

# 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?<sup>1</sup>

I hope to show in this dissertation that we have much to gain from looking at 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.<sup>2</sup> Yet there seem to be

- 
- 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 reasons (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."
  - 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

“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, 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.<sup>3</sup> 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 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

---

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.)

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.

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).<sup>4</sup>

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 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.<sup>5</sup> 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,"<sup>6</sup> 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 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

---

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.

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.

different from that of their American counterparts. “A server is a server,” they say.<sup>7</sup> 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 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

---

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

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.<sup>8</sup>

### *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.”<sup>9</sup> 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

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?

local projects the attention that such projects may 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.<sup>10</sup> 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 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

---

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).

quantitative analysis, believing that such analysis would likely be misleading.<sup>11</sup> 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 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.<sup>12</sup> 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 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.<sup>13</sup> Second, and more

---

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).

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.

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 than they are while applying for a job and less fact-checking is often needed.

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 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.<sup>14</sup>

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.<sup>15</sup> 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 engagement with Kepler brought me close to that point, requiring that I draw clear lines.<sup>16</sup> It is during that time, I believe, that I gained some of the most important insights about the project that I was studying.

---

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.

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.

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.

## 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.<sup>17</sup> 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.<sup>18</sup> (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.<sup>19</sup> To compensate, I present such chronology in this section.

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

---

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.

reader, but cannot necessarily be explained from the point of view of what would have made most sense to do.<sup>20</sup>

### *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.”<sup>21</sup> 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 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 to 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

---

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.

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.<sup>22</sup>

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 trained to do so.<sup>23</sup> 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,<sup>24</sup> 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

---

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).

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).

public and private sector. I talked to people of different ages,<sup>25</sup> and made an attempt to include a larger number of women than I believe a random sample would have drawn.

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.<sup>26</sup> 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 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,<sup>27</sup> 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

---

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

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.

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.

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 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.<sup>28</sup>)

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

---

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 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.

technology” and start looking at how the global nature of technology is constructed through local work.<sup>29</sup>

### *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.<sup>30</sup> 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 Ierusalimsky’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.)

---

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.

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

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 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.<sup>31</sup> 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 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

---

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).

“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).<sup>32</sup> 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.

## 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

---

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.

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 Ierusalimsky'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.