

3.2 Porting Lua

In 1993 a group of computer scientists working in a university in Rio de Janeiro developed a simple programming language called “Lua” to serve the needs of a Brazilian company based in the same city. Fifteen years later Lua is often ranked among twenty of world’s most popular programming languages²⁶³ (out of thousands) and has a user community spanning all five continents. While Lua has brought its authors rather modest financial rewards (it is distributed for free and brings little consulting revenue) its use in popular software such as Adobe Lightroom and World of Warcraft has made it in some ways one of the most successful software products ever developed in Latin America.

American users of Lua often credit it with being highly *portable*—Lua can run on many different computing platforms. While increased portability in this narrow technical sense is an important part of Lua’s story, I focus here on a different kind of “portability”: Lua’s gradual transformation from a highly local project to an international programming language that betrays little connection to the city where it was developed and where it is still based. I organize my discussion around Giddens’s (1990) notion of “disembedding”—the “lifting out” of social (or in our case socio-technical) relations from their local context, which then makes them mobile across time and space. Following Lua’s transition from a highly embedded project, developed as a solution for a specific set of problems, entangled in a web of local relations, goals and commitments and reliant on what we may call “tacit knowledge,” to an international programming language we observe the different mechanisms that enabled and facilitated this disembedding.

As we will see, Lua’s international success was not planned in advance. To a large extent the disembedding of Lua simply “happened,” in many ways without a conscious intention by its authors. It happened due to numerous decisions that most participants saw as quite natural. In some of the cases, acting otherwise, for example using Portuguese words as Lua’s keywords, would be nothing short of “ridiculous” according to some of my interviewees. It is important, however, to look closely at such “obvious” decisions. It is by understanding how such decisions come to be obvious, and why they are obvious to some and not others, that we can come to see the geographic logic woven into the professional culture of software development.

The story of disembedding told in this chapter complements the investigation of local re-assembly of a foreign practice presented in chapters 2.2 (the larger history of informatics in Brazil) and 3.1 (a look at a particular company dedicated to bringing foreign technology to local clients). After decades of work that helped establish the foundation of software practice in Brazil, the context was created that made it possible for some of the practitioners to engage in one of the most central roles in the world of software: authoring a programming language. This replicated context, however, is not identical to the original and is characterized by a distinct pattern of

263 For instance, Lua was ranked between fifteenth and twentieth in TIOBE TPCI index for most of 2007 and 2008, dipping to the twenty-second position in December of 2008.

connections. Brazilian academic computer science has strong connections to foreign computer science, which gave Lua's authors a good start in cultural capital. At the same time, much unlike American computer science, it is relatively isolated from both local and foreign computer industry and instead exists in a somewhat of an enclave, which makes the experience of Lua's authors quite different from those working for Alta (chapter 3.1).

I turn to the limitations of this process of disembedding in the next chapter, looking at Lua's changing relationship with the university, city and country where it was born. Fifteen years into its history, Lua appears to be less known in Brazil than in the United States. Few Brazilian companies use Lua in their products. Lua's lively online community interacts exclusively in English. *Programming in Lua*, written by Lua's author Roberto Ierusalimsky in English, is yet to be translated into Portuguese. (German, Korean and Mandarin translations are available as of late 2008. With the exception of one book written in Japanese, all other books dedicated to Lua or covering it in substantial depth are available only in English.²⁶⁴) In addition to having to learn Lua from a book written in a foreign language, a potential Brazilian Lua programmer would likely have to order the book online through Amazon.com and have it shipped from the United States — *Programming in Lua* is not sold in Brazil, except in the author's office. In the end of that chapter, I turn to the difficult question of whether Lua's success can be of any benefit to the city and country where it was developed.

The two chapters are based on around forty interviews with people involved with Lua or related projects, complemented by my analysis of the archives of the Lua mailing list (Lua-L), Lua publications and the historic Lua code. Eight of the interviewees were Lua users residing outside Brazil, five of whom were interviewed in person in California. Most of the remaining participants were interviewed in person in Rio de Janeiro. (One was interviewed by phone.) As in other chapters, I try to minimize the number of people who I quote to make it easier for the reader to keep track of the characters.

Those two chapters present a largely self-contained account of Lua's history, while avoiding repeating material presented in earlier chapters (in particular in chapters 1.2 and 3.1). Readers interested in additional historical details can find them in chapter 1.2, while those interested in additional perspectives on Lua's place in Rio de Janeiro today should look at chapter 3.1 (as well as chapter 3.4).

Choosing Lua

“Craig” is an engineer at a small startup in California developing an online computer game. Like most other users of Lua whom I interviewed in California, Craig encountered Lua

264 Jung & Brown (2007) is another introductory text in English, while Ueno (2007) provides an introduction to Lua in Japanese. Schuytema & Manyen (2005) focuses on use of Lua for computer game development. Gilbert & Whitehead (2007) and Whitehead et al. (2008) discuss using Lua to customize World of Warcraft. Varanese (2002), Gutschmidt (2003), Buckland (2004), and Dickheiser (2006) discuss computer game development more broadly, but dedicate a substantial attention to Lua.

online, while searching for a scripting language to embed²⁶⁵ in his application, coming across Lua “through [online] forums, googling and searching for ideas.” None of the people he knew personally had heard of Lua before. Craig cites Lua’s small size and simplicity as the reasons for choosing it over a “more mature” language such as Python. He was particularly concerned with security of his application and felt that “a sort of more mature language like Python” was too complex for him to be sure that his application would be safe from malicious hackers. Craig notes to an important weakness of Lua—the relative scarcity of libraries, a hurdle faced by all new languages. This factor, however, was of little relevance to him. “We were not building an application, like a web-server or something else that would need a whole bunch of specialized libraries,” says Craig. “We needed a language to primarily reference the objects inside our own system and to be able to script them.” (In chapter 3.4 we will look at Kepler—a project aiming to turn Lua into a platform for web development, exactly the thing Craig thinks Lua would be least suitable for.)

