SOCIOLOGY/SCIENCE, TECHNOLOGY, AND SOCIETY/GLOBALIZATION

"By examining software development in the 'wrong place' of Rio de Janeiro, Yuri Takhteyev shows us with vivid accounts and clear narrative how individuals who work far from the geographic hubs of their field create local connections and shape local environments even as they embrace global culture and pursue global dreams for themselves and their locations. The concept of a 'wrong place' proves an immediately beguiling and completely original approach for understanding work in the global setting; Takhteyev's choice of Rio, in particular, is nothing short of brilliant."

Diane Bailey, School of Information, University of Texas at Austin

"*Coding Places* opens the black box of 'globalization' to show us the pieces involved in that process—people, technical objects, government agencies, universities, businesses—in intimate detail: how they work, what they need to survive, what they furnish to others, the network of their connections, conflicts, and accommodations. We see the whole machine in operation: how the many possible inputs generate a variety of outputs, technically and organizationally. And we learn a way of thinking that we can apply to the arts, science, or business, to any kind of activity with worldwide extension and ramifications. It does all this with a depth of vision and a clarity in telling the story seldom found in the social sciences."

Howard S. Becker, author of *Outsiders* and *Art Worlds*

"Software development is no longer limited geographically but is expanding to different regions of the world. Yuri Takhteyev has produced an insightful work that provides a critical account of software developers and their role in the global knowledge economy. This is a fascinating story of knowledge workers in a region that has the potential to become the next Silicon Valley."

Alladi Venkatesh, Professor and Associate Director, Center for Research on Information Technology, University of California, Irvine

9 780262 018074 90000

The MIT Press
Massachusetts Institute of Technology
Cambridge, Massachusetts 02142
http://mitpress.mit.edu

978-0-262-01807-4

CODING PLACES TAKHTEYEV

CODING PLACES

Software Practice in a South American City

Yuri Takhteyev

0 The Wrong Place

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 in front of 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, at nearly 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 should suddenly become far from flat. Saying that I had come to Brazil to study *software developers* was a sure way to raise eyebrows further.[1]

I will try to show in this book that we have much to gain from looking at software development in this somewhat unlikely place, and more generally, from looking at high-tech work in "wrong" places. By doing so, we can learn a lot about *place* and its persisting importance in today's "knowledge economy." For over a decade, popular authors have declared that place will soon become unimportant for human activities, as people increasingly gain the ability to communicate and collaborate over distance (Cairncross 1997; Friedman 2006). In the age of the Internet, they have argued, where you are does not matter. Others have countered such claims, pointing out that the world might actually be becoming more "spiky," with a small number of places *growing* in importance as centers of global activities (Florida 2008). Picking the city to live and work in, they say, may be your life's most important decision. If you are in the wrong place, pack your bags quickly and move! And some people do exactly that. For decades, places like Silicon Valley have attracted (and continue to attract) people from all over the world. Eighteen years ago, I myself left a provincial Russian city for Palo Alto. Most people stay close to where they were born, however. This book is about those people, the work they do, and their role in globalization.

My story and analysis challenge both views outlined earlier. I argue that we should neither declare "the death of distance" nor fix our gaze on a handful of "spikes." Instead, we must look at globalization as an active process arising from the combined efforts of many people around the world working daily to defy space, building individual connections to remote places in pursuit of global dreams. To understand globalization we must look closely at such people: at their goals, their struggles, their failures, and their successes. We must pay attention to how their efforts reduce or increase differences between places. And we must look in the wrong places.

Practice and Place

The book looks at people who inhabit simultaneously two different contexts. One of those contexts is defined geographically—a metropolitan area in southeastern Brazil, consisting of the city of Rio de Janeiro that is home to around six million people known as *Cariocas*, and the adjacent municipalities inhabited by an equal number of *Fluminenses*, many of whom commute to Rio de Janeiro for work. The other context is an instance of what I call *worlds of practice*—systems of activities comprised of people, ideas, and material objects, linked simultaneously by shared meanings and joint projects. Such worlds vary in scale, but many of them are global, connecting people and objects spread around the planet. The world of software development is global in this sense, inhabited by around ten million people who are spread far and wide. I argue in this book that global worlds of practice are the key constitutive elements of globalization. In other words, to understand globalization we must look at not just the technologies that enable global communication, nor the structures of global governance. Rather, we must investigate the global "worlds" that form around specific systems of human activity, noting how globalization projects occurring within such systems reinforce each other and produce the overall experience of globalization.

