someone asked something about Microsoft email program [Outlook] [...], and I never heard about... he answered in ten minutes: "Oh, you go to that menu, you use that option, and then that solves the problem." How does he know that?! [Laughs.] And he knows everything about operating systems, about architecture: "The Pentium 4 has a specific hardware circuitry for that operation, but in the other one it gets off the other, and then the cache…" He knows... It's unbelievable how much he knows.

In addition to his incredible ability to answer the most obscure questions, Joe has developed a tool used by many Lua users. Despite those contributions, Joe remains a mystery to Roberto, who has never met him.

> Roberto: That's one of the reasons we think he's ten guys. [Laughs.] We never met him. He's very secretive. I never saw a picture of him, he doesn't have a home page.

---

324 The first of those workshops (at Harvard) focused narrowly on improvements to Lua's API and is typically not counted as one of the "Lua Workshops." The other three (in San Jose, CA in 2005, in Venlo, Netherlands in 2006, and at George Washington University in Washington, D.C. in 2008) were aimed for a more general discussion of things happening in the Lua community.

325 Luiz Henrique points out that the joke was actually made by someone else.

Joe, whom some Lua users assume to live in Europe (based on nothing but his email address), is "secretive" according to Roberto and has not attended any of the Lua workshops despite his active involvement with the language. He may well have remained a mystery to Roberto even if Roberto's home was in Palo Alto, California. Being in Brazil, however, ensures that most members of the community have much of the same mystery that surrounds Joe's persona.

To understand the community around Lua, Roberto finds that he has to get a better understanding of the "culture" of open source. He does so in part by reading a book about open source:

> Roberto: Yeah, because there is this *culture*. For instance, I have this... I just... [Rodrigo] Miranda gave me a copy of this book called [...] *How to Run an Open Source Project*.[326] It's assumed that an open source project is something that is open source decisions and... It even considers the possibility of... what they call... [pause] "the benevolent dictator." Is that the name? But it must be *benevolent*, that dictator. [...] And it's exactly because of that, he said in the book, because there is always this... the possibility of fork if the dictator is not benevolent, or if it's *really* a dictator. People will fork to another project and so... I mean, you cannot be a dictator, so I mean. So he puts that as almost a result of an open source project—that it will have an open source development, decision-making organization.

The book assumes, says Roberto, that the development of an open source project is driven by what Roberto calls "open source decisions"—a communal decision making process in which no single person can make unilateral decisions. Projects that do not adhere to those norms run the risk of "forking."

The book explains forking and its effect on the decision making the following

326 Fogel (2005), *Producing Open Source Software: How to Run a Successful Free Software Project*.

way:

> The indispensable ingredient that binds developers together on a free software project, and makes them willing to compromise when necessary, is the code's *forkability*: the ability of anyone to take a copy of the source code and use it to start a competing project, known as a *fork*. The paradoxical thing is that the possibility of forks is usually a much greater force in free software projects than actual forks, which are very rare. Because a fork is bad for everyone [...] the more serious the threat of a fork becomes, the more willing people are to compromise to avoid it.
>
> Forks, or rather the potential for forks, are the reason there are no true dictators in free software projects. This may seem like a surprising claim, considering how common it is to hear someone called the "dictator" or "tyrant" in a given open source project. But this kind of tyranny is special, quite different from the conventional understanding of the word. Imagine a king whose subjects could copy his entire kingdom at any time and move to the copy to rule as they see fit. Would not such a king govern very differently from one whose subjects were bound to stay under his rule no matter what he did? (Fogel 2005, p. 88.)

When software developers release code under a free software license, they do not simply share the *product* of their work. By sharing the source code and by growing a community around it, they share *nearly all there is to the project*. A free software license allows any recipient of the code to not only use or share it, but also to *modify* it as they see fit. Having modified it, they can make a case to other users that their version is better than the original. If the leaders of the project reject the modifications, the users can still make a choice to use the modified version.[327]

Convincing the user community to abandon its current leader is a hard task for the claimant, as it requires, at the very least, a combination of demonstrable technical

327 In fact, as Raymond (1999) points out, in a section called "Promiscuous Theory, Puritan Practice," the right to create forks is essential to an any open source license and such licenses can be said to "implicitly encourage" forking. At the same time, he argues, forking rarely happen in practice because of "an elaborate but largely unadmitted set of ownership customs" and a set of "taboos."

prowess and communication skills, lacking which the claimant will likely be ridiculed.[328] (One derivative of Lua was subject of such ridiculing by the mailing list members on a number of occasions in 2007 and 2008.) Those who create forks thus typically present them as subordinate projects, downplaying their significance, for instance by calling them "experiments." Over the years, the members of the Lua community have undertaken a number of such "friendly" forks. Most of them, says Roberto, had a positive effect on Lua, as they either helped develop new ideas later incorporated into Lua or demonstrated certain proposals to not be a good idea. The possibility of a successful adversarial fork that leaves the project leaders without the community, however, is ever-present. It looms especially ominously for leaders who may trust their own technical instincts but not their grasp of the *culture* of their users—or find themselves unwilling to accept what they see as the demands of this culture.

> Roberto: Yeah, I learned a lot about exactly... For instance, I didn't know that there... is so strong this culture, of exactly what the meaning of "open source" is for some people. I thought that Lua of course was open source. Now I say: "Lua is open source but it may not be what you think it means to be open source."

After reading the book, Roberto decided to take the risk and proceed with what he feels is best for Lua, disregarding what he understood to be the author's advice. He remained worried about the consequences of that decision.

Roberto's experience with open source illustrates the challenges that open source presents for software developers at the periphery of the software world. While open

328 Such skills represent the minimum requirements, since a key extra factor is usually needed: a justification for what is often perceived as a clear violation of a social norm.

source software development presents many opportunities for such participants, producing open source software successfully requires *higher* levels of competency in the software culture than other forms of engagement in software development. The developers working for Alta (chapter 3.1), who produce commercial software for their local clients, must only project a competent image to their clients and local peers. Their clients typically have limited grasp of the software culture and, in most cases have few options for looking for expertise outside Brazil or even Rio de Janeiro. The authors of Lua cast their lot with a community of developers based largely outside Brazil and comprised of many people who are *more* fluent in the software culture than Lua's authors themselves. A firm grasp on the software culture becomes crucial for success.

Ginga

When I returned to Brazil at the end of 2008, I learned of a new development that promised to substantially increase Lua's use in Brazil: the language had been included in the Brazilian digital television standard. In the short term, this meant that companies interested in selling receivers for digital television signal in Brazil ("set-top boxes") had to support Lua in their products. In the longer term, this promised the creation of a local labor market for Lua programmers in developing content for interactive television. While I heard of such plans during my visits to Brazil in 2005 and 2007, many of my interviewees seemed to dismiss them as just talk. At the end of 2008, however, Lua's place on Brazilian set-top boxes seemed all but assured.

Lua's inclusion in Brazilian digital television standard did not come from specific efforts of the Lua team. Rather, Lua entered the standard as a component of "Ginga"—a digital television system developed at PUC-Rio.[329] From the late 1980s, Luiz Fernando Gomes Soares, a PUC professor who once served as the Master's advisor for Roberto Ierusalimschy, had worked on the problem of representing the structure of multimedia presentation. In the late 1990s, this work resulted in NCL ("Nested Context Language") —an XML-based language for expressing hypermedia presentations, similar to a popular format called SMIL, but offering a different way of representing temporal relations between multimedia objects.[330] While NCL did not get adopted by the foreign industry as Lua was, Luiz Fernando and his collaborators published a large number of papers on their research, many of them in foreign computer science conferences.[331]

In 2004, Brazil's government announced a call for research proposals related to the future Brazilian Digital Television System (SBTVD). While the lower layers of the system (at the level of signal modulation) were to be based on the system earlier developed in Japan (ISDB), the government was looking for local innovation, in particular at the higher level of interactivity. Luiz Fernando's laboratory submitted a proposal and NCL was accepted as a research project and then as a basis for a "middleware" of SBTVD under the name "Ginga," in reference to a basic move in

329 To be more precise, Ginga represents a particular layer in the digital television system, typically referred to as "middleware."

330 The most important consequence of the difference in temporal semantics between NCL and SMIL is that NCL makes possible live editing of the presentation. That is, the editing of a SMIL presentation must be completed before the presentation is viewed, while NCL makes it possible to edit the later parts of the presentation while the earlier parts are already being displayed. This is ability is important for live digital television.

331 For example, Casanova et al. (1991), Santos et al. (1998), Costa et al. (2006).

capoeira.[332]

As with Lua, foreign success has been crucial in making NCL-based middleware accepted by the local industry, says Luiz Fernando.

> Luiz Fernando: The first big difficulty that we had was that people didn't believe that we had actually built something much better than what was being proposed abroad. It was very interesting that Brazil would not believe it and it took people abroad to believe and say so first, for Brazil to later accept it.[333]

Foreign approval of NCL, expressed in publications citing the work of Luiz Fernando and his colleagues helped gain local support.

NCL is a "declarative" language, which focuses on describing the multimedia objects and the relationships between them. To be truly interactive, however, the system needed a procedural language that could be used to program the behavior of the objects. Java was an obvious candidate. "Java is a language that is used by practically all TV systems, digital TV systems," says "Michel," one of the Ginga collaborators at PUC. "So, to maintain compatibility, Brazil decided to also have a Java API in its middleware."[334] A research group at the University of Paraiba, headed by a PUC-Rio graduate, had already

---

332 Ginga is designed to work with terrestrial television, that is signal that is sent by an antenna on the ground and picked up by a simple television antenna. Different countries use different several standards for digital terrestrial television (DTTV). Brazil is basically introducing the Japanese system, though adjusted for higher resolution. However, national DTTV systems in current use do not provide much interactivity. The interactive televisions systems that exist (e.g., those provided by private cable companies in the US) typically require a "return path," that is some form of broadband Internet connection (cable, DSL) or telephone. Ginga offers a degree of interactivity without a return path: the code is transmitted by air and is executed on the set-top box. Similar solutions have been proposed, but Ginga is actually the first open implementation.

333 See appendix C, "Original Interview Quotes," (Luiz Fernando, January 2009, "O Brasil não acreditava").

334 See appendix C, "Original Interview Quotes," (Michel, July 2007, "Pra manter interoperabilidade").

done much research on putting Java into set-top boxes. The two universities started working together, developing a system that combined NCL and Java.

PUC researchers, however, decided to also add Lua. Michel points out two reasons: the personal connections to the Lua group (located in the adjacent building) and Lua's obvious fit for the task.

> Michel: Well, Lua comes in first of all because it was easy for us to work with the Lua community, which also is a language developed here at this university. Second, because we started seeing that we needed for the terminal, for the set-top box, something that would be light, that would be efficient, that wouldn't occupy much memory. And we started to notice in the comparisons done by third parties, sites that do comparisons of scripting languages, that Lua is very well received for those characteristics, for being a light and efficient interpretor. [...] With digital TV, people soon start thinking of entertainment, of applications, games, those sort of things. Those are interesting applications to start thinking about, running them on the receiver. Lua being much used in the game industry, this again had much to do with it. So, it was because the sum of those characteristics.[335]

Indeed, set-top boxes provide a context in which Lua is strongest and quite similar to the context where Lua is used most often—computer games. Companies outside Brazil include Lua in their set-top boxes without any government mandate.[336]

Michel later offers a different explanation, however: there was little cost to including Lua:

> Michel: Java [...] has a heavy side in terms of the virtual machine. But if you don't put Java, you already limit applications. The broadcasters, they are worried also about importing programs, with compatibility. So, if you

335 See appendix C, "Original Interview Quotes," (Michel, July 2007, "Usado por essa indústria de games").

336 The examples include Verizon FiOS (http://www.verizon.com/fiostv, see Perez 2009), Active-TV (http://active-tv.org/) and Vudu (http://www.vudu.com/, see Hessel 2008).

> remove Java from the system, you already exclude a number of interactive applications. And Lua is something that doesn't weigh anything. So, whether you put Lua or not put Lua... there is practically no advantage to not putting Lua in. So, if there is already something that can do work well, that is already integrated into NCL, there is no harm in adding it. It's very light, just 50K. [...] So, it's there as one more option.[337]

While NCL was already struggling for acceptance as a local technology, inclusion of *another* local technology did not burden the proposal too much. Lua was just "one more option."

When I arrived for my interview with Luiz Fernando in January 2009, I found him in the midst of a meeting. As it turned out, he was meeting with representatives of a company that wanted to license PUC's implementation of Ginga to offer services to foreign hardware manufacturers interested in selling set-top boxes in Brazil.[338] All of those set-top boxes were to run Lua. As I waited for the meeting to end, I passed my time looking at the poster on the adjacent door—it advertised the course on NCL programming, which was offered at PUC in the Fall of 2007 and included an introduction to Lua. The existence of "a Lua course" at PUC was an important news, relayed to me diligently by a number of my former interviewees. The course used the Portuguese translation of the Lua manual, completed by one of Roberto's Ph.D. students in August 2007. After the interview, I passed by LabLua to chat with some of Roberto Ierusalimschy's students. I saw *Programming in Lua* open on one of the last chapters. The

337 See appendix C, "Original Interview Quotes," (Michel, July 2007, "O ganho de não botar Lua").

338 PUC's implementation of Ginga is available under GPL license. This means, that companies can use it without paying royalties to PUC, but are required to release their implementation under the same license (if they were to distribute the software). Those who do not want to do so have an option of obtaining the software from PUC under a different license but for a fee.

same student who translated the Lua manual was finishing the translation of the book.[339] The translation work was not related to Ginga—it was paid from Kepler's grants. Local efforts to promote software based on Lua, however, seemed to start adding up. Late that day I gave a well-attended talk about my own Lua project, organized by Chico, whose research lab, dedicated to running Lua on small electronic devices, had expanded to several rooms. Having gained success abroad Lua was starting to appear ready to come home.

339 The draft of the translation was completed later that month.

3.4 Glocal Dreams

"World domination," Rodrigo said with a smile, his tone suggesting that the answer was obvious. I had just asked him to clarify what purposes he had in mind when he jokingly raised the question of whether knowledge of Chinese or Japanese would be more important for "his purposes." It was Wednesday afternoon in May of 2007 and we were at an *a quilo* restaurant in Copacabana, a typical Rio lunch place selling food by weight. A block away was the office high-rise where Rodrigo's company occupied a few small rooms and where both of us had been spending most of our time for the last few months, sharing a tiny office, the backs of our chairs just about a foot apart. It was perhaps a somewhat odd place to be plotting world domination. We laughed. A few minutes later we got up, paid, and headed back to the office. We did have a world to conquer, after all.

Rodrigo, whom the reader has met on a few occasions in the earlier chapters, is a *carioca* in his late 30s who has dedicated the last decade of his life to building a web development platform based on Lua, the programming language discussed in the previous two chapters. Since 2005, Rodrigo has pursued this goal in the form of an open source software project called "Kepler." I first met Rodrigo during my time in Rio in 2005, he was in fact my first technical interviewee that year. While finding the project intriguing, I dismissed it as an outlier, choosing to dedicate my time to interviews with developers working in more typical companies. I stayed in touch with Rodrigo, however, and as I

learned more about Kepler over the course of the following year, I gradually came to the conclusion that observing a project aiming to develop a global product based on local innovation would provide a useful contrast to a study of a "typical" company, focused on bringing foreign technology to Brazil. (See chapter 3.1.) When I returned to Brazil in 2007, I decided to accept Rodrigo's invitation to study his project and his offer of a desk in his office, using it as a base for my investigation of Lua (see chapters 3.2 and 3.3) as well as for looking at Rodrigo's own project—the subject of this chapter.

Kepler and the company that has sponsored it stand in important contrast with both Alta and Lua. In contrast to Alta's founders, Kepler represents an attempt to create a global platform—a software product developed for the world rather than for the needs of specific local clients and meant as a foundation for other software development rather than a product in itself. While Rodrigo's talk about "world domination" is a joke (as are his colleagues' references to him as "the Brain" of "Pinky and the Brain"), the project does have global aims, positioning itself in competition to globally popular web development technologies.[340] Its audience was also primarily outside Brazil.

Unlike Lua, however, whose success can be attributed to a combination of pre-existing foreign ties and successful disengagement from the local context, Kepler had to create crucial local alliances, on which it came to rely. One of those alliances involved betting on Lua itself—a programming language developed just a few kilometers away from Rodrigo's current office. Rodrigo also decided to work together with a local

340 To fully appreciate the challenge, one must note the massive role of network effects for software platforms. In other words, the value of a software platforms depends on the number of people using it. See Katz & Shapiro (1986) for the general discussion of network effects.

company oriented towards Brazilian clients, and to rely on funding provided by the Brazilian government.[341]

Paradoxically, local use of local innovation is itself a starkly non-local move. Using local innovation is something software companies in Silicon Valley do all the time. It is not commonly done in Brazil. Rodrigo's project is thus simultaneously global and local in important ways—"glocal" to use Wellman and Hampton's (1999) term. This chapter shows some of the challenges involved in pursuit of such "glocal" dreams.

Attempting to bring together the resource of local universities, the local industry and the national government in pursuit of innovation that aims to be global in its significance, Kepler represents the kind of innovation that may be crucial for development. Understanding its challenges may thus provide valuable insights into the dynamics of technological development at the periphery.

Making Waves

Rodrigo Miranda grew up in Rio, as a son of an engineer and a journalist. As a child, he wanted to become an architect, but an encounter with a computer at the age twelve led him to a new passion. (See chapter 2.1 for a discussion of Rodrigo's early days as a "nerd.") By the time he was choosing a college program, Rodrigo knew he wanted to study computer science. He was planning to apply to the public Federal University of Rio

341 Lua of course was also to some extent funded by the Brazilian government and PUC, itself an actor tied closely to the local context. In Lua's case, however, the funding system appears to have stabilized a while back, now forming a part of the infrastructure that is almost invisible to the actors. By contrast, Kepler's alliances with the funding agencies are new and highly problematic, requiring re-thinking and re-negotiation.

de Janeiro (UFRJ), but PUC-Rio opened an undergraduate program in computational engineering at the last moment.[342] Since his mother was teaching at PUC at the time, Rodrigo managed to get a scholarship to study there.

Rodrigo sometimes describes Kepler as the third of the three "crazy ideas" that he had pursued over the years. The first, an idea for a video game, had to be put on hold back in college. Rodrigo did not believe he could do it all by himself and could not convince his friends to join forces.[343] The second vision, a hypertext database system, was abandoned in 1994 when Rodrigo saw the Mosaic web browser. The web, however, gave Rodrigo his third idea, one he had has pursued for over a decade since: a web development platform based on Lua.

Lua, whose story was presented in the two previous chapters, is a programming language developed in Rio de Janeiro by a team of three people, headed by Roberto Ierusalimschy, who advised Rodrigo during his undergraduate and master's studies, as well as through Rodrigo's two-year stint as a Ph.D. student at PUC. As described in chapter 3.2, Lua is today used primarily in computer games, a domain for which it is often seen as being particularly fit because of its performance and ease of integration with C. Few software companies in Rio de Janeiro, however, build products of that kind. Instead, most provide software services, which typically involve development of web

342 PUC earlier had an undergraduate program in "informática" but Rodrigo found it outdated. According to a PUC faculty member, department's "informática" program is generally more popular among lower middle class students, perhaps because it is taught at night. The newer computational engineering program is considered substantially more prestigious.

343 After reading a draft of this chapter, Rodrigo commented that while the project had been on hold for years, he had never fully given up on it. He could not find money at the time and his friends would not work for free. "But now there is the 'open source way,'" Rodrigo commented.

applications. Lua is rarely seen today as a good foundation for web development.

Yet web development in Lua has a long history. The first attempts were made in 1995, by a group of PUC students. According to "Cássio," one of the first students to apply Lua to web development in a project called "LuaForm," the effort started as "a game among friends," driven primarily by a desire to understand the emerging web technologies. Cássio was pursuing his Master's degree at the time, while working at Tecgraf, together with a number of friends. He and others working at Tecgraf at the time describe the laboratory as an intellectually stimulating place where work projects, academic research and things done just for fun transitioned one into another easily. "So, in this environment we produced this toy [*brincadeira*]. It was called LuaForm at the time. Just because we wanted to have a way of handing HTML forms," he explained to me. "Just simply to learn, to discover how things worked then."

Little by little the project grew features. "And from there, one thing leads to another, right?" Cássio explained in our interview.

> Cássio: Given that you already have the data in the Lua environment, with pretty parameters, why don't I... I could just do a *print*, right? And that would produce a response to this form. So, it would write the page based on what I would put in the "print" below. So, from here it could be made even easier, because instead of writing "print '<H1>'," it would be easier to have something generate those *strings* for me.[344]

The project was then picked up by another student, who developed it into a Master's thesis (completed in 1998) and published several papers about the project (Hester et al. 1997, 1998).

344 See appendix C, "Original Interview Quotes," (Cássio, April 2007, "Uma brincadeira entre amigos"). Cássio's quotes from the previous paragraph are included in the same original quote.

CGILua, as the extended system came to be called, had a number of advantages over the better known alternatives, and in particular over writing web applications directly in C, a common practice at the time. While the first iterations of many currently popular methods for building web applications were being developed around 1996, none were really well known. For a short time, CGILua was about the best approach one could use for building web applications.[345] As with Lua, CGILua used English keywords and the papers about it were written in English. Other documentation, however, was written in Portuguese and the software was never widely advertised outside PUC. It was not announced on the Lua list until 1999, when the documentation was finally translated into English.[346] At the time when CGILua started getting attention on the list, the student who was the primary author of CGILua had moved to United States and web platforms built in other languages were much ahead. "Lua lost the web wagon," says Roberto Ierusalimschy.

Meanwhile, Rodrigo had quit his Ph.D. program at PUC in 1996 and went to work as a manager in a web development company, which at the time built web applications in C and had a strategy not very different from Alta's. Being acquainted with the students who were working on CGILua, Rodrigo understood the time savings that CGILua offered over building applications in C and proposed using it for the client's projects. Rodrigo says that the manager refused to even consider the idea, finding Lua to be a toy language. Rodrigo proceeded to introduce Lua in the company surreptitiously, while also starting to

345 See Hester et al. (1997) for a discussion of CGILua in comparison to a number of the better known alternatives. The paper does not, however, compare CGILua to PHP—a system that had been available since 1995 but was relatively unknown at the time. PHP was very similar to CGILua in a number of ways and came to dominate web development a few years later.

346 Two messages mentioned it in passing in 1997 and 1998, the first of which was written in Portuguese.

think about the possibility of building a complete web development platform based on Lua. A year later he left the company. Another year later, Rodrigo found a new opportunity to pursue his dream of a web development platform based on Lua.

While Rodrigo was looking for opportunities to pursue his technological dream, his brother "João" was pursuing his own. Unlike Rodrigo, João saw himself as an entrepreneur. His dreams, therefore, focused on starting and running a successful business. In 1997, having earned a small amount of money in a venture that provided software localization services to foreign software companies, João wanted to move to a new level: launch a *product* company based on proper development methodologies. I refer to this company here as "Nas Nuvens," a self-deprecating pseudonym suggested by Rodrigo. "Nas Nuvens" means "in the clouds" in Portuguese, a name that suggests at the same time a degree of disconnection from reality (much like its English equivalent) and an association with Lua through an allusion to the phrase "no mundo da Lua" (lit. "in the Moon world"), a more idiomatic translation for the English "head in the clouds."[347]

João wanted to start a US-style company, hoping to distinguish himself from the local competition by a strict adherence to foreign software development methodologies and, as far as possible, foreign business methods. In a particularly stark attempt at following foreign methods, João decided to build his company around local research—a practice standard in California but rarely pursued in Rio. Having heard Rodrigo's praises of Lua, João decided that Lua could prove to be a key strategic advantage for a software firm based in Rio: close access to the Lua team would help the company stay abreast of

347 The name captures Rodrigo's doubts about whether the venture was ever realistic, but should not be interpreted as signifying my own position as to whether the efforts were worthwhile.

any changes and perhaps influence Lua's development to its own advantage, while PUC would provide a steady stream of interns and employees skilled in the use of Lua. João also decided to start with a focus on the needs of the domestic market, perhaps expanding internationally in the longer term.

As João saw it, the local market needed an easy way to build websites with dynamic content. Rodrigo's experience with CGILua had proven that this could be done with Lua. Using his ties to PUC, João acquired a web publishing system based on CGILua that was built by another PUC student, using it as a starting point for Nas Nuvens' future product. He also managed to find start-up capital from among relatives, friends and a PUC professor. Soon after, Rodrigo joined the company as a chief software architect, seeing the venture as a chance to take CGILua to the next level and build web development platform based on Lua in pursuit of his own technological vision. The two brothers were pursuing two different dreams—one looking to create a global platform with relatively little interest in the financial aspects of the venture, the other looking build a high tech company using a foreign model, less interested in the details of technology to be built. Their parallel pursuit of their professional dreams, attempting to reproduce the practice centered in the same place, allowed for an alliance, much like many of the alliances described in chapter 2.2.

The company's product was innovative for the time and the company successfully obtained small innovation grants from FINEP, an agency that funds industry research. João's superb skills as a salesman then allowed him to quickly attract substantial capital

(about a million reals[348]), hire development and actually build a product, something few entrepreneurs could do in Brazil even at the height of the dot-com boom. As the product was nearing completion, however, João faced a new challenge: finding additional money to actually launch the product, another two million reals in João's estimate. In 2000, as the clouds were starting to gather over the Internet industry, attracting additional money turned out to be a challenge even for João. Banks he turned to asked for no less than 50% annual interest, João told me, while some wanted considerably more. A large foreign company (a household name) offered to invest as much as two million reals, but did not want to do so alone. João eventually found what appeared to be a solution: the Brazilian Development Bank (BNDES) offered to fund the project at the interest rate of 30% a year —an exceptionally low interest rate for Brazil. João knew that the bank would take months to make a final decision, but decided to take the risk, seeing few alternatives. The bank eventually offered João the money, but demanded that he would personally guarantee the loan, while giving the bank an option to convert it into equity if the venture was to succeed. Such terms left João with a risk of finishing with millions reals of personal debt[349] while giving the bank most of the profit if the venture were to succeed. In João's view, this perverted the meaning of "risk capital." He decided to not take the loan and accept that the key ingredient of the California startup recipe was missing in Rio de Janeiro and the plan had to be rethought.

Nas Nuvens was soon out of money and had to lay off many of the developers,

348 A US dollar was worth around $1.80 reals in 1999 and 2000. So, a million reals was a little over half a million US dollars.

349 Note that Brazil has much stricter personal bankruptcy laws than the United States, so an individual with a large personal debt cannot easily start his life over again.

including nearly all of the most skilled ones, who were also the most expensive. João ended the lease on the two floors that Nas Nuvens used to occupy, limiting the company to the few rooms it has occupied since. The company shifted its focus to providing services to the customers it already had—the approach that many consider the only realistic route for a software company in Rio de Janeiro anyway. Even this option, however, was turning out to be problematic. Unlike its competitors, Nas Nuvens had build up expertise in building web solutions using Lua and thus had to rely on CGILua, a rapidly aging platform, and could not provide much of what the clients were starting to expect. The author of CGILua had by then departed for the United States and the CGILua was largely abandoned by the PUC Lua community, which was looking to solidify Lua's success in other domains (see chapter 3.2). (João's expectations of steering Lua's development had proved unrealistic—a tiny local venture was too insignificant, considering that Lua was getting adopted by major players such as Adobe and Microsoft, whose interests were quite different from those of Nas Nuvens.) Shifting towards services and a closer relationship with local clients was also making the job less and less attractive to Rodrigo, who had wisely abstained from financial involvement with João's venture and could find better opportunities elsewhere.

The solution came in the form of government support. By that point the company had already received small grants from FINEP—a government agency tasked with supporting industry research.[350] By 2001, FINEP showed less interest in the company's

350 Observing the lack of collaborative relationships between industry and local research and unwilling to wait for companies to start building such relationships on their own initiative, the Brazilian government requires companies to contribute money to "sectoral funds" which are then used to fund collaborative projects such as the one described in this chapter. (Though typically the funding occurs

product, no longer considering it particularly innovative. In 2002, however, FINEP started providing additional funding earmarked for open source projects.[351] It was thus decided that the company would apply for FINEP funding for an improved web platform based on Lua to be released as open source. The project would make it possible to hire skilled developers from among PUC students and alumni and would satisfy Rodrigo's desire to pursue his grand vision. It would also give Nas Nuvens a better Lua platform, which it could use to provide more competitive services. The project was called "Kepler." As the project's website explained, the name alluded to Johannes Kepler's discovery that tides on Earth were caused by the Moon, and was to suggest that Lua ("moon") was about to cause some tides. For the six years since, Rodrigo has been seeing himself as primarily the leader of this project which he runs in something of a partnership with Nas Nuvens.

The funding provided by FINEP was expected to be rather modest, and the company could not use it to hire as many qualified programmers as it expected. Rodrigo's connections to PUC networks, however, helped him find students and recent graduates already interested in Lua and willing to dedicate part-time efforts for less than the market wage. One of the new project members was "Márcio"—a recent PUC graduate working as a programmer, on a PUC web application based on CGILua. A dedicated fan of CGILua, Márcio had worked privately to improve the platform, without advertising his

---

on a much larger scale, according to my conversation with a FINEP grant officer.) The distribution of those grants is managed by FINEP. In addition to FINEP's sectoral funds, Kepler has relied on money from other agencies, such as CNPq (an agency responsible for funding academic research) and funding agencies of the State of Rio de Janeiro. I avoid a discussion of the complex web of funding relationship, as such a description would require a tome of its own, focusing instead on FINEP, the major source of funding.

351 See Schoonmaker (forthcoming) for a discussion of the Brazilian governments rationale for supporting open source.

efforts. He had little interest in Rodrigo's plans for "world domination." Rodrigo's offer of paying him to work on Kepler, however, provided a good opportunity to dedicate more time to improving CGILua by not having to take on other consulting projects. Others joined the project partly out of interest in Lua and partly as a favor to Rodrigo. (See chapter 3.1 for Alta's early participation in Kepler.)

Rodrigo also returned to the Lua list and resumed his efforts to transform the Lua community to fit his needs. Lua is typically used as a programming language embedded inside a larger C application, taking advantage of Lua's small size, efficiency and ease of integration. Lua's community had consequently emphasized minimalism and use of highly customized solutions. The members of the list had thus often preferred to share ideas rather than actual code, stressing that one project's code would rarely provide a perfect fit for another project's needs. Development of web applications, on the other hand, required a large collection of software modules that could be readily used. Realizing that he could no longer hope to fund the development of all the modules he needed, Rodrigo attempted once again to convince the Lua community of the importance of sharing code and assembling a larger collection of modules. He refers to his efforts as "culture farming." Since Kepler's own software was now going to be released under an open source license, Rodrigo used this as an opportunity to jump start the sharing culture, actively announcing the release of each Kepler module on the mailing list and setting up a website called LuaForge for sharing modules.

When I first met Rodrigo in 2005, Kepler was relatively well-funded and Rodrigo had a number of skilled developers working on the project. A later gap in funding

decimated the team, as most Kepler developers felt that working for free was a luxury they could not afford. (Márcio took a somewhat different approach, continuing his involvement in the project but refusing to accept money for it. This gave him the freedom to pursue other sources of money without constraining himself with a commitment to Kepler, which was too unreliable as a source of income. It also started, however, his gradual disengagement from Kepler.)

While the development had slowed down substantially, the work done in 2005 was starting to pay off by early 2007 as the project was slowly attracting some users. In Spring 2007, the project's mailing list (run in English) had around one hundred people, who were increasingly involved in the project, at least to the point of asking questions. Additionally, the Lua community was gradually warming up to the idea of sharing modules. LuaForge included around two hundred projects, with several new projects getting registered on a typical week. This was giving Rodrigo hope that the project was going to succeed. "My friends used to call me crazy, now they just say I am insane," said Rodrigo.

Opening Kepler

On one of my first days in Rio in March of 2007, Rodrigo told me he had a habit of going to PUC roughly once a week, to talk to the two Kepler contributors who worked there—Márcio, the PUC graduate who was working in PUC's IT department, maintaining an administrative web application based on CGILua, and "Tiago," a Ph.D. student of Roberto Ierusalimschy. (A third major contributor, "Alan," used to be there as

well but had just recently moved to Porto Alegre, about a thousand kilometers south-west of Rio de Janeiro.) Occasionally, he also met with Roberto, Chico and others members of the local Lua community.

A week later, I was in a taxi with Rodrigo, going to PUC where I expected to sit through his typical meetings, observing the routine. As soon as I got into the taxi, however, Rodrigo handed me a printout of a web article, in English, entitled "Financing Volunteer Free Software Projects" (see Hill 2005). I scanned it briefly. The article argued that paying open source developers is a dangerous practice, since paid labor would tend to "crowd out" volunteer contributions. In other words, paying some people would make others less willing to work for free, and even those that are paid might work less, since they would now see their work as an economic transaction. This article, explained Rodrigo, had helped him identify the problem that had plagued his project for a long time. He was releasing his source code under an open source license, but was not running Kepler as a "real" open source project. He was hiring developers with the money provided by FINEP, and they worked while they were paid. When the money dried up (as was the case for most of 2006), the work stopped. Plus, he simply could not hire all the developers he needed: FINEP funding was limited and could only be used to pay people in Brazil. He had hoped to get people from outside to participate, but this had not happened. He had to get them involved. Perhaps this would require that he stopped paying his current developers, even though the project depended on them.