Craig’s use of Lua is quite typical: most Lua users in the United States employ Lua for scripting applications written in C, a programming language developed in the early 1970s, which had come to dominate software development by the early 1990s. Over the last decade, many developers have moved to newer languages such as Java, Python or more recently Ruby, which make it easier to develop software quickly. C and its close relative C++ remain popular, however, as they often make it possible to write software that runs faster. While most of the new languages can in theory be combined with C in a single application, potentially allowing the developer to get the best of both worlds, such usage is often complicated and frowned upon. For example, the authors of Java, a highly popular programming language designed by Sun Microsystems, pursued a targeted campaign to eradicate such mixed applications, encouraging the programmers to write their code in “100% pure Java.” Lua on the other hand, presents itself as the language primarily designed for hybrid use.

Lua has become particularly popular in the development of computer games, where efficient use of computer hardware is often crucial and the developers frequently work closely with C modules for handling graphics. Lua allows such developers to use “easy” Lua for parts of the code that are likely to change often, while relying on the more efficient C for the tasks that are most likely to put strain on computer’s resources. Lua thus strives in a relatively small niche, where it has positioned itself as a complement to a well-accepted technology, offering certain unique features that shield it from devastating competition with the “more mature” languages such as Python. Developers like Craig note and appreciate Lua’s close fit for the needs of the computer graphics developments.

Like other interviewees, Craig found it quite easy to get started with Lua. My question about how he learned to use Lua and what kind of resources he used takes him by surprise. “I am sure I downloaded everything, ran the command line, found out how that works,” he says after a pause. Another interviewee “Steve,” a lead engineer for a team of software developers working for a large software company in California, reports similar ease, which he then contrasts with JavaScript, a programming language developed by Netscape that my interviewees often consider

265 Underlined terms and abbreviations are defined in the glossary, which also includes some terms not marked in the text.

as an alternative to Lua. He has not needed to look for people who understand Lua, says Steve. Had he decided to use JavaScript instead, Steve would have gone and talked to people in his company who had a JavaScript implementation. “But that’s because JavaScript is messy,” he explains.²⁶⁶ “The great thing about Lua is that you don’t need any of that.” Lua’s elegant simplicity makes its foreign origin irrelevant—Steve and Craig can use Lua even if nobody else in California does so.

Like Steve, Craig did not feel constrained by his lack of contact with other Lua users. Just as he was starting working with Lua in the summer of 2005, however, a Lua Workshop was organized at the Adobe office in San Jose, about twenty miles south of where Craig’s startup is located. (Adobe itself was extensively using Lua in one of its projects—Adobe Photoshop Lightroom—which was released two years later.) The talk included presentations by two of Lua’s authors and Craig attended a part of it to see if Lua was “serious”:

Craig: Well, my reason to go was to get some sense of how serious this is. To ask a few questions. To talk with some people about it. Curiosity. More and more now I am very confident about our choice but [unclear]. Who are the other people who use it and why do they use it? [...] I enjoyed it. I think I only came for half a day. I think it was a multi-day event. It was very nice.

Seeing live users of Lua helped Craig feel more confident about his choice. When I ask him whether we was looking to make contact with local users of Lua, however, Craig tells me this was not his objective:

Craig: I don’t know if I met any *local* users. I met *people*, I didn’t... My interest was not in establishing long-term relations with people there, so I didn’t find out where they lived or if they were local.

Yuri: Why?

Craig: Because I am busy. I am getting plenty of information and valuable help through the mailing list. So... That’s not as important.

Craig finds that he can easily use Lua in the absence of any local community.²⁶⁷

266 JavaScript is similarly designed for embedding and is widely used due to the fact that its implementation is included with all modern web-browsers. Unlike, Lua, however, JavaScript does not have a standard implementation. For a long time no free implementation of JavaScript was available at all. At the moment, several such implementations are available but the most popular of them (Mozilla’s Spidermonkey) is typically found by my interviewees to be overly complex. (Spidermonkey source code includes around 140,000 lines—eight times more than Lua’s 17,000. Lua’s source includes the code for an interactive shell and builds with a single command in just a few seconds.)

267 One might suspect that Craig relies on the community of Lua users inside his own startup. This is only true to an extent—while a small number of engineers in his company write Lua code, he is the only person involved with the more complicated task of linking Lua with C.

A Global Perspective

All American users of Lua to whom I talked say that Lua's Brazilian origin was nearly irrelevant to them. "I take a global perspective on those things," says Rich, a software contractor who uses Lua in his projects when he has a chance. Taking a global perspective on Lua luckily takes little effort.

Like most interviewed users, Craig cannot read Portuguese (though he says he speaks German fluently and could maybe "make out" text in French). This of course does not present a problem for his use of Lua, since the Lua community interacts in English:

Craig: I guess English language is the *lingua franca* for Lua as well. From what I can tell. I haven't seen any Brazilian, or Portuguese, emails coming up. So... So, I never had any... Or, I should say I've never had any concerns...