The world of software development makes an interesting context for a study of globalization because it exemplifies its paradoxes like no other field. Software development is often seen as a quintessential example of "knowledge work," a global profession, freed from the constraints of geography by the immaterial nature of its inputs and outputs. Whereas traditional industries convert material inputs into material outputs, and moving those inputs and outputs costs money, "knowledge work" focuses on 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 a privileged few, in practice it is often seemingly rendered free for all by the collective generosity of "communities of geeks," which Friedman (2006) sees as an example of a broader "uploading" of knowledge.

Given the abundance of uploaded knowledge, engaging in software production seemingly requires little more than a computer, stable electricity, and Internet access—all of which are available in places like Rio de Janeiro even to the relatively poor. Armed with those tools, developers can access vast repositories of code and documentation from across the globe. Brazilian developers sometimes spare no words when describing the significance of the Internet to their work. They speak of it as "the world's greatest library," full of "all the imaginable and unimaginable resources." Developers can use code and documentation found on the Internet to build their own solutions. They can then distribute the products of their labor to people around the world, again using the Internet. Occasionally we read news stories that seem to illustrate the ease of this scenario. For example, in 2009 a seventeen-year-old Moscow high school student built Chatroulette—a video chat system that soon had over a million users from around the globe and was discussed in the news all over the planet.

Such stories, however, must not distract us from another notable feature of the world of software: its stark and persistent centralization. Over the last several decades, the world of software has revolved around a handful of places. One of those places—Silicon Valley—has in fact become a textbook example for illustrating the idea of regional clustering of industry. In addition to being home to a large number of software practitioners, Silicon Valley and the greater San Francisco Bay Area also serve as a base for some of the world's largest and most successful IT companies that control the work of developers around the world. Together, market capitalization of IT companies headquartered around San Francisco comprises over a third of the world's total (see chapter 4).

This concentration of valuation is indicative of the difference in the *kind* of software work that gets done in different places and the geography of control over software work. 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, which would, if successful, bring their companies big rewards. Such rewards can be both financial and symbolic: the successes of Oracles, Apples, and Googles make up a good part of the global software lore. Software developers working outside such major clusters recognize the preeminence of remote centers. Stories such as those of Chatroulette often have a little-noted ending: the developers moving to San Francisco Bay Area or selling their venture to a company based there.

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 most likely could only have arisen—in a highly predictable place, biking distance to Silicon Valley's Sand Hill Road.

By most counts, Rio de Janeiro is a peripheral place in the world of software. In terms of sheer numbers, Rio de Janeiro likely has about one-tenth the number of programmers of the San Francisco Bay Area; in terms of IT valuation, the difference between the two regions likely approaches a factor of *one thousand* (see chapter 4). Developers who work in Rio usually dedicate their efforts to the smaller problems faced by local organizations. The most successful address the needs of Brazil's national market (though many usually find that such work is better done elsewhere, in the larger São Paulo). "This is not Silicon Valley," Rio developers often explain when talking about the possibility of taking on more ambitious projects.

Yet it is precisely this peripheral position in the remarkably centralized world of software development that makes Rio an interesting place for looking at knowledge work. After all, while the software developers working in Rio are fewer than those in Silicon Valley, the overwhelming majority of people who write software do so in places that are more similar to Rio than to Silicon Valley.[2] To understand the truly exceptional position of centers such as Silicon Valley, perhaps it helps to spend some time contemplating the periphery. What do software developers *do* in such places? Why do they do it? Answering such questions will help us better understand the nature of ties that bind together the world of software and today's global society.

To make sense of the paradox between software's seeming independence from geography and the centralization of its production, we could try to understand why software development remains so concentrated in the era of unrestricted knowledge flows—a popular road that seems to almost inevitably lead one to ask what is *wrong* with all the places that fail to produce a thriving software industry. I touch upon this question at several points in this book. For most of it, however, I take a different approach. Instead of assuming that technical knowledge is naturally fluid and trying to understand what barriers keep software development so concentrated, I take the concentration as a given and seek to understand how the practice of software development moves in space *at all*, investigating the work that is needed to establish this practice in new places. How the seeming universality is achieved *in spite* of this geographic concentration then becomes one of the key questions.