We arrived to PUC and went to meet Tiago, a Ph.D. student of Roberto Ierusalimschy, whom Rodrigo paid to work on Kepler a few hours a week. We found

Tiago in a small room that he shared with three other Ph.D. students. The three of us went into a conference room across the hallway and sat down under a white board covered in set-theoretic formulas, half in English and half in Portuguese. Rodrigo told Tiago that he wanted to start running Kepler as a "real" open source. They were a project with open-sourced code, he said, but a closed-source development method. He wanted to change this. Perhaps the fact that Alan had just moved to Porto Alegre would make this easier: Alan's departure already made it impossible to resolve all the issue in face-to-face meetings at PUC, and they had to discuss most of the decisions by email and IM. Now they just had to take it a step further. He wanted to start discussing more things on the mailing list, openly, explained Rodrigo. Of course they would use English for those discussions. Kepler already had a mailing list operating in English, used primarily for announcements. This list could become the center of the Kepler project, the new forum for the discussions that up until now occurred inside PUC walls. Tiago listened to Rodrigo speak, occasionally asking clarifying questions. He looked concerned, though it was hard for me to tell whether he was unhappy with Rodrigo's plans or with my presence there, after all, we had just met a few minutes ago. (As I learned from my later interview with Tiago, he supported Rodrigo's idea to "open" the project, but was concerned that Rodrigo was spending too much time focusing on the process.)

"This finishes the Weird Ideas of the Week part," finally said Rodrigo, "Now the practical part." Rodrigo and Tiago spent the next half hour talking about specific problems with the launch of Kepler 1.1. Even so, Rodrigo's suggestion of changing the model repeatedly seeped back into the discussion. They needed people to test, Rodrigo

said. Perhaps people at Nas Nuvens could help, but this was also a sort of thing that list members could assist with. Perhaps there should be a list of tasks on the website where people could go and see what needs to be done. Rodrigo gave an example: a new person had appeared on the list, asking what he could help with, but Rodrigo did not know what to tell him. While yearning for outside participation, the project was not organized so as to take advantage of it. There should be a page with tasks, said Rodrigo, some of them in red. In this case, they could tell new people: go to that page, pick a task.

We walked Tiago back to his room, chatted briefly with other Ph.D. students there (as I later found out, Rodrigo was working on recruiting one of them to work on Kepler), then headed to meet Márcio, the other contributor. We were going to meet Márcio outside, near a kiosk that served coffee. Márcio liked short meetings, Rodrigo explained, and thus preferred to meet on a bench rather than in a comfort of a conference room. Márcio had recently asked to not be paid for his work on Kepler, to have fewer obligations to the project. The meetings, therefore, had to be done in the way he wanted to do them.

While waiting for Márcio, I asked Rodrigo about the move to English. It was simply pragmatic, he explained. He had to draw on the developers outside Brazil, so he needed a mailing list in English. He did not have the time to maintain two lists. Brazilians interested in Kepler would know how to read English and usually would know how to write it as well. Plus, he continued, there simply was little benefit to discussing things in Portuguese. Kepler had almost no documentation in Portuguese, so there was no meaningful way of engaging with the developers who did not read English. He illustrated with an example. *Let's say someone would write to the Portuguese list and ask, in*

Portuguese, "Where is the documentation?" What would Rodrigo tell them? He could only point them to the documentation in English. Requiring Kepler contributors to discuss all the project plans in English on the mailing list would perhaps inconvenience them somewhat. Rodrigo was confident, however, that they were capable of doing this and would agree to do it if asked. When Márcio came down, Rodrigo presented him with a much condensed version of what he said to Tiago. Márcio nodded, seemingly finding no problems with this. In somewhat of a contradiction to the new plan of the moving decision making from face-to-face discussions to the mailing list, Rodrigo and Márcio then proceeded to quickly discuss the state of specific sub-projects.

A week later I had a chance to talk to Márcio about the proposed changes. He appeared divided. He did not share Rodrigo's desire to conquer the world, he explained. He did not care if Kepler was popular, he said. He wanted it to be *good*. That was the difference, he repeated: "Rodrigo wants Kepler to be popular, I want it to be good." When Márcio started working on Kepler, he did not expect it to be successful—it was a fun project that also paid some money. Now it had succeeded and Márcio hoped that it would succeed more. It was good for the world to know, he continued, that here in Rio de Janeiro they were doing something interesting. For this reason, he supported Rodrigo's plan, even as he did not see Kepler's future success important to him personally. He agreed that further success would require opening the project to outside participation and moving all decision making to the English mailing list. Doing so meant more work, but the effort was worth it for Kepler's success. But sometimes, he then added, he found himself just deciding not to write.

A week later, Rodrigo and I were again at PUC, sitting at a picnic table not far from the Informatics Department, together with two other people. One was a potential future contributor whom Rodrigo was courting at the time. Another one was Renato, a close friend of Rodrigo always willing to lend Rodrigo an ear when Rodrigo needed a sympathetic listener. Rodrigo talked again about the idea of moving on an open source model and getting people on the list to contribute more. He did not really know how to do that, he said, since he had never run this kind of project before and his knowledge was only theoretical. ("I have read books, articles on the web, talked to Yuri," he said, naming three foreign sources of information.[352]) But it had to be done. Kepler could not just rely on the efforts of local developers funded by FINEP. He had to look for people outside. This "outside" (*lá fora*) was potentially ambiguous, as it could mean either outside the project, or abroad. In this case, however, the two largely coincided. Looking for help outside the project meant looking for it abroad, through Lua's mailing list. As Rodrigo talked, the rest of us just sat listening and asking clarifying questions. Our job was to help Rodrigo feel that his plan was not altogether crazy.

When we got up, it was clear that the decision had been made. Next day, an email to the Kepler list informed the subscribers that the list was now going to function as a "real" open source project. The lengthy message, entitled "Opening Kepler," started as

352 In this particular case, Rodrigo was referring to our conversations in the day between our first visit to PUC and this meeting, in which he had asked me for my opinions about what he was planning to do. More generally, throughout my time in Rio, Rodrigo and I had numerous conversations about his project and he occasionally asked for my opinion. In cases when my advice was requested, I provided it without reservations. I also allowed myself to express unsolicited opinions to the same extent as I saw other contributors doing it, but trying to limit such suggestions to specific technical matters rather than the larger strategic concerns. According to Rodrigo, however, the *questions* I asked may have been the most important source of influence on the project, as thinking about the answers had led him to occasionally rethink his approach.

follows:

> Hi,
>
> This mail got a lot bigger than I imagined at first... :o)
>
> As you have noticed, the conversion of Kepler to Lua 5.1 is taking a lot longer than expected.
>
> Not only we have found that this involves more work than we assumed, but also that the development model being used until here is not working as well as we would like.
>
> The Kepler team is currently using a model that involves too much communication outside the public channels (this list and the site for example). We are trying to educate ourselves in order to change that, but this is not exactly easy for a team used to rely on interpersonal communications.

After laying out a detailed plan for specific Kepler components, Rodrigo concluded by saying:

> As I hope you can notice by this mail, we are trying hard to move to a more open development model. That includes using this list in a different way and opening the site wiki for others to contribute.
>
> Before going on, we would like to know what you think about the general idea and what other ideas could be added to this "new vision".
>
> Thanks in advance for any suggestions and thanks for reading that much...
>
> Rodrigo

The three subscribers who responded to Rodrigo's message expressed support for the plan. For some of them, Rodrigo's decision to "open" Kepler may have seemed obvious and natural.

Working the List

As we came back from lunch on Wednesday, we turned our attention to the task at hand. Rodrigo and I were working on a chapter for an upcoming book about Lua programming. The chapter presented an opportunity to promote Kepler, putting Lua's use for web development side by side with such accepted uses of Lua as embedding it in games or running it on micro-controllers. Our chapter focused on implementing in Kepler a "Model-View-Controller" (or MVC) application. MVC was a popular approach to organizing software applications and had been growing in popularity in web development for the last few years. Allowing MVC development in Kepler was one of Rodrigo's projects for this year. This goal, which seemed natural to me, was not shared by all of Kepler's contributors, Rodrigo complained. Márcio, in particular, felt no need for it, preferring the methods that Rodrigo considered outdated. Rodrigo talked about having to explain to Márcio that he had to implement MVC "to please his friends." Unsure whether the phrase referred to the foreign users of Kepler, on the mailing list, or to people in Rio, I decided to ask. Being in perhaps too jovial of a mood, I phrased the question in a somewhat unfortunate way: "Are you talking about your real friends or your imaginary friends?" Rodrigo was taken aback by the question and asked me if I knew what "imaginary friends" means in Brazil. I told him it meant the same in the United States and apologized.

I do have a real friend, said Rodrigo. He mentioned Renato—a close friend who was with us at PUC when Rodrigo was finalizing his decision to open up the project.

Renato was always willing to listen and be supportive. Perhaps too much so. The problem was that Renato would say “This is very interesting” to almost anything. Renato was too busy with his own job to follow Rodrigo’s interest in depth. He was willing to express support, but had no ability to accompany Rodrigo in the journey.

The conversation moved back to Kepler contributors—not exactly friends, but perhaps colleagues, Rodrigo pointed out. For some of them, such as Tiago, Kepler only made sense with MVC. Others, like Márcio, saw no value in this approach. Rodrigo had to find a way to navigate between them.

We returned to work. I read a section Rodrigo had drafted, then proceeded to work on the next section, which we had agreed I would write. Rodrigo, meanwhile, dedicated himself to implementing a change to the Kepler code that we had discovered we needed to present a more elegant example in our chapter. (The chapter was describing a version of Kepler that was yet to be released, so we had the freedom to change Kepler to fit our description of it.)

By 4:30 p.m. Rodrigo had completed the change and sent out a lengthy email to the Kepler mailing list, describing the problem we had encountered and the proposed solution. It concludes with a request for comments. “What do you think about the change?” As he finished the message, he shouted out, as if talking to the people on the list: “Answer! Because I am here lonely, sitting in this room with a crazy Russian guy!” After reading his message, I posted a brief response, pointing out an additional benefit of the proposal and raising the possibility of an additional change. Rodrigo saw the new message in his inbox. “Oh, someone answered!” he exclaimed. “Oh, my friend Yuri!”

As I read Rodrigo's message, however, I noticed a problematic ramification of the change we had proposed, realizing that it would break something else that our examples depended on. I turned around and mentioned this to Rodrigo. We went to Nas Nuvens' lobby and planted ourselves in the two bean bags. We proceeded to discuss the problem for a long time, eventually coming up with a better solution. Rodrigo returned to his seat and wrote another email, describing the problem we had found and the new solution. "Again, if someone is still with me, comments are welcome," concluded his message.

Another hour later, while Rodrigo and I were having a late dinner around the block (a different place from where we had lunch, since we were celebrating the day's success), a list member "John" responded to Rodrigo's second message with a short follow up question. His message ended up being the only one in this thread written by someone outside Rio de Janeiro. (Of course, we did not know where John was, but it was safe to assume he lived abroad.) Rodrigo responded to John's question after getting home; I replied to Rodrigo from home in the middle of the night, this time disagreeing with his response. Another message from Rodrigo, just a few minutes after mine was the last one in the thread. Next morning Rodrigo committed the change to the code repository.

This email conversation on an English mailing list, in which all but two sentences were written by the two of us, who spent most of our day sitting in the same small room, may seem strange and almost farcical. We were resolved, however, to continue developing Kepler in an "open" way, hoping that someone will join us eventually. Turning Kepler into a global project, a trans-local place where space would seem to no longer matter, would require substantial work on the ground. We did not expect it to be easy, and even

small successes counted.

As we were having dinner, I asked Rodrigo if he really felt that he was alone, talking to himself, as some of his earlier comments suggested. He disagreed. He had a hundred people on that mailing list, after all. He did not expect them to write code, it was enough that they read what he wrote and comment, if only occasionally. This, pointed out Rodrigo, was *much* more than what they did up until March!

So, they are becoming a good audience, I thought to myself. They were not good collaborators, at least not yet, but they were now behaving as a good audience. Before that Rodrigo was performing without audience, now he had one. Instead of thinking of myself as his only interlocutor, I concluded, I should perhaps think of myself as a side-kick in his performance.

Noting that Rodrigo had said that he did not expect the list members to contribute code, I asked him if he thought that opening Kepler had the effect he hoped for and whether the list members were as involved as he wanted them to be. (The project had been "open" for over a month, and it was perhaps too early to expect great results, but it was worth a discussion.) He did not expect them to do much, said Rodrigo. But maybe they will, he then added. The distinction between expectations and what *might* happen reminded me of another conversation I had that day.

In the early afternoon, I stepped outside of Rodrigo's office to get some water. There I ran into Pedro, one of the Nas Nuvens' employees who worked in customer support as well as some software development. While he was still working on his

undergraduate degree—and was doing it through a night program at a university that can hardly be compared to PUC—Pedro was considered by Rodrigo to be one of the most promising people at Nas Nuvens. (All PUC graduates had long left for better salaries. A few months later Rodrigo helped Pedro get into PUC's prestigious Master's program in Computer Science. Another year later Pedro left the company, having become overqualified for the salary Nas Nuvens could pay him.)

Pedro was dressed up to a surprising degree: black pants, white shirt, a tie. He had sunk himself into a bean bag, which made his attire seem even more out of place. It turned out that he was presenting the final project for his undergraduate degree later that day—a web application written in Ruby-on-Rails, a recently popular web framework for Ruby, a programming language sufficiently similar to Lua to be seen by some as one of its main competitors. (Ruby was developed in Japan, was little known for years and then suddenly exploded in popularity. As another *peripheral* language and a former underdog that suddenly made it, Ruby has a special place in the imagination of the Lua community.) We talked about the report Pedro had to turn in for the writing project. I asked in which language he wrote it. "In Portuguese," he replied, "It *has* to be in Portuguese." Intrigued by his "has to be," I asked in what language he had written the code. The code was all in English, explained Pedro. Unlike the report, the code could be in English, since only the advisor had to look at it, and the advisor knew English. So it was all in English, he continued: the names of functions and variables, as well as the comments—all but the report. "Why?" I asked. Pedro laughed before answering. "Out of megalomania," he said then. He explained that he and his partner wanted to think that one

day people abroad would be using their code, perhaps even contributing. Writing the code in Portuguese would exclude all of those people.

Of course, making their code in English excluded some Brazilians too, continued Pedro. But those people were already excluded. The code was based on Ruby-on-Rails, which was documented only in English. Ruby-on-Rails function names were in English too. One could not work with it without knowing English. If Pedro were to start a company around his final project, he would not consider hiring anyone who did not know English. But again, he summed it up, it was also about megalomania. What if he decided to hire a developer in India? If their code were in Portuguese, they would not be able to do this.

Was Pedro joking? On the one hand, he had to be. Someone in California talking about "hiring a developer in India" has most likely worked closely with developers from India (perhaps having found himself on more than a few occasions as the only non-Indian in a conference room) and may know people who have traveled to India's outsourcing capitals. The decision to hire a developer in India would thus be a practical question, a matter of cost-and-benefit analysis. Things could not be more different for Pedro, who had never met anyone from India or anyone who had traveled there.[353] Pedro's wages (and those of most programmers in Rio) were hardly high enough to justify hiring programmers in India as a matter of cost savings. The talk of outsourcing was thus not a matter of planning for cost savings, but a matter of dreaming about one day entering the same field with the big global players.

353 My own planned trip to Bangalore later that year provoked active interest from many developers, as did my stories from India when I returned to Brazil later.

This dream was so far-fetched that Pedro himself called it “megalomania.” Having learned those global dreams from the global technical culture in which he engages virtually, Pedro maintained an ambiguous attitude towards them. This was not something he was ready to defend publicly as a plan for action and he was ready to mock his own global imagination calling it “megalomania.” Yet, he did write all his software in a foreign language, entertaining the possibility that those dreams *just might* come true.

Pedro’s story makes explicit the ambiguity of global imagination that came up in more subtle ways in numerous conversations with Brazilian software practitioners. This ambiguity could be understood in two ways. One approach is to liken it to what Favret-Saada calls “I know, but still...” in her discussion of witchcraft in France (Favret-Saada 1980). Favret-Saada describes the ambiguous approach to witchcraft, which combines the public acceptance of the rational view (I *know* there are no witches) and suppressed belief in witchcraft, rarely verbalized and quickly withdrawn upon questioning, yet strongly affecting the practices. Global dreams show a similar duality, being powerful enough to actually affect practice, yet not defensible in public and eagerly labeled “megalomania” upon interrogation.

A somewhat different, but closely related, approach is to look at such global dreams as a game, a make-belief, a simulation of a global practice. Regardless of whether Pedro’s project has a global future, writing the code in English and imagining future plans for hiring programmers in Bangalore could similarly be *entertaining,* a way of doing in imagination what could not be done in reality.

Like Pedro, Rodrigo had a “megalomaniac” dream of running from Rio a major

international open source project, a web development platform that could compete with the popular frameworks for Ruby and Python. Like Pedro, he repeatedly described the dream as a "crazy," a part of his half-humorous plan towards "world domination." The word "crazy" (*maluco*), was in fact one of the most common words that he applied to himself, his work and the people he respected. Unlike Pedro, Rodrigo had actually invested a number of years of his life into his "crazy" dream *publicly*. He thus could not as easily dismiss it as a joke. Instead, he embraced the image of a crazy dreamer. Even so, many of his plans fit within the "I know, but still" space.

One way to deal with this gap was to maintain a healthy distinction between things that were to be discussed and things that were actually to be done. As I started getting involved with Rodrigo's project hands-on, I was quickly overwhelmed by the flood of ideas Rodrigo had for how my part of the project could be improved. As new ideas were coming before old ones could be implemented, I told Rodrigo that we had to slow down —I only had so many hours in the day and had to reserve time for writing field notes and conducting interviews. In response, Rodrigo pointed out, that the problem was arising simply because I was taking his suggestions as things to be *done*. "You have to learn to talk about things but not work on them," he explained. He did see the need for action, but the action was to inevitably fall behind the big dreams. Comparing them too closely would lead to nothing but frustration. We could talk about what Kepler could become one day. Meanwhile, we had to take the small steps: carrying out discussions on the mailing list, appreciative of the few short comments we occasionally got in response.

The Windows Build

As we celebrated the day's success on Wednesday, Rodrigo was in a positive mood, focusing on what Kepler had achieved rather than on the project's troubles. This contrasted sharply with many of the days from the previous month, during which the themes of "being wrong," being "alone" and just "giving up" had come up again and again in our conversations. (The difference, I suspected, had much to do with the fact that the two of us were now actively working together.) Two weeks earlier Rodrigo had shown me a checklist page he had created on a wiki. The problem, he said, was that nobody seemed to want to follow it. Tiago and others did not even think Kepler should be making releases. Perhaps they were right and he was wrong. He had fought with the Lua community for a very long time, trying to make Lua into something that the majority of the users seemed to have little interest in. Perhaps he was just wrong. He was all by himself—even those who were working on Kepler with him were not in agreement. (He was not *planning* to give up, Rodrigo explained on a number of occasions, but the possibility was always there on the table. After all, while Nas Nuvens perhaps could not give up on Kepler, *he* could do just that, perhaps joining Renato as a manager of Java developers.)

One particular source of contention was the installation process. Like other software systems with a part written in C, Kepler's code had to be compiled or *built* before it could be used. This process could in theory be left to the users, and the Lua community had historically stressed this approach, since most users of Lua were

themselves skilled developers working on software products written in C and were assumed to be capable of handling the compilation step. It meant, however, that compared to other languages used for web development, Lua was quite difficult to install.[354] This created a problem for Nas Nuvens, since most customers did not have the expertise to compile software on their own. Simplifying the installation procedure was therefore one of Rodrigo's main projects in 2007.

The problem could be broken in two: one approach for users of Unix-based systems such as Linux and another for users of Windows. The first problem was easier: users of Unix could be expected to have the software tools needed for software compilation and to be accustomed to compiling source code into executable software on their own machines, at least as long as the process was automated and did not require too much tweaking. Rodrigo also had access to Unix enthusiasts among PUC students. In 2006, he used FINEP money to hire "Alan," a PUC Masters student. By April 2007, Alan had written a build script—a program that could be used to compile and install Kepler. With the new script, a user could download, build and install Kepler in less than a minute, making the process of installing Kepler on Unix computers quite trivial. As almost all Kepler contributors used Linux or other variations of Unix for their own server needs, the general opinion seemed to be that Kepler 1.1 was done. As a Linux user myself, I was of the same opinion.

---

354 While Rodrigo's work described in this section helped this problem somewhat, installing Lua and Kepler continued to be much harder until 2008 (about a year after the events described in this chapter), when a different group made an effort to make Lua easy to install on Windows. The existence of this project could be attributed in part to Rodrigo's "culture farming" efforts (for example, the project was hosted on Rodrigo's LuaForge website) and drew heavily on Kepler's modules, though it arose without direct participation of the Kepler developers.

Rodrigo, however, wanted to have a version of Kepler that could be used on Windows—partly due to the fact that he believed this would open a wider "market," but mainly because nearly all of Nas Nuvens' costumers ran Windows. (Rodrigo and many others sometimes attributed this to their being in Brazil, not yet on the Linux bandwagon.) Making Kepler work on Windows was a challenging task, however, both because of the quirks in the Windows build system and because of the Windows users' higher expectations for the ease of the installation process.

A further challenge lay in finding people to do the work. The developers to whom Rodrigo had access appeared to be divided into those who did not have the skills for the task (for example the developers employed by Nas Nuvens) and those who had the skills but lacked interest. Tiago, in particular, had the knowledge necessary to solve this problem (and had done this for the earlier release), but was more interested in other aspects of the project. Following the new policy of relying on developers' intrinsic motivation rather than simply just hiring them to do tasks, Rodrigo decided to not press, hoping instead find volunteers among the list subscribers. A few people expressed interest in helping (a big improvement from February, noted Rodrigo), but none were willing to lead the task. By early May Rodrigo had to face the fact that, if there were a Windows build, it would have to be done by *him*.

Rodrigo's attempt to do the Windows build, however, made it clear that despite good high-level understanding of the technology behind Kepler, he was unable to make sense of its internals and occasionally had to rely on *me* to help him out. Being "out of shape" in this way concerned him greatly. While working on an example for our chapter,

for example, I at one point turned to Rodrigo to ask for a suggestion on how to do in Lua a particular thing which I expected to be simple. A quick discussion showed this problem to be non-trivial, though perhaps doable. Rather than looking for a solution, which could take some time, I told Rodrigo I would avoid the problem by taking a different approach. Some time later, however, I saw that Rodrigo was still thinking about the problem. I asked him why he was wasting his time. “This problem shows,” Rodrigo said, “how out of practice I am.” He wanted, he then explained, to use opportunities like this to get his programming skills back.

Frustrating as it was, the Windows build proved a blessing in disguise. As Rodrigo was starting to understand, getting his programming skills back was not just a matter of solving the specific practical task of completing the Windows build. It was also a matter of adjusting to the change in the rules of the game brought about by “opening” Kepler, following through on the course he had already chosen.

As many people pointed out to me, in Brazil, where less educated technical workers are abundant and cheap and where many needs of the domestic market do not require high skill, highly educated people like Rodrigo get quickly promoted out of programming jobs into management. Those who are thus promoted often lose their hands-on programming skill quickly, in part due to the fast pace of change in the software technology. They often lament the loss, but stay in management. Rodrigo’s friend Renato, working as a manager in a local software company, often talked about his desire to get back to programming, carrying a copy of *Programming in Lua* on in the back seat of his car. (He had also at one point volunteered to translate the book into Portuguese, hoping

that this would help him get back into the actual practice of software development.) Renato's management job, however, kept him quite busy and left him exhausted at the end of the day. Renato still wanted to think of himself as a "computer scientist," but appeared to be "stuck" in management work indefinitely.

Rodrigo, whose first job after leaving the university similarly involved managing software developers rather than writing code, had stayed more closely involved with technology, but had similarly become more skilled at bringing together people, ideas and resources than at getting code to compile. Rodrigo's desire to develop (or, rather, lead the development of) something more than a customized system for a local client, required software developers with higher level of skill than the financial resources of the project could afford. In a recorded interview around the same time, Rodrigo explained:

> Rodrigo: I wanted to build a web platform for Lua. But I didn't have the technical capacity for this. So, I had to think like this: "How could I structure things so as to bring together a bunch of people and this way build a web platform?" The first attempt was Nas Nuvens. When Nas Nuvens appeared, I said: "Look, this is a place where I can make a web platform—because they want to use Lua." That's good, but it didn't work. [...] Because the level of the developers wasn't the level of the developers for a *platform*. Those are different levels. For developers of applications on top of a platform there are certain expectations. If you want to do a platform itself, you have to know a lot more about the operating system, a lot more of C, and about the integration with other things to do the connectors. This is a lot of work.[355]

If Nas Nuvens had been successful in attracting the additional money, the problem would perhaps been solved: Rodrigo would have the money to hire the right developers and could focus on directing the work. Without money, however, he faced a daunting task. "As

355 See appendix C, "Original Interview Quotes," (Rodrigo, May 2007, "Queria fazer uma plataforma web").

far as C programmers go, I know a few," said Rodrigo, "As far as C programmers who want to work for free on a Lua platform—I don't know any."

Rodrigo thus decided that he had to attract foreign open source developers. He discovered, however, to get their help he needed an entirely different currency, one he had in short supply: respect earned through demonstrated technical competence.

> Rodrigo: I noticed that on a free [software] project, there are two currencies of exchange: credit and respect. And there is a tangible good, which is the *source* [i.e., source code, says in English]. So, you have two abstract currencies, and one concrete. I can provide *source* to the community and with this earn respect. Or someone can provide me *source* and I could pay with credit. [...] If I give someone credit, they get respect. And I am trying now is to earn respect.[356]

His skills as a coordinator who had managed to bring together people and money to lay the foundation of web development in Lua did not easily convert into respect. While Rodrigo's quote above suggests the possibility of a simple conversion of code into respect into more code, Rodrigo understood that the free software community assigned respect primarily to the people who wrote the code, not to those who organized and funded its production.

Rodrigo thus found himself in a paradoxical situation:

> Rodrigo: I was in a very delicate situation, because I had a platform that I had come up with—the idea of the platform was mine—but that I didn't know how to use, because I didn't know all the...
>
> Yuri: Didn't know how to develop?
>
> Rodrigo: Not even how to use! I didn't know the internal details. "How does LuaExpat work?" I don't know! I never stopped to look. I said: "I

356 See appendix C, "Original Interview Quotes," (Rodrigo, May 2007, "Duas moedas").

> need a LuaExpat!" And it appeared. Ah, good, now I have this piece. "Now I need a LuaZip!" It appeared. "Now I need MD5." It appeared. So, now I have all the squares. Good.

Rodrigo was familiar with the theory of open source and the idea that such projects are supposed to proceed differently, originating in an individual programmer's desire to "scratch an itch"—solve a specific problem through actual programming work.[357] They are not supposed to proceed top down from, as a government-funded pursuit of a grand technological vision. (Some projects do start this way in Rio—many if not most of my interviewees have pursued one at some point. Such projects rarely go very far, however, for lack of time and inability to find others willing to participate.) Rodrigo's project, as he himself saw it, was proceeding "backwards."

Running the project by the local rules did not require Rodrigo to understand the details of the platform that was being built under his leadership. Local participants had accepted Rodrigo's role as the provider of an engaging vision and the resources they themselves could not obtain. Such resources, however, had no power in an international free software community to be supported by volunteer efforts. Rodrigo had to account the new currency:

> Rodrigo: Because this is something I already noticed on the Lua list. [...] If I ask a question, my question shows a great deal of ignorance—because I am ignorant. Because there is a huge quantity of things I don't know. [...] And this shows in the question. And people's reply tries to be polite, but is... kind of *patronizing* [says in English]. Kind of like: "Look, eh... Try this here. Go see how this is done." And when I send a more technical response, I feel that this produces a connection with the person

357 This idea was codified by Raymond (1999), who wrote: "Every good work of software starts by scratching a developer's personal itch." (Rodrigo had read Raymond's "The Cathedral and the Bazaar" when it was still an online article, and then again when it came out as a book, Raymond 1999.)

> on the other side. They say: "Oh, I am not talking to a stupid manager. I am talking to someone who will at least understand what I am thinking." So, you feel in the reply a different vibe.[358]

Rodrigo thus had to learn to engage in his own project in a new way: as a developer capable of discussing the minute details of the code with the members of the list and making changes to this code when necessary. The Windows build of Kepler was forcing him to do this work. In our interview a few months later, Rodrigo described the time as a difficult but important experience. The difficulty, he explained to me, corresponded to a "deconstruction of a mental model" of how the project worked and of what his place was in it, a time when he had already realized that his old approach was not working yet was not ready to accept the need to make a transition.

Ups and Downs

I left Rio in early August, but continued to follow the project remotely, partly out of researcher's desire to know what happened later, but also as a simple matter of commitment. In April 2007 I had volunteered to write a wiki based on Kepler. The wiki became the first public application built on top of Kepler, and quickly came to be seen as a demo of Kepler's capabilities and a proof that Kepler could actually be used to build real web applications.[359] I now had a role in the project that could not be easily dropped.

---

358 See appendix C, "Original Interview Quotes," (Rodrigo, May 2007, "Meio patronizing").

359 My wiki application ("Sputnik") was not the first web application built on top of Kepler—the administrative application maintained at PUC by Márcio had been using Kepler since 2005 and many list members reported using Kepler for their own applications. However, Sputnik was the first application that a member of the general public could see live and obtain the code for.

By September, I had started to notice a substantial growth in list traffic, finding myself barely capable to keep up with it. I called Rodrigo to catch up on what had happened since my departure. Rodrigo started the conversation by communicating his excitement over the recent changes. The opening of Kepler was finally bringing fruits. The activity on the list was an important part of that. There was a clear spike in traffic. The increased list activities were matched by the growth in interest among local companies. In particular, João was talking to local firms and FINEP about the possibility of bringing Kepler into the digital TV projects. (See previous chapter for the discussion of Lua's role in digital TV.) This was made possible, Rodrigo was sure, in part because Kepler was finally easy to use on Windows.

Rodrigo offered an example of collaboration that illustrated the list's new dynamic. Two weeks before our interview, while working on one of Nas Nuvens' projects (which were starting to use Kepler), Pedro discovered a particular problem specific to Windows. He sent a message to the Kepler list, which went unanswered. After thirty to forty hours of work, by Rodrigo's estimates, Pedro managed to identify two potential causes of the problem. (This work included talking to Tiago off the list.) He sent a new message to the list, now much more specific, and two foreign members of the list joined in the discussion—one of them from Uruguay and another from Southern California. In the course of a lengthy discussion that involved Rodrigo, Pedro, Alan and the two foreign members, the California engineer proposed using a library developed by Luiz Henrique, one of the authors of Lua. The library did not work with Windows, which led to an additional conversation, off the list, with Luiz Henrique. In the end, the California

engineer adapted the library for Windows. Rodrigo and Tiago then jointly made the necessary changes to Kepler. This highly collaborative process, which brought together a Nas Nuvens employee, FINEP-funded Kepler contributors, a member of the Lua team and two external participants served as an example of the project's new dynamic, helped Nas Nuvens solve a specific problem it faced, and resolved a serious underlying problem in Kepler, improving the quality of the platform on Windows.

External participation was also affecting Kepler in more fundamental ways. In April, Rodrigo talked about two new ideas: a radical re-design of the installation method that would make it easier for the user to install Lua modules and a re-design of Kepler's method for connecting to the web server. As the first major changes to Kepler after the decision to open it, Rodrigo sought to discuss the ideas at length on the mailing list, and both ideas have in fact led to lengthy discussions, with active participation of a few list members outside Rio. While this early success had gotten obscured by the troubles with the Windows release, both proposals started attracting interest over the following months. Rodrigo was starting to receive more and more comments and code submissions. As both subprojects were nearing completion, Rodrigo found an opportunity to bring back some of the members of the Lua list who had interest in the project in 2005 but appeared to have lost it later.

The project was also helped by arrival of Jason, a Rio developer whom we met in chapter 2.1. In Spring of 2007 I noticed Jason's name on the Lua mailing list, as one of a small number that sounded Brazilian yet had not come up in my map of the local Lua community built around PUC. I emailed Jason, discovered that he was working for a local

company in Rio. I decided to use the opportunity to talk to him in person. (I contacted two other people in a similar manner. One of them turned out to be a Brazilian graduate student in New Zealand, while another one was a software developer working in São Paulo. In both cases, we exchanged some messages but did not do an oral interview.)