Emails in Portuguese *do* occasionally come up on the Lua mailing list, and are typically treated politely, usually receiving a reply in English (sometimes quoting the original question run through an online translator) and occasionally even in Portuguese. Displaying a global perspective simply requires treating such infrequent occurrences with humor. The discussion on Lua list show that some of the list members have certain curiosity about Lua's unusual origin and sometimes allow themselves friendly questions about Brazilian practices that they find surprising—for example the use of a pair of brackets to close email messages.²⁶⁸ Those who want, however, can simply ignore such cultural differences. Most of the translation work is already done by the Brazilian members of the list.

Perhaps the only way that Lua's foreign base entered into Craig's calculation, is because of the increased difficulty of understanding how "academic" Lua was. Craig knew that Lua was produced by university researchers and was worried that it was "an academic exercise":

Craig: My reaction was more... I don't know if it had to do with Brazil or not, my question was "Hmm..." Actually, I think at the same moment I realized that it sounded like it started out as an academic exercise, I think I read this there as well. And of course the question is: What are the primary interests of the caretakers of the language? Is it to satisfy their academic ambitions? And I don't mean that in... academic ambitions not as in a career but as in academia. Is it a language to highlight their work on the programming language theory? Or is it something for practical use?

Making this evaluation would have been somewhat easier in case of an American university, Craig explains, since he could meet the authors or would perhaps "know somebody who knows somebody who knows them":

268 On one occasion, a Brazilian member of the list was asked to explain his use of "[]," in the end of the message: "Does this mean something? Some people sign their mails 'thanks' or 'regards' or 'good day'. You're the first I've seen sign 'boxes'." ("[]" is used in Brazil as a shorthand for "abraços"—"hugs.")

Yuri: This question of whether it's an academic exercise, how does it relate to it being in Brazil?

Craig: Perhaps it is harder, it might be harder for me to determine than if it were for example at Stanford, then I could... A) It would be easier to meet the people or I would know somebody who knows somebody who knows them or something like that. So, in this context it is certainly more opaque.

While understanding whether Lua is “academic” takes Craig somewhat more effort, he ultimately finds other ways to understand the intentions of Lua’s authors: reading about the language, using it, later attending the workshop, learning that Lua was used in World of Warcraft, one of the most successful games at the time.

Other interviewees similarly express the desire to understand the people behind the language as a way of learning where the language may be going in future. Some mention that not being able to see the authors speak (at least on video) makes it harder to make this judgment. For many, however, Roberto Ierusalimsky’s book provides enough answers. For instance, Rich, who started using Lua several years before having a chance to see Roberto live (at the same workshop as Craig) says:

Rich: Like Perl, the book was written by the architect of the language. *Programming in Perl* is written by Larry Wall. And the Lua book was written by Roberto. So, by reading the book you get the sense of both the designer’s personality and the language itself. And so just reading the book it was pretty clear that the guy who made this language was a really smart guy, and he valued principles that I valued: simplicity, elegance. And at that point it doesn’t really matter where he lives or what nationality he is. He is just a smart guy who made a good language, that’s all that matters to me. If he wasn’t able to speak English it probably would have been a problem. But obviously there wasn’t any communication barrier. So the fact that it was made in Brazil wasn’t anything to me.

Lua authors’ ability to successfully communicate (in English) their commitment to the principles that the potential users share makes Lua’s geographic origin irrelevant as far as most of the foreign users are concerned. American users of Lua also find that they can understand Lua by placing it in the larger genealogy of programming languages, most of which were developed either in the United States or with strong involvement of American computer scientists. “Lua’s syntax is based on Algol and Algol’s syntax is based on algebra” says Rich. Anyone who is familiar with algebra has already made many steps towards learning Lua.

Steve draws a somewhat different genealogy, describing Lua as “Lisp-like.” Some aspects of Lua’s design closely resemble Lisp, an influential programming language developed at MIT in 1960s, which is rarely used today but is often hailed as an example of good programming language design.²⁶⁹ Describing Lua as “Lisp-like” presents Lua as a descendant of a language that

269 Lisp occupies a paradoxical position in the world of programming languages: a language often admired but

all programmers are expected to respect, giving Lua substantial gravitas. Others describe Lua by producing a list of Lua's features. "Suffice it to say," says a recent article on Lua, "that Lua is an elegant, easy-to-learn language with a mostly procedural syntax, featuring automatic memory management, full lexical scoping, closures, iterators, coroutines, proper tail calls, and extremely practical data-handling using associative arrays" (Hirschi 2007). Such a description again helps make Lua's geographic origin irrelevant by placing the language within a classification system developed primarily by American computer scientists.

The classification system that makes it possible to describe Lua using a list of eight concepts, as Hirschi does in the quote above, and the larger system of meaning to which it is linked provide an important disembedding mechanism. A programming language that is designed in such a way that it can be explained in the terms of this shared system is relatively free to travel. Such academic sophistication is not expected of all programming languages—some of them are notorious for being "messy" and "ugly" and become wide-spread primarily through a strong association with a powerful actor. For Lua, however, the academic credentials become crucial. (It should be noted that not all programmers would be expected to understand what all of those terms mean. "Coroutines," in particular, are a relatively rare concept outside of academic computer science. A description of Lua in terms of coroutines and tail calls may thus do more to legitimate the language than to explain it to a typical programmer.)