In doing so, I put aside the term "knowledge" for the sake of another one: "practice." To understand how knowledge comes to new places, 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 can travel (Latour 1987). 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 rich body of social theory from which I borrow the concept of "practice," and in part because I feel that the tendency to think of "knowledge" as something akin to the content of computer files is so strong today that I cannot expect the reader to ever fully leave behind this unfortunate metaphor.

The concept of "practice" provides us with a useful analytic layer between the more abstract, propositional notions of knowledge and the messy details of daily life. As I explain in more detail in the next chapter, I understand "practice" as a system of activities, a collective way of doing certain things, or a system of "doings and sayings" (Schatzki 1996). A practice maintains continuity through a mutually sustaining relationship between patterns of interactions, material resources, 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. I also look at how such doing interacts on the one hand with ideas and discourse, and on the other hand with the material elements of the practice. This nexus of relations creates a *context* for individual actions, a context that individuals can "inhabit" in ways that can be likened to how they inhabit physical places, and to which they can have commitments—commitments that must be balanced with those to the local place and the national community. Such contexts are bounded and often named. The developers sometimes talk about being in "the world of software." For this reason, I describe such systems of activities as *worlds* of practice.[3]

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, since it helps us recognize the many different pieces that would need to be put together to re-create the practice in a new place. For someone who adopts this perspective, 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.

My discussion of practice in place focuses on several themes. The first is *the process of disembedding* and *reembedding* (Giddens 1991) involved in its reproduction across space: people engaged in a practice that is based somewhere else often have to reassemble the practice around imported elements, substituting for missing pieces what happens to be available. (And if they want to get involved more centrally, i.e., *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 reproduction of the American software practice and frame my observations as a particular moment in the history of this practice—a moment when many elements have already been brought in and reassembled (hence the need to look at history in chapter 4), while others are still missing. I also look at this reproduction as one of many parallel efforts to re-create 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 globalizing world of the practice. (Those engaged with the practice at its centers may face this issue as well, but the gap between the two worlds 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 practice*: local practitioners must often decide whether to cast their lot with their local colleagues or focus on their individual connections to 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. Finally, I stress the importance of paying attention to *actors' reflexive understanding of the world*, the possible futures they can imagine individually and collectively, and the factors that influence this imagination (Giddens 1979; Appadurai 1996). Together those themes provide us with a view of globalization that highlights individual agency of peripheral actors, situating their actions in the context of cultural and economic structures, while also showing how their individual attempts to engage in global systems of activities add up, collectively and over time, to create the seeming universality of global practice.

By bringing to light the work that peripheral practitioners must do to give software development its seeming universality, I hope to offer them the credit 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" (e.g., Graham 2006). While smart people are undoubtedly 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 the re-creation 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 intend to show how such peripheral work contributes to the continued dominance of remote centers. Like many other knowledge products, software production is characterized by strong network effects: software that is used becomes more useful and will often gain in popularity because of its popularity. This is often especially true for open source software (which I discuss in the next section), where products that are widely used often actually become *better* as they attract more contributions. By fixing their gaze solidly on foreign technology and investing efforts into making it work locally, peripheral developers often deny to local projects the attention that such projects may need. Such lack of attention and, more important, 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 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 invite policy makers to follow me on a visit to 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, an open source software project described in chapter 8, read together with the two alternatives to the approach Kepler exemplifies (chapters 5–7), will be useful for thinking about innovation policy.

Peculiarities

Understanding the way a universal practice is made to work in a concrete place requires looking at the many peculiarities of that 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. While looking at one specific place, however, I seek to show relations I believe exemplify the patterns we can find in many other places. Every place has a history and every place has a local context. In every place concrete work must be done to turn abstract knowledge into a living practice.

While I believe the patterns I explore in this book 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 has yet to take root would help us see the earliest steps in this process, but would shorten the history available to exploration, leaving us to imagine all sorts of possible scenarios for the future. 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 (e.g., 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 in this history far enough back to allow for critical analysis.