Jason was working for a company that was using Lua to develop applications to run on old, underpowered computers. Jason originally discovered Lua several years earlier, while working on a computer game as a hobby project together with a friend, discovering at one point that Lua was developed in Rio de Janeiro. The project was abandoned, however, when Jason's friend left Brazil. Several years later, while looking for a programming language to use for his company's applications, Jason ran into the developers of SuperWaba, a Java-like development platform maintained by a software developer on Rio. SuperWaba reminded Jason of Lua, which became his platform of choice.[360] After introducing Lua at work Jason became a fan of the language. Having used some of the modules maintained by Kepler, Jason also talked with much interest about

---

360 Jason's discussion of his choice to use Lua shows a complex mix of pragmatism and interest in local technology:

> Jason: Those Waba guys [actually, SuperWaba]... Waba is famous. I discovered that they are used there abroad, right? [...] So, I took Waba, looked and said: "Wow, that's cool." And having gotten something done by a guy in Rio de Janeiro, the bell went off in my head. I was like: "Damn, there are the people at PUC!" I have already heard of Lua before, right. I was researching something and found that there was a game that used the Lua language and discovered that Lua was done in Rio de Janeiro, was done by PUC-Rio. And I was like: "Man, why would I use something super-sophisticated like Java, when there is this thing, which is Lua, which is made for the core of game mechanisms and must be very efficient. Java is much more versatile, sophisticated. So it's slower, it has a lot of requirements. So, man: Lua is very lean, must be perfect for my needs. And it even is a national product, is something... It's like this: it's not enough... It's not just because it's a national product. It's a good product. It's something good, done in Brazil. So, look, for this reason I have to at least look at it and understand it better.

See appendix C, "Original Interview Quotes" (Jason, June 2007, "Negocio muito bom, feito no Brasil").

this project and his own desire to participate in something like this. Noticing that Jason seemed to have the exact combination of expertise and interests that Rodrigo was desperately seeking, I asked him if he thought practically about participating in Kepler and suggested that he should meet with Rodrigo.

Despite his excitement about the fact that Lua was developed locally, Jason had never approached any members of the local Lua community in person, considering himself too far removed from that community socially:

> Jason: Look, I think it would even be interesting to see that they are real people, right? They are people whom I really came to admire, seeing the amazing job those guys did. But I... It's like this story with you meeting a great musician, who plays amazing music. You go and meet him personally. Damn... [Laughs]. He can play for you, that's cool, you know. But man, to go there and talk to him is different. To exchange ideas about music, etc... You have to understand music as well as he does to be able to go and take advantage of this.[361]

Jason did hope to meet the members of the Lua community eventually, but expected to only get there slowly, step by step.

> Jason: I'll go there [to take PUC courses] and will be there, transfixed, wanting to see what they are doing and how it works. Maybe something I could help with. So, when I discovered that the guys were at PUC, that I started using something and the guys were from PUC, I was like: "Damn. This is there. I think I'll go there and see what it's like." But I wouldn't even know were to meet those guys. I think if there was a course they were offering, I would go immediately, running. But I wouldn't go and knock on the door to learn where the guy is.[362]

After returning from our interview, I sent Jason an email encouraging him to talk to Rodrigo. A day later they met for lunch. A few weeks later Jason was actively involved

361 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Ver que eles são gente").

362 See appendix C, "Original Interview Quotes" (Jason, June 2007, "Vou ficar amarradão").

in Kepler. For Jason, meeting Rodrigo was a transforming experience. Rodrigo showed him, he said, that one could live in Brazil doing something outside Java and .Net:

> Jason: The fact that I am now working with Lua, and seeing this potential, it makes me want to find a way a little into this intersection that I find amazingly cool. Now, there is little motivation, if you look at this from the economic point of view, right? So... I mean, we don't just live by the economy, but speaking like modern thinkers, it's a factor that cannot be ignored. So, I have a lot of interest in becoming a researcher. I think if there is a calling that I haven't yet pursued, it's this. And I think participating in this project might count, in some way, for this. If only for networking, to meet the people and to be contributing, entering a community that I always wanted to enter, and now a door opened. [...][363]

> Jason: With the PUC people, I mean. Like how I met Rodrigo, for example, who is a person who showed me that it's possible to work like this. Because this is a question that has always been with me, you know. Me thinking: "Well, great, I've always wanted to do this, but how would I make a living? If I were to be doing those highly experimental and advanced things, which do not have... are not commercial in a mainstream way? How would I live outside Java and .Net?"[364]

Rodrigo was a living proof that this was possible: "Well, he is there and living, right?"[365]

Jason's entry into the project, however, also proved a life-saver for Rodrigo. While a number of new sub-projects were generating a lot of interest, several others were stagnating, in part due to the fact that Alta's Fabio and Rogerio were increasingly busy with Alta's expanding projects and had less and less time for Kepler (see chapter 3.1). Rodrigo's reliance on FINEP funding, however, meant that projects could not be simply dropped for lack of interest. Jason's adoption of one of such lagging project helped Rodrigo focus on the other ones, the ones he felt were most promising. Another month

---

363 See appendix C, "Original Interview Quotes" (Jason, July 2007, "Me tornar pesquisador").

364 See appendix C, "Original Interview Quotes" (Jason, July 2007, "Está lá e vive").

365 See appendix C, "Original Interview Quotes" (Jason, July 2007, "Está lá e vive").

later, Jason left his job and came to Nas Nuvens, taking over Rodrigo's *de facto* role as the company's technical director. This further freed Rodrigo to focus on Kepler: he was now able to spend as much as 80% of his time working "as a developer." Another month later Rodrigo managed to dedicate some of this time to do a side project, for which he wrote half of the code.

The new dynamic continued for several months. After the end of the calendar year, however, the project found itself without money again. A grant had been awarded for the new year, but no money had arrived, having disappeared somewhere in the complicated network of funding transactions. (The work, of course, was expected to proceed on schedule.) Over the next many months Rodrigo had to suspend his newly acquired career as a developer, dedicating much of his time to the work that he thought he was starting to put aside: finding out what had happened to the money the project was owed and what would need to be done to get it back. The team's morale was also seriously hurt, as some of the contributors had to go for months without getting paid for their work. The team managed to release the next version of Kepler in June 2008, but after that the project was effectively suspended. When I returned to Brazil in December of 2008, Rodrigo had managed to finally get access to the money awarded a year earlier and was starting to bring the project back to life, but a lot of momentum had been lost and a lot of work had to be re-done.

Expanding the Practice

After reading a partial draft of this dissertation, Rodrigo Miranda commented that my exposition had shown to him that his project had been futile, what he had tried to do over the last ten years was simply too hard to do, at least in Brazil. Indeed, this chapter attempted to show the many difficulties involved in such a project, arising from the seeming improbability of the alliances that must be made and maintained for such a project to succeed. One may wonder if I have led Rodrigo into diabolical experiment, encouraging him through my display of interest in his project, only to use it in the end as demonstration of why a project like this cannot succeed.[366] Difficult, however, is not the same as impossible. While the structural difficulties faced by a project like Kepler may see daunting, I believe it is important to look at the project historically, as incremental efforts to adapt the local context for the increasingly more central forms of the global software practice, looking at the local practice that the project has enabled, rather than just its outcome. Jason's quote presented above is key for this view of Kepler. "I met Rodrigo," Jason said, "who is a person who showed me that it's possible to work like this. [...] He is there and living, right?"

By trying to make Lua into a practical option for web development, Kepler had sought to change Lua (understood in the broader sense, as a techno-social system[367]) from

366 In a different conversation Rodrigo suggested that our early conversation in 2005 had influenced him by convincing him that what the project was doable, that he was "just crazy, not maniac." "Because you came from the Valley," he wrote, "and you looked at Kepler as it was viable, even if remotely. Nobody had looked at it this way before. Maybe not even me." When I pointed out that my willingness to listen could hardly be interpreted as a sign that I truly believed in his project, Rodrigo explained out that such a belief was not expected from me. "You just took it seriously enough," he said.

367 In other words, Kepler did not bring many changes to Lua *itself* — that it is, to Lua understood in the

a highly specialized niche programming language, strong in very specific areas of applications but assumed impractical for everything else, into a mainstream language that could in principle be used for anything. While early specialization was undoubtedly key to Lua's success, it also has put a limit on the cultural significance of Lua, suggesting too simple of a story: a software project done in Brazil can succeed, but only through a combination of extreme specialization and luck. Kepler has attempted to broaden Lua's success by turning it into a general purpose platform. While Lua's future in web development is still uncertain, the larger effort to widen Lua's use by building a community of people willing to share reusable code and gradually undoing working to undo the network effects that make Lua successful, was perhaps starting to succeed.

---

narrow sense of the programming language specification and implementation. (It did help bring one important change, though —Lua's packaging system, which made it easier to write additional modules for Lua. The new system come form a combination of efforts, which included, in particular, Rodrigo's lobbying for its need and Márcio's early implementation.) Kepler's efforts, however, have made substantial changes to Lua understood in a broader sense, as a techno-social system of elements that enable different uses of Lua, comprising the language itself, a collection of modules, a community, the documentation, etc.

Conclusion

This dissertation has presented an account of software work in a city in Brazil that challenges some of the common views about the globalization of work and knowledge. On the one hand, my account challenges the trivialized notions of the "flattening" of the world, pointing out not only that place continues to matter today, but also that globalization can even strengthen some of the asymmetries. On the other hand, it demonstrates the extent and importance of transnational connections in modern work — uneven as such connections may be. The dissertation shows globalization as an active process and as comprised of numerous individual globalization projects. Drawing on contemporary ethnographic data and historical accounts, I show how individuals build alliances in personal pursuits of globalization, stressing that such pursuits must be understood through a combination of a cultural and an economic perspectives. I show how such alliances transform local contexts, first establishing landing strips that enable arrival of additional elements of remote practices, then gradually synchronizing local contexts with remote ones. I also discuss the challenges faced by such alliances, stressing the partial and uneven state of globalization, and the combination of isolation and hyper-connectedness that sometimes emerges in the process.

My work takes the form of a "thick description" of software work in Rio de

Janeiro (Geertz 1973), combined with an analysis that relies on a theoretical perspective that I believe can help us make sense of globalization of work and knowledge in other contexts. In pursuit of such "thick description," I have tried to present not just the observed behaviors and the recorded stories of the developers, but also the meaning, rationale and context of their actions. While such representation is inherently interpretative (as is any representation), I have attempted, to the extent possible, to separate my thoughts from those of the participants, making it clear when we disagree and presenting the different perspectives that I have encountered. (Software developers in Brazil, after all, do not always agree on everything.) This multiplicity of stories and perspectives thus presented does not allow for an easy summary, and I will not attempt it here.

Instead, I focus in this conclusion on the theoretical framework used in the dissertation, the idea of looking at knowledge and work as a *practice,* pointing out the value this perspective brings to analysis of work, knowledge and innovation. I also highlight my specific contributions to the refinement of this perspective: first the smaller ones (corresponding to the discussion in chapter 1.1) and then what I see as the most important one —my attempt to globalize the practice perspective by looking at reproduction and synchronization of practice between places (chapter 1.3). I then discuss the extent to which this perspective may be applicable for understanding other contexts.

The Practice Perspective

In this dissertation I have looked at work, knowledge and innovation through the lens of *practice*. While my reliance on the idea of practice cannot be seen as innovative *per se*, the dissertation fortifies this concept and provides an example of this perspective in action. While the primary contribution of my work lies in my attempt to globalize the idea of practice (as described in the next subsection), it is important to first pause to discuss the value of the concept of practice itself.

In my usage, a practice is a historically developed system of activities involving people and objects, and held together by culture and politico-economic relationships.[368] My notion of practice draws on several sources, including, first of all, the fusion of the work studies literature with the literature on communities of practice, and, to a lesser extent, the actor-network theory, the theory of structuration, and politico-economic perspectives on labor. (See chapter 1.1.)

My approach to understanding work as practice stresses looking at work as simultaneously a cultural and an economic phenomenon. In other words, the work of software engineers cannot be understood without considering, on the one hand, the extent to which it is affected by the *culture* of software development, i.e. by the shared dispositions and techniques, acquired through active engagement with communities of practitioners. As we saw throughout the dissertation, what software developers do is affected strongly by what they see as "cool," "fun," or "elegant" vs what they see as

368 Cf. with Schatzki's (1996) definition as "the temporary unfolding and spatially dispersed nexus of doings and sayings" (p. 89).

"boring" or "ugly." (We saw this most clearly in chapter 2.1, though the same idea has run through all chapters in parts 2 and 3.) On the other hand, it would be wrong to ignore the fact that in most cases such work is done in the context of employment — a politico-economic relationship in which the developers offer their labor in exchange for resources that they can use both for basic sustenance and for the acquisition of objects required for the participation in the cultural side of the practice. (This would include, for example, the acquisition of the latest gadgets, or using money to hire others to assist in a pursuit of a culturally-motivated technological vision.) This idea shows most clearly in chapters 3.1 and 3.4, though it is exemplified throughout the dissertation.

While both perspectives on work are common, they are rarely combined. This dissertation shows the need to do so, by demonstrating how neither of the two is sufficient by itself. A purely cultural perspective (which seems to dominate, for example, the discussion of open source software development), would lead us to politico-economic naiveté and unjustified expectations about the upcoming "flattening" of the world. (It also, can lead us to accepting too quickly the assumptions of the culture we are studying, as we would have no basis from which to critique it. This again is common in the literature on open source.) On the other hand, purely economic approaches to work, and the focus on the control over the labor process, would lead us to put too much stress on the formal organizational systems, such as firms and industries. While such entities do play an important role, I believe it is important to recognize that knowledge and innovation in software often (and increasingly) produced and shared through lateral ties between individual developers, who are often driven by cultural as much as just economic

motivations.

The combined perspective means, among other things, looking at how the developers unify the cultural economic sides of practice, a task that is rarely easy. “Why will no one ever pay you to do anything interesting?” asks a recent message to the Lua mailing list. The question is asked in jest —many software developers stress than being paid to do what is interesting is the biggest attraction of software development as a profession. (Needless to say, this means doing what *software developers* find interesting, since the desire to spend long hours “mapping interrupts” is hardly a universal human trait.) It highlights, however, the frequent challenge of making the work bring simultaneously cultural and economic benefits. I tried to show the interaction between such factors in several chapters, starting with Jason’s stories in chapter 2.1.

My approach to practice also stresses its systemic nature. I see practice as a system of interrelated activities, which requires specific components (people, tools, documents) that must be made to work together. Additionally, each practice is itself an element in a larger system of related practices. My approach here draws to some extent on that of Hughes and Abbott (in particular the division of labor between the practices), but is also influenced strongly by the actor network theory (and in particular Latour), especially when it comes to looking at how the elements are made to work together—the process that actor-network theorists call “enrollment.” This aspect of practice was illustrated in chapters 2.2 and 3.4.

I also highlight the role of reflexivity and imagination in modern practice. In other words, practice changes in part because the practitioners reflect on their actions (in part

by reading what is written about them) and because they can imagine alternative futures. This aspect is illustrated most clearly in chapters 3.2 and 3.4. To fully appreciate the importance of this aspect of practice, however, we need to consider the processes of globalization, as described below.

Globalizing Practice

The second task undertaken by this dissertation involved extending the practice perspective to account for the phenomena which I believe it was not fully prepared to explain. I attempted in this work to *globalize* the practice perspective by seeking to show how a practice developed in one place takes root in another. This issue is theorized in chapter 1.3 and illustrated throughout the rest of the dissertation, using both historical examples (chapter 2.2) and contemporary qualitative data collected (primarily in part 3).

My discussion of software development in Rio de Janeiro positions the city as *a peripheral site in a widely dispersed but highly centralized world* of software development practice (chapter 2.2), which is dominated by a small number of "Meccas." Local processes orient themselves towards such Meccas in an attempt to draw on their symbolic power and to bring the local practice closer to the remote standards. I stress the incomplete state of the reproduction process: the practice of software development in Brazil can be seen as *a partial replication* of the American practice of software development. (For most places, most certainly including Rio de Janeiro, this incomplete state is permanent, since the continued change of the practice in the central sites ensures

that the job of reproduction is never finished.) This leads me to highlight the "diasporic" situation of the peripheral practitioners, who engage simultaneously in two cultures: the local mainstream culture and the foreign culture of the practice, shown best by chapter 1.2 by looking at the developers' use of English and Portuguese.

My discussion of the process of reproduction focused on *disembedding* and *re-embedding* (Giddens 1990) as a key element of this process. A practice, understood as a system, cannot move to a new place all at once. Individual elements of the practice, however, can be detached from the system ("disembedded"), moved and inserted ("re-embedded") into a new context. Such mobile elements may include material objects (see the story of the UNIVAC's arrival to Brazil in 1960 in chapter 2.2), people ("the Wallauscheks"), ideas ("the Smith Plan" imported from the United States), and documents. As we saw in parts 2 and 3, people attempting to engage in the practice in a new place must reassemble it from disjoint elements brought from other places, and that such re-embedding is often a non-trivial task.

The same applies in reverse: peripheral participants who want to make a contribution to central practices must thoroughly disembed their innovations, making them mobile. As we saw in the case of Lua (chapter 3.2, 3.3), such disembedding does not involve conversion of contextualized elements into some neutral and context-free medium. Rather, it involves removing them from the local context and linking them to the global context of the practice, which, however, is *local* for those in central sites. Knowledge once shared through Portuguese conversation, for example, takes the form of a global book, written not in Esperanto or Volapük, but in English, the language spoken

fluently in California but significantly less so in Rio de Janeiro. The price of such disembedding is borne not only by the peripheral innovators, but also by other peripheral users. Luciano's struggle with reading *Programming in Lua* —an English book written in Brazil just a few kilometers away from where he works —is indicative of this (see chapter 1.2). It also shows how peripheral participants in many ways bear the burden of maintaining the predominance of central sites. It can also draw new boundaries locally.

Building on the idea of practice as a system, I stress *the cumulative nature of the reproduction process*. The process of reproduction of practice happens over time, as a gradual synchronization of context. At each step, elements are brought together and local work is done to makes the context more similar to the central sites, thus laying the "tracks" that Latour stresses must be in place for knowledge to move between places (Latour 1988a). Alternatively, we can think of such efforts as creating landing strips for future elements —enclaves of the practice in the midst of otherwise unconquered territory.[369] Once the tracks and landing strips are there, importing additional elements becomes easier. Chapter 2.2 shows us how over a number of decades Brazil's context was brought closer to that of the central sites of the computing world. Establishment of connections to the Internet, for example, transformed the methods for keeping practices in sync. The different projects described in part 3 all contribute to continued synchronization of context.

---

369 Morais's (2006) account of the history of aviation in Brazil stresses the difficulty of bringing aviation to remote places, which can be compared to the "bootstrapping" problems described by Staa (2003, see my discussion in chapter 2.2). Places that had built landing strips were easy to include in the growing aviation networks. Adding a new place, however, presented a "bootstrapping" challenge, requiring the aviators to either look for the closest alternatives (roads, fields) or to arrive to such places by means other than airplane.

The parallel nature of the reproduction process leads to *a complex relation between individual and collective efforts of reproducing foreign practices*. The local practitioners must often make a decision whether to cast their lot with local colleagues or to focus on their individual connections to the remote centers. We saw examples of this both within and between practices. For example, the Brazilian government and other Brazilian users of computers had to decide at several points whether to rely on local makes of computers (in the hope of eventually benefiting from cheaper technology more attuned to local needs) or to focus on acquisition of the better foreign machines. Within the practice, software developers in Rio de Janeiro must decide whether to go with the globally established programming language such as Java, or to dedicate their efforts to supporting a local one. Chapters 3.2–3.4 use the case of Lua to show the challenges this presents for local innovation, which must often succeed abroad before being accepted at home.

Seeing software practice as itself an element of a larger system of practices has led me to stress *the parallel nature of the reproduction process*. As we saw most clearly in chapter 2.2, the reproduction of the foreign software practice cannot be understood in isolation from the parallel efforts of people engaged in different practices, all of them pursuing their own globalization projects. The fact that the centers of many practices coincide simplifies this task tremendously. The work of Alta's engineers, who have mastered technology developed on the west coast of the United States, is made much easier by the fact that their clients seek to emulate business practices originally developed in the same place.

Finally, reflexivity becomes particularly important for understanding the reproduction process. We saw throughout the dissertation that software developers know quite a bit not only about the social context they inhabit (as do most people, argues Giddens 1979), but also about the remote social context. This knowledge of a remote social structure becomes an important structuring tool. Knowledge of how things are done elsewhere can help bring out the same structure locally. It becomes important to look at the sources of such knowledge, and in particular at the developers' use of foreign books and websites not just as a source of technical knowledge but also as a source of ideas for social organization. I have tried to note the use of sources throughout the text.

Foreign structure, however, is not always deemed relevant to the local activities. The developers often know what happens abroad but consider it silly to attempt the same in Brazil. It thus becomes important to pay attention to what outcomes can be *imagined*, and how the dubious nature of such imagination is negotiated. Rodrigo's repeated attempt to draw the distinction between "crazy," "insane" and "maniac" dreams (chapter 3.4) illustrates this point.[370] Change often comes from plans that are sufficiently "crazy" to present an ambitious step forward, yet imaginable enough to build a coalition in their pursuit.

370 I do not believe Rodrigo ever meant to properly order those terms and affix each of them a specific degree of craziness. He most certainly understands that those words sound the same to the listener. (The joke would be easier delivered in English by use of the same word with different prosody: "just crazy, not *crazy*.")

Other Places, Other Practices

This dissertation has relied primarily on an observation of a particular practice in a particular place. What can this account tell us about other contexts of work? I start with the question of what this dissertation may tell us about software work in other places. I then briefly discuss whether it may be useful for understanding other kinds of work.

While additional work would be needed to examine the extent to which this perspective would fit the practice of software developers in other places, I expect that for many places such work would discover a substantial fit with the general perspective, if not with the details. This would particularly be the case for other "semi-peripheral" sites such as Rio, places where the software practice has been assembled to a substantial degree but where continued work must be done to keep it up to date with remote standards.

For example, chapter 1.2 would look quite different if it were based on observations of software developers in one of the software capitals of South India. In many parts of India, English is not the language of software, but simply the language of education. It may again be somewhat different if written in Russia, where the local language is in a much stronger position vis-à-vis English than Portuguese is in Brazil, and where programming languages using non-English keywords *have* been developed. In this sense, I believe Brazil represents an intermediate position and a case worth understanding.

The notion that becoming a software developer often has more to do with learning

to love the computer than a pursuit of lucrative employment would also not hold for India, where economic considerations appear to be the most important reason for becoming a software developer for many people. Even so, however, the larger perspective taken in this dissertation may well apply to understanding software work in India. While Brazilian software developers learn to love software early on, but may then struggle to find "proper" jobs, software developers in India, for whom exceptional grades in high school often become a ticket to the world's largest software companies, will likely have to learn to love software after getting their jobs. (If they do not, their status in the global software world will likely remain marginal, as they would be seen by their foreign colleagues as low-cost mercenaries rather than fellow practitioners.) Such differences, fit within the broader perspective presented in my work, though understanding those two different ways of entering the world of software may be a particularly fruitful direction for future work.

This dissertation has focused on a place separated by the centers of the software world by several kinds of distance: the cost of traveling, the differences of languages, national boundaries that limit the movement of both people and things, differences of government policy, national identity, local and national culture. While such different kinds of "distances" quite often coincide, they do not always do so. Looking at places that present specific combinations of those different kinds of distance would help refine the notions of "place" and "peripherality."

Another question concerns the extent to which the perspective taken in this chapter can by applied to other fields of endeavor. One may wonder if software is unique in the extent to which its practitioners in places such as Rio de Janeiro engage with the

global world of practice, including its global culture. For many other lines of work, the local communities of practice may be all that matters to the individual practitioners. This requires a two-fold answer.

The more abstract aspects of the framework would likely be immediately applicable to a wide range of professions. Even for practices where individual practitioners rarely venture out of their local community, the substantial degree of similarity in practice points to the existence of processes that lead to synchronization of those practices between places, which likely draw on some of the same mechanisms. The perspective taken in this dissertation would in the very least provide a starting point for analysis. For example, I have stressed that the global culture of software development provides a set of "perceptions of judgments," which include the understanding of software work as interesting and worth pursuing for the sake of intellectual stimulation. This particular way of seeing the practice is most certainly not shared by all other practices. The practitioners of a trade can instead understand their work as a matter of "service," of "honest work," as a game, or as form of political action. What is likely to be found across many different practices, however, is the pursuit of a shared understanding of the activity, whatever that understanding may be.

Software may well be nearly unique in the extent to which the use of foreign documents by individual practitioners is important for synchronization of the practice. Software is unique today in the abundance and accessibility of documents describing the practice. It is also unique in the extent to which such documents are *useful*. This likely has to do with the relative immateriality of the software practice. Traditional accounts of

science practices, for example, commonly stress the importance of direct access to the material tools and artifacts. Software developers, on the other hand, work with few physical objects apart from their computers. Their work is a disembodied, textual art. Repositories of software code and mailing list (on which code can be shared by simply being pasted into the message) serve as virtual environments in which the objects of work reside and can be observed. Such repositories can be "visited" at little cost, as such visits do not disturb the work that occurs in them.

While there is no equivalent of this for many professions, software represents represent an example of a *class* of occupations that primarily involve manipulation of digital representations. Software developers, simultaneously help create the technologies that enable work based on digital representations and become pioneers of such work. I expect that a number of such occupations will grow over time, and new technologies will make it possible to increasingly interleave such representations with texts. The reliance on such non-rival representations, may also lead to increased free sharing of elements of practice, enabling non-software equivalents of open source production. As the analysis presented in my work suggests, however, this does not mean that place would cease to matter and may instead add power to the central sites.

Glossary

Adobe
: A software company headquartered in San Jose, California, most famous for Adobe Photoshop.

Adobe Lightroom (Adobe Photoshop Lightroom)
: Image management software by Adobe, written mostly in Lua. See Adobe.

Algol (also ALGOL)
: An early programming language (developed in the mid 1950s), which greatly influenced the later programming languages.

Alta (pseudonym)
: A software consulting company based in downtown Rio de Janeiro, described in chapter 3.1.

Amiga
: A personal computer popular in the 1980s.

analyst
: see systems analyst.

ANSI C
: The standard version of the C programming language. (ANSI is the American National Standards Institute).

API (application programming inferface)
: A set of functions that a software module offers to other software interacting with it.

applet (or "Java applet")
: A Java program embedded in a web page. (This method of developing dynamic websites was popular in the 1990s. Today JavaScript is typically used instead.)

application
: Software used directly by the user, rather than by other software. (See also library.)

architect
: See software architect.

assembler (assembly language)

A low level programming language. Programming in an assembly language was common before the use of C became widespread.

Atari

A brand of video game and personal computers popular in the late 1970s and the early 1980s.

BASIC

A simple high-level programming language designed for beginners, popular in the 1980s, and especially among hobbyists and children.

BBS (Bulletin Board System)

A computer running software that allows users to connect to it (usually through a phone line) for the purpose of downloading or uploading files and reading news. (See Carvalho 2006 for a discussion of BBS use in Brazil in the 1980s.)

benevolent dictator

A charismatic leader of an open source project, who often wields substantial authority but must rely on the support of the community. The term was originally applied to Guido von Rossum (the inventor of Python) by the Python community.

bit twiddler

A programmer seeking to change the behavior of software by changing individual bits in its binary representation. The term may be used as derogatory, since bit twiddling can take a lot of time while bringing relatively minor benefits. The Portuguese equivalent is "escovador de *bit*."

BITNET

A computer network popular in the 1980s and the early 1990s, until the rise of the Internet.

bug

An defect in the program's source code. Finding and removing such defects is called "debugging."

C

A programming language developed in the 1970s and by some measures the most popular language to this day. C and the closely related C++ have become the standard solution for low level programming, replacing the use of the assembler. High-level programming languages such as Java, Lua and Python are themselves implemented in C.

C++

A programming language closely based on C, developed in the 1980s.

cachaça

Sugarcane rum produced in Brazil.

caipirinha
: A mixed drink based on cachaça.

CAPES
: A Brazilian government agency that funds post-graduate education.

capoeira
: An Afro-Brazilian physical art that combines elements of a martial art and a dance.

carioca
: an adjective meaning “of Rio de Janeiro.”

CGILua
: A system for creating dynamic web applications in Lua, described in chapter 3.4. CGILua served as a basis for Kepler.

client
: An application that makes a request to another application, typically over the network. For instance, a web browser is a “client” to the application run by the maintainers of the website that it accesses. See also server.

CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico)
: A Brazilian government agency that funds academic research.

COBOL
: An early programming language popular for business applications in the 1980s and still used today, to a lesser extent.

coder
: A software developer. This is an insider term in the United States, which has no equivalent in Brazil.

comment
: Text included in source code solely for the purpose of making the code understandable to the software developers who will work with it later. Comments are ignored by the compiler and do not affect the behavior of the software.

compiler
: Software that translates a human-readable source code into a format that can be executed by the computer.

Copacabana
: A mixed-use neighborhood in Rio de Janeiro, about ten kilometers from Downtown.

Cray
: A super computer brand.

cruft

Code of poor quality, often connoting unnecessary complexity where a simpler solution could be possible. It may more specifically suggest the complexity that is introduced through the long period incremental changes and could be potentially removed if the code were re-written from scratch.

database
A computer system dedicated to storing data.

DBA (database administrator)
A computer professional whose work involves managing complex databases.

DEC
A 1980s manufacturer of minicomputers, based in Boston, MA.

DEL
A predecessor of Lua. See chapter 3.2.

DOS
An operating system for personal computers used in 1980s and the early 1990s.

Downtown (Centro)
The main commercial area of Rio de Janeiro, where many of Rio's software consulting companies are located, including Alta.

Dr. Dobb's Journal
A US periodical aimed at software developers with circulation of 120,000.

Eclipse
An open source application for editing computer code, especially popular among developers working with Java.

EIT (pseudonym)
A software company based in California, with which Alta had a "partnership."

eiWeb (pseudonym)
A proprietary Java web development platform developed by EIT.

embedding
Putting software inside an application.

ENIAC
One of the very first computers, built for the US Army in the 1940s.

Estácio de Sá, Universidade
A private university based in Rio de Janeiro, with many campuses throughout the state.

favela
A shanty town. Rio's favelas house a substantial portion of the city's population.

FINEP (Financiadora de Estudos e Projetos)
A Brazilian organization under the Ministry of Science and Technology that funds

scientific and technological research. In particular, FINEP supports projects that involve collaboration between industry and universities.

forum
: A web site that allows visitors to engage in a discussion by posting questions and answers.

framework
: A collection of software code that implements standard functionality for a particular domain. A framework typically provides a more complete solution than a platform, but the difference between the two terms is not essential for this dissertation.

free software
: Same as open source. See open source vs. free software for an explanation of the terminological differences.

FTP (File Transfer Protocol)
: A method for making files available over the Internet, now declining in popularity.

function
: A sequence of instructions that tells a computer how to perform a specific task. A function is a basic unit of software source code in most modern programming languages.

Ginga
: A layer in the Brazilian digital television system, using Lua as its component. See chapter 3.3.

graphics
: A subfield of computing dedicated to making computers display images on the screen.

Grim Fandango
: A game released by LucasArts in 1998s. Grim Fandango used Lua and helped make Lua popular among computer games developers.

gringo
: A non-Brazilian or a person from outside Latin America.

hacker
: 1. Expert programmer.
2. Someone who gains access to computers to obtain data or use them for other purposes.

hardware
: The physical computer and those aspects of its behavior that cannot be changed through software. The hardware defines the basic behavior of computers, in particular by encoding certain instructions in patterns of semiconductor etched on the surface of computer chips. (Semiconductors —such as silicon —have an

important electric property: their ability to conduct electricity depends on whether an electric field is applied to them. This makes it possible to use combination of semiconductors to build basic logical operations such as “or” and “and.” Those can then be further combined to enable more complex processing of data.) Software developers typically see hardware as material and given, though the actual process of hardware design has many similarities to software development.

Herculoids
: A private mailing list maintained by Rodrigo Miranda.

high-level
: Software code relying on existing libraries and isolated from the details of the hardware or operating system.

hypertext
: Text with cross-references.

IBGE (Instituto Brasileiro de Geografia e Estatística)
: A Brazilian agency responsible for collection of statistics, including the census.

IMPA (Instituto Nacional de Matemática Pura e Aplicada)
: A research and higher education institution in Rio de Janeiro focused on mathematics.

informática (informatics)
: Speakers of Brazilian Portuguese use the term “informática,” borrowed from French (“informatique”) to refer broadly to disciplines related to information technology. There is no single English term with the same meaning. It can sometimes be appropriately translated with the English abbreviation “IT” (“a career in IT” for “careira em informática”), though it can in other cases be used to refer to what in English would be called “computer science” (“estudou informática” can be used to mean “studied computer science,” to avoid the more formal term “ciência de computação”). Authors writing on the history of Brazilian IT policy typically translate “informática” as “informatics,” and I do the same when talking about the policy of 1970s and 1980s. It should be noted, however, that the English term “informatics” is often used today in other senses.

InterJX (pseudonym)
: Alta’s first product, never completed.

Intermercado
: Alta’s main client, a major Brazilian retailer.