Users of Lua often also point to another important disembedding mechanism: Lua's internationalized credentials, typically expressed in the form "Lua is used by X," where X is a major company such as Microsoft or Adobe, or "Y was written in Lua," where Y is an internationally known software product, such as World of Warcraft or Adobe Lightroom. Steve, for example, says that his choice of Lua of course attracted questions within the company. "What is this? Where does this come from?" asked his colleagues. Steve replied that Microsoft had already shipped products with Lua, so they would not be the first big company using it.²⁷⁰ "Used by Microsoft" can be understood as an instance of what Giddens (1990) calls "symbolic tokens." Symbolic tokens represent known units of value recognized across space and act as an important disembedding mechanism, by removing the need for situated trust. "Used by Microsoft" deflects the original question about Lua's origin ("Where does this come from?") and whether its authors can be trusted. If major companies use Lua, where it comes from and who wrote it is a lot less relevant. (Of course, what is important is not the actual use of Lua by such companies, but rather public *knowledge* of such use. We will return to this distinction later.)

rarely used. Lisp is typically seen by programmers as elegant but hard to learn, requiring a highly abstract and somewhat unintuitive approach to programming. Fans of Lua often point out that Lua implements some of Lisp's ideas while also allowing a more intuitive style of programming used by other popular programming languages. (This more popular style is what Rich calls "Algol syntax.")

270 This was somewhat of an exaggeration, he tells me, since it was actually Microsoft Games that used Lua (in "Baldur's Gate"). Still, Steve explains, Microsoft shipped something with Lua, so, there were "respectable, known companies" using the language.

Legacy Stuff

Over its history, Lua has undergone some substantial modifications, breaking backwards compatibility many times. Craig remembers this issue being discussed at the workshop that he attended. As a new user, he had little interest in the topic:

Craig: During the meeting a lot of people were worrying about legacy stuff, and Roberto was saying “Here is the new...”—Roberto, right?—“Ok, here are the new things in Lua 5” or whatever, 5.1. And then there were all of those people, “Oh, will this break compatibility? Will this break compatibility?” And in my case, I was just like: “Oh, I don’t care, please break compatibility, make it good for me!”

Existence of old code (“legacy stuff”) creates a serious problem in software development. When programming languages and software libraries are put to use, their limitations eventually become clear. While those limitations can often be overcome by additional code, such incremental additions lead to increasingly “ugly” and “ad hoc” design. The authors face the temptation of rethinking their design and “cleaning it up.” To be truly effective, such “cleaning up” often requires radical changes, which would make the new version incompatible with code written earlier. Changing old code to work with the new version of the programming language or a library often requires a lot of work and introduces new opportunity for bugs. It is therefore not taken lightly. (Many companies still use software written in 1960s in COBOL.) Avoiding the need for such changes is called “backwards compatibility.” The need for backwards compatibility typically impedes the evolution of a language or a library and leads to increase in size of the code (“bloat”) and unnecessary complexity (“cruft”).

While other Lua users in California express somewhat more interest in backwards compatibility than Craig, most consider Lua’s willingness to break with the past as a major source of strength, comparing Lua explicitly with JavaScript, a language embedded into most modern web-browsers.²⁷¹ Lua devotees sometimes describe JavaScript as “nasty,” comparing it to Lua’s “elegance.” Some point out, however, that this “nastiness” is very much connected with JavaScript’s wide-spread adoption:

Rich: JavaScript suffered from premature standardization. There is this web-browser, the web blew up, so everybody is using JavaScript. And then they thought: “Oh, we have to standardize this, make it interoperable.” And the language hadn’t really stabilized at that point. So there is a lot of cruft and nasty things. Both in JavaScript, that is the language itself, such as the “with” operator, and especially in the object libraries. [...] *Because* it was standardized so early, *because* it had a huge community. Whereas Lua didn’t really have those forces. It had a small community and less momentum. So the designers could, when they realized they made a mistake, throw it out, unify the concepts under a different way of thinking. The different abstractions that went from Lua 3 to 4 to 5. So,

²⁷¹ For additional information, see the glossary entry for “JavaScript.”

JavaScript *could* have been as good of a language as Lua, I think, if it hadn't this clutter on, by the huge community, a huge user base.

For Rich, Lua is simple and elegant because it is not weighed down by commitments to any existing body of code. Not having such commitments is, however, a choice that Lua's designers made, as they broke compatibility with Lua's original applications. Comparing Lua and JavaScript, we should consider that while popular web browsers prevent JavaScript from achieving Lua's elegance, it is those browsers that make JavaScript relevant and cutting the links with those browsers would undermine JavaScript's popularity. Lua's early commitments were to custom software written for a company located in Rio de Janeiro. Most Lua's users agree that leaving such "legacy" behind was crucial for Lua's international success. As we will see, some users in Rio de Janeiro had to pay for this luxury.

DEL, SOL, LUA

If we went back in time to the early 1990s, when the first version of Lua was created, we would find a highly embedded project, tied to local goals, relationships and commitments. Had Craig tried to use an early version of Lua, he would have likely found it a daunting task, facing a piece of code written for a specific purpose (quite unlike his), offering him little in terms of social support and no justification if his choice of the language were to be questioned. At the same time, as we will see, from the earliest days of Lua, the authors had made a number of choices that made future disembedding easier.

In 1992, Roberto Ierusalimsky returned to Rio de Janeiro from a one year post-doc at the University of Waterloo, in Canada, and started working as an assistant professor at PUC-Rio, a private catholic university that is home to the most prestigious computer science (*informática*) program in Brazil. A native of Rio, Roberto had earlier completed Bachelor's, Master's and Ph.D. at PUC. As we saw in chapter 2.2, PUC had become a key center of computer science in Brazil after a long and curious history of "bootstrapping," when pioneers of Brazilian computer science worked to assemble the practice of computer science research out of whatever elements could be found. By the late 1980s, however, the department was well established and had been granting doctoral degrees for a decade. Roberto followed up his PUC studies with a post-doc at University of Waterloo—the department that had long served as the North American counterpart for PUC's Department of Informatics and had been instrumental in helping develop computer science research in Rio de Janeiro.