In addition to the peculiarities of Rio de Janeiro, two other aspects of the book may strike the readers as unusual and thus call for a brief introduction.

Free / Open Source Software

The book focuses disproportionately on a specific form of software practice known as "open source" or "free" software development. Although those two terms vary substantially in connotation, both refer to software that is distributed in a manner that allows the recipient to modify it, and then redistribute it to others without paying royalties to the original author. While such distribution of software has been common since the earliest days of software, it has come to particular prominence since the development of Linux, an open source operating system, in the 1990s.[4] In recent years, the development of free / open source software has attracted substantial attention from social scientists, including sociologists and anthropologists who have often looked at it as a political and cultural movement (e.g., Kelty 2008) and economists who have looked at efficiency gains associated with this form of software production. In this book, I look at cases of open source software development through the lens of *practice*, highlighting the interrelations between culture and material production, and positioning open source within the context of the global world of software practice.

Open source software development presents in perhaps the clearest form the paradox between placelessness and centralization described earlier. Open source communities are intentionally open, and the apparent generosity of those "communities of geeks" provides much of the motivation for Friedman's discussion of "uploading" as one of the key factors contributing to 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 global culture of such communities is based largely on the "hacking" culture that originally developed in American universities.[5] Their practices are today 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 open source projects involves a complex negotiation of culture, language, and geography, and is often *harder* than engaging in other forms of software practice, since 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 a new opportunity—and a challenge—to participate 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 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 we could call "synchronization work." Looking at how Rio developers respond to this challenge may therefore help us understand how people engaged in other worlds of practice respond to changes that take place in those worlds.

Lua

Several chapters of the book look closely at a particular open source project that would be unusual by most measures: Lua, a programming language developed in Rio de Janeiro that has recently gained substantial global popularity around the world—in particular, among software companies based in California. For example, Lua was used extensively in *World of Warcraft*, a networked computer game played by over ten million people (a number sufficiently high to secure an entry in the Guinness World Records), and more recently in *Angry Birds*, a game that was downloaded over one hundred million times in its first fifteen months. Lua has also been used in products made by Google, Adobe, Microsoft, Verizon, Cisco, and other technology companies.

Lua's global success is surprising, not the least to those people in Rio de Janeiro who are familiar with the scale of its use abroad. It is particularly stunning when we consider the powerful network effects that ensure that the number of programming languages in common use remains quite small. Lua is the only entrant into this exclusive club from a developing country.

Lua's position in Brazil, however, presents us with an even larger puzzle. Almost no local companies make use of Lua in their products. Lua's large and active community interacts primarily in English. Software developers in Rio de Janeiro who wish to learn Lua can do so using a book written by one of the authors of the language (a professor at a local university), but they will need to read the book in English, because no Portuguese

translation of the book is available. Unless they know the author personally, they will likely also need to order the book from Amazon.com and have it shipped from the United States, since no Brazilian bookstores carry it on their shelves. Lua's global success has so far done little to rescue Rio de Janeiro from its position as a "wrong place" for developing software. Programming in Lua has just become another activity that is better done in Silicon Valley.

I present Lua as a case of a particular strategy of engagement with global technology: a focus on global connections, in the name of which local linkages may have to be sacrificed. I show the reasons for such disengagement from the local context, as well as some of the efforts to reconnect Lua to Rio. I discuss the strengths and weaknesses of this approach, presenting a range of perspectives on Lua's past, present, and future. I contrast this case with two others: the localization of global technology by a successful IT firm in Rio (chapter 5) and an attempt to bridge the gap undertaken by a government-funded open source project aiming to make use of Lua locally (chapter 8).

The Project

This book is based on an ethnographic project—an attempt to understand the experience of a group of people through an extended engagement with them. In my case, this meant a combination of over one hundred interviews, extended presence in places where software work was being done, and at times active engagement in the members' projects.[6] 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 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. They normally do so out of the 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 readers better understand what was observed by being told who did the observing and how. It also helps the ethnographers consider their own biases, as it encourages them to think more closely (and explain to their readers) about their own role in the events.[7] (It also, as Van Maanen points out, helps establish the ethnographers' authority by showing that they have gone to places where the readers have not been.) Though I include myself in the account whenever appropriate, I avoid the extremes of confessional ethnography, finding it potentially distracting from the larger points that I want to make. In particular, the order of the chapters reflects the theoretical logic of the book rather than the chronology of my fieldwork. To compensate, I present a brief "confession" in this section.