Internet
: A computer network that has become popular world-wide in the 1990s. Note that while the Internet has since become synonymous with networking technology, in the 1980s and the early 1990s it was one of the large computer networks among several, BITNET being another example.

interrupt
: A signal sent within the computer.

intranet
: The network of computers inside an organization and not accessible from outside or the websites only accessible from inside the organization.

IP address
: A numeric address that identifies a computer on the Internet and allows other computers to communicate with it.

IT
: information technology

ITA (Instituto Tecnológico de Aeronáutica)
: An education institution established in Brazil in 1950 to train aeronautics engineers. See chapter 2.2.

J2EE (Java 2 Enterprise Edition)
: A collection of Java APIs for doing web development in Java.

Java
: A programming languages developed by SUN Microsystems in the mid 1990s and one of the most popular programming languages today. SUN Microsystems is headquartered in Santa Clara, CA.

JavaScript
: A programming language embedded in most modern browsers and used to add client-side functionality to web applications. Modern web applications typically divide the work between the server and the client. In a typical setup, the web browser sends a request to the web server, which pulls data from the database, then sends the processed data and some additional instructions to the web browser, which then processes the data further before displaying it to the user. While a range of languages are used for processing the data on the server side, Java and PHP being the most popular at the moment, client-side processing is typically done in JavaScript, which is the only programming language most web browsers can understand.

JavaWebStart
: A method for launching Java applications by clicking on a link on a web site.

Kepler
: An open source software project aiming to develop a web development platform based on Lua. See chapter 3.4.

keyword
: A word that has a specific meaning in a given programming language.

Kubix (pseudonym)

A company where Alta's founders worked before starting Alta.

legacy
: Old code with which new code must stay compatible.

library
: Software that is used by other software (which delegates to it some of the tasks) rather than directly by the user.

license
: The legal terms under which software is distributed.

Linux
: An open source operating system.

Lisp
: A programming language developed in the 1960s, often prized for conceptual elegance.

localization
: Adapting software for a particular human language, country or region.

low-level
: Software that interacts directly with the computer's hardware or operating system rather than relying on multiple layers of libraries. (Cf. high level.) Note that "low-level" software development does not imply low level of skill, but in fact is often understood to be more challenging.

Lua
: A scripting language developed in PUC-Rio and used in a number of popular software products, including Adobe Lightroom and World of Warcraft. Lua's history is described in chapters 1.2, 3.2 and 3.3.

Lua 3.2
: The last version of Lua before Lua 4.0. Because of the changes introduced by Lua 4.0, some users of Lua continued using 3.2 until today. (Nas Nuvens' software uses Lua 3.2.)

Lua 4.0
: A version of Lua released in 2001, which introduced a substantial change from the previous versions. Many earlier users of Lua never transitioned to Lua 4.0 and continued using Lua 3.2.

Lua 5.1
: The latest version of Lua in 2007 and 2008.

Lua book
: See Programming in Lua.

Lua-L (Lua mailing list)
: The official mailing list for the Lua community.

LuaForge
: A website for sharing open source libraries and applications written in Lua. (See chapter 3.4.)

mainframe
: Large computers (sometimes occupying multiple rooms) used from the 1960s until mid 1990s. The term is also sometimes used today to refer to the somewhat smaller modern computers (starting at around one hundred kilograms) compatible with the older mainframes.

many eyes
: A shorthand for Eric Raymond's pronouncement that "given enough eyeballs, all bugs are shallow" (Raymond 1999) —that is, the idea that exposing the source code to other developers helps discover and fix its defects.

microcomputer
: A generation of small computers (typically small enough to be put on a under a desk) introduced in the 1970s, which has grown in popularity through the 1980s. The term is almost never used in the US today except in historical context, since nearly all modern computers are microcomputers. In Brazil, the term "micro" is occasionally used as a synonym for "personal computer," especially by older computer professionals.

minicomputers
: A generation of "compact" computers smaller than mainframes introduced in the 1960s and popular in the 1970s and the 1980s. A typical minicomputer was the size of a refrigerator. Brazil's Market Reserve policy focused on domestic production of minicomputers.

module
: Same as a library.

Nas Nuvens (pseudonym)
: A software company in Rio de Janeiro offering web development services using Lua.

newsgroup
: A system for exchanging messages over the Internet, that allows an individual to start a discussion topic to which other users can contribute. Newsgroups were popular in the 1980s and the 1990s. Today mailing lists and forums are often used for the same purposes.

off-shore
: See outsourcing, second sense.

open source software
: Software distributed together with its source code and a permission to modify and redistribute it. For an introduction to open source see Schwarz & Takhteyev

(2009). See also open source vs. free software.

open source vs. free software

The terms "open source software" and "free software" do not differ in denotation. Both terms refer to software that is distributed together with its source code and a permission to modify and redistribute it. However, there is a substantial difference in connotation between the terms, and the choice of one term over the other may sometimes be suggestive of the speaker's position in regards to when and why source code should be shared. The term "free software," introduced in the mid 1980s, is typically preferred today by those who see sharing of source code as a ethical (or even political) movement, which they may perceive as antagonistic to the interests of for-profit corporations. The term "open source" was introduced in the late 1990s as an alternative to "free software" in order to make the concept more palatable corporate users and contributors. Those who use the term "open source" typically see room for co-existence between free software / open source software and proprietary software. They stress that open source licensing is often (not always!) the most sensible solution, without seeing it as a matter of moral obligation.

outsourcing

1. Delegating software development to a third party, which may or may not be in a different country. For example, Intermercado's reliance on Alta for its software needs is an example of "outsourcing." In Brazilian Portuguese, this is typically "terceirização" (lit. "third-partying"), though the term "outsourcing" may be used occasionally.
2. Moving software development to a third country, which may or may not imply delegating it to a third party. For example, in chapter 3.3 Carlos talks about the prospects of Rio's outsourcing economy —that is, getting foreign companies to rely on Brazil for their software development needs. In Brazilian Portuguese, this is typically referred to by the English term "outsourcing."

performance

Efficient use of available hardware resources. Lua is prized for performance, which means that a program written in Lua runs faster than an equivalent written in many other programming languages, for example Python.

Perl

A popular scripting language.

Petrobras

The largest Brazilian oil company. The Brazilian government owns more than 50% of the shares of the company making it subject to some of the same rules as the government agencies. Petrobras is headquartered in Rio de Janeiro.

PHP

A popular scripting language for development of web applications.

platform

A collection of software libraries and tools that facilitate the development of applications for a particular domain. Kepler is a platform for web application development in the sense that it provides a collection of Lua libraries and tools that make it possible to develop web applications in Lua. Platforms are subject to substantial network effects. (See Katz & Shapiro 1986 for a general discussion of network effects.)

porting

Modifying software for using it on different hardware.

product

Software written for use by multiple clients, rather than narrowly aimed for attending to the very specific needs of a particular one.

programmer

Someone who writes software code, a software developer. In the US, "programmer" is an outsider term, rarely used by the software developers themselves, who often prefer such terms as "developer" or "coder." The Portuguese cognate "programador" is used even less often by software developers in Rio where it is seen as connoting a low level of skill.

programming

Writing source code. Other popular English terms include "coding" and "hacking." The equivalent Portuguese terms are "programar," "escriver código," and "hackear."

Programming in Lua (PiL, the Lua book)

A book written by one of Lua's author (Ierusalimschy 2003, 2006) and generally considered the definitive introduction to Lua. The first edition was published in 2003 and documented Lua 5.0. The second edition ("the Blue PiL") was published in 2006 and documented Lua 5.1.

programming language

An artificial language for expressing instructions to a computer. Modern programming languages are designed so as to be meaningful to a human programmer while also easy to translate into a format that can be used by the machine.

proprietary software

Software distributed without a permission to modify or redistribute. Cf. open source.

pt

A code for Portuguese language.

PUC-Rio (PUC)

Pontifícia Universidade Católica do Rio de Janeiro (Pontifical Catholic University

of Rio de Janeiro), a private catholic university in Rio de Janeiro, one of the most prestigious in the city. PUC-Rio is referred to as "PUC" inside Rio de Janeiro, but not in the national context, since there are several other pontifical universities in Brazil. PUC's Department of Informatics is the highest-rated computer science department in Brazil (using CAPES ratings). Lua was developed at PUC-Rio (at Tecgraf).

Python
: A scripting language, one of the most popular today.

RCS
: A system for storing multiple revisions of source code.

real, Brazilian
: Brazil's currency since 1994. In the recent years the exchange rate between the Brazilian real and the US dollar has typically been 2–3 BRL for 1 USD.

reserved words
: For the purpose of this dissertation, same as keywords.

RPG (role playing game)
: A game in which the players, take roles of fictional characters and enact a story that is determined by players decisions and rolls of dice. Such games can be played on a computer, but do not have to be.

Ruby
: A scripting language developed in Japan, one of the most popular today. Ruby is particularly popular in the development of web applications, which are often used based on a framework called Ruby-on-Rails. Ruby was developed in Japan, though Ruby-on-Rails was developed in the United States.

sandboxing
: Constraining software code so as to only allow it to use a specific subset of computer resources.

scripting
: Programming in a high-level programming language or a "scripting language." Programming in a scripting language requires less knowledge of the operating system and computer's hardware.

serial cable
: A type of computer cable used in the 1980s and 1990s.

server
: A computer or a program that performs tasks in response to requests from other software (usually through the network) rather than from the user directly. Cf. client.

service
: Development of software for the needs of a specific client rather than for the

market. See product.

set-top box
: A device that connects to a television set to allow display of additional content, often offering a level of interactivity that is higher than that of traditional television and more similar to that of a computer.

Smalltalk
: A programming language developed in the 1970s, rarely used commercially but highly influential.

software
: A collection of instructions that tells a computer how to process data. Computers are machines that manipulate data in accordance with given instructions. Modern computers follow "stored-program" design, which means that while basic instructions for processing data are fixed in the machine's "hardware," the primary purpose of such fixed-instructions is to load additional instructions from the computer's malleable memory. Further behavior of the computer then depends on the instructions found in memory. This design makes it possible to modify the machine's behavior by changing the instructions in its memory, without replacing any of its hardware. Such stored and modifiable instructions are called "software." The work of software developers focuses on production of software, in the form of code. (See also hardware.)

software architect
: A computer professional tasked with the overall design of a software system.

Software: Practice and Experience
: A computer science journal focused on practical experience with software. A 1996 publication in this journal was one of the two articles that brought Lua to the attention of the American software development community.

source code
: Textual representation of software, which can be read and edited by human programmers. See chapter 1.2 for examples of source code.

spec
: To define, to specify.

strings
: Sequences of text manipulated by the software.

syntax
: The formal rules of a programming language.

systems analyst
: A computer professional whose primary responsibility involves writing specifications for the software to be written by others.

Tcl

A scripting language popular in the 1990s, often used as an embedded language.

Tecgraf
: A research and consulting laboratory in PUC-Rio. Lua was originally developed as a part of a Tecgraf project for Petrobras.

TK 82, TK 85
: Brazilian clones of small personal computers manufactured in the 1980s by Microdigital Eletrônica. TK 82 was very similar to Sinclair ZX-80, though with some enhancements. TK 85 was a clone of Sinclair ZX81.

Ubuntu
: A variant of Linux, one of the most popular in 2007.

UNIVAC
: A brand of early commercial computers produced in the 1960s.

Unix
: A class of operating systems, of which Linux is the most well known representative.

user
: The person who uses or will be using the software. In software development, "the user" is often hypothetical. See Woolgar (1991) for an analysis of "the user" as a social category.

user base
: The people who use the software. Due to the strong network effects in software, the existence of a large user base is usually seen as providing a substantial advantage.

VAX
: A family of minicomputer produced by DEC in the 1970s and the 1980s.

Visual Basic
: A programming language developed by Microsoft.

web application
: A software application running on a web server and responsible for some of the functionality provided by the website.

web server
: A server that powers a website. When the user enters a URL into the browser or clicks on a link, the browser makes a request to a remote computer (the web server), which returns the content to be displayed.

wiki
: A web application that gives the user the ability to edit the content they are presented with.

World of Warcraft

The most popular multi-player online computer game. (http://gamers.guinnessworldrecords.com/records/pc_gaming.aspx). World of Warcraft's interface was developed using Lua and the users can install plugins written in Lua. World of Warcraft is probably the most high-profile use of Lua today.

Bibliography

Abbott, A. (1988). *The System of Professions: An Essay on the Division of Expert Labor*. Chicago: The University of Chicago Press.

Adler, E. (1986). "Ideological guerillas and the quest for technological autonomy: Brazil's domestic computer industry." *International Organization,* 40(3), pp. 673–705.

Adler, E. (1987). *The Power of Ideology: The Quest for Technological Autonomy in Argentina and Brazil*. Berkeley: University of California Press.

Alkema, H. and K. McLaughlin (2007). "A chronology of computing at the University of Waterloo." http://www.cs.uwaterloo.ca/40th/Chronology/index.shtml, last accessed May 3, 2009.

Anderegg, D. (2007). *Nerds: Who They Are and Why We Need More of Them.* New York: Jeremy Tarcher / Penguin.

Anderson, B. (1991). *Imagined Communities: Reflections on the Origin and Spread of Nationalism.* Revised edition. London: Verso.

Appadurai, A. (1996). *Modernity at Large: Cultural Dimensions of Globalization*. Minneapolis, MN: University of Minnesota Press.

Audretsch, D.B. and M.P. Feldman (1996). "R&D spillovers and the geography of innovation and production." *The American Economic Review*, 86 (3), pp. 630–640.

Azevedo, M. (1989). "Vernacular features in educated speech in Brazilian Portuguese." *Hispania*, 72 (4), pp. 862–872.

Bagno, M. (2001). "Português do Brasil: herança colonial e diglossia." *Revista da FAEEBA*, 10 (15), pp. 37–47.

Barley, S. (1996). Forward to Orr, J., *Talking About Machines: An Ethnography of a Modern Job*, Ithaca, NY: Cornell University Press.

Barley, S. and B. Bechky (1994). "In the backrooms of science: Notes on the work of science technicians." *Work and Occupations*, 21, pp. 85–126.

Barley, S. R. and G. Kunda, G. (2001). "Bringing Work Back In." *Organization Science*, 12 (1), pp. 76–95.

Barley, S.R. and G. Kunda (2004). *Gurus, Hired Guns, and Warm Bodies: Itinerant Experts in a*

Knowledge Economy. Princeton, NJ: Princeton University Press.

Bastos, M. I. (1994). *Winning the Battle to Lose the War: Brazilian Electronics Policy under US Threat of Sanctions.* Ilford, Essex, England; Portland, OR: F. Cass.

Becker, G.S. (1962). "Investment in human capital: a theoretical analysis." *The Journal of Political Economy*, 70(5), pp. 9–49.

Becker, H. (1972). "A school is a lousy place to learn anything in." *American Behavioral Scientist,* 16, pp. 85–105.

Becker, H.S. (1951). "The professional dance musician and his audience." *The American Journal of Sociology*, 57(2), pp. 136–144.

Becker, H.S. (1953). "Becoming a marihuana user." *The American Journal of Sociology*, 59(3), pp. 235–242.

Becker, H.S. (1963). *Outsiders: Studies in the Sociology of Deviance.* London: Free Press of Glencoe.

Becker, H.S. (1972). "A school is a lousy place to learn anything in." *American Behavioral Scientist*, 16, pp. 85–105.

Becker, H.S. (1982). *Art Worlds*. Berkeley: University of California Press.

Becker, H.S. (1986). *Writing for Social Scientists: How to Start and Finish Your Thesis, Book, or Article*. Chicago: The University of Chicago Press.

Becker, H.S. (1998). *Tricks of the Trade: How to Think About Your Research While You're Doing It.* Chicago: The University of Chicago Press.

Behrens, A. (2005). "Brazil." Chapter 5 in S. Commander (ed.), *The Software Industry in Emerging Markets.* Cheltenham, UK: Edward Elgar. Pp. 189–219.

Berger, P.L. and T. Luckmann (1966). *The Social Construction of Reality: A Treatise in the Sociology of Knowledge*. Garden City, NY: Anchor Books.

Biber, D. (1993). "Representativeness in corpus design." *Literary and Linguistic Computing,* 8 (4), pp. 243–257.

Botelho, A. (1999). "Da utopia tecnológica aos desafios da política científica e tecnológica: o Instituto Tecnológico de Aeronáutica (1947–1967)." *Revista Brasileira de Ciências Sociais,* 14 (39), pp. 139–154.

Bourdieu, P. (1977). *Outline of a theory of practice*. London: Cambridge University Press.

Braverman, H. (1974). *Labor and Monopoly Capital*. London: Monthly Review Press.

Brown, J.S. and P. Duguid (1991). "Organizational learning and communities of practice: toward a unified view of working, learning, and innovation." *Organization Science*, 2(1), pp. 40–57.

Brown, J.S. and P. Duguid (2000). *The Social Life of Information.* Boston, MA: Harvard Business School Press.

Brown, J.S. and P. Duguid (2001). "Knowledge and organization: a social-practice perspective." *Organization Science*, 12 (2), pp. 198–213.

Buckland, M. (2004). *Programming Game AI by Example.* Jones & Bartlett.

Burawoy, M. (1979). *Manufacturing Consent*. Chicago: University of Chicago Press.

Burawoy, M. (1997). "The anthropology of industrial work." *Annual Review of Anthropology,* 8, pp. 231–266.

Burawoy, M. (1998). "The extended case method." *Sociological Theory,* 16 (1), pp. 4–33.

Bush, V. (1945). "As we may think." *The Atlantic Monthly*, 176, pp. 101–108.

Cairncross, F. (1997). *The Death of Distance: How the Communications Revolution Is Changing Our Lives.* Cambridge, MA: Harvard Business School Press.

Cardoso, F.H. (1972). "Dependency and development in Latin America." *New Left Review,* 74, pp. 83–95.

Carvalho, M.S.R.M. (2006). *A Trajetória da Internet no Brasil: do Surgimento das Redes de Computadores à Instituição dos Mecanismos de Governança.* Masters Thesis. Universidade Federal do Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil.

Casanova, M.A., L. Tucherman, M.J.D. Lima, J.L.R. Netto, N. Rodriguez, and L.F.G. Soares (1991). "The nested context model for hyperdocuments," *Proceedings of the Third Annual ACM Conference on Hypertext.* New York, NY: ACM.

Castells, M. (1996/2000). *The Rise of the Network Society*. 2nd edition. Oxford: Blackwell.

Ceruzzi, P. (2003). *A History of Modern Computing.* 2nd edition, Cambridge, MA: The MIT Press.

Collins, H.M. (1974). "The TEA set: Tacit knowledge and scientific networks." *Social Studies of Science,* 4(2), pp. 165–185 .

Collins, H.M. (2001). "Tacit knowledge, trust and the Q of sapphire." *Social Studies of Science,* 31(1), pp. 71–85.

Contu, A. and H. Willmott (2003). "Reembedding situatedness : the importance of power relations in situated learning theory." *Organization Science*, 14 (3), 283–295.

Contu, A. and H. Willmott (2006). "Studying practice: situating *Talking About Machines,*" *Organization Studies*, 27 (12), 1769–1782.

Corbet, J. (2007). "Who wrote 2.6.20? LWN.net." http://lwn.net/Articles/222773/, last accessed October 30, 2008.

Coser, L.A. (1994). "Introduction." In E.C. Hughes, *On Work, Race, and the Sociological Imagination* (edited by L.A. Coser). Heritage of Sociology Series. Chicago: The University of Chicago Press.

Costa, R.M. de R., M.F. Moreno, R.F. Rodrigues, L.F.G. Soares (2006). "Live editing of hypermedia documents." *Proceedings of the 2006 ACM Symposium on Document*

Engineering. New York, NY: ACM.

Cowan, R., P.A. David, and D. Foray (2000). "The explicit economics of knowledge: codification and tacitness." *Industrial and Corporate Change,* 9 (2), 211–253.

Creswell, J. W. (1994). *Research Design: Qualitative and Quantitative Approaches.* Thousand Oaks, CA: SAGE.

Crowdy, S. (1993). "Spoken corpus design." *Literary and Linguistic Computing,* 8 (4), 259–265.

Dantas, M. (1989). *O Crime de Prometeu: Como o Brasil Obteve a Tecnologia da Informática.* Rio de Janeiro: Abicomp.

Dantas, V. (1988). *Guerrilha Tecnológica: A Verdadeira História da Política Nacional de Informática*, Rio de Janeiro: LTC. [Technological Guerrilla War: The True History of the National Informatics Policy, in Portuguese.]

Davies, D.W., K.A. Bartlett, R.A. Scantlebury, and P.T. Wilkinson (1967). "A digital communication network for computers giving rapid response at remote terminals." SOSP '67: Proceedings of the first ACM symposium on Operating System Principles, ACM Press, 2.1–2.17.

Dedijer, S. (1961). "Why did Daedalus leave?" *Science*, New Series, 133 (3470), pp. 2047–2052.

Dees, T. 2006. "Tradition vs. ten codes." http://www.officer.com/interactive/2006/11/13/tencodes/, last accessed May 19, 2009.

Delorey, D.P., C.D. Knutson, and C. Giraud-Carrier (2007). "Programming language trends in open source development: an evaluation using data from all production phase SourceForge projects." Second International Workshop on Public Data about Software Development, Limerick, Ireland.

Desmond, M. (2007). *On the Fireline: Living and Dying with Wildland Firefighters.* Chicago: The University Of Chicago Press.

Dickheiser, M. (2006). *Game Programming Gems 6.* Charles River Media.

Didier, D., E.T. Weber, J.A. Pimenta-Bueno (2005). "Gávea angels: the birth of an angel group in Rio de Janeiro." Chapter 8 in E.F. O'Halloran, P.L.Rodriguez, and F. Vergara (eds.), *Angel Investing in Latin America.* Pp. 51–60.

Dos Santos, T. (1970). "The structure of dependence." *The American Economic Review,* 60(2), pp. 235–246.

Drucker, P.F. (1969). *The Age of Discontinuity: Guidelines to Our Changing Society.* New York: Harper & Row.

Duguid, P. (2005). "'The Art of Knowing': social and tacit dimensions of knowledge and the limits of the community of practice." *Information Society*, 21 (2), pp. 109–118.

Duguid, P. (2008). "The community of practice then and now." In A. Amin and J. Roberts (eds.), *Organizing for the Creative Economy: Community, Practice, and Capitalism.* Oxford: Oxford University Press.

Elias P. and M. Birch (1994). "Establishment of Community-Wide Occupational Statistics. ISCO 88 (COM): A Guide for Users." Institute for Employment Research, University of Warwick.

Emerson, R., R. Fretz, and L. Shaw (1995). *Writing Ethnographic Fieldnotes.* University of Chicago Press.

Engelbart, D. (1962). "Augmenting human intellect: a conceptual framework." SRI Report, [Online] http://www.bootstrap.org/augdocs/friedewald030402/augmentinghumanintellect/ahi62index.html, last accessed February 27, 2009.

Engelbart, D. (1962). "Augmenting human intellect: a conceptual framework." SRI Report.

Engeström, Y. (2001). "Expansive learning at work: toward an activity-theoretical reconceptualisation." *Journal of Education and Work,* 14 (1), pp. 129–152.

Evans, P. (1979). *Dependent Development: The Alliance of Multinational, State, and Local Capital in Brazil.* Princeton, NJ: Princeton University Press.

Evans, P. (1995). *Embedded Autonomy: States and Industrial Transformation.* Princeton, NJ: Princeton University Press.

Favret-Saada, J. (1980). *Deadly Words: Witchcraft in the Bocage.* Cambridge: Cambridge University Press.

Fenwick & West, LLP (2008). "Trends in Terms of Venture Financings In the San Francisco Bay Area (Fourth Quarter 2007)"

Ferguson, C. (1959/1971). "Diglossia." In A.S. Dil (ed.), *Language Structure and Language Use: Essays by Charles A. Ferguson.* Stanford, CA: Stanford University Press.

Figueiredo, L.H. (1992). "DEL: uma linguagem para entrada de dados." TeCGraf/ICAD.

Figueiredo, L.H. de, R. Ierusalimschy, and W. Celes (1996). "Lua: an extensible embedded language." *Dr. Dobb's Journal,* 21 (12), pp. 26–33.

Figueiredo, L.H., C. S. Souza, M. Gattass, and L. C. G. Coelho (1992). "Geração de interfaces para captura de dados sobre desenhos." Proceedings of SIBGRAPI '92 (Brazilian Symposium on Computer Graphics and Image Processing), pp. 169–175.

Figueiredo, L.H., R. Ierusalimschy, and W. Celes. (1996). "Lua: an extensible embedded language." Dr. Dobb's Journal, 21(12), pp. 26–33.

Fine, G.A. (1995). Preface. In Fine. G.A., ed. *A Second Chicago School?: The Development of a Postwar American Sociology.* Chicago: The University of Chicago Press.

Fogel, K. (2005). *Producing Open Source Software. How to Run a Successful Free Software Project.* Sebastopol, CA: O'Reilly.

Form, W. (1987). "On the degradation of skills." *Annual Review of Sociology*, 13, pp. 29–47.

Freire, F.R.F. (1993). *Pró-censo: algumas notas sobre os recursos para processamento de dados nos Recenseasmentos do Brasil.* Memória Institucional–3. Rio de Janeiro: IBGE.

Fritz, W.B. (1996). "The Women of ENIAC." *IEEE Annals of the History of Computing,* 18 (3), pp. 13–28.

Gallini, N. and S. Scotchmer (2002). "Intellectual property: when is it the best incentive mechanism?" In: A. Jaffe, J. Lerner and S. Stern (eds.), *Innovation Policy and the Economy.* Vol 2. Cambridge, MA: MIT Press. Pp. 51–78.

Galpin, V. (2002). "Women in computing around the world." SIGSCE Bulletin, 34 (2).

Garfinkel, H. (1967). *Studies in Ethnomethodology*. Englewood Cliffs, NJ: Prentice Hall.

Geertz, C. (1973). "Thick description: toward an interpretive theory of culture." In *The Interpretation of Cultures: Selected Essays.* New York: Basic Books. Pp. 3–30.

Giddens, A. (1971). *Capitalism and Modern Social Theory: An Analysis of the Writings of Marx, Durkheim and Max Weber.* Cambridge: Cambridge University Press.

Giddens, A. (1979). *Central Problems in Social Theory.* London: Macmillan.

Giddens, A. (1990). *The Consequences of Modernity.* Stanford, CA: Stanford University Press.

Giddens, A., M. Duneier, and R.P. Appelbaum (2007). *Introduction to Sociology.* New York: W. W. Norton.

Gieryn, T. F. (1983). "Boundary-work and the demarcation of science from non-science: strains and interests in professional ideologies of scientists." *American Sociological Review,* 48 (6), pp. 781–795.

Gilbert, D. and J. Whitehead (2007). *Hacking World of Warcraft.* Indianapolis, IN: Wiley.

Glaser, B.G. (1992). *Basics of Grounded Theory Analysis: Emergence vs Forcing.* Mill Valley, CA: Sociology Press.

Glaser, B.G. (1998). *Doing Grounded Theory: Issues and Discussions*. Mill Valley, CA: Sociology Press.

Glaser, B.G. and A.L. Strauss (1967/1999). *The Discovery of Grounded Theory: Strategies for Qualitative Research.* New York: Aldine de Gruyter.

Goldhaber, M. (1997). "The attention economy and the Net." *First Monday*, 2(4), available at http://www.firstmonday.dk/issues/issue2_4/goldhaber/.

Goldhaber, M. (2006). "The value of openness in an attention economy." *First Monday*, 11(6), available at http://www.firstmonday.org/issues/issue11_6/goldhaber/.

Goldman, A. (1998/2002). *Knowledge in a Social World.* Oxford: Oxford University Press.

Graham, P. (2004/2005). "Great hackers." In J. Spolsky (ed.), *The Best Software Writing I.* Apress, pp. 95–110. The original version is available at http://www.paulgraham.com/gh.html, last accessed on May 19, 2009.

Graham, P. (2006). "How to be Silicon Valley." http://www.paulgraham.com/siliconvalley.html. Last accessed June 19, 2008.

Grosjean, F. (1982). *Life with Two Languages: An Introduction to Bilingualism.* Cambridge, MA: Harvard University Press.

Gutschmidt T. (2003). *Game Programming with Python, Lua, and Ruby.* Course Technology PTR.

Harvard Law Review (1997). "Contracts. Independent contractor agreements. Ninth circuit finds that misclassified employees are eligible for federally regulated employee benefits. Vizcaino v. Microsoft Corp., 120 F.3d 1006 (9th Cir. 1997) (En banc)." *Harvard Law Review,* 111 (2), pp. 609–614.

Hayek, F.A. (1937). "Economics and knowledge." *Economica,* 4(13), pp. 35–54.

Hayek, F.A. (1945). "The use of knowledge in society." *The American Economic Review,* 35, pp. 519–30.

Hessel, E. (2008). "Opening your TV to the web." Forbes.com, http://www.forbes.com/2008/12/15/vudu-hulu-tv-tech-personal-cz_eh_1216vudu.html, last accessed March 16, 2009.

Hester, A., R. Borges and R. Ierusalimschy (1997). "CGILua: a multi-paradigmatic tool for creating dynamic WWW pages." In *XI Simpósio Brasileiro de Engenharia de Software.* Fortaleza, CE. Pp. 347–360.

Hester, A., R. Borges and R. Ierusalimschy (1998). "Building flexible and extensible web applications with Lua." *Journal of Universal Computer Science,* 4(9), pp. 748–762.

Hildreth, P.M. and C. Kimble (2004). *Knowledge Networks: Innovation through Communities of Practice.* Idea Group Publishing.

Hildreth, P.M., C. Kimble, and P. Wright (1998). "Computer mediated communications and international communities of practice." Proceedings of Ethicomp '98, Erasmus University, The Netherlands. Pp. 275–286.

Hill, B.M. (2005). "Financing volunteer free software projects." *Advogato.* http://www.advogato.org/article/844.html, last accessed March 12, 2009.

Hirschi, A. (2007). "Traveling light, the Lua way." IEEE Software, September/October 2007, pp. 31–38.

Hochschild, A.R. (1989). *The Second Shift: Working Parents and the Revolution at Home.* New York: Viking.

Hoffmann, E. (1999). "International statistical compariosns of occupational and social structures: problems, possibilities and the role of ISCO-88," International Labour Office.

Hughes, E. (1958). *Men and their work.* Glencoe, IL: Free Press.

Hughes, E. (1959/1995). "The Study of Occupations." in E.C. Hughes, *On Work, Race, and the Sociological Imagination* (edited by L.A. Coser). Heritage of Sociology Series. Chicago: The University of Chicago Press. Pp. 21–36.

Huskey, V.R. and H.D. Huskey (1980). "Lady Lovelace and Charles Babbage." *Annals of the History of Computing,* 2(4), pp. 299–329.

IBM1130.org (n.d.). “All about the IBM 1130 Computing System.” http://ibm1130.org, last accessed on May 19, 2009.

Ierusalimschy, R, L.H. de Figueiredo, and W. Celes. (2007). “The evolution of Lua,” *Proceedings of ACM HOPL III.*

Ierusalimschy, R. (2003). *Programming in Lua.* Rio de Janeiro: Lua.org.

Ierusalimschy, R. (2006). *Programming in Lua.* Second edition. Rio de Janeiro: Lua.org.

Ierusalimschy, R. (2007). *Programmieren mit Lua*. Open Source Press.

Ierusalimschy, R., L. H. de Figueiredo , W. C. Filho (1996). “Lua—an extensible extension language.” Software—Practice & Experience, 26 (6), pp. 635–652.

Ierusalimschy, R., L.H. de Figueiredo, and W. Celes (2001). “The evolution of an extension language: a history of Lua.” *Proceedings of V Brazilian Symposium on Programming Languages,* pp. B14–B28.

Ierusalimschy, R., L.H. de Figueiredo, and W. Celes (2006). *Lua 5.1 Reference Manual.* Rio de Janeiro: Lua.org.

Johnson, H.G. (1965). “The economics of the ‘brain drain’: the Canadian case,” *Minerva*, 3(3), pp. 299–311.

Jorgensen, D.L. (1989). *Participant Observation: A Methodology for Human Studies*. Applied Social Research Methods Series, Volume 15. SAGE Publications: Newbury Park, California.

Jung K. and A. Brown. (2007). *Beginning Lua Programming.* Wrox.

Katz, M.L. and C. Shapiro (1986). “Technology adoption in the presence of network externalities.” *The Journal of Political Economy,* 94(4), pp. 822–841.

Kendall, L. (1999). “Nerd nation: images of nerds in US popular culture.” *International Journal of Cultural Studies,* 2(2), pp. 260–283.

Knorr Cetina, K. (1999). *Epistemic Cultures: How the Sciences Make Knowledge.* Cambridge, MA: Harvard University Press.

Knuth, D. (n.d.). “Email (let’s drop the hyphen).” http://www-cs-faculty.stanford.edu/~knuth/email.html, last accessed on May 19, 2009.

Koss, A.M. (2003). “Programming on the Univac 1: a woman’s account.” IEEE Annals of the History of Computing, pp. 48–59.

Kuhn, T. (1962). *The Structure of Scientific Revolutions.* Chicago: University of Chicago Press.