In addition to his job as a professor, and his academic research in programming languages (an experimental language that "completely failed," according to him), Roberto started doing consulting work at a PUC-based consulting venture called Tecgraf.²⁷² In the early 1990s, Tecgraf

272 At the time and throughout the 1990s Tecgraf spelled its name as "TeCGraf," highlighting the fact that the name was an acronym for "Tecnologia em Computação Gráfica" ("Computer graphics technology"). The transition from "TeCGraf" to "Tecgraf" happened some time between 2001 and 2003. In this chapter, I use the new spelling throughout.

was a fairly small group of PUC students and professors offering IT consulting services to a number of organizations, including Petrobras—Brazil’s main oil company and one the country’s largest corporations. Tecgraf was founded in 1987 in partnership with Petrobras, at the time when Brazil was pursuing the policy of self-sufficiency in computer technology and software (see chapter 2.2) and Petrobras had to rely on local consultants for its computing needs.²⁷³ Petrobras was an unusual client—a semi-public company responsible for reducing Brazil’s dependence on foreign oil by developing capacity to extract deep sea oil off the coast of Brazil (and especially the areas surrounding Rio de Janeiro). Petrobras thus faced substantial technological challenges and was an important consumer of scientific expertise.

Also at Tecgraf was Luiz Henrique de Figueiredo who had also just recently received his Ph.D. in Rio de Janeiro at the Institute for Applied and Pure Math (IMPA) and was employed by Tecgraf full time. Also a native of Rio and a graduate of PUC-Rio, Luiz Henrique had earlier spent three years pursuing a Ph.D. in England and another year working at the University of Waterloo. Luiz Henrique was trained as a mathematician, thus benefiting from a different and considerably longer history of efforts to transplant a scientific practice in Brazil. From his early days as an undergraduate at PUC, however, Luiz Henrique had been interested in computing. (PUC did not offer an undergraduate computer science course at the time.) After returning from England, Luiz Henrique later spent a year in Canada, working at University of Waterloo’s Computer Systems Group. He focused his doctoral research on computer graphics, while working as a software developer at Tecgraf.

In 1992, Luiz Henrique turned to the problem of providing a unified way of configuring graphic interfaces for large number of software applications that Petrobras used for simulations related to oil extraction. “These were huge programs, that were very old and very refined and they didn’t want to give them up,” says Luiz Henrique, “But at the same time, because they were very old, the interface was very clunky.” Tecgraf was asked to provide a better interface to this old simulation software, something that would allow the users to simply click on a diagram, enter a value and request a simulation. Realizing that Tecgraf would need to provide such an interface for a wide number of simulators, Luiz Henrique started thinking about developing a language for expressing the configurations:

Luiz Henrique: So, we talked to them and came up with an idea: why not design a language so that we can write a single program that would capture this data. This was kind of a typical problem. You would write a simple text file, that would say: I want this diagram and in this diagram when I click this entity you should show this kind of a menu and do this kind data validation, things like that. And then when I am done I want this data to be output in this format. So we wrote this language to spec this kind of task.²⁷⁴

273 “Tecgraf’s clients could not afford, either politically or financially, to buy customized software from abroad: by the market reserve rules, they would have to go through a complicated bureaucratic process to prove that their needs could not be met by Brazilian companies” (Jerusalimschy et al., 2007).

274 All my interviews with Luiz Henrique de Figueiredo were conducted in English. See appendix B for the treatment of English interviews with non-native English speakers.

The language, written in 1992,²⁷⁵ was described in a Tecgraf report (Figueiredo 1992) and the larger system of which it was a part was then described in a paper presented at a conference in Brazil (Figueiredo et al. 1992). The language was called “DEL,” short for “data-entry language.” It was what would today be known as “a domain-specific language” and was then called “a little language” (Ierusalimschy et al. 2007), designed to be used with a specific application. While DEL was a success in Tecgraf and among its users in Petrobras (Ierusalimschy et al. 2007), it soon became clear that it was too limited to build all the applications that Petrobras wanted.

At the same time, Roberto Ierusalimschy and Waldemar Celes (then a Ph.D. student at PUC), developed another domain-specific language called “SOL” (“Simple Object Language”) for another of Petrobras’s many specific problems. SOL was finished, but never delivered to Petrobras, as it became clear that, just like DEL, it lacked the expressiveness wanted by some of the Petrobras projects. In mid 1993 Roberto, Waldemar and Luiz Henrique met to discuss the possibility of replacing both DEL and SOL with a new language, which was soon implemented by Waldemar as a course project. They called the language “LUA,” meaning “moon” in Portuguese. This was as a pun on “SOL” (Portuguese for “sun”), but also, as somewhat of a joke, an abbreviation for Portuguese “Linguagem para Usuarios de Aplicação”—“Language for Application Users.” LUA was a success and was quickly picked up by other projects at Tecgraf.