In the summer of 2003, after spending three years working as a software developer in Mountain View, California, the heartland of the region known worldwide as "Silicon Valley" (but referred to locally as just "South Bay" or "the peninsula") and before starting my PhD program at Berkeley, forty minutes away by car, I spent a month in my hometown in Vladivostok, Russia, on the other side of the Pacific. (As I learned a few years later, this city is known to many Brazilians primarily as a base for attacking Alaska in *War*, a board game based on the American *Risk*.) 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 seemed in some ways quite provincial, yet at the same time was 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—in particular, the United States, which seemed to be visible from Vladivostok in the way no country is from California. I started developing an interest in understanding how people who work in "peripheral" places maintain ties to the places they consider more "central" to their field. Trips to Brazil and Finland the following year solidified this interest.

By September 2004 I had decided to focus my dissertation research on software developers in Brazil and their access to software knowledge from the foreign centers of software practice. At the time, 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 places such as Silicon Valley. Both assumptions 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 thinking and to which I will turn shortly.

After another year at Berkeley, spent learning Portuguese and reading social theory and economics, I arrived in Rio de Janeiro in June 2005 for a six-month stay and started building my sample of "software professionals." I defined the term loosely, including in it people who were trained to write

software, regardless of whether 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. (I later switched to the term "software developers" to avoid the presupposition that software developers are "professionals.")

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" they hope will challenge their preliminary assumptions and lead to further development of the theory (Glaser and Strauss [1967] 1999). Such a sampling technique often aims to increase the diversity of the sample, in order to compensate for its small size. A common way of building such a sample is by asking interviewees to recommend additional people who could be interviewed (a "snowball" technique), either specifically asking for people matching certain characteristics or selecting them from among the nominees. In my case, I attempted to include among my interviewees every type of software developer 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-geeks" until I interviewed two of the authors of the Lua programming language, to which I dedicate chapters 6 and 7. 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, and I made an attempt to include women in what otherwise was turning out to be a heavily male-biased sample.

I came to Rio with basic 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 better 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. This awkward start and the subsequent change in the language of the interviews turned out to be a blessing in disguise. My own struggles with Portuguese made me somewhat more sensitive to my interviewee's struggles with English (and more appreciative of their successes with it), while alternating between English and Portuguese exposed me to the different discourses invited by each language.

During my second month in Rio, I learned a methodological lesson that greatly affected the rest of my project. Most developers that I had interviewed up to that point assured me that they never discussed 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, was 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 did 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 "ethnomethods"—developers I interviewed in Brazil became a great source of explicit advice on how to interview other Brazilians.[8])

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 took this criticism seriously. Though my investigation into Brazil's history and its relation to the current practices did not come together until after my return, I did use my visit to discuss my interests with a number of scholars affiliated with the Federal University of Rio de Janeiro, including some who observed firsthand Brazil's technology policy in the 1970s and 1980s—or even helped shape it. Those conversations 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. This 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.[9]

In January 2006 I returned to Berkeley to analyze my data and to prepare for my qualifying exam before another five-month trip to Rio. 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 the 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." The project aimed to build a web development platform based on Lua, the programming language developed in Rio de Janeiro that I referred to earlier.[10] 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 them, I found myself puzzled and surprised at every step. I was also starting to get a new 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 web applications for local clients, using Java, a popular programming language and a software development platform by Sun Microsystems, a California company.

In February I joined the Lua mailing list, spent some time reading its archives and did six interviews with Lua users in California. I then went to Rio to start a new round of fieldwork, having already secured not only Rodrigo's invitation to study Kepler and the Lua team'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: Rodrigo was about to try a new approach to the project that would aim to "open" it in order to draw in a larger number of remote participants.