Kunda, G. (1992). *Engineering Culture: Control and Commitment in a High-Tech Corporation.* Philadelphia, PA. Temple University Press.

Laird, C. and K. Soraiz (1998). “1998: breakthrough year for scripting,” SunWorld, http://sunsite.uakom.sk/sunworldonline/swol-08-1998/swol-08-regex.html, last accessed on May 19, 2009.

Lakatos, I. (1968/1970). “Falsification and the methodology of scientific research programmes.” In I. Lakatos and A. Musgrave (eds), *Criticism and the Growth of Knowledge.* Cambridge: Cambridge University Press. Pp. 91–196.

Lakoff, G. and M. Johnson (1980). *Metaphors We Live By.* Chicago, IL: University of Chicago Press.

Lamont, M. (2000). *The Dignity of Working Men.* Harvard University Press: Cambridge, MA.

Lamont, M. and V. Molnár (2002). “The study of boundaries in the social sciences.” *Annual Review of Sociology,* 28, pp. 167–195.

Latour, B. (1988a). *Science in Action: How to Follow Scientists and Engineers Through Society.* Cambridge, MA: Harvard University Press.

Latour, B. (1988b). *The Pasteurization of France.* Cambridge, MA: Harvard University Press.

Latour, B. and S. Woolgar (1979/1984). *Laboratory Life: The Construction of Scientific Facts.* Princeton, NJ: Princeton University Press.

Lattes, Sistema de Currículos (2009a). “Arndt von Staa.” Available at http://lattes.cnpq.br/3387018530825694, last accessed on May 19, 2009.

Lattes, Sistema de Currículos (2009b). “Carlos José Pereira de Lucena.” Available at http://lattes.cnpq.br/2491891766759477, last accessed on May 19, 2009.

Lave, J. and Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation.* Cambridge: Cambridge University Press.

Law, J. (1986). “On the methods of long distance control: vessels, navigation and the Portuguese route to India.” In J. Law (ed.), *Power, Action and Belief: a New Sociology of Knowledge?* Sociological Review Monograph. London: Routledge. Pp. 234–263.

Leontiev, A.N. (1972/1981). “The problem of activity in psychology.” In J.V. Wertsch (ed.). *The Concept of Activity in Soviet Psychology.* M.E. Sharpe. Pp. 37–71.

Levine, E. (1972). “Chicago’s art world: the influence of status interests on its social and distribution systems.” *Urban Life and Culture,* 1(3), pp. 293–322.

Luhn, H. (1961). “The automatic derivation of information retrieval encodements from machine-readable texts.” In A. Kent (ed.), *Information Retrieval and Machine Translation, part 2, Interscience Publication,* 3, pp. 1021–1028.

Luzio, E. (1996). *The Microcomputer Industry in Brazil: The Case of a Protected High-Technology Industry.* Westport, CT: Praeger.

MacKenzie, D. and G. Spinardi (1995). “Tacit knowledge and the uninvention of nuclear weapons,” *The American Journal of Sociology,* 101(1), pp. 44–99.

Marques, I. da Costa (2000). “Reserva de mercado: um mal entendido caso político-tecnológico de "sucesso" democrático e "fracasso" autoritário.” *Revista de Economia da Universidade Federal de Paraná,* 26 (24), , pp. 91–116.

Marques, I. da Costa (2003). “Minicomputadores brasileiros nos anos 1970: uma reserva de

mercado democrática em meio ao autoritarismo." *História, Ciências, Saúde Manguinhos,* 10 (2), pp. 657–81.

Marshall, A. (1890/1927). "Book IV: The Agents of Production. Chapter X. Industrial organization, continued. The concentration of specialized industries in particular localities" In *Principles of Economics: An Introductory Volume,* London: McMillan and Co. Pp. 267–277.

Marx, K. (1867/1977). *Capital.* Volume One. New York: Vintage Books.

Marx, K. (1978). "Economic and Philosophic Manuscripts of 1844." In R.C. Tucker (ed.), *The Marx-Engels Reader.* Second edition. New York: W.W. Norton and Co.

Mayo, E. (1933). *The Human Problems of an Industrial Civilization.* New York: Macmillan.

Meyer, J., J. Boli, G. Thomas, G. and F. Ramirez (1997). "World Society and the Nation-State." *American Journal of Sociology,* 103(1), pp. 144–181.

Mills, C.W. (1959). *The Sociological Imagination.* Oxford: Oxford University Press.

Morais, F. (2006). *Montenegro: As Aventuras do Marechal que Fez uma Revolução nos Céus do Brasil.* São Paulo: Planeta.

Muysken, P. (2000). *Bilingual Speech: A Typology of Code Mixing.* Cambridge: Cambridge University Press.

Nelson, T. (1974/1987). *Computer Lib/dream Machines.* Revised edition. Redmond, WA: Tempus Books of Microsoft Press.

NRC (1941). *Tour of industrial exploration — South America.* Washington, DC, National Research Council.

Ó Riain, S. (2000). "Networking for a living: Irish software developers in the global workplace." In M. Burawoy et al (9 co-editors), *Global Ethnography.* Berkeley: University of California Press. Pp. 175–202.

Olinto, G. (2007). "Internet access in Brazil: social context and science and technology professionals." Paper presented at the American Society for Information Science and Technology Annual Meeting, Milwaukee, October 21–24.

Orr, J. (1996). *Talking About Machines: An Ethnography of a Modern Job.* Ithaca, NY: Cornell University Press.

Otlet, P. (1925/1990). "The preservation and international diffusion of thought: the microphotic book." In W.B. Rayward (ed.), *International Organization and Dissemination of Knowledge: Selected Essays of Paul Otlet.* Elsevier. Pp. 204–210.

Perez, S. (2009). "Verizon unveils their vision for the Web-Connected TV." *ReadWriteWeb.* http://www.readwriteweb.com/archives/verizon_unveils_their_vision_for_the_web-connected.php, accessed March 16, 2009.

Petersen, W.E.P. (1994). *Almost Perfect: How a Bunch of Regular Guys Built WordPerfect Corporation.* Rocklin, CA : Prima Publishing.

Polanyi, M. (1952). *Personal Knowledge: Towards a Post-Critical Philosophy.* New York: Harper Torchbooks.

Polanyi, M. (1966). *The Tacit Dimension*. Doubleday.

Popper, K. (1959). *The Logic of Scientific Discovery.* New York: Basic Books.

Powell, W., K.W. Koput, J.I. Bowie, and L. Smith-Doerr (2002). "The spatial clustering of science and capital: accounting for biotech firm–venture capital relationships." *Regional Studies*, 36 (3), pp. 291–305.

Qualis (2009). "Ciência da computação," an Excel file downloaded from http://qualis.capes.gov.br/webqualis/ConsultaListaCompletaPeriodicos.faces, last accessed on May 20, 2009.

Rasmussen D. (1979). "Marx: on labor, praxis and instrumental reason." *Studies in Soviet Thought,* 20, pp. 271–289.

Raymond, E.S. (1999). *The Cathedral and The Bazaar.* Sebastopol, CA: O'Reilly.

Reddy, M. (1979). "The conduit metaphor: a case of frame conflict in our language about language." In A. Ortony (ed.), *Metaphor and Thought.* Cambridge: Cambridge University Press. Pp. 284–324.

Ritzer. G. (1989). "Sociology of work: a metatheoretical analysis." *Social Forces,* 67 (3), pp. 593–604.

Rogers , C.R. (1945). "The nondirective method as a technique for social research." *American Journal of Sociology,* 50 (4), pp. 279–283.

Romer, P. (1990). "Endogenous Technological Change." *The Journal of Political Economy,* 98(5), pp. S71–S102.

Romer, P. (1993). "Idea gaps and object gaps in economic development." *Journal of Monetary Economics,* 32, pp. 543–573.

Ryle, G. (1949). "Knowing how and knowing that." In *The Concept of Mind.* Chicago: The University of Chicago Press. Pp. 25–61.

Santos, C.A.S., L.F.G. Soares, G.L. de Souza, J.-P. Courtiat (1998). "Design Methodology and Formal Validation of Hypermedia Documents." *Proceedings of the Sixth ACM International Conference on Multimedia.* New York: ACM.

Saxenian, A. (1996). *Regional Advantage: Culture and Competition in Silicon Valley and Route 128.* Cambridge, MA: Harvard University Press.

Saxenian, A. (1999). "The Silicon Valley-Hsinchu connection: technical communities and industrial upgrading." Working Paper No. 99-10, Stanford Institute for Economic Policy Research.

Saxenian, A. (2005). "From brain drain to brain circulation: transnational communities and regional upgrading in India and China." *Studies in Comparative International Development,* 40(2).

Saxenian, A. (2006). *The New Argonauts: Regional Advantage in a Global Economy.* Cambridge, MA: Harvard University Press.

Schatzki, T. (1996). *Social Practices: A Wittgensteinian Approach to Human Activity and the Social.* Cambridge, UK: Cambridge University Press.

Schmookler, J. (1954). "The level of inventive activity." *The Review of Economics and Statistics,* 36, pp. 183–190.

Schmookler, J. (1965). "Technological change and economic theory." *The American Economic Review,* 55, pp. 333–341.

Schoonmaker, S. (2002). *High-Tech Trade Wars: U.S. Brazilian Conflict in the Global Economy.* Pittsburgh, PA: University of Pittsburgh Press.

Schoonmaker, S. (forthcoming). "Software politics in Brazil: toward a political economy of digital inclusion." *Information, Communication and Society.*

Schuytema, P. and M. Manyen (2005). *Game Development With LUA.* Charles River Media.

Schwarz, M. & Y. Takhteyev (2009). "Half a Century of Public Software: Open Source as a Solution to the Hold Up Problem." NBER Working Paper No. 14946, http://www.nber.org/papers/w14946.

Senra, N. (2007). *História das Estatísticas Brasileiras.* Volume 3, *Estatísticas Organizadas (c.1936–c.1972).* Rio de Janeiro.: IBGE.

Shaw, R.R. (1949). "The Rapid Selector." *The Journal of Documentation,* 5, pp. 164-171.

Sheridan, M. B. (2006). "Va. state police swap '10-4' for 'Message Understood.'" http://www.washingtonpost.com/wp-dyn/content/article/2006/11/12/AR2006111201098.html, last accessed May 19, 2009.

Shibutani, T. (1955). "Reference groups as perspectives." *American Journal of Sociology,* 60, pp. 562–569.

Silberman, S. (2001). "The geek syndrome." http://www.wired.com/wired/archive/9.12/aspergers_pr.html

Simpson, I. (1989). "The sociology of work: where have the workers gone?" *Social Forces,* 67 (3), pp. 563–581.

Solow, R.M. (1957). "Technical change and the aggregate production function." *The Review of Economics and Statistics,* 39, pp. 312–320.

Spence, M. (1973). "Job market signaling." *Quarterly Journal of Economics,* 87 (3), pp. 355–374.

Spolsky, J. (2000/2004). "The guerrilla guide to interviewing." In J. Spolsky, *Joel on Software: And on Diverse and Occasionally Related Matters That Will Prove of Interest to Software Developers, Designers, and Managers, and to Those Who, Whether by Good Fortune or Ill Luck, Work with Them in Some Capacity.* Apress. Pp. 153–166. The original blog post is available at http://www.joelonsoftware.com/articles/fog0000000073.html, last accessed on May 19, 2009.

Staa, A. v. (2003). “Fase heróica (67 a 70).” In A. von Staa, A.L. Furtado, and S.D.J. Barbosa (eds), *Carlos José Pereira de Lucena: Pioneiro da Informática.* Rio de Janeiro: PUC-Rio.

Stallman, R. (2002). Miscellaneous essays in J. Gay (ed.), *Free Software, Free Society: Selected Essays of Richard M. Stallman.* Boston, MA: GNU Press.

Strauss, A. (1978). “A social world perspective.” *Studies in Symbolic Interaction,* 1. Greenwich, CT: JAI Press, pp. 119–128.

Strauss, A. (1982). “Social Worlds and Legitimation Processes.” *Studies in Symbolic Interaction,* 4. Greenwich, CT: JAI Press. Pp. 171–190.

Strauss, A. and J. Corbin (1990). *Basics of Qualitative Research: Grounded Theory Procedures and Techniques.* Thousand Oaks, CA: Sage.

Suchman, L. (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication.* Cambridge: Cambridge University Press.

Takhteyev, Y. (2005). “Googling Across the Equator,” http://www.takhteyev.org/papers/Takhteyev-2005-Googling-Across-the-Equator.pdf

Tigre, P. (2003). “Brazil in the age of electronic commerce.” *Information Society,* 19(11), pp. 33–43.

Time (1994). “Diagnosing Bill Gates.” http://www.time.com/time/magazine/article/0,9171,979990,00.html

Torvalds, L. (2001). *Just for Fun: The Story of an Accidental Revolutionary.* New York: HarperCollins.

Traweek, S. (1988/1992). *Beamtimes and Lifetimes: The World of High Energy Physicists.* Cambridge, MA: Harvard University Press.

Turner, F. (2006). *From Counterculture to Cyberculture: Stewart Brand, the Whole Earth Network, and the Rise of Digital Utopianism.* Chicago: University of Chicago Press.

Turner, S. (2008). “The social study of science before Kuhn.” In E.J. Hackett, O. Amsterdamska, M. Lynch, and J. Wajcman (eds.), *The Handbook of Science and Technology Studies.* Third Edition. Cambridge, MA: MIT Press. Pp. 33-62.

Ueno, Y. (2007). *Introduction to Lua Programming* [in Japanese]. Softbank Creative.

Unruh, D. (1980). “The nature of social worlds.” *Pacific Sociological Review,* 23, pp. 271–296.

Van Maanen, J. (1988). *Tales of the Field: On Writing Ethnography.* Chicago: University Of Chicago Press.

Van Maanen, J. and Barley, S. R. (1984). “Occupational communities: culture and control in organizations.” *Research in Organizational Behavior,* 6, pp. 287–365.

Van Maanen, J. and S. Barley (1985). “Cultural Organization: Fragments of a Theory.” In P.J. Frost, L.F. Moore, M.R. Luis, C.C. Lundberg, and J. Martin (eds), *Organizational Culture.* Beverly Hills, CA: Sage Publications. Pp. 31–54.

Varanese, A. (2002). *Game Scripting Mastery.* Course Technology PTR. Cincinnati, OH: Premier Press.

Veloso, F., A.J. Botelho, T. Tschang, and A. Amsden. (2003). "Slicing the knowledge-based economy in Brazil, China and India: a tale of 3 software industries." Technical report, SOFTEX Brazil and Ministry of Science and Technology, Brazil.

Vygotsky, L. (1978). "Interaction between learning and development." In M. Cole (ed., transl.), *Mind in Society.* Cambridge, MA: Harvard University Press. Pp. 79–91.

Vygotsky, L. S. (1930/2002). "Orudie i znak v razvitii rebenka." [Tool and sign in child development, in Russian.] In L.S. Vygotsky, *Psikhologiya.* [*Psychology.*] Moscow: EKSMOPress.

Weiss, R.S. (1994). *Learning from Strangers: The Art and Method of Qualitative Interview Studies.* New York: Free Press.

Wellman, B. and K. Hampton (1999). "Living Networked On and Offline" *Contemporary Sociology,* 28 (6), pp. 648–54

Wells, H.G. (1938). *World Brain.* London: Meuthuen & Co. Limited.

Wenger, E. (1998). *Communities of Practice: Learning, Meaning, and Identity.* Cambridge: Cambridge University Press.

Whitehead II, J., B. McLemore, and M. Orlando (2008). *World of Warcraft Programming: A Guide and Reference for Creating WoW Addons.* Indianapolis, IN: Wiley.

Williams, Sam. (2002). *Free as in Freedom: Richard Stallman's Crusade for Free Software.* Sebastopol, CA: O'Reilly.

Willis, P. (1977/1981). *Learning to Labor: How the Working Class Kids Get Working Class Jobs.* Morningside Edition. New York: Columbia University Press.

Woolgar, Steve (1991). "Configuring the user: the case of usability trials." In J. Law (ed.), *A Sociology of Monsters: Essays on Power, Technology, and Domination.* London, UK: Routledge.

Zook, M.A. (2002). "Grounded capital: venture financing and the geography of the Internet industry, 1994–2000." *Journal of Economic Geography,* 2, pp. 151–177.

Appendices

Appendix A: The Participants

Major Participants

Roberto Ierusalimschy (real name)
: A professor at PUC-Rio, author of Lua.[371]

Luiz Henrique de Figueiredo (real name)
: A researcher at IMPA, author of Lua.

Rodrigo Miranda
: The project leader for Kepler, former Ph.D. student at PUC-Rio (with Roberto Ierusalimschy), affiliated with Nas Nuvens, a company sponsoring Kepler.

Jason, also **Zé Luis**

: A software developer originally from Nova Iguaçu, a city near Rio. Now working at Nas Nuvens.

Other Participants Mentioned in the Text

The participants are listed in alphabetical order by first name, as referred in the text. With one exception (noted below), all names are pseudonyms. The participants were interviewed in 2007, except where stated otherwise.

371 See the glossary for the underlined terms.

Alan
: A Kepler contributor living outside Rio (interviewed by phone in 2007 and 2008).

Antônio
: A manager at Tecgraf, graduate of PUC-Rio, a user of Lua since 1993. (Interviewed twice: in 2005 and 2007.)

Bruno
: A PUC professor and a researcher at Tecgraf. An early user of Lua.

Carlos
: A higher-up official at PUC working to promote Lua at PUC, Chico's boss.

Cássio
: A developer at Tecgraf, the original author of LuaForm, the predecessor of CGILua. (Interviewed twice: in 2005 and 2007.)

Célio
: A Senior Systems Analyst for a foreign consulting company.

Chico
: A Lua evangelist at PUC, friend of Roberto Ierusalimschy.

Craig
: An engineer at a startup in Silicon Valley, a user of Lua.

Daniel
: A partner at Nas Nuvens.

Edmundo
: A software developer working for a large foreign hardware company in Barra da Tijuca (interviewed in 2005).

Eduardo
: A founding partner at Alta (not formally interviewed).

Fabio
: A partner at Alta and a former contributor to Kepler.

Felipe
: A founding partner of Alta.

Jaime
: A software freelancer working with Delphi and looking to move to Java.

João
: A brother of Rodrigo Miranda, the founder and head of Nas Nuvens.

Jorge
: A former electronics researcher now teaching computer science.

Leonardo

A minority partner at Alta.

Luciano
A software developer working at Nas Nuvens.

Luis
A founding partner of Alta

Luiz Fernando Soares (real name)
A PUC professor heading development of Ginga.

Márcio
A Kepler contributor working for PUC IT.

Marcos
An employee at Alta (not formally interviewed).

Mauricio
An Alta employee

Michel
A researcher working on Ginga.

Miguel
A developer in a software company.

Pedro
A software developer working at Nas Nuvens.

Renato
A friend of Rodrigo Miranda, now working as a manager for a software company using Java.

Ricardo
A software developer at PUC working on a project based on Kepler.

Rich
An software consultant in Berkeley, a user of Lua.

Rogerio
A minority partner at Alta, working on a Kepler project together with Fabio in 2007.

Silvio
A researcher at PUC and an early user of Lua.

Steve
A team lead in a large software company in California, a user of Lua.

Tiago
A Kepler contributor and a Ph.D. student at PUC.

Other Participants Interviewed in 2007

Table 1 lists participants interviewed in 2007 not included in the list above (but including those who were also interviewed in 2005). Basic demographic information is provided for whom it was recorded. The list does not include informal conversations.

Table 1: Other participants interviewed in 2007.

Age	Sex	Education	Description
30s	M		An independent Mac shareware developer interviewed in San Francisco.
50s	M		An embedded systems developer in Southern California (interviewed by phone).
	M		A scripting consultant in Berkeley.
20s	M	in college in the US	An American university student in Missouri (interviewed by phone).
30s	M	MS in the US	A Brazilian software development who recently returned from the Unites States.
30s	M	in college (private)	A software developer at company in partnership with Nas Nuvens.
	M		A software developer at IBGE in the 1970s, now running a small software consulting company.
60s	M	college	A software developer at IBGE, working there since the 1970s.
70s	M	high school	A manager at IBGE in the 1960s.
20s	M	PUC	An independent software consultant who occasionally works on Kepler.
30s	M	PUC	An independent software consultant who occasionally works on Kepler.
30s	M	Ph.D. in Canada	A software developer for a large Brazilian company.
	M		A partner of a small company working together with Nas Nuvens .
20s	M	PUC	A software developer at Alta.
50s	M	Ph.D. in the US	A professor at PUC.
20s	M	in college (private)	A software developer at Alta.
20s	M	incomplete college (private)	A software developer at Nas Nuvens.
20s	M	in college (private, in progress)	A software developer at the same company as Jason's company.
30s	M		A software developer from another Latin American country now working in Rio.
40s	M	PUC	A manager of a web design company.
20s	M	pursuing MS at UFRJ	A Masters student at UFRJ.
			An officer of a funding agency.

Other Participants Interviewed in 2005

Table 2 lists participants interviewed in 2005, excluding those included in the list or tables earlier in this appendix. Basic demographic information is provided for whom it was recorded. The list does not include informal conversations.

Table 2: Additional participants interviewed in 2005.

Age	Sex	Education	Description
	M	Ph.D.	A professor at UFRJ
30s	M	UERJ	A system administrator at an industry research center.
30s	M	college	A system administrator at an industry research center.
30s	M	PUC	A software developer for a cell phone provider.
30s	M	pursuing MS at UERJ	A Java developer.
40s	M	PUC	A manager for a small software company.
50s	M	Ph.D. in the US	A high level manager for a large US-based company.
40s	M	Ph.D. at PUC	A manager for a large US-based company.
50s	M	MS at PUC	A software contractor.
50s	M	Ph.D. in Europe	A professor at UFRJ.
30s	M	in college (private)	A software developer for a large telecommunications company and a free software activist
40s	M	MS abroad	A partner at a electronics company.
40s	M	UFRJ	A former COBOL programmer, now a Java developer at a research center.
20s	M	UERJ	A software contractor at Petrobras.
20s	F	college (private)	A computer trainer.
20s	M	UERJ	A Java developer working for Petrobras.
20s	M	in college (private)	A web developer working with Python.
40s	M	some PUC	A software consultant for a foreign company.
30s	M	in college (UFRJ)	A Java developer trying to start his own company.
20s	M	college (private)	A database administrator for a consulting company.
40s	M	UFRJ	A former COBOL programmer, now a Java developer at a research center .
40s	M	MS in Brazil	A technology consultant.
30s	M	college (private)	A sales engineer for a foreign company.
20s	M	a federal university outside RIo	A systems analyst for a small company.
40s	M	USP	A manager for a consulting company, working with Java.
20s	M	college (private)	A Java developer for a consulting company.

40s	M	PUC	A head of a software company.
30s	M	high school	A Visual Basic developer working for an NGO.
50s	M	UFRJ	An independent software contractor, web developer and an adjunct professor.
20s	M	PUC	A Java developer for a consulting company.
40s	M	UFRJ	A Java software architect working for a consulting company.
20s	M	some college (private, in progress)	A Java instructor and independent contractor.
30s	F	PUC	A software developer using Java and Python, working for a small office of a foreign company.
20s	M	college (private)	A manager at a training company.
20s	M	PUC and some MS	A Java developer, working for a small office of a foreign company.
30s	M	MS at UFRJ	A systems analyst for a government organization.
20s	M	MS at PUC	A partner at a small software product company (grew substantially by 2007).
20s	M	MS at PUC	A partner at a small software product company (closed by 2007).
20s	M	high school	A Python developer for a small company in Barra.
30s	M	MS abroad	A manager for a small office of a foreign company.
	F	some Ph.D. at PUC	A Ph.D. student at PUC .

Appendix B: Transcription and Quoting Method

Quoting

In addition to written sources, there are three sources of quotations in the text: (1) the transcripts of recorded interviews, (2) phrases written down during interviews for which I do not have recorded audio, (3) specific utterances remembered during my presence in the field (including sometimes during interviews) and recorded in my field notes *after the fact*, sometimes on the same day, sometimes one or two days later.

Interviews

Quotes taken from recorded transcription are typically represented as long quotations and are always presented verbatim, with the following exceptions:

I correct minor disfluencies when they do not appear to be relevant to the interpretation of the quote. This includes removal of almost all *uhs*, and *hms*, my feedback utterances (*uhu*'s) that overlap with interviewee's speech, and some of the minor self-corrections or repetitions. For instance, "and that that that it would be strange" would typically be quoted as "and that it would be strange." For more serious self-correction, which may represent a major change in what the interviewee was saying, I leave the utterance as is, using ellipsis to show the place where the grammatical flow is broken: "I

can maybe try to find some... but I think that I usually already wrote many things in English." Alternatively, I truncate the corrected part, replacing it with [...]. (In my usage, "[...]" represents an omission, but "..." does not, and may or may not correspond to a pause.)

I correct minor grammatical errors for interviews conducted in English and in my own Portuguese utterances in Portuguese interviews. I do not, however, correct non-standard or even plain wrong use of vocabulary (either for by myself or by my interviewees). E.g., if an interviewee uses the word "propaganda" in place of English "ads" and I leave it this way. I also leave the speech uncorrected if the usage is relevant to the flow of conversation (e.g., if it creates a confusion between me and the interviewee).

I frequently remove parts of the utterance to shorten quotes and make them easier to read. I represent such removals with [...]. Such removals may span several dialog turns, though I do not splice together distant parts of in interview.

I mark laughter, chuckles and pauses selectively, when they appear to be relevant to understanding the conversation.

For recorded interviews, I sometimes also include shorter quotes in the paragraph, inside quotations marks. Those shorter quotes are transcribed in the same way as long quotes.

Some of the interviews were not recorded. Such quotes follow conventions described in the next section.

Quotes from field notes

Quotes taken from field notes, either based on interviews or observed scenes, are obviously not as reliable as those from recorded interviews. When I place a phrase in quotes while describing a scene, this means that the quote was recorded in my notes, and that at the time of recording it I was reasonably sure that it represented the utterance verbatim or nearly verbatim. Some of those quotes were spoken and recorded in my notes in Portuguese, while others were spoken in Portuguese but recorded in English. In the latter cases, I typically took time to make a mental note of the phase right after it was uttered, and could remember at that point my on-the-fly translation of it, but not the exact Portuguese phrasing.

Direct speech based on paraphrases

Additionally, I occasionally introduce direct speech that is based on paraphrases recorded in my notes. I include such speech *without quotes*, italicizing it if it helps understanding:

> *How would you even do it?* asks Fabio. How would you translate "DIM?" What does "DIM" stand for anyway? Dimension? So, perhaps it would be "Dimensão." This would be so strange and verbose!

or

> Yes, the Java code should be in English, he says again, but the database tables should be in Portuguese.

Such direct speech always represents utterances that actually occurred, but may not match those utterances verbatim. In the example above, Fabio may have said "could" rather than

"would," or may have even said "How would this even work?" In such cases I intend to capture speaker's meaning and overall manner of speaking, but not make assertions about the exact words. I sometimes choose to drop quotes around utterances that *were* actually recorded verbatim, to blend them with the adjacent utterances, which are based on paraphrases. E.g., in the quote above I could put quotes around Fabio's question "How would you translate *DIM*?" but choose to include it without quotes since the adjacent utterances are paraphrases.

Representing the vernacular

There are substantial differences between spoken Brazilian Portuguese and the formal Portuguese as it is often written in Brazil. There is a strong tradition in Brazil of translating speech into formal register when representing it in writing. I do not follow this tradition and instead attempt to represent the spoken vernacular as is. At the same time, I try to keep the text readable. I use the following convention as a compromise:

I always preserve speaker's actual grammar and vocabulary. For instance, I do not correct "a gente trabalha" to "nós trabalhamos," ou "pegava" to "pegaria."

I represent common verbal abbreviations. I write *tar* (*tá*, *tô*) instead of *estar* (*está*, *estou*), *cê* instead of *você*, *pra* instead of *para*, *vamo* for *vamos*, *cabou* instead of *acabou.* Such abbreviated forms are not used in standard written Portuguese, and the Brazilian convention is to formalize them when representing direct speech. At the same time, most Cariocas seem aware of the fact that they use such abbreviated forms and recognize them when they see them written. Needless to say, I only use such abbreviated

forms when the speaker does.

I do not represent other aspects of Brazilian or Carioca pronunciation. For instance, I do not represent the pronunciation of "s" as [ʃ], "r" as [x], or the insertion of extra vowels between consonants. I also do not attempt to represent the nearly ubiquitous substitution of *-ô* for *-ou* in the past tense of verbs ("ele falô" instead of "ele falou") except in "tô," or the dropping of final "r" in verbs ("quero falá" vs. "quero falar"). Since such features are common to nearly all Carioca speakers, the reader can fill them in. Representing such features in quotes, on the other hand, would make the quotes a lot harder to read.

Translation into English

Identifying translated quotes

Long quotations translated from Portuguese are identified by footnotes to the original Portuguese quotes (given in appendix C). Long quotes from interviews in English are similarly identified as such with footnotes. (I only use such footnotes once per speaker in each chapter.) Shorter quotes should be assumed to be translated from Portuguese, unless otherwise indicated. Quotes from Rodrigo Miranda represent an important exception to the rule: our conversations switched between English and Portuguese so often that it is not always possible for me to be sure in what language a particular phrase was said.

Choice of translation

When translating vernacular Portuguese into English, I try to pick vocabulary from the appropriate register, but erring on the side of the more formal English. (For example, I translate "cara" as "guy" rather than "dude.") I do not attempt to represent in English the *grammatical* informality of Portuguese vernacular. I include the original Portuguese in brackets when the translation is not obvious (e.g., "unlock the value [*valorizar*]") or where the original choice of word may be relevant.

Representing English used by Portuguese speakers

When Portuguese quotes include English words, I include those words in English in the quote and italicize them. When translating the quote into English, I again italicize the English words. I add "[says in English]" or "[English]" if this English phase is not common in Portuguese, for example: "Since Lua is *late late late binding* [says in English], it is very versatile to..." I do not do this, however, for English words that can be considered already a part of Portuguese vocabulary: "the *download* would be automatic."

In some cases, it becomes hard to translate without changing the relevant English word. For example, "era bem *nerd*" could be translated either as "was quite *nerdy*" (replacing "nerd" with "nerdy") or "was quite a *nerd*" (turning "nerd" into a noun, though it is used as an adjective in Portuguese). I either use the second option or change the word and show the original in brackets ("was quite nerdy [*era bem nerd*]").

When Portuguese speakers use English words in the context of a Portuguese sentence, they nearly always adjust their pronunciation to the rules of Portuguese

phonology, which makes them quite different from the original English: “late binding” would be pronounced as [ˈleɪʧɪ ˈbaɪŋʤɪŋ] (“leichi bine-jing”), “download” as [daʊnˈloʊʤɪ] (“dawnloaji”) and “nerd” as [ˈnexʤɪ] (“nehji”). The way such words are pronounced varies from one speaker to another and between different contexts: the pronunciation can be more Portuguese or closer to English.

I avoid representing pronunciation in most such cases, since it would make the text hard to read and may create an impression that my interviewees are less competent in their use of such words than they actually are. Instead, I present them in their standard English spelling, which how all of my interviewees would *write* such words in the context of a Portuguese sentence. I make a parenthetical comment about the pronunciation in cases when it is particularly relevant:

> Mauricio: Since I was quite a *nerd,* I spent most of my time in the computer lab.
> Yuri: What’s “quite a *nerd*”?
> Mauricio: *Geek* [says as in English].

This may then be followed by a comment such as:

> Mauricio pronounces “nerd” as it is usually said in Rio: “nehji.” When I ask him what he means by that, hoping that he would elaborate on the meaning of “nehji,” he replies with another English word, one that is somewhat less common in Rio and which he this time carefully pronounces in nearly proper English: “Geek.”

Appendix C: Original Interview Quotes

Jaime, October 2005

A maior biblioteca mundial

> Jaime: A Internet, hoje em dia a maior biblioteca mundial é a Internet, hoje ela dispõe de todos os recursos imaginários de dúvidas, eu já resolvi muitas situações.

Jorge, October 2005

Migrou pra software

> Jorge: Então, quando abriu a Reserva de Mercado [unclear], eles [companies] não tinha como competir com estrangeiros. [...] Então a gente, engenheiros eletrônicos, nos percebemos que a gente tava com nosso horizonte fechando. A eletrônica não tinha muito mais pra onde avançar no Brasil. Então, nós tínhamos vários centros de microeletrônica no Brasil, e hoje em dia só tem um —o único cara que foi persistente, eles continuaram, tá. Eles são um tipo de reserva intelectual nessa área. [...] Só no Rio Grande do Sul. [...] Eles por exemplo tem um chip Java. [...] Mas a gente aí migrou pra software.

Edmundo, October 2005

80% em Inglês

Edmundo: Eu uso Google... Por exemplo, eu tô com uma duvida de ler um arquivo binário. Eu esqueci, como é que eu faço? Ou eu perco tempo fazendo uma classezinha aqui dentro da [company name] pra ler o arquivo binário, ou dou um *search* "*java binary file*". Aí eu vejo mais ou menos como o pessoal fez o código, e faço o meu.