Comments in English

DEL, SOL and LUA (soon renamed “Lua”) were all written in the C programming language, one of the most popular programming languages in the 1990s. In addition to being the most popular (the distinction that it is only starting to lose today), C was and still is the *lingua franca* of programming languages.²⁷⁶ Most programming languages provide mechanisms to develop “bindings” to libraries written in C, giving the programmer an option of calling C code. Some (but not all) provide ways of being called from a C program. Like other “scripting languages” that emerged around the same time,²⁷⁷ Lua aimed to be easier to work with than C. Unlike most of those languages—with the exception of Tcl—Lua was primarily designed for use inside a C application, making it possible to develop parts of the system in C and other parts in Lua. Rather than trying to serve as an alternative to the *lingua franca*, Lua was designed to work *with* it. The ease with which Lua and C could be mixed, is in fact often considered one of Lua’s greatest strengths vis-à-vis similar programming languages.

Similarly, from early on, Lua displayed a commitment to another *lingua franca*: English. Lua and SOL were both coded “in English,” in the sense that all entities had English names—a

275 According to Ierusalimschy et al. (2007) and the comments in DEL code, the language was developed jointly by Luiz Henrique de Figueiredo and Luiz Cristovão Gomes Coelho.

276 TIOBE declared C “the programming language” for 2008, acknowledging the fact that C has grown in popularity in 2008, despite being one of the oldest languages in TIOBE’s top twenty.

277 Perl in 1987, a variety of “structured” dialects of BASIC in the late 1980s, Tcl in 1988, Python in 1991, Ruby in 1992, to name some of the more successful ones.

relatively common practice among Brazilian software professionals.²⁷⁸ (See chapter 1.2.) Somewhat more unusually, and contrary to Tecgraf’s general practice, Lua’s error messages and the comments in the code were written in English as well. One of Tecgraf developers who worked with Lua in the early 1990s (“when it was still called SOL”), talks about the differences between Lua and other Tecgraf projects:

Yuri: I know that in the first version of Lua, and even the code of SOL was written in English, even with comments in English...

Antônio: Yes, even today we do this. Here in our group too. Even apart from Lua we try... No, sorry, the comments are in Portuguese... obviously, right? But variable names, we usually try to do all in English. But our comments are in Portuguese. Lua is different.²⁷⁹

Antônio starts by saying that Lua’s use of English is no different from the general practice at Tecgraf, but immediately corrects himself, pointing out that while using English for variable names is the standard, Tecgraf *obviously* uses Portuguese for comments.

When I ask Antônio whether usage of Portuguese for comments is really so “obvious,” considering that Lua’s comments are written in English, he explains, contrasting Lua’s case with that of most Tecgraf applications:

Yuri: You said the comments are “obviously” in Portuguese. [...] It seems that sometimes people *do* write comments in English.

Antônio: No, no. We... in Portuguese. Because... In Lua’s case I think it even makes sense for everything to be in English. It was born to fly, for other... It has a much more globalized use than our applications. We, frankly, don’t even encourage... we don’t even want people here to try to write comments in English, because most of them are not fluent in English, so it would end up being broken English (*inglês capenga*)... Our products are not... Our source code is not for export, it’s not open. What we do is not open. The code we write is for Petrobras and is their property, and they don’t want to have to also... to know English to read our code. So, for several reasons, in applications we don’t write comments in English. Variable names—yes, in the code. You read it more or less in English. But comments...

Yuri: But then why are variable names in English?

278 SOL implementation had minimal comments, a total of fifty five words. All of those comments were in English, however. The earliest available implementation of Lua (from July 28, 1993) contains a small number of comments in Portuguese, but is overall consistently written in English. DEL implementation consists of twenty three files, twenty of which are strictly in English, including all eighteen files attributed to Luiz Henrique de Figueiredo. The other three files use a mixture of English and Portuguese for both variable names and comments.

279 See appendix C, “Original Interview Quotes,” (Antônio, May 2007, “Os comentários nossos são em português”).

Antônio: This is a question we've been debating, but we feel that it ends up... For example, the code syntax would, even when you just read it, would be weird, no? We find it strange to mix things that way. But comments—no. And now that you said it... For me this is obvious, but looking back at it, maybe not, huh? Why is it natural for comments to be in Portuguese but not for other things, right?²⁸⁰

Choosing English variable names is standard practice at Tecgraf and many other Brazilian organizations. This practice already gives software a certain degree of mobility. Lua authors, however, take an extra step, using English not only for naming variables but also for comments in their software.

Antônio alludes to two differences between the actual code and the comments that may explain why Brazilian developers often use English for variable names and Portuguese for the comments. Code already typically includes English words, whether or not a Brazilian programmer wants it to (see chapter 1.2). Additionally, code uses English in a constrained way, with heavy reliance on abbreviations (e.g., “apl_unsel_group”), making the programmer’s lack of English fluency less obvious. Comments on the other hand, are written in full sentences, and offer little opportunity to hide the weakness of programmer’s English.

Lua is different, however, says Antônio, and he does not remember anyone ever finding strange the use of English in Lua’s comments and error messages:

Antônio: No, no. It’s like this. I don’t think this is questionable. I mean, everyone kind of understands that Lua... The idea is that Lua would be used really in other contexts, in other places. That it would have... coverage. Would reach beyond Brazil. Rio is [just] it’s base. I think it’s natural for Lua to be that way.²⁸¹

In retrospect, using English appears quite natural. The language “was born to fly,” says Antônio.

World-wide success, however, was hardly a part of the Lua team’s plan, according to Roberto Ierusalimsky, who instead explains his use of English as a matter of habit and convenience.²⁸² Compared to the Tecgraf developers who, as Antônio fears, might write comments in broken English, Luiz Henrique and Roberto were both comfortable with English, as well with the English-speaking academic culture more broadly, having spent some time abroad. Lua’s authors could thus write their comments in English without the risk of embarrassing themselves with simple mistakes. This competence (and confidence) with English gave Lua an additional early start on disembedding.