Despite having seemingly open access to the project and getting a chance to meet most of the participants early on, I soon confirmed my suspicions that mere physical observation does not go very far when studying 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, e.g., with instant messaging, a more private medium). Without literally looking at the developers' monitors over their shoulders, both at work and at home, and keeping track of their solitary work, private emails, and instant messenger 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.[11]

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 began to doubt whether anything was actually even happening. I decided to start helping Rodrigo with the project's web site, but felt that even this was giving me too secondhand of a view. Our discussions of the web site, however, soon arrived at the conclusion that we wanted to run it as a wiki—a web site that allows visitors to make changes to the content. 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. Even though 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—and immediate demand for improvements. As I started spending time making changes, 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 an answer to many of the problems of studying software work that troubled me at first. While no method can reconstruct the project in its entirety, active *participant* observation provided me with a partial solution: a situated and integrated picture that weaved together *some* private emails and instant messenger conversations, *some* late night conversations over pizza, and quite a few hours alone in front of the monitor making sense of debug traces.[12]

Such engagement also created a number of challenges. While my own technical background proved a blessing because it allowed me to get engaged, I soon came to face the challenge of getting involved without "going native." A certain degree of resocialization 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 limit time available for reflection. 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 in a "serious" manner, 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. This led to a struggle to maintain balance between my life as an ethnographer and my life as a Kepler developer, but in the end I felt it was worth it.

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. Since leaving Brazil in August 2007, I have stayed involved with the project, following it through its ups and downs, finding it impossible to disengage from it even after Rodrigo himself decided to move onto other things.

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 and also substantially less participatory, because I continued to be involved with Kepler and conduct interviews related to both 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 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 a more typical, work environment.

I returned to the United States in August 2007, bringing with me 150,000 words of field notes, not counting notes and recordings for over a hundred interviews.[13] Over the course of the following years, I sifted 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

This introduction is followed by eight chapters and a conclusion. Chapter 1 lays a theoretical foundation for the book, outlining with more precision analytical concepts employed later in the book. (Readers who would prefer to avoid a strong dose of social theory upfront may consider starting with chapter 2, perhaps returning to chapter 1 later.) I argue that to understand Rio developers' engagement with "software development," we must conceptualize software development as "a world of practice." I show how Giddens's theory of structuration can be extended to analyze the spatial expansion of worlds of practice and how such analysis can be applied specifically to software development. In chapter 2, I add some ethnographic flesh to the theoretical skeleton developed in chapter 1, taking a slice through many of the contexts explored in the book while focusing on a particular theme: the use of English and Portuguese by Rio software developers, which illustrates the developers' position between the world of software and the local world of Rio de Janeiro.

Chapter 3 explores developers' early steps toward the software profession, looking at biographies of a small number of developers. I first show the adolescents' cultural entry into the world of computer "nerds" that often precedes the engagement with labor markets. I then turn to the transition from the cultural to the economic engagement with the world of software practice. Throughout this chapter, I look at how neophyte software practitioners build both local and global connections, entering the world of software simultaneously *from* and *in* a peripheral place. Chapter 4 switches from the situated perspectives of Rio's young nerds to a broader look at the world of software development as a whole, outlining its history and looking at its current geographic organization, pointing out a strong asymmetry between the kind of work that is done at the "central" and that which is done at the "peripheral" sites. I then turn to the history of computing in Brazil to explain the local politico-economic structures encountered by the young nerds seeking to convert their passion for global software into a local career. Despite Rio's relative isolation from the larger centers of software production, a myriad of ties link the city to other parts of that world. This chapter shows how some of those ties were built historically and the work that went into their construction.

Chapter 5 starts a discussion of the opportunities faced by Rio software developers today by looking at Alta—in some ways a typical software company in Rio de Janeiro, engaged in building local applications using software produced in California. Chapter 6 turns to a very different kind of project: Lua, the globally successful programming language developed in Rio de Janeiro that I introduced earlier. I first discuss Lua's use abroad, drawing on my interviews with users of Lua in California. I then present the history of Lua, focusing on the ways in which Lua had to gradually separate itself from the local context to achieve its global success. Chapter 7 turns

to Lua's relationship to Rio de Janeiro and Brazil in the recent years. I show the continued tensions between Lua's adoption in Brazil and its status as a global programming language. I present a number of conflicting—and often *conflicted*—opinions on the possibility of viewing Lua as a "patriotic" artifact and a potential vehicle for local development. Chapter 8 looks at Kepler, a project that aims to bridge the two different worlds: the mostly local world occupied by Alta and the mostly global community inhabited by Lua. The themes of the book are summed up in chapter 9.