Yuri: E você sempre usa palavras ingleses?

Edmundo: Sempre. Palavra inglês. Ó... Perdão, perdão. 80% em inglês, 20% em português. E te falo por quê. Por que os 20% em português? Porque aqui no Brasil, nós temos uma das maiores comunidades de Java do mundo, que é a comunidade Sou Java, que fica em São Paulo. E 80% porque no mundo inteiro, independente de comunidade Java, chama JUG, Java User Group, que independente de comunidades, tem programador Java pelo mundo inteiro, então é muito mais fácil você encontrar uma palavra que você post em inglês do que em português.

Yuri: Então, às vezes você usa inglês, às vezes português?

Edmundo: 20% português, 80% inglês.

Yuri: Mas como você escolhe?

Edmundo: Boa pergunta. Como eu escolho isso? [Pause.] Quer saber mesmo? Quer saber mesmo? Quando eu tô cansado de escrever em inglês, aí eu boto em português. [Chuckle.] Tô cansado escrever inglês. [Chuckle.] Não existe nenhum padrão, "Ah, agora eu vou fazer em português". Não, não. Não existe esse padrão.

Yuri: Tá bom, então quando você tá cansado?

Edmundo: Quando tô cansado. [Laughs.] Quando eu não quero mais ler em inglês, não quero ler nada, não quero escrever inglês.

Yuri: Ler em inglês é mais difícil pra você?

Edmundo: [Pause.] Com certeza, porque não é a minha língua-mãe, entendeu? Não é a minha língua-mãe. É tipo você, mesmo que você morando cinco anos já nos Estados Unidos, três anos nos Estados Unidos. Cinco anos você morou nos Estados Unidos, né?

Yuri: Dez.

Edmundo: Dez anos, né? É dez anos, pô... Ainda assim, ler em russo é muito mais fácil do que ler em inglês.

Bruno, April 2007

Aprendi pelo código

Bruno: Aprendi pelo código.

Author: Vendo o código já escrito?

Bruno: Já escrito. Pelo Luiz Henrique, pelo Roberto, e por outras pessoas que começaram antes de mim. Provavelmente vi código do Rodrigo [pseudonym], provavelmente eu vi código de outras pessoas. [...] Como a Tecgraf no inicio era pequena as pessoas ficavam no mesmo laboratório. Então por exemplo, quando eu fiz doutorado aqui, eu sentava numa máquina, do meu lado tava o Luiz Henrique que trabalhava aqui na época... Ele tinha acabado o doutorado mas ainda ficava por aqui. Quando ele terminou o doutorado foi que eu comecei. O Rodrigo [pseudonym] tava terminando a graduação, sentava do meu outro lado. Entendeu? Então assim, você tomava contato. A minha tese por exemplo eu fiz toda usando C. Então quando eu resolvi migrar, tomar contato com Lua é porque eu tava fazendo um troço, escrevia, sei lá, duas paginas de código, eu olhava pro lado, o cara do meu lado fazia isso em, sei lá, um quarto de pagina. “Pô, peraí, como é que você fez isso... tão simples?” Então esse contato dentro de laboratório é que as pessoas aprendiam. Entendeu? Hoje em dia a gente cresceu muito. Nao existe mais um laboratório. São pequenas baias e cada baia tem seu projeto. Então se cê tá num projeto que não usa Lua, cê não vai ter essa facilidade de ter o contato e aprender desse jeito. Mas naquela época era quase que natural.

Carlos, April 2007

Percebi que existia um movimento

> Carlos: Então, eu sabia de Lua, mas... Sabia de alguns sucessos de Lua, mas também... Sabe como... sabe aqui e ali. Como, o professor ganhou um prêmio... Mas quando eu assumi aqui essa função de planejamento, olhando o potencial aqui da universidade, o que ela tem de ..., percebi que existia um movimento bastante grande até utilizando a linguagem Lua.

Evangelização

> Carlos: A partir de Chico conhecia e tinha contato conversando com Rodrigo Miranda, que é uma pessoa que tem um papel importante também [...] no pensar de estratégia de evolução de Lua. [...] E ouvindo de Chico, que também tinha essa posição, eu tinha oportunidade pra levar isso pra administração da universidade, que... A universidade estava percebendo pouco a importância que a Lua tava ter no contexto internacional. E membros de administração da universidade concordaram com isso, né. E eu mencionei que deveria tentar ver quais eram maneiras de a gente reforçar essa associação—entre Lua e a PUC. [...] O que a gente chamou de "evangelização", você tem que ganhar o coração das pessoas, a mente das pessoas.

Em benefício de todos

> Carlos: A percepção de que se nós conseguirmos valorizar esta linguagem, ela vai servir em beneficio de todos, do Rio de Janeiro, e do Brasil. Porque sempre um país periférico tem... Sempre... "Ih, não, isso é uma coisa só de brasileiro." Você compra um produto no Brasil, se dar defeito, você diz: "Ah, é brasileiro." Cê compra um equipamento importado, um automóvel, você diz: "Ah, olha que maravilha." Quando dá defeito, a pessoa quase não fala nada. Mas se você compra um produto Brasileiro, você diz: "É porque é brasileiro". E isso vale pro software também, né. Apesar da nossa destacada posição na área de software bancário, na área financeira, e tal... Mas o Brasil é pouco agressivo nessa questão de *off-shore*, não?

Programadores de Lua

> E também o que eu tenho colocado, né, de talvez a necessidade de a gente fazer um esforço de buscar recursos, de trazer agências financiadoras aqui, dizer: “Vamos formar programadores de Lua.”

Parceiros de peso

> Carlos: E disso também começar a procurar grandes parceiros, né. Porque se Lua tem uma projeção no nível internacional, se ela tem uma serie de qualidades que a comunidade reconhece, por que não atrair parceiros de peso, grandes *players* internacionais, como Softnet, IBM, para conversarmos sobre esse assunto?

Uma proposta

> Carlos: ...onde uma das propostas que nós vamos colocar pra ele pra agenda de desenvolvimento seria a questão de tecnologia de informação, mas centrado em Lua, como diferencial brasileiro pra *off-shore*. Java pode ser na China, na Índia... A Índia tem a facilidade da língua, que aqui no Brasil... não temos talvez isso. Mas sem dúvida nenhuma Lua é um diferencial, onde você tem condições de formar uma massa crítica rapidamente, se nós tivermos uma articulação. E é isso que, nessa vinda do Secretário, nós vamos convidar os órgãos financiadores da área de ciência e tecnologia, o Banco Nacional do Desenvolvimento Econômico e Social, pra virem aqui, e vamos trazer empresas que já estão em processo de catequese... [laughs] pra [talvez?] falarem sobre isso e ter o seu interesse... Ou seja, você de uma certa maneira vai sublinhando e certificando que isto é um produto brasileiro, *open source*, e, ainda, dessa criatividade aqui no Rio de Janeiro.

Felipe, April 2007

Deixado cada vez mais de lado

> Felipe: Então, de cara a gente já saiu um pouco do foco de investir no InterJX, investir em integração e isso foi só aumentando, e, com o passar do tempo, a gente sempre foi, cada vez mais trabalhando com o desenvolvimento de aplicações web, né, com o desenvolvimento de soluções sob encomenda por os clientes, né? E isso fez InterJX uma coisa deixada cada vez mais de lado.

Vamo começar a faturar

> Felipe: Começou bem no começo, porque foi um contato de Eduardo. Um amigo dele que trabalhava na Petrobras tava precisando uma empresa que pudesse dar manutenção nesse contrato, na Intranet no caso, e aí perguntou se a gente estava disponível e tal. E a gente "Ó, lógico, vamo começar a faturar e entrar na Petrobras, que é uma grande empresa..." E a gente começou. Aí cobramos baratinho pra entrar e começar a gerar alguma receita.

Nem existiria

> Felipe: Acho que se a gente não tivesse pego aquele projeto acho que a Alta nem existiria hoje.

Fechou a caixão

> Felipe: E chegou um momento também que a gente consegui uma parceria com uma empresa grande, multinacional, que é "EIT." A gente consegui uma parceria com eles e eles são uma das principais empresas que vendem software de integração. Então a gente priorizou aprender a plataforma deles, e tentar gerar serviços em cima da plataforma deles. E aí acho que foi que fechou a caixão de InterJX. Nunca trabalhava mais com InterJX. Foi o momento quando a gente priorizou trabalhar com uma tecnologia que é líder de mercado. Que era a tecnologia de EIT. [...] Lá tem desenvolvedores, no Estados Unidos, Indianos, tudo mundo evoluindo o sistema. A gente aqui, nos não temos ninguém. [EIT] 'tava

anos luz a frente.

Cássio, April 2007

Uma brincadeira entre amigos

Cássio: Na verdade não foi um projeto, foi uma brincadeira entre amigos, digamos assim. Que... A gente usava Lua, na verdade, pra... O interesse na época era pegar dados de *forms* em HTML, né, porque passa por um CGI, né? Então, na época a intenção não era nem gerar as páginas efetivamente, nesse começo, lá no começo. Era eu submeter um, digamos assim, tem um *form* escrito em HTML normal e aí botar, este programa que a gente chamou na época de LuaForm pra ser o *script* CGI que vai responder... que ia receber, digamos assim, o nosso *request*. A aí o interesse maior era em conseguir pegar os parâmetros, que era uma coisa complexa na época, porque tinha que programar em C para extrair os parâmetros, né, que vinham no *request*, no formulário. Então, originalmente foi pra isso, pra basicamente você escrever... pegar o resultado do form e aí produzir...

Yuri: Pra depois passarem estes parâmetros pra alguma coisa escrita em C que geria a página mesma...

Cássio: É. E daí uma coisa puxa a outra, né? Dado que você já tenha os dados no ambiente Lua com os parâmetros bonitinho, porque que eu não faço um... eu podia simplesmente fazer um *print*, né? E era como se fosse a resposta pra aquele form. Então ele vai escrever uma página como se eu escrevesse no *print* embaixo. Então daí seria mais fácil ainda porque ao invés de eu escrever print abre colchete H1 fecha, era mais fácil ter uma outra coisa que gerasse essas *strings* pra mim. Aí provavelmente daí deu origem ao CGILua...

Yuri: Isso começou como brincadeira entre amigos em que sentido?

Cássio: Na época a gente estava aqui fazendo o mestrado, ou graduação, não me lembro exatamente agora, e estava começando a surgir, pelo menos aqui no Brasil, no meio acadêmico essa coisa de geração de páginas. As pessoas usavam um editor “careta” pra escrever uma página, não tinha muito essa coisa desse conteúdo dinâmico...

Yuri: Editor de que?

Cássio: De texto pra escrever um HTML simples, né? E justamente por as coisas serem estáticas quando você fazia um form, você precisava escrever um programa. Então: "Como é que a gente faz isso de uma maneira melhorzinha, né, pra pegar esses parâmetros?" Daí você podendo fazer print, já faz a resposta da pagina. Então as coisas acabaram, tipo assim, dando origem aí ao CGILua. Mas aí já foram outras pessoas que pegaram e transformaram num produto e até numa tese.

Yuri: O código começou com você e algumas outras pessoas que estavam fazendo...

Cássio: Aqui na PUC, no laboratório.

Yuri: Não tinha nada com Tecgraf?

Cássio: Era dentro do laboratório, dentro do ambiente do Tecgraf. Digamos assim, todos dentro do laboratório naquela época estavam fazendo mestrado, doutorado ou eram alunos. Então, dentro deste ambiente a gente produziu essa brincadeira. Na época se chamava LuaForm. Justamente porque o interesse era só em dar um tratamento de *forms* de HTML.

Yuri: A idéia era usar ele pra fazer algum tipo de site, ou só...

Cássio: Só pra aprender mesmo, só pra descobrir como as coisas funcionavam na época.

Antônio, May 2007

Os comentários nossos são em português

Yuri: Então, voltar pra parte mais antiga. Quando você começou a usar Lua você já sabia inglês?

Antônio: Sabia. Sim, sabia ler bem e falava... razoável. Sim.

Yuri: Como foi escrito...? Eu sei que a versão primeira do Lua até o código do SOL mesmo foi escrito em inglês, com comentários em inglês...

Antônio: Sim, até hoje a gente faz isso. Aqui no nosso grupo, também.

Mesmo não sendo Lua, a gente tenta... Não, desculpe, os comentários são em português, claro, né. Mas os nomes das variáveis, tudo a gente tenta fazer em inglês mas os comentários nossos são em português... Lua, não...

Uso muito mais globalizado

Yuri: Você disse que os comentários são em português, "claro." Mas eu tenho a impressão que isto não é óbvio. Não estou dizendo que não é comum, mas não é óbvio. Eu acho que, às vezes, as pessoas até usam comentários em inglês.

Antônio: Não, não. A gente—é em português. Até porque... No caso de Lua eu acho até perfeitamente compreensível que seja em inglês, ela nasceu pra voar, para outros... tem um uso muito mais globalizado do que as nossas aplicações. A gente, na verdade, nem incentiva e nem quer que as pessoas aqui tentem fazer os comentários em inglês, porque a maioria não é fluente em inglês, então ia ficar um inglês capenga... Nossos produtos não são... O código fonte não é feito para exportação, não é aberto. O que a gente faz não é aberto. O código que a gente faz para a Petrobras é propriedade deles e eles não querem que tenha que tambem... saber inglês pra ler o nosso código...Então, sob vários aspectos, nas aplicações, a gente não comenta em inglês. Os nomes das variáveis até são, pro código; você lê ele mais ou menos em inglês. Mas, os comentários...

Yuri: Mas, então, por que os nomes de variáveis são em inglês?

Antônio: Isto é uma questão que a gente vem debatendo, mas, a gente acha que fica... por exemplo, a sintaxe do código mesmo para você ler fica esquisito, né. A gente estranha um pouco misturar nesse aspecto, mas, os comentários, não. É, agora que você falou... para mim é óbvio mas, se você olhar em retrospecto, talvez não, né? Por que é natural comentar-se em português e o resto não, né?

Em outros contextos, em outros lugares

Yuri: Alguém achava isso estranho, não?

Antônio: Não, não. É... Assim... Acho que isso não é questionável, quer dizer, acho que todo mundo meio que entende que Lua... A idéia é que Lua fosse usada, realmente, em outros contextos, em outros lugares. Tivesse uma... abrangência, né... um alcance maior do que Brasil, o Rio e

a sede. Eu acho natural Lua ser assim.

Vou botar no LuaForge

Yuri: Mas, na época não era, né? Na época ela era um projeto...

Antônio: Sim, ninguém poderia pensar o estouro, o sucesso, que Lua ia ter. A aceitação para a indústria de games, etc. Mas, talvez eles, o Roberto e o Luis Henrique, tivessem essa idéia, eu não sei. Eu, por exemplo, esse filtro de spam que eu fiz, todo ele é em inglês. Comentário é em inglês. Porque, no meu caso, por exemplo, eu pensava ah, sei lá, um dia, eu vou botar no LuaForge, sei lá, e alguém pode querer baixar. Eu gosto dessa, da idéia do "open source," de "many eyes," etc... todo mundo tar olhando... Então, eu acho que quando você se propõe a fazer uma aplicação ao "open source" você deve falar a língua mais abrangente possível; mais, mais comum possível, talvez. Mais facilmente entendida, né? No caso das nossas aplicações não, nós não... no Tecgraf não, na verdade a gente não quer que isto aconteça.

Um software nacional

Yuri: Mas não é questão de que Lua é uma linguagem brasileira ou não sei o que...

Antônio: Tem um pouco disso, também. É porque eu conheço as pessoas que fizeram... Eu queria ajudar a promover de alguma forma. [...] Mas tem um pouco disso, sim, sem dúvida, de orgulho, de saber de onde veio, de conhecer, de promover um software nacional, isso tem. Sem dúvida.

Rodrigo, May 2007

Queria fazer uma plataforma web

Rodrigo: Eu queria fazer uma plataforma web pra Lua. Mas eu não tinha capacidade técnica de fazer. Então, eu tive que pensar assim: "Como é que eu vou estruturar as coisas pra conseguir juntar um monte de gente e aí fazer uma plataforma web?" A primeira tentativa foi [Nas Nuvens], quando [Nas Nuvens] apareceu, eu falei assim: "Olha, esse é um lugar

onde eu posso fazer uma plataforma web. Porque eles querem usar Lua." Já é bom, mas não funcionou...

Yuri: Porque não?

Rodrigo: Porque o nível dos desenvolvedores não era o nível de desenvolvedores de plataforma. São níveis diferentes. Desenvolvedores de aplicações numa plataforma tem algumas expectativas. E se você quer fazer a plataforma em si você tem que saber muito mais do sistema operacional, muito mais da parte de C e de integração com outras coisas pra fazer os conectores. Isso dá mais trabalho.

Duas moedas

Rodrigo: Eu percebi que numa lista de um projeto livre existem duas moedas de troca: crédito e respeito. E existe um bem palpável que é *source*. Então, você tem duas moedas abstratas e uma concreta; eu posso fornecer *source* pra uma comunidade e com isso ganhar respeito ou alguém pode me fornecer *source* e eu pagar em crédito ou... [...] Se eu dou crédito pra uma pessoa, ela tem respeito, o que eu estou tentando agora é ganhar respeito.

Meio patronizing

Rodrigo: Porque isso é uma coisa que eu já reparei na lista de Lua. Se alguem faz uma pergunta, ou se eu faço uma pergunta, a minha pergunta transparece com um grau de ignorância muito grande. Porque eu sou ignorante. Porque eu não sei uma quantidade enorme de coisas. Eu não sei como funciona o *Makefile*, eu não sei como funciona o sistema de *build*, não sei como [unclear]... Tem muita coisa de Linux que eu não sei. Então isso aparece na pergunta. E a resposta das pessoas tende a ser educada mas é... meio *patronizing*. Meio tipo, "Olha, ah... Então faz aí isso aqui. Vai lá descobre como faz isso." E quando eu dou uma resposta mais técnica, eu sinto que isso gera uma conexão com a pessoa do outro lado. A pessoa fala assim: "Ah, eu não estou falando com um gerente burro. Eu estou falando com alguém que, pelo menos, vai entender o que eu estou pensando." Então você sente na resposta uma vibração diferente, que eu imagino que seja o que a pessoa sente quando vê a minha resposta.

Yuri: No final você acha que isso é importante pro projeto?

Rodrigo: Sim, claro. Mas é importante pra mim, eu estou fazendo por mim porque eu quero aprender a mexer em Linux, Windows e OSX.

Yuri: Só pra saber ou pra saber como obter respeito nesse tipo de lista?

Rodrigo: Não, só pra eu saber fazer o Kepler. Eu estava numa situação muito delicada onde eu tinha uma plataforma que eu tinha bolado, a idéia da plataforma era minha, mas eu não sabia usar porque eu não conhecia todos...

Yuri: Não sabia fazer?

Rodrigo: Mesmo usar. Os detalhes internos eu não sabia. "Como funciona o LuaExpat?" Eu não sei. Eu nunca parei pra olhar. Eu falei: "Eu preciso de um LuaExpat." E aí apareceu. Ah, bom, essa parte já tem. "Agora eu preciso de um LuaZip." Apareceu. Agora preciso de um MD5. Apareceu. Então, eu já tenho todos os quadradinhos que eu preciso. Tá bom.

Silvio, May 2007

Sem se incomodar com a plataforma

Silvio: Isso aconteceu, de uns seis anos pra cá, que a gente começou a aumentar significamente a quantidade de projetos em Java. [...] Em 2001 a gente teve uma demanda do nosso cliente... De "Ah, a gente queria fazer um sistema com essas, essas, e essas características. E eu achei que era mais interessante usar Java do que... Porque você tem uma parte... É aquele ditadinho: "Pra quem tem martelo, tudo é prego." A gente tem que ter uma caixa de ferramentas e saber usar cada ferramenta na situação adequada, né? Lua é uma excelente ferramenta, mas não é a ferramenta mais adequada pra tudo. Ninguém espera que seja. E eu acho pro trabalho que a gente tinha em mãos, o ideal era uma mistura. A gente realmente fez isso, a gente usou Java pra construir o cliente, que a gente queria rodar pela intranet, etc, etc, no navegador. Na época não existia JavaWebStart, mas existia já o plugin pra Applet. Então a gente conseguia, por exemplo, rodar na intranet, uma aplicação com uma interface gráfica super-legal, sem se incomodar com a plataforma lá do cliente, do usuário. Então a gente optou por fazer o cliente e o servidor em Java, mas a parte que está distribuída nas máquinas, que precisa mais dessa coisa operacional, a gente implementou em Lua. A gente fez esse

corte. Só o que é que acontece é que a parte do servidor e do cliente demanda muito mais implementação do que a outra, por isso que hoje o projeto é prioritariamente Java.

Lua 3.2 e Lua 4

Silvio: Eu acho que a gente pulou o 4.0. Não tenho certeza. Se quiser até te mando isso depois, se isso for importante pra você.

Yuri: Tá bom. Eu estou perguntando porque eu já ouvi as pessoas falando que tinham algumas dificuldades no Lua 3.2 e Lua 4 e que algumas pessoas naquele momento desistiram.

Silvio: É verdade. Basicamente porque a API de comunicação entre C e Lua mudou drasticamente, né. Foi em 4, eu acho, que foi introduzida a máquina de pilha na comunicação, no bind de Lua com C. E aí quem tinha muito código C chamando Lua, teve que fazer muitas mudanças. E até teve gente que realmente reclamou, "A, pô, chato, mudar," não sei o que. Mas eu não tenho essa visão, acho que faz parte, acho que a gente tem que andar pra frente. Lua tem que evoluir, a gente não vai parar, estagnar o Lua ou ficar numa interface que a gente entende que é pior, só porque tem gente que tá usando e tá com preguiça de mudar as aplicações. Não faz sentido. Estagnar por estagnar... Quem não quiser evoluir congela lá na versão 3.2 e fica com ela o resto da vida, que ela vai funcionar bem, obrigado.

O outro motivo

Silvio: Eu continuo usando Lua nos projetos TecGraf, então tem um aspecto prático aí de que conforme as coisas acontecem em Lua eu tenho interesse. [...] Então eu quero tar a par de tudo o que tá acontecendo porque pode me afetar no projeto. O outro motivo é porque eu adoro isso, eu sou um entusiasta da linguagem, adoro Lua, acho um barato e gosto de ver as discussões, aí é mais por gosto mesmo. Então tem esses dois aspectos.

Yuri: Barato em que sentido?

Silvio: Uma coisa muito legal, "cool." Em inglês "cool."

Yuri: [...] Você pode me falar um pouco mais sobre essa parte de adorar Lua, o que significa?

Silvio: Sim, porque eu acompanhei Lua desde o início do desenvolvimento, né. O Roberto foi o meu orientador tanto mestrado... Quer diser graduação, mestrado e doutorado. Eu tenho um vínculo de amizade com o Roberto, a gente vira e mexe, vai almoçar juntos, ele me conta o que tá acontecendo, não sei o que. Então eu vivo muito esse mundo da linguagem e eu gosto sim, de tar a par, de ver o que está acontecendo, conversar com o Roberto, trocar idéia. Então assim. É uma coisa que eu gosto, eu gosto mesmo, por causa das pessoas envolvidas. É um trabalho que eu acho fantástico.

Por puro preconceito

Silvio: Eu acho muito legal o Roberto conseguir... Tem o aspecto aí... É assim. Eu já vi muito preconceito contra a linguagem. É impressionante como isso acontece. Assim. A gente tem um cliente. Eu já vi cliente nosso, uma pessoa dentro de uma empresa dizer o seguinte: "A não, eu tô aqui na dúvida entre usar Lua ou usar essa ferramenta da Microsoft. Mas acho que vou voltar da Microsoft. Porque se usar Lua e der errado, meu chefe vai dizer que eu sou maluco. Se eu usar da Microsoft e der errado, não tem problema porque isso acontece todo dia." É *ridículo* alguém pensar assim. Eu acho assim... Eu sinto *vergonha* de ver alguém falando um negócio desse tando aqui, quer dizer... Ao invés de... podendo usar... Ao invés de incentivar um trabalho que é totalmente desenvolvido no país, o cara não incentivar, deliberadamente por puro preconceito... Eu acho uma coisa totalmente absurda. Então assim... Essas coisas me motivam muito a acompanhar o Lua, tentar usar. Porque eu vejo que a lista, a maior parte das pessoas na lista não são do Brasil, a maior parte são estrangeiros. Eu vi que Lua teve que fazer sucesso lá fora, pra poder ser bem aceito aqui dentro. [Pause.] Ou seja, é um projeto que eu acho fantástico, que eu gosto muito. Vejo a luta que é do Roberto de fazer Lua funcionar, vejo o trabalho que ele tem. Lua reflete, no fundo, essa genialidade dele. Que acho que Roberto é um cara muito bom... [Pause.] Então, por esses aspectos todos, eu gosto muito da linguagem e tenho prazer em acompanhar o que tá acontecendo. Por isso que eu não saio da lista, continuo a ler. Por mais que eu não fale nada, eu continuo na lista vendo o que tá acontecendo.

Pedro, May 2007

Curso de Inglês

Pedro: Porque o que eu fiz, eu não fiz [minha graduação] em quatro anos. Eu fiz em quatro anos e meio, porque eu decidi reduzir a carga para estudar inglês. Então reduzi a carga para estudar inglês, pra estudar inglês um ano e meio, no curso comercial de inglês. *If you want we can talk.* [Laughs.] O meu inglês ainda tá... ainda tá... Eu ouço melhor do que falo. [...]

Yuri: E como você aprendeu?
Pedro: Curso de inglês. Curso semi-intensivo de um ano e oito meses, que tô fazendo. Continuo lá fazendo. Então reduzi minha carga na faculdade no sexto período. Há oito períodos. Reduzi no sexto, planejando ficar um período a mais, um semestre a mais. Porque no sexto período parei: se eu não estudar inglês, não vou poder fazer mestrado. No [...] curso pra poder fazer mestrado. Porque com mestrados aqui [esperam] que a pessoa já seja fluente... fluente não, mais capacitado em leitura. [...] Deve saber ler e escrever inglês. [...]

Yuri: E foi bom?
Pedro: Pra mim foi bom, funcionou.
Yuri: Antes desse curso você não sabia...
Pedro: ...nada de inglês.
Yuri: Até não sabia ler?
Pedro: Nada. Nem ler. A gente tem inglês aqui na escola fundamental, só que é muito pobre. É inglês que é baseado em tradução. E aí você não aprende nunca. Eu sabia muito pouco. De leitura muito pouco. Posso dizer que hoje eu consigo ler qualquer artigo em inglês. [...]

Pedro: Segundo semestre de 2005, né. Foi mais ou menos mesma época que eu comecei mexer com Rails, estudar inglês, e aí já tava melhorando meu inglês. O curso é muito bom, o curso que eu fiz. Em seis meses eu já tava lendo bem. Não tava falando também, mas tava lendo bem.

Jason, June 2007

TK 85

Jason: Eu mexo com computação desde os oito, oito anos de idade. Eu comecei trabalhando com pequenos computadores de lógica Sinclair que no Brasil eram comercializados com o nome de TK 85, TK 82. Eram computadores bem pequenininhos e meu pai comprou um desses pra mim e eu desenvolvia joguinhos neles e os meus primos, da mesma idade que eu, jogavam estes joguinhos e sugeriam mudanças que eu ia lá e implementava. Eu aprendi a linguagem BASIC no manual do próprio computador, né, que vinha com a linguagem BASIC nativa, e eu aprendi lá, mais ou menos sozinho mesmo, e fiquei muito interessado. Mas não persegui muito mais isso não porque, na verdade, eu queria ser escritor, escrever ficção. Eu sempre tive interesses muito diversificados em várias áreas, né? Só anos depois, quando... Nos anos 80 a educação teve uma série de problemas com governo naquela época, então houve muitas greves, tinha um período de alguns anos. Tinha repetidas greves que deixavam os anos letivos com lacunas às vezes de quatro meses, assim. Então a gente começava a estudar e ficava quatro meses parados e só depois ia concluir. Numa dessas o meu pai achou importante me colocar em algum tipo de curso para eu não perder um ano sem estudar. E aí me colocou num curso de informática. [...] Aí no curso de informática eu fui apresentado a outras tecnologias, bancos de dados, essas coisas. Aí finalmente eu me interessei pela carreira. Pra ser mais preciso, eu acho que foi 1987. [...]

Meu pai

Jason: Meu pai era engenheiro eletrônico, mas ele nunca trabalhou como engenheiro eletrônico. Ele é formado em Engenharia Eletrônica, mas ele trabalha dando aula de eletrônica. Quando eu era criança ele trabalhava consertando equipamentos eletrônicos, ele consertava televisão, rádio, essas coisas assim. E minha mãe era professora. Na verdade meu pai não tinha um emprego fixo quando eu era criança, ele trabalhava fazendo trabalhos esporádicos. E a minha mãe sempre foi professora e aí ela ganhava pouco... [...] A gente sempre teve pouco dinheiro, eu não sei como é que foi a sorte de ter podido comprar esse pequeno computadorzinho.

Yuri: E como?

Jason: Não sei, ele [o pai] deve ter feito algum trabalho, lá que deu algum dinheiro, houve mais na época e acabou conseguindo.

Yuri: Ele comprou esse computador pra você? Ele mesmo não usava?

Jason: Não, ele não usava. Era um computador de recursos muito limitados né?

Yuri: Mas foi você que pediu ou...?

Jason: Não, ele deve ter achado interessante e comprou. Ele é muito interessado em informática até hoje, mas ele realmente não é da área.

Uma época complicada

Jason: Era uma época complicada, eu me lembro, porque obter informação era difícil, você descobrir como se fazia uma coisa, você ficar fazendo tentativa e erro. Os livros eram caros e difíceis de encontrar, você precisava encomendar livros importados e você ia nas livrarias, pedia o catálogo, o cara te mostrava o livro "Então, existe um livro sobre isso?" "Existe" "Então, traz pra mim" Aí demorava meses até chegar e você ir lá e comprar e eram caros às pampas. Então, a solução, normalmente, era conseguir os programas de alguma maneira, alguém que tinha, copiava e tal e aí você ia lá e tentava, testando ele e descobria como é que ele funcionava. E aí usava os recursos dele, aí de repente descobria alguém que já tinha feito uma coisa mais avançada do que aquilo. "Pô, como você fez isso?" Aí o cara explicava e ia lá e aplicava também no seu programa.

Em vez de ir no shopping

Jason: Eu tinha alguns amigos que eram da mesma... se formaram com mais ou menos mesmo perfil do que eu, no curso de informática. O curso se chamava "Datacenter," lá em Nova Iguaçu. O curso em que a gente tinha estudado, o panorama era interessante tecnologicamente porque era um grupo de jovens que tinham...

Yuri: Mais ou menos da mesma idade?

Jason: Mais ou menos da mesma idade. Eu devia ter 17 anos, 16, 17 anos,[372] que ficavam *hanging around*, em vez de ir no shopping, no curso de informática. E aí os professores do curso de informática eram pessoas experientes, profissionais experientes, sabiam muita coisa, eram uns caras bons e aí a gente ia lá e ficava pegando macetes e dicas com eles. O pessoal que programava coisas de baixo nível, um cara sabia assembler outro sabia C++, o outro sabia C. C++ acho que naquela época era muito incipiente, não era muito conhecido ainda. E aí a gente tinha saído do BASIC.

Num lugar lá bem longe

Jason: A idéia da gente era construir aplicativos, nós queríamos fazer aplicativos porque era uma era em que os aplicativos eram poucos. Eram pocas coisas. Então, como nós entendíamos um pouco de programação nós achávamos que a gente tivesse condições de fazer um aplicativo daqueles e ficar ricos e famosos.

Jason: E a emoção ainda era maior ao ver que a gente conseguia fazer coisas que eram boas. Eu tinha um amigo, que chamava Rogério, na época, que ele... cada um tinha um perfil assim, e ele era o cara que era "fissurado" por editores de texto. Ele queria fazer um editor de texto porque estava frustrado com o editor de texto que existia naquela época chamado WordStar. E aí o WordStar, ele tinha uns defeitos que eram óbvios, que eram coisas, é... de decisões de projeto que tiveram ao fazer o software que não era um, é... que estavam se tornando inaceitáveis naquele mundo ali alguns anos depois, né? Era um software grande, que já tava no mercado, não ia poder sair trocando. Aí ele começou a construir um editor de texto que começou a ter funcionalidades melhores do que a do WordStar, um garoto lá de 16 anos, enfurnado num lugar lá bem longe. E isso era legal, esse prazer era...

Yuri: Longe onde?

Jason: Nova Iguaçu, longe com relação... até longe com relação a metrópole mais próxima, que era o Rio de Janeiro e longe com relação ao lugar que se fazia software comercial, que é lá nos Estados Unidos, lá no *Silicon Valley* e tal. Então, numa cidade periférica de um país de Terceiro Mundo, o cara foi lá e conseguia fazer um software, que era, que comparando o software comercial, você considerava "Esse software é

372 In a later email exchange Jason says that and his friends were actually 14 to 15 at the time. The number 17 is an incorrect on-the-fly calculation based on the year in which he remembers those things occurring (1987-1989) and his year of birth.

bom.” Esse potencial motivava muito a gente de estudar, de aprender as coisas.