When I point out to Antônio that Lua was clearly *not* meant to “fly,” but was ostensibly written to solve a specific problem faced by Tecgraf, he offers a different explanation of Lua team’s possible rationale by referring to his own project, which he sees as *potentially* open.

280 See appendix C, “Original Interview Quotes,” (Antônio, May 2007, “Uso muito mais globalizado”).

281 See appendix C, “Original Interview Quotes,” (Antônio, May 2007, “Em outros contextos, em outros lugares”).

282 See chapter 1.2 for Roberto’s own explanation for this decision.

Yuri: But at the time it wasn't [intended for use abroad], right? At the time it was just a project...

Antônio: Yes, nobody could think about... the explosion, the success that Lua would have. The acceptance in the *games* [said in English] industry. But maybe they, Roberto and Luiz Henrique, had this idea, I don't know. For example, this [hobby project] that I wrote, that's all in English. Comments are in English. Because in my case I was thinking, I don't know: one day I'll put it on Lua Forge, someone will want to download. I like this idea of *open source* [English], of "many eyes" [English]. Everyone will be looking. I think that when you do an open source application, you have to speak the most wide-spread language, the most common language, most easily understood. In case of our applications, no, us no. Not at Tecgraf. We actually don't want this to happen.²⁸³

While Antônio says that he has never written comments in English when working on Tecgraf products, he uses English comments when he works on a hobby project. He makes it clear that he neither expect this project to become anything big, and in fact he has not gotten around to releasing the code to anyone. "One day," however, he may put it on LuaForge, a website where users of Lua put code that they want to share. And who knows, it may become popular. Using English keeps open this possibility.

Antônio's project and Lua are both different from the products that Tecgraf built for Petrobras by their relative isolation from the local power relations. Neither was written to immediately become property of Petrobras. While both projects may technically belong to PUC, as an academic institution PUC appears to be content to let the developers to treat their code as free. (See the next chapter on the question of Lua's ownership by PUC.) In addition to simply giving them the freedom to write such projects as they want, this makes the possibility of a global success somewhat more imaginable, since the projects' destinies are not as obviously in the hands of the bureaucracy of a Brazilian corporation

To understand Lua's early use of English we must thus consider the *potentiality* of Lua's later success, the inherent ambivalence of such projects, and the notion of "subvocal imagination" discussed in chapter 3.3. Created for specific needs (of those who pay the bills), projects such as Lua may from the beginning carry the imaginable possibility of global success. While such global future is rarely *planned* for explicitly, it can be *imagined*, and this may be sufficient reason for developing the project in a way that would not altogether preclude the possibility of a global success. (See a later discussion of this issue in chapter 3.4). This means, among other things, writing comments in English.

When explaining his rationale for writing his hobby project in English, Antônio draws explicitly on *open source* terminology. Today, the open source paradigm provides developers with a ready vocabulary and an accepted framework for explaining such decisions. While Lua eventually becomes "free software," it did not *start* this way nor does it appear that the free software / open source vocabulary was available to Lua's authors in 1993. We will come back to

283 See appendix C, "Original Interview Quotes," (Antônio, May 2007, "Vou botar no LuaForge").

Lua's relationship with the world of "open source" software later in this chapter and again in the next one.

Wait, How Did You Do This?

Like code and comments, Lua's manual was similarly written in English, though none of the early users whom I interviewed remember using it. (Some seem to believe it was written in Portuguese.) As Tecgraf was quite small and all the developers were located in the same room, simply talking to other members of the lab was the easiest way to learn Lua. As another early user of Lua says:

Bruno: I learned from the code.

Yuri: Reading code that was already written?

Bruno: Already written. By Luiz Henrique, by Roberto, and by others who started before me. I probably saw Rodrigo's [pseudonym] code, probably saw other people's code. [...] Since Tecgraf in the beginning was small, the people all stayed in the same lab. So for example, when I did Ph.D. here, I sat at a computer, next to me sat Luiz Henrique who worked here at that point. He just finished his Ph.D. but was still here. I started when he finished his Ph.D. Rodrigo was finishing his undergraduate; he sat at my other side. You see? So, we were all in contact. My dissertation, for example, was all done in C. So when I decided to move, to get into Lua, it was because I was following what other people were doing. I would write, I don't know, two pages of code and then look to my side and see that the guy sitting there did the same in, I don't know, a quarter of a page. "Wait, how did you do this... in such a simple way?" So, it was through this contact inside the lab was that people learned. You see? Today we've grown. There is no more lab. There are now groups and each has their own project. So, if you are with a project that doesn't use Lua, it won't be as easy to stay in contact and learn the same way. But back in the day it was almost natural.²⁸⁴

Others who started Lua at the time similarly say that they learned Lua by simply talking to its authors and other early users, rather than having to rely on the early version of the manual. Lua similarly did not have a mailing list—things that could not be resolved in person could be discussed on the general Tecgraf list.

While Bruno stresses that much has changed at Tecgraf due to its growth (Tecgraf now employs more than 250 people) he still prefers to have lunch with Luiz Henrique once a month than to follow the discussion on the Lua mailing list. In fact, as Bruno and I leave Tecgraf's

²⁸⁴ See appendix C, "Original Interview Quotes," (Bruno, April 2007, "Aprendi pelo código").

office and head to get lunch with Rodrigo Miranda (who sat on Bruno’s other side in 1993), we run into Luiz Henrique. The days of face-to-face contact at Tecgraf are not quite gone.