Interceptar no relógio a parada

Jason: Esse grupo de pessoas trocava figurinhas né? A gente dizia “Mas você conseguiu fazer isso como?” “Ah, eu descobri que na interrupção tal do DOS [pronounces “De-Oh-Ese”] você pode botar um negocinho e o cursor passa a notificar cada vez que tá... você pode interceptar no relógio a parada e você consegue pegar as teclas do cara e aí você consegue passar um software por cima do outro e tal.” Umas decisões legais.

Tinha ficado rico

Jason: Eu lembro que na época eu era um cara que gostava de fazer gráficos, eu queria uma coisa que pudesse permitir que você rodasse vários programas ao mesmo tempo em janelas, um por cima do outro, assim. E aí pô, eu queria fazer isso de forma gráfica, na tela do DOS, mas era muito lento, muito ruim, eu queria ficar tentando soluções melhores, colocar buffers de vídeo mais espertos, copiar os dados mais rapidamente. Então, cheguei à conclusão que pra fazer isso eu ia ser obrigado a usar o disco e aí ia ficar muito lento e considerei inviável, desisti disso. Aí comecei a perseguir outras coisas. Fiquei muito indignado quando depois surgiu o Windows, alguns anos depois surgiu, justamente, usando o disco que era a idéia que eu tinha descartado como inviável, aí eu pensei “Sacanagem, se eu tivesse perseguido isso, tinha ficado rico.” [Laughs.] Ou não, né? [Long pause.]

Bem thorough

Jason: A informação era difícil de obter, a gente tinha que mesmo explorar os softwares, por isso era muito lento, né, o desenvolvimento dos nossos horizontes né; a gente... Em compensação ele era bem *thorough*, bem minucioso, a gente conseguia coisas que às vezes espantavam os professores. “Nossa, mas você conseguiu isso?” “É, tive que esquadrinhar todas as interrupções ali e descobri que essa faz isso, essa faz assim; tive que dar uma solução pra poder contornar o negócio que eu não conseguia fazer”. E aí acontecia...

Jason: Eu estudei informática no curso lá e terminei o 1º, terminei o 1º grau em 89 eu acho. Aí eu já programava em BASIC, dBase, Clipper, Pascal, um monte de linguagens. Aí eu resolvi fazer o 2º grau técnico de Informática, né? O 2º grau foi *peace of cake,* assim. Passei (...) completo, porque eu já entendia de tudo que estavam dando ali, por causa do meu contato com as coisas. O meu 2º grau era dividido em três anos: o primeiro ano era básico, estudava todas as matérias, biologia, física, química e só uma introdução à informática; a partir do segundo ano é que começavam as matérias técnicas mesmo, programação, algoritmos, essas coisas todas. E aí no primeiro ano eu identifiquei uma necessidade de automação, de desenvolvimento de sistema pra escola, e aí comecei a desenvolver o software para controle de alunos, para o curso deles né? Isso até era uma coisa engraçada porque eles tinham um aluno do primeiro ano, que nem era técnico, desenvolvendo um sistema bastante avançado já né; o que acabou me dando muito acesso às pessoas, aos professores, aos laboratórios, eu tinha acesso aos laboratórios pra fazer, e me rendeu até um apelido na época, que é o apelido que eu uso profissionalmente até hoje, que me chamam de "Jason," porque acho que naquela época tinha aquele filme "Sexta-feira 13," o Jason com uma máscara e tal e ele não morria de jeito nenhum, você dava tiro nele e aí tinha... e eu era uma pessoa extremamente obsessiva com programação, então, eu ia lá e programava, passava os dias... eu estudava pela manhã nesse ano, acho que foi 89 ou 90, acho que 1989, eu estudava pela manhã, de 7hs da manhã ao meio dia e durante a tarde o laboratório de informática tava liberado pra mim, porque ele tinha acesso restrito, nem todo mundo podia usar os computadores, mas eu tinha acesso porque eu fazia um sistema pra escola. Aí, às vezes, eu emendava e começava a usar os computadores meio-dia e pouco, assim que abria o laboratório, e continuava usando até dez horas da noite, quando me chutavam de lá pra fora. [...] Eu lembro que uma vez eu fiz uma prova de inglês pela manhã, eu também era bom de inglês, então fazia a prova bem rápido, eu fiz uma prova de inglês e saí oito horas da manhã da prova e aí os laboratórios só abriam uma hora da tarde. E aí eu: "Pô, como é que eu vou fazer, pra fazer hora até a hora de começar a programar né?" Aí fui como quem não quer nada, por desencargo de consciência, fui lá no laboratório e por acaso tava aberto, estavam dando manutenção nos computadores, eu "Legal, então eu vou entrar" Entrei e fiquei lá sentado programando e as turmas entravam e saiam, entravam e saiam, e eu continuava lá fiquei lá de oito da manhã até às dez da noite sem levantar da cadeira. É, como 2º grau é uma coisa que a fofoca é uma coisa natural, no dia seguinte a escola inteira já sabia do garoto que tinha ficado de oito horas da manhã

às dez da noite no computador. Então, me passaram a me chamar, ao longo da semana, de vários apelidos: “zumbi,” “vampiro,” “morto vivo,” “sem funções vitais.” No final o que pegou foi o “Jason”. Aí todo mundo: “Ah, o Jason que não morre, que tá ali no computador, como sempre, lá.”

Subestimei as cadeiras

Jason: Só fiz seis meses dessa faculdade lá. [...] Naquela época, eu era um cara extremamente técnico e muito pouco... eu era extremamente “um escovador de bit” que a gente chama, né? Um cara de muito baixo nível. [...] E aí eu subestimava conhecimentos como análise de alto nível. Até bancos de dados mesmo eu achava uma coisa muito trivial, “Pô, é um negócio que você pega o dado, bota lá e depois pega lá e tira, e depois bota de novo e tira, bota, tira. Não tem nada de especial nisso.” Então eu subestimei um pouco as cadeiras que eram disso, eu meio que analisei o currículo sob a ótica das tecnologias. Pensei assim: “Pô, não vão me ensinar tecnologia nova nenhuma nesse lugar, então não vou ficar aqui não.” E aí eu fui pra casa estudar mais coisa, né. E aí passei a fazer meio que um mosaico de diversas coisa, estudei essa coisa gráfica, estudei coisas de linguagem de programação, aprendi C++ nessa época, entre 91 e 94, e outras coisas. [...] Agora sim tinham livros decentes, né. Já era 91, 92, eu já podia entrar numa livraria, eles inauguraram, até aqui no Centro, era o *playground* da gente, Livraria Ciência Moderna aqui no edifício Avenida Central...

Um mercado com uma cara diferente

Jason: Na Petrobras foi quando eu meio que comecei a sentir a crise do meu super tecnicismo. Porque eu chegava lá e descobria que o mundo tinha mudado um pouco, as coisas eram mais fáceis de fazer no mundo do 1994, 1995, meio dos anos 90. A tecnologia já estava um pouco facilitada, o acesso a informação já tava facilitado e eu tinha uma bagagem técnica de “escovação de bit” muito grande mas ela já não era tão valiosa como era cinco ou seis anos antes, no final do anos 80. Então, me vi num mercado que precisava de aplicações de bancos de dados, basicamente, sistemas de informação, e eu não era uma pessoa com uma bagagem teórica grande sobre isso, né? [...] Eu vinha de um mundo de tecnologia de software básico, de coisas de baixo nível e manipulação de escovação de bit, e, de repente, eu encontrei o pessoal que vinha de sistemas lá da antiga de informação, de COBOL, pessoal de bancos de dados; e a gente meio que se encontrou no afunilamento do mercado, né.

Na verdade não foi exatamente um afunilamento, o mercado foi expandido. [...] No começo dos anos 90, lentamente, os primeiros frutos do fim da reserva de mercado começaram a nascer e a informática abriu, aflorou, as empresas surgiram e o mercado foi crescendo, né? E em 94 ele já estava com outro formato, já era um mercado com uma cara diferente. Essa época tinha uma demanda de sistemas muito maior, então, não foi exatamente um afunilamento, foi uma explosão.

Fiquei a fim de estudar

Jason: De repente, o mercado de sistemas de informação era grande e dava pra trabalhar, e dava pra viver disso bem e tinha um profissional disponível pra trabalhar com isso, isso meio que transformou o cenário, né. Eu cheguei lá, migrando, eu encontrei os caras que estavam vindo do COBOL e tal, né. E aí foi assim: "Pô." Eu tinha uma desvantagem com relação aos caras porque eu não tinha bagagem teórica de modelagem de dados, de análise de requisitos, era uma coisa que nem se falava muito naquela época, né? Então, pô, fiquei a fim de estudar e tal, fui estudar na universidade Estácio de Sá, aqui no Centro.

Vende ele pra mim

Jason: Então, ela [Petrobras] tem todo aquele ciclo de concursos públicos pra pegar funcionários e tal, aí tem os funcionários da Petrobras; mas quando ela precisa de uma coisa feita, ela contrata o externo, uma pessoa externa, que fica lá trabalhando com um contrato temporário; só que a lei não permite que você faça isso muito tempo, as leis trabalhistas do Brasil são pesadíssimas. Então, o que ela faz? Ela contrata uma empresa externa, terceirizada e obriga essa empresa: "Contrata esse cara aqui e vende ele pra mim." Aí a empresa contrata o cara, ele faz o contrato, depois ele é obrigado a abrir licitação, né? Abre a licitação, aí uma outra empresa ganha e "Demite essa cara aí. Você—contrata esse cara aqui." Aí o cara é contratado, participa de umas três empresas mas, na verdade, ele trabalha pra Petrobras. Eles queriam a mim e abriam concorrência pras empresas me contratarem. [Laugh.] Eu e assim milhares de pessoas que trabalhavam na Petrobras, pra agilizar os processos, fazer as coisas funcionarem. Porque se dependesse de abrir concurso pra entrar funcionário, aí demora, um ano, dois ano aí têm as falcatruas, o que a gente chama de "peixadas" né, que são os políticos que indicam quem é que vai entrar ou não, fraudam os concursos, é uma série de corrupções e tal.

Naturalmente averso a burocracia

Jason: Como profissional de informática eu já sou *naturalmente* averso à burocracia. Quando ela atinge a mim eu fico possesso. Eu fiquei zangado e saí de lá, resolvi ir embora. E aí voltei a trabalhar por conta própria.

Negocio muito bom, feito no Brasil

Jason: Não sei se eles estão aqui no Rio, não sei se ainda tão, mas originalmente foi um pessoal do Rio de Janeiro. E aí esse pessoal do WABA... WABA é famoso, eu descobri que eles são usados por aí fora, né. É uma máquina virtual pequena que funciona em Palm e outros dispositivos móveis. O WABA é free, mas acho que o SuperWABA não. E aí eu peguei o WABA olhei e falei assim "Pô, que legal, né." E por ter pego um trabalho de um cara do Rio de Janeiro isso explodiu na minha mente. E eu: "Caramba, tem o pessoal da PUC!" Eu já tinha ouvido falar do Lua antes, né? Quando eu tava pesquisando na época dos jogos, eu tava pesquisando alguma coisa e descobri que tinha um jogo que usava linguagem Lua e descobri que linguagem Lua era feita no Rio de Janeiro, foi feita pela PUC-Rio. Eu falei assim "Pô, pra que que eu vou usar um negócio super sofisticado como o Java se tem uma parada que é o Lua que é feito pra núcleo de mecanismo de jogo e deve ser super eficiente." A parada... O Java é um bocado mais versátil, né, sofisticado. Então ele é mais lento, tem um monte de outros requisitos. Assim, pô: o Lua é super enxuto, deve ser perfeito pras minhas necessidades. E ainda é um produto nacional, é um negocio que... Assim: não basta... não é só por ser nacional, é um produto bom. Negocio muito bom, feito no Brasil. Então, pô, quero pelo menos ver isso e entender isso melhor."

Ver que eles são gente

Jason: Olha eu acho até que eu me interessaria de ver que eles são gente, né? São pessoas que eu passei a admirar pra caramba, de ver o trabalho maravilhoso que os caras fizeram. Mas fico... é aquela história de você encontrar lá um músico maravilhoso, que faz aquelas músicas maravilhosas, você vai lá encontrar ele pessoalmente. Porra... [Laughs.] Ele pode tocar pra você, aí é legal, sabe. Mas pô, vai lá conversar com ele e tal é diferente, né. Trocar uma idéia sobre música e tal, você tem que entender de música tanto quanto ele pra você ir lá e poder aproveitar aquilo.

Vou ficar amarradão

Jason: Vou chegar lá e vou ficar amarradão querendo ver o que eles estão fazendo e se funciona, algum tipo de ajuda que eu possa dar e tal. Então, quando eu descobri que os caras eram da PUC, que eu comecei a usar o negocio e os caras eram da PUC, eu "Pô... É isso aí. Acho que eu vou lá, ver qual é." Mas eu não ia saber onde encontrar os caras. Acho que se eu visse algum deles oferecendo um curso, eu ia fazer imediatamente, correndo, mas eu não ia lá bater na porta pra descobrir onde é que está o cara.

Jason, July 2007

Me tornar pesquisador

Jason: O fato de eu estar trabalhando com Lua agora, e vendo todo esse potencial também lá, faz com que eu queira me enfiar um pouco nessa interseção que eu acho legal pra caramba. Agora, é uma motivação fraca, quando você olha pelo lado econômico, né? Então, quer dizer, nem só de economia se vive, mas modernamente falando, é um fator que não pode ser ignorado. Então, eu tenho muito interesse de me tornar pesquisador, e acho que se tenho ainda alguma vocação que eu ainda não persegui é essa. E acho que estar participando desse projeto pode contar ponto, de alguma maneira, para isso. Nem que seja só pelo networking, só por estar conhecendo as pessoas e estar contribuindo, adentrando uma comunidade que eu sempre quis adentrar e agora apareceu uma porta.

Está lá e vive

Jason: Com as pessoas da PUC, quer dizer, como eu conheci o Rodrigo, por exemplo, que é uma pessoa que me mostra que é viável trabalhar assim. Porque era uma dúvida que sempre me acometeu, sabe. Eu pensando: "Pô, beleza, eu sempre quis fazer isso, mas como é que eu vou ganhar a vida?" Se eu ficar fazendo essas coisas altamente experimentais e avançadas que não têm, não são mainstream comercial? Como eu vou viver fora do Java e do .Net?"

Yuri: E como você vai viver? O que ele te mostrou?

Jason: Ele mostrou... Bom, ele está lá e vive, né? [Laughs.]

Michel, July 2007

Pra manter interoperabilidade

Michel: Porque o Java, o Java é a linguagem que praticamente todos os sistemas de TV, sistemas de TV digital, eles oferecem uma API Java. Por ser uma linguagem a princípio independente de plataforma, que favorece interoperabilidade. Então Java tem sido uma API procedural utilizada nos middlewares de TV digital. Então, o Brasil pra manter essa compatibilidade, optou por ter também no seu middleware o API Java. E essa experiência em ... quais seriam as melhores APIs, como é que seria essa... a integração disso num set-top, isso o grupo de pesquisa da UFPB, a Universidade Federal da Paraíba, eles já tem, tão há muito tempo, já há mais de cinco anos, seis anos já bem focados nesse assunto. Então na época em... que foi mais ou menos em 2003, 2004, quando o governo brasileiro decidiu investir num projeto pra definir um sistema brasileiro e convocou as universidades e as... Então nessa hora os grupos de pesquisa daqui da PUC e o grupo de pesquisa da UFPB, eles trabalharam juntos nessa questão do middleware, um mais voltado pra parte procedural e o outro mais voltado pra declarativa.

Usado por essa indústria de games

Michel: Bom, Lua entra primeiro pela a facilidade que a gente tinha de trabalhar com a comunidade de Lua, que também é uma linguagem desenvolvida aqui na universidade. Segundo, porque a gente começou a ver que a gente precisava pra terminal, pra set-top box, de algo que fosse bem leve, que fosse eficiente, que não ocupasse muita memória. E a gente começou a perceber nas comparações feitas por terceiros, sites que fazem comparações de linguagens script, que Lua tem grande aceitação por essas características, por ser um interpretador muito leve, muito eficiente. Um ponto também que chama muito pra... TV digital o pessoal logo começa a pensar em entretenimento, aplicações, jogos, essas coisas. São aplicativos interessantes pra você começar a pensar nesse, rodar nesse receptor, Lua vem sendo muito usado por essa indústria de games, então de novo tinha tudo a ver. Então foi por essa soma de características.

O ganho de não botar Lua

> Michel: O Java ele entra, ele tem o lado mais pesado da máquina virtual. Mas se você não põe o Java, você já limita a questão das aplicações. As emissoras, elas ficam preocupadas com a questão da importação de programas também, né? Da compatibilidade. Então se você tira o Java do sistema, você já exclui uma quantidade de aplicações interativas. E Lua e uma coisa que não pesa. Então você botar Lua ou não botar Lua... O ganho de não botar Lua não existe praticamente. Então já que tem uma coisa que pode funcionar legal, que já está integrada com NCL, não é prejuizo você acrescentar. É muito leve, 50K. Com menos de 100K você bota interpretador Lua. Já tem o player, já tem tudo integrado com a linguagem, com o NCL. Então fica como mais uma opção.

Luciano, July 2007

O único livro em Inglês

> Luciano: O único livro em Inglês que eu li foi de Lua. É, único.
> Yuri: Leu mesmo, do inicio até o final?
> Luciano: Uhu.
> Yuri: E entendeu?
> Luciano: [Laughs.]
> Yuri: Levou tempo?
> Luciano: Muito, muito tempo.
> Yuri: Mas quando já tinha lido um livro deve ser mais fácil ler outro, né?
> Luciano: É, mas não arisco mais. Demora muito!

Jogos RPG

> Luciano: Jogos, jogos RPG. Conhece? Jogava muito—jogos RPG. Os jogos RPG me deram o minimo, e aí... Por exemplo para comprar coisas, eu ia atribuindo [...] a palavra "comprar", e a palavra "vender," entendeu? [...] Se tem que abrir aquela porta, é "abrir a porta." Atribuir as palavras "abrir a porta." [Pause.] É... é "buy", né?[373]

373 Luciano's "buy" in the end of the quote appears to refer to the discussion of buying/selling a few sentences earlier, rather than confusing the English words "buy" and "open."

Fácil me perder

Luciano: Pra duvidas—qualquer coisa. Entendeu? Se tiver uma duvida... Agora lá: [unclear] resolução de meu monitor [unclear], sei lá. Vou colocar no Google em inglês. [...] Mas em termos de aprendizado, quero um tutorial sobre [unclear], aí vou botar em português. [...] Com inglês é muito mais fácil me perder, entendeu, uma palavra vai mudar completamente tudo. Eu não garanto ainda.

Mauricio, July 2007

Como era bem nerd

Mauricio: E como era bem *nerd*, eu passava a maior parte de tempo no laboratório de informática.

Yuri: "Bem nerd" significa que?

Mauricio: *Geek*. Eu gostava muito disso. Não era pessoa muito social.

Destruir mouse pra fazer o cabo

Mauricio: Ele [o professor] entrava, dava uma aula, liberava tudo mundo e a turma ia jogar futebol. A turma toda saía e a gente ficava lá no laboratorio. É que saiu Doom, né, então assim. A grande onda da gente era pegar, destruir mouse pra fazer o cabo no modem. Jogar Doom contra...

Yuri: Explica...

Mauricio: Cabo de... no modem... Na verdade cabo serial, pra conectar computadores. Pra poder se comunicar. Não tinha rede naquela epoca. Ou até tinha mas não funcionava lá. [...] O mouse tinha um conector certo—seriál. [...] Era mais barato pegar um mouse, quebrar e fazer. Chegou um ponto quando a gente tinha tanta prática de fazer isso. A gente arrancava o mouse [picks up an imaginary mouse, rips off its cord and removes the isolation with teeth], juntava os fios, colocava... Demorava menos de dez minutos pra fazer um cabo.

Ricardo, July 2007

Que bobagem!

Ricardo: Eu lembro que a gente fez um projeto em C e ela incinou uma linguagem nova, tinha poucos anos, inventado aqui na PUC e chamada Lua. E eu olhei praquilo e foi assim: “Ahhh, uma linguagem inventado aqui? Que bobagem! Não vou aprender esse negocio. Nunca vou usar esse negócio na minha vida profissional. O que vou fazer com isso? E eu lembro que tinha duas partes da matéria que ela mandou. Uma parte era em C, uma parte era em Lua. E a menina que tava fazendo uma parte desse trabalho comigo... “Faz a parte em Lua porque não vou aprender esse negocio não, não quero saber de Lua, eu vou fazer a parte em C, que vai ser mais interessante porque vou usar.”

As coisas mudam

Ricardo: Eu sempre olhei pra esse negocio de Kepler e sempre achei muito interessante. O Rodrigo sempre passava, assim, pela PUC, e a gente ficava batendo papo, ele me contando como é que tava.

Yuri: Mas, interessante em que sentido?

Ricardo: [Long pause.] Aí você vê como é que as coisas mudam, né? A ideia de ser uma tecnologia nacional... [pause], bem estruturada [pause]. Uma proposta pra um framework mesmo, de desenvolvimento pra Internet. Pra competir, ou talvez o Rodrigo vai dizer que não é pra competir mas vamos dizer assim, com as tecnologias que existem por aí... Eu me interessei.

Prestigiando uma coisa que começou aqui

Yuri: A ideia de tecnologia nacional quer dizer que?

Ricardo: Oi?

Yuri: Você disse que uma parte que foi interessante foi essa ideia de tecnologia nacional.

Ricardo: Ah, é... [Pause.] Porque é que eu acho interessante? [Pause.] É

difícil dizer. É... [Pause.] Eu não sei muito bem. Talvez um pouco de patriotismo mesmo também. De achar que a gente pode desenvolver tecnologia muito boa, de nível mundial, entendeu? Pra ser usado por pessoas de todos os cantos, e que funciona.

Yuri: A gente nesse caso quer dizer quem?

Ricardo: Eu posso dizer "a gente Brasileiros" ou posso dizer "nós," "PUC," "Nas Nuvens," "comunidade *open source*." Não sei, não sei direito. É isso.

Yuri: Mas você tinha qual em mente?

Ricardo: Eu acho que [pause]... essa coisa de ser tecnologia nacional, ou seja desenvolvida, mesmo que inicialmente, por pessoas daqui, do Brasil. Eu acho que isso me empolgava. Mesmo que, "Ok, tem participação de gente do mundo inteiro." Melhor ainda! Cê tá entendendo? É gente do mundo inteiro prestigiando uma coisa que começou aqui. [Long pause.]

Rodrigo, October 2007

1981

(07:48:03 PM) RMiranda: (EC) I started with this computer thing around 1981
(07:48:22 PM) RMiranda: but I was doing Texas calculators before, so I already new something about programming
(07:48:36 PM) YuriOnLaptop: you said you bought a computer in college, right?
(07:48:46 PM) YuriOnLaptop: where did you use one before?
(07:48:51 PM) RMiranda: at the computer store
(07:48:59 PM) YuriOnLaptop: ?
(07:49:20 PM) RMiranda: there was one (really one) computer store in Rio at the time
(07:49:41 PM) RMiranda: so we used to skip classes at school and go there
(07:49:44 PM) YuriOnLaptop: but how could you use a computer there?
(07:50:05 PM) RMiranda: stading in front of it
(07:50:06 PM) RMiranda: :)
(07:50:13 PM) RMiranda: nobody went there
(07:50:16 PM) YuriOnLaptop: but for how long would they let you use it?
(07:50:23 PM) RMiranda: and it was a TImex sinclair zx81
(07:50:27 PM) RMiranda: hours

(07:50:33 PM) RMiranda: like 4-6 a day
(07:50:41 PM) RMiranda: but we were sharing it
(07:50:46 PM) RMiranda: like for 4 people
(07:50:50 PM) YuriOnLaptop: they didn't kick you out?
(07:51:00 PM) RMiranda: no, they were amused by those kids
(07:51:08 PM) RMiranda: everybody had that IBM look
(07:51:28 PM) RMiranda: and they wanted to show to CEOs how that TRS80 really would revolutionize their companies
(07:51:44 PM) RMiranda: so I guess pointing at us and saying, see? even kids can use these new computers
(07:51:50 PM) RMiranda: that could even help them
(07:51:51 PM) RMiranda: hehe
(07:52:09 PM) YuriOnLaptop: and how did you get interested in that?
(07:53:31 PM) RMiranda: A guy in my classroom was into electronics and he was trying to learn how to program in assembler. when he heard that the other nerd used to program a TI58 he assumed I would be able to learn assembler too
(07:53:41 PM) RMiranda: so he gave me the Z80 ref man
(07:55:04 PM) YuriOnLaptop: and what did you say?
(07:55:17 PM) RMiranda: about the manual? I simpy loved it
(07:55:25 PM) YuriOnLaptop: how so?
(07:55:33 PM) RMiranda: it was so easier than the TI! :)
(07:55:59 PM) YuriOnLaptop: I had that feeling too when I learned assembly in college.
(07:56:08 PM) YuriOnLaptop: but that was in 98
(07:56:38 PM) YuriOnLaptop: but how did you get into programming TI58 in the first place?
(07:57:40 PM) RMiranda: I got one as a present from a cousin
(07:57:47 PM) RMiranda: that heard I was into electronics
(07:57:55 PM) RMiranda: and he used it no more
(07:58:48 PM) RMiranda: then I somehow upgraded that 58 for a 59, in order to enjoy the amazing world of persistent storage (magnetic cards!) :)
(07:59:06 PM) YuriOnLaptop: you were into electronics before that? in what way?
(07:59:18 PM) RMiranda: just as a hobby
(07:59:36 PM) RMiranda: but I was considering it as a profession
(07:59:44 PM) RMiranda: the initial plan was architecture
(07:59:50 PM) YuriOnLaptop: that was at what age?
(07:59:58 PM) RMiranda: 12
(08:00:24 PM) YuriOnLaptop: but what is electronics as a hobby? (I am the wrong generation, perhaps...)
(08:00:51 PM) RMiranda: small devices like sirens, led based dices, door locks, alarms, digital clocks, nothing too fancy

(08:00:57 PM) RMiranda: but at 12 this is amazing
(08:01:03 PM) YuriOnLaptop: and how did you get into that?
(08:01:11 PM) RMiranda: magazines in the newstand
(08:01:40 PM) RMiranda: oh, and a beginners course by mail
(08:01:44 PM) YuriOnLaptop: there are also magazines on doing makeup in the newstand - you didn't get into that
(08:01:55 PM) RMiranda: if you say so
(08:01:58 PM) RMiranda: :):):)
(08:02:04 PM) YuriOnLaptop: there must have been other influences
(08:02:20 PM) RMiranda: well, people around me were nerdy too
(08:02:32 PM) RMiranda: so everybody discovered those things around the same age

Luiz Fernando, January 2009

O Brasil não acreditava

Luiz Fernando: A primeira dificuldade que a gente teve muito grande era que as pessoas não acreditavam que a gente tinha feito realmente uma coisa que era muito melhor do que tava sendo proposto lá fora. E foi muito interessante que o Brasil não acreditava de jeito nenhum e precisou pessoal aí fora acreditar e dizer isso primeiro, pra que depois o Brasil aceitasse.

Appendix D: Doctoral Degrees of PUC-Rio Department of Informatics Faculty

Table 3 shows which institutions have awarded doctoral degrees to the people working as full-time professors at PUC-Rio's Departamento de Infomatica in 2007. (All full-time professors had a Ph.D. in 2007.) The list of faculty members was obtained from the department's website.[374] The degrees were looked up in Sistema de Currículos Lattes.[375] The data was collected in October of 2007.

Table 3: Doctoral degrees of the professors of Departmanento de Informatica at PUC-Rio

Year	Institution awarding the doctoral degree	Post-docs abroad
1974	U. of Toronto (Canada)	
1974	U. of Waterloo (Canada)	Canada (Waterloo, 1994)
1975	UCLA (USA), earlier MS from U. of Waterloo (no MS from PUC)	the US (IBM Research, 1975)
1976	ITA (Brazil), undergraduate and MS from ITA	
1979	Harvard (USA)	
1979	PUC-Rio (Brazil)	
1981	UCLA (USA)	Italy (POLIMI, 1990/1991)
1982	Cornell (USA)	
1983	PUC-Rio (Brazil)	France (ENSTEL, 1985)
1985	UC Berkeley (USA), earlier MS from UFRJ (Rio de Janeiro)	
1987	PUC-Rio (Brazil)	in the US twice (Stanford, 1991/1992 and UMBC, 2002)
1988	UC Irvine (USA)	
1988	Univ. of London (UK)	
1990	PUC-Rio (Brazil)	Canada (Waterloo, 1991/1992)
1990	PUC-Rio (Brazil)	Denmark (Aarhus, 1994), Germany (EKUT, 2003)
1991	Univ. of London (UK)	

374 http://www-nt.inf.puc-rio.br/cgilua/cgilua.exe/pessoa_si.htm?cxid=professor

375 http://lattes.cnpq.br/

1992	Tech. Univ. of Berlin (Germany)	
1993	École Polytechnique de Montréal (Canada)	
1993	PUC-Rio (Brazil)	the US (UIUC, 2001/2002)
1994	ENST (France),	the US (UC San Diego, 2004/2005)
1995	PUC-Rio (Brazil)	the US (Cornell, 1995-1997)
1999	PUC-Rio (Brazil)	
1999	PUC-Rio (Brazil)	France (ISIMA, 2003)
1999	PUC-Rio (Brazil)	
2000	PUC-Rio (Brazil)	the US (UIUC, 2001/2002)

With the exception of one person, all faculty members had obtained their PhDs either at PUC or abroad. All but three of the faculty members earned an MS from PUC-Rio, and most did their undergraduate degree there. Only two do not have any PUC degrees and only three did their undergraduate studies outside Rio de Janeiro. This likely reflects a confluence of factors: (1) a number of faculty members obtained their doctoral degrees while already teaching at PUC, (2) PUC-Rio has few rivals in Brazil, (3) few people move to Rio de Janeiro today, as the city is considered violent. Of those who got their PhD at PUC all but three did a post-doc abroad later.

The percentage of PUC degrees is noticeably higher among the recent PhDs. Of those who obtained their Ph.D. before the 1990s, only three did so at PUC. (Note that PUC did not award doctoral degrees in computer science until the late 1970s.) On the other hand, all five of the most recent Ph.D. recipients got their degrees at PUC.

Appendix E: Length of the Wikipedia Article on Lua in Different Languages

Table 4 presents the length and the age of the Wikipedia articles on Lua in different language. The English article is by far the longest and the second oldest (after Polish, which preceded it by two months), it is also the most revised. The Portuguese article is surpassed by those in eight other languages. (Note the total number of Wikipedia articles in some of those languages is smaller than the number of articles in the Portuguese Wikipedia.)

Table 4: The length of the Wikipedia article on Lua in different languages.

Language	World Count (on 2007.05.28)	Number of Revisions (on 2007.05.28)	Month of the First Version
English	3181	~500	2002/03
French	1065	~100	2004/10
Czech	735	14	2005/09
German	667	102	2004/04
Mandarin Chinese	~660[376]	16	2005/08
Korean	645	21	2005/04
Spanish	547	37	2004/07
Russian	518	59	2005/04
Portuguese	**475**	**44**	**2004/07**
Galician[377]	431	9	2005/08
Italian	387	14	2006/04
Esperanto	383	9	2006/10
Danish	349	5	2006/10
Japanese	~340[378]	10	2007/02

376 1315 characters, divided by 2.

377 Galician is very similar to Portuguese, and much of content of the Galician article had been copied from the Portuguese version with minimal modifications.

Finnish	266	17	2004/11
Polish	128	31	2002/01

378 850 characters, divided by 2.5.

Appendix F: Counting Software Developers in the US and Brazil

This appendix looks at statistics for computer professionals working in the US and Brazil. While it would be more desirable for the purpose of this dissertation to look specifically at statistics for *software developers,* such numbers cannot be easily isolated. Generally, making cross-country comparisons between the numbers of people engaged in specific occupations is not trivial. For this reason I describe in some detail the process used to obtain the data and the caveats that should be kept in mind when interpreting the numbers.

This appendix focuses on occupational statistics, which should not be confused with employment by industry or type of economic activities. For example, a software developer (an occupational classification) may be employed by a bank, thus working in "banking" (a type of economic activity). Similarly, a software company may employ an accountant, who represents a different occupation. Industry employment statistics are cited quite often in the literature on software development. (See appendix G.) Processed numbers for software developers as an occupation, on the other hand, are not readily available.

Occupational statistics can be obtained from several data sources, which include complete population censuses, studies that specifically aim to measure and categorize the labor force, and representative sampling-based studies of the population (often conducted

more often). This appendix draws on the first two kinds of data for both the US and Brazil.

For the census, I used microdata provided by IPUMS[379] for the US and Brazil censuses conducted in 2000, extracted records for certain categories of computer professionals (see below) and grouped them by metropolitan area.

I also include tables based on the main occupational surveys for each country. Occupational Employment Statistics (OES) is a semi-annual mail survey conducted by Bureau of Labor Statistics in the US in order to estimate employment and wages for about 800 occupations. Around 200,000 establishments are surveyed during every six months cycle. OES reports data by location, occupations, and industry. The data does not cover self-employed people. I tabulated the data from May 2006 using the tool on the BSL website.[380] RAIS ("Relação Anual de Informações Sociais" = "Annual Report of Social Information") is a dataset compiled by IBGE based on reports by employers (which includes the self-employed). In theory all employers are required to supply the data for RAIS, though in practice it is unclear whether all do that, especially in case of the smaller employers. I tabulated 2006 data using RAIS website.[381] The tables were then merged by hand. For both countries occupational surveys for 2006 produced somewhat smaller numbers than the 2000 censuses.

---

379 http://www.ipums.org/international/

380 http://www.bls.gov/oes/

381 http://www.mte.gov.br/pdet/Acesso/RaisOnLine.asp (Login is required, which was obtained from MTE.)

Occupational Classifications

Despite the attempts to standardize occupational classifications, and the existence of an international standard called “ISCO88” (referred to as “CUIO88” in the Spanish and Portuguese literatures), national statistics continue to rely on their own occupational classifications, which makes cross-national comparisons difficult. This work is often simplified by reducing the precision of the classification, and world-level statistics usually only count each country’s workers by broad “skill-level” category (Hoffmann 1999, p. 6), generally classifying software developers as “professionals” (ISCO88 Category 2) or “technicians and associate professionals” (ISCO88 Category 3). Such reduced statistics are useful for broad studies of inequality and stratification, but cannot help understand how many people in the world actually write software code. For this reason, I discuss the specific occupational categories used in the US and Brazil.

While no country uses ISCO88 for collecting its statistics and data that is re-coded into ISCO88 typically lacks precision for counting people by specific occupation, national classifications are increasingly adjusted to be closer to ISCO88 in their overall design, and for this reason it is useful to look at ISCO88 first.

ISCO88 organizes occupations into a hierarchy, which starts with a division into ten “major groups” which are associated with four skill levels (Elias & Birch 1994). (Groups 1 and 10 are not assigned skill level, because those groups are seen as following their own logic of membership and promotion.) The ten groups are show in table 5.

Table 5: ISCO88 major groups.

Skill Level	Major Groups
does not apply	1. Legislators, senior officials and managers
4 (highest)	2. Professionals
3	3. Technicians and associate professionals
2	4. Clerks
	5. Service workers and shop and market sales workers
	6. Skilled agricultural and fishery workers
	7. Craft and related workers
	8. Plant and machine operators and assemblers
1 (lowest)	9. Elementary occupations
does not apply	10. Armed forces

In this approach, which is followed in national classification schemes for the United States and Brazil, people who write software may end up divided between several categories: while most of them will be likely classified as “professionals” (major group 2, highest skill level), some may be classified as “technicians and associate professionals” (major group 3, second highest skill level) and others as “legislators, senior officials and managers” (group 1). Looking at the terms used for sub-groups of major groups 2 and 3, however, it seems likely that *most* programmers will be classified as “professionals” and for this reason I only look at subcategories of ISCO88 major group 2 and corresponding national categories.

Within major group 2, ISCO provides two third-level codes that may be used for people who work with software: 213 (“Computing professionals”) and 231 (“College, university and higher education teaching professionals”). Most national datasets, however, do not separate their analogues of group 231 by field, so it is impossible to separate computer science professors from chemistry professors. For this reason, I ignore group

231 and the corresponding categories in national classification systems.

ISCO88 defines 3 subgroups of group 213 that may include people who write software:

2131 - Computer systems designers, analysts and programmers
2132 - Computer programmers
2139 - Computing professionals not elsewhere classified

While most national classification systems have clear equivalents of group 213, the way this group is subdivided varies greatly from country to country, as well as from agency to agency and from census to census.

The United States

In the United States, the 2000 revision of "Standard Occupational Classification" (SOC 2000) is used for most recent studies by federal statistical agencies.[382] A different but compatible system (typically referred to as "Census 2000") was used in the 2000 US Census. It uses three digit codes, but such codes map either onto specific single SOC 2000 codes or onto a collection of them. Table 6 presents the SOC 2000 and Census 2000 categories for occupations that may involve writing software code.

Table 7. SOC 2000 and Census 2000 codes for occupations that may involve writing software:

Census 2000 category		SOC 2000	
Code	Description	Code	Description
100	Computer Scientists and System Analysts	151011	Computer and Information Scientists, Research
		151051	Computer Systems Analysts

382 See http://www.bls.gov/soc/

		151099	Computer Specialists, All Other
101	Computer Programmers	151021	Computer Programmers
102	Computer Software Engineers	151031	Computer Software Engineers, Applications
		151032	Computer Software Engineers, Systems Software
104	Computer Support Specialists	151041	Computer Support Specialists
106	Database Administrators	151061	Database Administrators
110	Network and Computer Systems Administrators	151071	Network and Computer Systems Administrators
111	Network Systems and Data Communications Analysts	151081	Network Systems and Data Communications Analysts

Table 9 shows the data for the US Census of 2000, cross-tabulated by the 2000 Census occupational category and metropolitan area. (Only the top fifteen metropolitan areas are shown, but the total count shows the total for the whole dataset.) The table shows a total of 3.4 million "computer professionals," distributed fairly evenly between the seven subcategories used by the US Census, with the exception of category 106 ("Database administrators") which has substantially lower counts. About 70% of the computing professionals are assigned to the top three categories (100 "Computer Scientists and System Analysts", 101 "Computer Programmers", and 102 "Computer Software Engineers"). Computer professionals comprise as much as 1.2% of the total US population (and 2% of the employed population). Their concentration goes to nearly 5% of the population in San Jose and to nearly 3% in Seattle (7.7 and 4.2 percent of the employed population respectively).

Metropolitan areas with the largest numbers of software developers vary substantially in density. Some, like New York/NJ, Los Angeles or Chicago have large totals with relatively low density. Others, such as Washington D.C, San Jose, or Boston

have fairly high concentration. (Note that the US Census separates San Francisco/ Oakland and San Jose into two metropolitan areas. If they were treated as one, their total number of computer professionals would be 199,073 —the second highest after New York/NJ, and ahead of Washington, D.C.)

Table 10 shows a similar table based on the OES data for May 2006, tabulated using the tool on the BSL website.[383] The OES numbers are fairly similar, though somewhat lower—a total of 2.7 million computer professionals rather than 3.4 million for Census 2000.

Brazil

In Brazil, variations of a classification system known a CBO ("Classificação Brasileira de Ocupações" = "Brazilian Occupational Classification") are used for most federal statistics. CBO-based classifications correspond to ISCO88 at the two-digit level, but diverge after that. While the 2002 revision of CBO (CBO 2002) is the "official" classification system and is used in RAIS, a somewhat simplified version of it, known as "CBO-Domiciliar" ("Household CBO"), was used in the 2000 Census and the subsequent household studies such as PNAD.[384]

CBO-Domiciliar uses five four-digit codes for groups within 212 (which corresponds to "213" in ISCO88). CBO uses only three four-digit categories but additionally defines six-digit categories. However, I am not aware of any statistics

383 http://www.bls.gov/oes/

384 See http://www.ibge.gov.br/concla/classocupacoes/classocupacoes.php

available at the six-digit level, and I ignore the six-digit codes for this reason. The codes and their definitions are shown in table 8.

Table 8. Categories for computing professionals in CBO 2002 and CBO Domiciliar

Code	CBO Description	CBO Domiciliar Description
2121	n.a.	Computing Specialist (Espercialista em Computação)
2122	Computation Engineers (Engenheiros em computação)	Computation Engineers - Software Developers (Engenheiros em Computação - Desenvolvedores de Software)
2123	Network, system and database administrators (Administradores de redes, sistemas e banco de dados)	Computer Specialists (Especialista em Informática)
2124	Analysts of Computational Systems (Analistas de sistemas computacionais)	System Analysts (Analistas de sistemas)
2125	n.a.	Computer programmers (Programmadores de Informática)

Of the CBO categories shown in table 8, two may be interpreted as applicable to people who would be called "software developers" in the United States: 2122 and 2124. Unfortunately, category 2124 ("Analistas de sistemas") *also* includes "database analysts" and "system analysts," as well as people who offer technical support. Additionally, 2124 is the most commonly used category in Brazilian statistics, outnumbering any other category by a factor of ten or more in all data sets. Therefore, all computing professionals in Brazil appear to be de facto lumped as "2124" and cannot be separated.

Table 11 shows the number of computer professionals in the census of 2000 cross-tabulated by CBO Domiciliar category and by metropolitan area. There is a total of about 158,000 computing professionals—a fraction of the US total. While this difference arises in part due to Brazil's smaller population, computer professionals also account for a much smaller percentage of the population in Brazil than in the US: 0.09 percent compared to

1.2 percent (0.24% vs 2% of the employed population respectively), a difference of an order of magnitude.

There is roughly a 1:1:2:100:10 ratio between categories 2121, 2122, 2123, 2124 and 2125. In other words, the overwhelming majority of Brazilian computer professionals are classified as 2124 ("Analistas de sistemas"). I find this finding an anomaly and suspect that it may reflect a problem with the classification method.

São Paulo and Rio de Janeiro have the highest number of computer professionals. Like the overall population, Brazilian cities have much fewer computer professionals than American cities of the same size, both in terms of total numbers and as the percentage of the population. For example, São Paulo has about 1/5 the number of computer professionals of New York, and Rio de Janeiro has about 1/6 the number of computer professionals of Los Angeles. In smaller cities that are typically described as being important centers of software development in Brazil (Curitiba, Campinas, Florianopolis) computer professionals make up 0.2 to 0.3 % of the population and 0.5 to 0.7 % of the labor force. In particular, Campinas, which is often hailed as "Brazilian Silicon Valley" has about 1/15 the number of computer professionals of San Jose by total numbers and 1/14 by density. (Campinas and San Jose have roughly the same population, with 2.3 and 1.7 million people respectively.)[385]

Table 12 shows the counts for categories 2122 ("Engenheiros de computação")

385 The Federal District (which would be natually comparable to D.C.) is missing from this list, despite being one of the most important areas in other statistics. This appears to be due to the fact that it simply is not classified as a "metropolitan area" in IPUMS data. It is likely that the Federal District professionals make up the bulk of what went into "Unidentified Metro."

and 2124 (“Analistas de Sistemas”). Those numbers are quite similar to those in the census. Rio again occupies the second place for Brazil, with a total of around 18,000 computer professionals.

Table 9: Computer professionals in the United States 2000 Census

Code	Metro	Total CPs	100	101	102	104	106	110	111	Population (millions)	Employed (millions)	CPs per 1000 people	CPs per 1000 employed
	Total/Average	3384372	764917	741048	766563	448295	85114	217879	360556	281.4	169.2	12.03	20
	Unidentified Metro	330345	75889	75371	56293	55230	7754	24293	35515	67.4	39.6	4.9	8.34
560	New York-Northeastern NJ	254253	62947	71231	44951	24852	6083	14987	29202	17.2	10.1	14.74	25.17
884	Washington, DC/MD/VA	177979	38361	26899	52019	21764	4778	10038	24120	4.7	3.1	37.6	**57.81**
448	Los Angeles-Long Beach, CA	136256	32469	33007	27391	15769	3149	7598	16873	12.4	7.1	11.02	19.15
160	Chicago-Gary-Lake IL	132052	34711	32610	27107	13211	3594	7107	13712	8.8	5.3	15	24.86
736	San Francisco-Oakland-Vallejo, CA	116770	22224	21624	37651	11701	2801	6783	13986	4.6	2.9	25.13	39.83
192	Dallas-Fort Worth, TX	108372	22290	21388	27397	17478	3246	5857	10716	5	3.2	21.49	34.34
112	Boston, MA	101567	20570	18711	35676	9286	2111	5439	9774	4	2.5	25.7	**40.56**
740	San Jose, CA	82303	10250	10715	41819	6128	1348	4255	7788	1.7	1.1	48.76	**77.21**
52	Atlanta, GA	82213	19227	15700	19087	11762	2509	4967	8961	4	2.6	20.62	32.21
616	Philadelphia, PA/NJ	74987	18178	18617	15849	8915	1846	4367	7215	5.1	3	14.76	24.75
760	Seattle-Everett, WA	65378	11834	10236	23425	6949	1913	3726	7295	2.3	1.5	28.03	**42.84**
512	Minneapolis-St. Paul, MN	62603	15716	15027	12675	7884	1755	4183	5363	2.9	1.9	21.92	33.58
208	Denver-Boulder-Longmont, CO	58244	11185	10351	16489	7611	1899	4284	6425	2.2	1.4	26.49	40.42
336	Houston-Brazoria, TX	56725	14859	11905	11569	7950	1666	3847	4929	4.4	2.6	12.85	21.67
216	Detroit, MI	53910	15806	11857	9888	6238	1281	4178	4662	4.4	2.6	12.17	20.56

Table 10: Computer professionals in the US Occupational Employment Statistics (OES) by BSL.

Area	Total CPs	151011	151021	151031	151032	151041	151051	151061	151071	151081	151099	Population (millions)	CPs per 1000
Total / Average	2722600	18890	373690	449970	264420	496750	409980	100400	276030	190790	141680	243.6	11.17
New York-Northern New Jersey-Long Island, NY-NJ-PA	201320	1940	39880	37570		34170	34580	9020	21360	16490	6310	18.8	10.70
Washington-Arlington-Alexandria, DC-VA-MD-WV	194080	3140	16680	32980	29730	22210	30200	6340	19350	13560	19890	5.2	36.69
Los Angeles-Long Beach-Santa Ana, CA	123340	830	13020	27020	12350	21810	16700	4170	11350	8240	7850	12.9	9.52
Chicago-Naperville-Joilet, IL-IN-WI	98650	1030	19760	14450	12960	17180	12790	4270	10360	5850		9.5	10.38
Boston-Cambridge-Quincy, MA-NH	94440	1410	11010	23040	20470	14400	12590	3500		5570	2450	4.4	21.20
Dallas-Fort Worth-Arlington, TX	93140	250	16080	12540	14570	14480	15840	4000	8080	5050	2250	6.0	15.51
San Francisco-Oakland-Fremont, CA	76710	1670	6430	18960	8650	9430	11310	2380	6680	5520	5680	4.1	18.35
Philadelphia-Camden-Wilmington, PA-NJ-DE-MD	75270	600	11650	11470	7190	13050	11220	3220	9330	4440	3100	5.8	12.92
Atlanta-Sandy Springs-Marietta, GA	70480	400	7890	9120	7050	14770	13360	2670	6380	5930	2910	5.1	13.72
San Jose-Sunnyvale-Santa Clara, CA	70040	1130	5160	20030	18680	7960	5960	1150	3700	3070	3200	1.7	39.19
Seattle-Tacoma-Bellevue, WA	64850		9450	20550		8890	9090	2280	6330	3830	4430	3.2	19.87
Minneapolis-St. Paul-Bloomington, MN-WI	60310		4750	13880	5830	8200	7770	2490	7010	4510	5870	3.1	19.00
Houston-Sugar Land-Baytown, TX	54240	450	7810	7420	5530	9950	10130	2240	5660	3050	2000	5.5	9.79
Denver-Aurora, CO	46240	190	3440	7860	6050	9010	6760	1700	4360	2960	3910	2.4	19.20

Table 11: Computer professionals in the Brazilian 2000 Census

Code	Metropolitan Area	Total CPs	2121	2122	2123	2124	2125	Population (millions)	Employed (millions)	CPs per 1000 people	CPs per 1000 employed
	Total/Average	158198	1660	1436	3699	138947	12455	169.8	65.6	0.93	2.41
	Unidentified Metro	27235	488	288	655	23201	2603	101.9	38.8	0.27	0.70
28	Undocumented	7647	0	66	245	7086	251	3	1.2	2.59	6.51
14	São Paulo	55646	624	402	983	49812	3826	17.9	7.2	**3.11**	**7.73**
13	Rio de Janeiro	21177	138	308	549	19085	1097	10.9	4.2	**1.94**	**5.07**
8	Belo Horizonte	7097	39	0	168	5989	900	4.3	1.8	1.63	4.01
17	Curitiba	5999	52	72	140	5350	385	2.7	1.2	**2.2**	**5.16**
26	Porto Alegre	5715	42	54	157	4704	758	3.7	1.6	1.56	3.67
16	Campinas	5468	134	131	128	4697	378	2.3	1.0	**2.34**	**5.51**
7	Salvador	3270	18	16	48	3030	156	3	1.1	1.09	2.92
5	Recife	3135	0	30	44	2832	230	3.3	1.1	0.94	2.85
15	Santos Coastal Region	2047	0	40	63	1775	169	1.5	0.6	1.38	3.61
3	Fortaleza	2039	0	0	100	1700	239	3	1.1	0.69	1.93
27	Goiânia	1934	0	0	46	1616	273	1.6	0.7	1.18	2.63
12	Grande Vitória	1930	0	0	22	1807	100	1.4	0.6	1.35	3.37
20	Florianópolis	1444	18	29	34	1251	112	0.7	0.3	**2.04**	4.68
1	Belém	1128	22	0	24	1009	73	1.8	0.6	0.63	1.79

Table 12: Computer professionals (categories 2122 and 2124 only) in RAIS 2006.

Metropolitan Area	2122	2124	Total
São Paulo	850	54580	55430
Rio de Janeiro	114	18149	18263
Federal District	16	10525	10541
Campinas	347	5591	5938
Belo Horizonte	110	5819	5929
Curitiba	83	4762	4845
Porto Alegre	60	4308	4368
Salvador	12	3041	3053
Recife	167	2458	2625
Fortaleza	8	1641	1649
Vitória	22	1415	1437
Florianópolis	21	1173	1194
Goiânia	5	1112	1117
Belém	1	813	814
RM Norte/Nordeste Catarinense	0	743	743
Others	184	18194	18378
Total	2040	139142	141182

Appendix G: Software Industry Employment

The number of software developers can be roughly estimated by looking at the employment statistics for economic sectors that are likely to employ them, and in particular at the numbers for software industry employment. While such numbers are more readily available than actual occupational statistics, it is important to note that not all software developers are employed in the software industry and not all people employed in the software industry are software developers. Additionally, such numbers need to be taken with tremendous caution, since they are often assembled by industry associations without clear indication of methodology. As Veloso et al. (2005, p. 101) point out, "statistics on software are illusive."

Table 13 collates a number of estimates for software industry employment for different countries.

Table 13: Software industry employment in different countries.

Country	**Estimate**	**Metric and Year**	**Source**
United States	1,042,000	"software industry employees" in 2001	Veloso et al. 2005 (p. 101), "authors' compilation from various sources"
India	350,000	--//--	--//--
	690,000	employment in "IT Services" and "Engineering Services and R&D and Software Products" in 2007	NASSCOM (http://www.nasscom.in/upload/5216/Domestic%20Services.pdf)
	1,068,000	same as above + employment in "Domestic Market (including user organizations)"	--//--
Japan	534,000	"software industry employees" in 2001	Veloso et al. 2005 (p. 101), "authors' compilation from various sources"

Germany	300,000	--//--	--//--
China	300,000	“software professionals” in 2002	Tschang & Xue 2005 (p. 156), citing China Software Industry Association
	186,000	“software industry employees” in 2001	Veloso et al. 2005 (p. 101), “authors’ compilation from various sources”
Brazil	158,000	--//--	--//--
Russia	70,000	“programmers” employed in the software industry in 2003	Hawk (2005)
	50,000–80,000	people who “work in the programming industry ”	Terekhov (2001)
Israel	14,500	software industry employment in 2000	Breznitz 2005 (p. 75), citing IAEI (http://www.iaei.org.il/) and IASH (http://www.iash.org.il/content/softwareInds/StatisticalInformation.asp)
	35,000	“software industry employees” in 2001	Veloso et al. 2005 (p. 101), “authors’ compilation from various sources”
Ireland	10,710	software industry employees in 2003	Sands 2005 (p. 44), citing National Software Directorate (http://www.nsd.ie/)
	15,000	software industry employees in 2003	Sands 2005 (p. 44), citing National Software Directorate (http://www.nsd.ie/)
	14,000	software industry employment in Irish-owned companyes in 2000	Breznitz 2005 (p. 75), citing National Software Directorate (http://www.nsd.ie/)
	25,000	“software industry employees” in 2001	Veloso et al. 2005 (p. 101), “authors’ compilation from various sources”
Spain	20,000	“software industry employees” in 2001	Veloso et al. 2005 (p. 101), “authors’ compilation from various sources”
Finland	20,000	--//--	--//--
Argentina	15,000	--//--	--//--

Appendix H: Market Capitalization of Software Companies

While software industry employment and occupational statistics show that a disproportionate amount of software work is done in the United States and India, they do not show the full extent of centralization. Looking instead at the location of the headquarters of the most important software companies gives us a much starker image.

Table 14 was derived by taking a list of 372 public companies with highest market capitalization traded in the US and classified as engaging in "software programming" and "computer services." The data was obtained from Google Finance on May 15, 2008. The addresses of each company's headquarters were obtained from the same site. The addresses were then mapped onto metropolitan areas. (For the US addresses, I used ZCTA-to-MA mapping database provided by the US Census Bureau. For foreign addresses I did the grouping by hand.[386]) I then counted total market capitalization for each metropolitan area.

The companies in the dataset have a combined market capitalization of about 1.2 trillion dollars. Note that this count excludes many companies that are traditionally associated with the computer and software industry, as they may be assigned to other categories—for example, IBM is assigned to "business services" and Yahoo! to "advertising." It does include most large foreign software companies, such as Infosys,

386 Note, though, that most foreign metropolitan areas have only one company, so metropolitan area boundaries do not matter.

SAP, Wipro and Satyam, which are either listed on American exchanges directly or indirectly through ADR. Two metropolitan areas account for 37.2 and 23.7 percent of the total, with the rest of the world accounting for the remaining 39.1 percent.

Table 14: Market capitalization of "software programming" and "computer services" companies by metropolitan area of corporate headquarters.

Metropolitan Area Code	Metropolitan Area Description	Market capitalization (US$ millions)
7362 CA	San Francisco and San Jose, CA[387]	444089.34
7602 WA	Seattle, WA[388]	282305.31
	The rest of the world combined	466675.34
	Germany-Walldorf[389]	60040.00
5602 NY		53932.66
	India-Bangalore[390]	44910.00
4472 CA		25667.51
	UK-London	25623.96
1922 TX		25371.02
	China-Beijing	22037.30
Other Zip MA		15619.59
5483 CT		12760.00
5602 NJ		12354.21
	China-Shanghai	9654.07
8872 VA		9450.82
	India-Hyderabad	8710.00
5082 WI		8460.00
7040 MO		7792.00
4992 FL		7481.15
	Israel-Tel Aviv	7325.97
1122 MA		7322.80
3362 TX		7218.08
	Japan-Tokyo	5792.20
1602 IL		5484.42

387 Google is $181 billion (40.7% of the metropolitan area's total).

388 Microsoft is $278.75 billion (near all of the metropolitan area's total)

389 SAP is 100% of the metropolitan area's total.

390 $25.36 billion from Infosys and $19.55 billion from WIPRO.

8872 MD		5134.88
1800 GA		5000.00
7320 CA		4975.86
6280 PA		4464.29
1642 OH		4340.00
5120 MN		4337.17
6640 NC		4285.29
Other Zip OH		4220.00
	Canada-Montreal	3571.71
2162 MI		3560.00
0520 GA		3146.34
0640 TX		3140.27
	Israel-Other	2401.83
	Argentina-Buenos Aires[391]	2380.00
2082 CO		2273.35
	Canada-Waterloo	2084.29
8960 FL		1978.16
	India-Mumbai	1939.48
Other Zip PA		1923.57
6520 UT		1904.59
Other Zip NH		1327.78
6442 OR		1159.60
8872 DC		1120.00
1440 SC		1110.00
	China-Xiamen	1000.00
	Taiwan-Taipei	979.24
Other Zip AR		949.95

391 Mercadolibre Inc.

Appendix I: Programming Language Popularity

While thousands of programming languages have been designed[392] only a small number are actively used. This happens in part because of massive network effects associated with programming languages. There is no reliable methodology for measuring the use of programming languages, but several measurements of popularity have been proposed. TIOBE Programming Community Index[393] measures popularity by number of Google search results for queries related to programming in particular languages. Table 15 shows TIOBE's list of twenty most popular languages as of May of 2009. We can see in this table that there is a quick drop off after a small number of programming languages.

A study of programming languages used by open source software projects (see below for a discussion of open source) by Delorey et al. (2005) shows a remarkably similar distribution, as shown in table 16. The only notable difference from the TIOBE list is the absence of Visual Basic (#5 on TIOBE's list), a programming language that is unpopular among open source software developers since it works only on Microsoft Windows. Lua does not appear on Delorey et al.'s list. This is most likely due to a combination of reasons: the list only includes the twelve most popular combinations (not

392 The History of Programming Languages project website (http://hopl.murdoch.edu.au/) listed 8,512 as of May 2009.

393 http://www.tiobe.com/tpci.htm

twenty like TIOBE's), Lua was much less known in 2005 and projects using Lua tend to be housed on LuaForge.org rather than the website from which the data was collected.

Table 15: Tiobe TPCI ranking for May 2009.

Position in May 2009	Position in May 2008	Delta in Position	Programming Language	Ratings May 2009	Delta	Status
1	1	unchanged	Java	19.537%	-1.35%	A
2	2	unchanged	C	16.128%	+0.62%	A
3	3	unchanged	C++	11.068%	+0.26%	A
4	4	unchanged	PHP	9.921%	-0.28%	A
5	5	unchanged	(Visual) Basic	8.631%	-1.16%	A
6	7	+1	Python	5.548%	+0.65%	A
7	8	+1	C#	4.266%	+0.21%	A
8	9	+1	JavaScript	3.548%	+0.62%	A
9	6	-3	Perl	3.525%	-2.02%	A
10	10	unchanged	Ruby	2.692%	+0.05%	A
11	11	unchanged	Delphi	2.327%	+0.30%	A
12	14	+2	PL/SQL	1.101%	+0.34%	A
13	13	unchanged	SAS	0.801%	-0.12%	A
14	15	+1	Pascal	0.776%	+0.18%	A
15	26	+11	RPG (OS/400)	0.678%	+0.43%	B
16	27	+11	ABAP	0.670%	+0.43%	B
17	12	+5	D	0.628%	-0.82%	A-
18	23	+5	MATLAB	0.517%	+0.25%	B
19	21	+2	Logo	0.504%	+0.17%	A-
20	19	-1	Lua	0.486%	+0.12%	B

Table 16: Programming languages most often used by open source projects (Delorey et al. 2005)

Programming Language Combination	Percentage of Open Source Projects (2005)
Java	21%
C	16%
C++	8%
PHP	7%
C, C++	4%
JavaScript, PHP	4%
Perl	3%
Java, JavaScript	3%
Python	3%
C, Perl	2%
Pascal	2%
C#	2%

Appendix J: Lua-L Statistics

Tables 17 and 18 show the distribution of message sent to Lua-L (the Lua mailing list) among different top-level domains (TLDs), between February 19, 1998 and August 12, 2008. Tables 17 shows the counts of messages by TLD. Tables 18 shows the distribution of individual senders. Note that .br accounts for a higher percentage of messages (9.3% in 2008) than of senders (4.2% in 2008). This is largely due to the fact that a small number of PUC members contribute at a disproportionate rate.

In both tables the domains are sorted by the total number of messages sent from them. The numbers for 1997 and 2008 show a part of the year. Popular email providers (gmail, hotmail, yahoo) are shown separately, since they are used by people in different countries. (Gmail in particular is quite popular among Brazilian software developers.)

Table 17: Messages to Lua-L per top-level domain

	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008
all TLDs	341	475	683	1886	3748	4164	5688	5803	6399	8396	7736	5918
com	66	124	187	491	1001	985	1564	1750	1417	2102	1472	1172
br	64	90	213	540	728	797	1009	878	949	1104	988	551
gmail.com								249	1209	1684	1990	2062
net	30	80	83	114	396	366	357	632	923	1097	1007	496
org		1		80	312	346	336	387	412	456	435	400
de	8	5	20	167	250	224	313	347	373	289	273	193
uk	31	19	16	111	379	274	211	150	119	188	172	125
edu	48	46	37	65	205	204	124	51	67	171	115	141
yahoo.com				19	54	98	89	100	46	75	171	76
fi	3			3	2	34	152	377	140	11	4	
nl			22	12	10	16	192	42	91	82	79	63
hotmail.com	2	2	3	5	46	81	69	52	56	123	106	22
au				18	13	40	162	6	14	68	31	138
fr	4	1	32	9	25	78	68	15	29	93	58	36
it			3	40	23	63	63	66	19	40	67	53

ca	3	4		30	20	45	160	57	33	32	28	8
ru	3	1			11	33	87	74	57	33	10	
se	5	1	2	4	21	10	38	19	24	95	36	12
ua				16	23	7	29	13	2	12	124	32
il								4	39	55	74	70
be		1	1			125	66	8		19	18	3
es				1	12	65	30	38	12	14	32	8
pe					9		128	71				
cx									7	5	98	38
at					2	1	4	9	19	15	48	26
no	1	10	4	31	30	2	13			1	10	4
pl	1		5	2	2	16	20	29	8	4	6	6
local	25	53		2	5	9	4					
ar						1	21	43	16	17		
cz				13	7	2			9	31	16	9
gr				1			46	32		2		
cn							1	2		60	17	
za					1	2	8	25	28	8	1	4
my							5	6	14	39		
info							5	6	1	36	8	7
uy						8	3	9	11	23	3	
dk		1	2	1		14	5	8	12	1	8	2
jp		2	1	9	3	6	1	6	6		1	13
cc										37	6	1
nz				3	2		7	3	2	22	1	
gov			5		2		13			18	1	
fm							3		1	3	13	15
mx	12	5	3	7	5		1					
ch			1	4	1	2	2	6		3	10	1
pt						2	10	15			1	
nu					2				24			
ro					1	17	5	2				
yu								2	15	1		
hu						1	10		3	2		
bg				1	1	3	2			2		
is					1						1	6
si							8					
kr						6	1					
bm								2	4			
sk					1		3	1			1	
cl						1			3			
er										4		
ph					1		3					
ws									4			
name											3	
sg					3							

hr				3								
cu											3	
st										1	1	1
us						1				2		
id								3				
1	2											
tr						2						
lt							2					
in									2			
tw			2									
other	1					1	2	1	1			

Table 18: Number of senders (unique addresses) per top-level domain (TLD).

	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008
all TLDs	57	78	93	190	349	391	510	479	478	629	614	522
com	13	33	26	64	122	113	150	148	120	154	144	127
br	15	13	19	18	42	44	49	37	28	29	26	22
gmail.com				1	4	7	7	23	84	155	173	169
net	4	4	9	21	38	40	57	49	52	53	39	44
de	3	3	7	12	18	22	32	34	28	28	38	32
org		1		7	11	11	20	29	18	27	36	29
uk	3	4	5	12	14	25	25	15	16	19	21	12
edu	8	8	7	12	12	16	18	13	16	15	13	12
fi	1			2	2	2	5	4	4	4	4	1
pe					1	1	1	1	1	1	1	
hotmail.com	1	1	1	3	11	14	20	16	15	22	15	7
yahoo.com				4	10	19	16	21	12	22	19	9
fr	1	1	1	6	6	7	10	6	8	9	12	6
au				2	7	4	6	2	5	4	4	6
it			2	2	3	6	5	5	5	6	4	5
nl			1	1	3	4	6	4	4	7	4	8
ru	1	1			6	4	12	10	5	5	2	
ca	1	2		4	6	8	12	5	10	11	4	3
se	2	1	1	1	5	6	5	7	3	8	5	4
ua				1	1	2	2	2	1	2	3	1
ar						2	2	2	2	3	1	1
be		1	1			3	2	3		3	2	2
il								1	1	1	1	1
es				1	4	3	3	4	2	2	2	
cx									1	3	2	1
at			1	1	2	1	2	5	4	4	5	3
no	1	2	2	2	4	1	2			1	1	1

pl	1		3	1	1	3	8	7	4	1	2	1
cz				2	2	2			1	3	3	3
gr				1			1	3		1		
cn							1	1		2	4	
za					1	2	2	3	3	3	1	2
my							1	1	1	1		
info							1	2	1	3	2	2
uy						1	1	2	2	2	2	
dk		1	1	1		2	3	1	2	1	1	1
bg			1	1	1	2	2	1	1	2	1	
jp		1	1	1	1	2	1	2	2		1	2
cc										1	1	1
gov			1		2	1	3			1	1	
nz				1	1		1	1	1	2	1	
fm							1		1	1	2	1
mx	1	1	1	1	1		1					
nu					1				3			
ch			1	2	1	1	1	2		2	4	1
ro					1	1	1	2				
yu								2	4	1		
hu						1	2		1	1		
si							3					
is					1						1	1
kr						2	1					
sk					1		1	1			1	
bm								1	1			
pt						1	1				1	
cl						1			1			
ws									1			
ph					1		1					
er										1		
cu											1	
st										1	1	1
hr				2								
name											2	
all other	1		1		1	4	3		3	1		