Let’s See What Happens

Lua’s authors stress that the language was developed to solve specific problems Tecgraf faced in their work for Petrobras, and that Lua’s later international success came as a major surprise to them:

Yuri: Have your personal goals for Lua changed over time? I mean...

Roberto: Completely, completely. Completely! This is so huge, I can’t... It changes everything. When we started Lua... This is one of the things that people do not... When we say in our paper, that paper about history of Lua,²⁸⁵ that it went beyond our most optimistic expectations, this is not very true. Because we didn’t have *any* expectations. For instance, when we started, we never created a language for ta-ta-ti-ta-ta-ta. We really created a language to solve this specific problem we had at the time. That’s why we joke, but it’s true: There was no “Lua 1.0.”²⁸⁶ There was “Lua.” We did code, and that worked and “oh great, it solved our problems here.” We didn’t even have an RCS project. We didn’t have *anything*.²⁸⁷

Lua represented a specific solution for a specific problem. Lua’s authors also did not attempt to be innovative for the sake of innovation. Lua represented a pragmatic combination of features, informed by academic research on programming language design, but not aiming to contribute to it.²⁸⁸

The pressure to publish, however, combined with Lua’s early success at Tecgraf, soon led the authors to present the language outside Tecgraf,²⁸⁹ starting with a short presentation at a conference in Brazil in 1993:

285 Ierusalimschy et al. (2007)

286 The manual distributed with Lua 1.1 contains a link to ftp.icad.puc-rio.br:/pub/luas/luas_1.0.tar.Z, which presumably represented a distribution of “Lua 1.0.” The link is currently dead and it appears that its inclusion in the manual for Lua 1.1 was a mistake. A snapshot of a pre-1.1 version of Lua was later released in 2004 as “Lua 1.0” to commemorate Lua’s ten year anniversary.

287 All interviews with Roberto Ierusalimschy were conducted in English.

288 Ten years later, Lua 5.0 (Ierusalimschy et al. 2005) became the first scripting language to use a register-based virtual machine, which brought Lua substantial academic interest. Additionally, in the more recent years, Roberto Ierusalimschy and his students have used Lua as a base for experimental work in programming language research.

289 DEL was similarly described in a Tecgraf technical report (Figueiredo 1992) and a papers presented at a conference in Brazil (Figueiredo et al. 1992).

Roberto: Yeah, I think it's exactly that: we are academics. [...] There was a symposium explicitly asking for a kind of industry... "real stuff," as they said. I think they still have this in software engineering. At least at that time they were very concerned about this division of academic and "real" applications. I remember sometime later them asking us to review papers for instance, and there was one or two grades related to how does these apply to real situations or how much this is really used, or things like that. So I thought, "Oh we're going to show Lua that way," I mean, it's a real tool, there are people, even... That was very funny, this I remember until today that even that very, very small base of users at that time, of still Lua 1.0, was considered big inside the conference and we said: "Oh we have some eight, ten people really programming in that language." [Laughs.] And that's much bigger than a lot of papers where, I don't know, only the writer [author] ever programmed, used that tool for anything. But that was not, I mean, it was academic because we always have this pressure to publish so if we can generate a publication, we always try to do that. But so this was just kind of: "Oh, they are asking for that, we have Lua ready here, let's try to stage and let's prepare Lua" and we gave a presentation there about Lua.

According to Roberto, the presentation was extremely well received.

As Roberto explains, Lua was not written to advance computer science research, but its design did benefit from Roberto's academic background in programming language theory. Roberto thus borrowed design ideas from a wide range of programming languages with which he was familiar, including some that few software developers know or use. More importantly, Roberto's work as a researcher allowed him to *explain* Lua in terms of current computer science research, helping Lua travel away from Rio using academic papers as a vehicle. (A few years later the same ability to explain Lua in proper terms became important in giving Lua credibility in the eyes of American software developers working outside academia.)

In 1994, the team wrote a longer paper for another Brazilian conference. This time, the paper was written in English and referred to the language as "Lua" rather than "LUA,"²⁹⁰ avoiding the hassle of explaining a Portuguese acronym in an English paper. The paper also included a link to download Lua ("Lua 1.1"). Roberto says that a colleague at Tecgraf "pushed" them to put Lua up on a website and to include a link in the paper "to show that this wasn't just vaporware." ("Vaporware" is developers' term for software that is described but does not actually exist or does not work as described.) Encouraged by Lua's success so far, the team was also curious to see what might happen next, and Luiz Henrique announced Lua on a number of newsgroups.²⁹¹

Roberto: I don't remember, but someone kind of pushed us to put it on the Internet. We didn't even think about that. They started: "Put it on the Internet,

290 The 1993 presentation was entitled "LUA: uma linguagem para customização de de aplicações" ("LUA: a language for customizing applications").

291 <http://compilers.iecc.com/comparch/article/94-07-051>

that's good, people are going to use it." "Ok, let's put it on FTP." And then I have no idea of how some people heard about it, we didn't have a website or anything like that. I think... I think Luiz sent mail to [long pause] to some groups. But I don't know when that was. 94? 95? Just announcing that there was Lua. And then some people started using it. But it was kind of, "Let's see what happens."

The paper and the announcement started a slow trickle of questions, some of them from abroad.

Lua 1.1 was packaged with an informal license that allowed free academic use, but reserved the rights for commercial use:

Roberto: Something we wanted, that I remember... Again someone gave us this idea to try to sell Lua. In the beginning we put it on the Internet with a free academic license, and "Please contact us for commercial use." So there was this idea "Let's try to sell Lua." And then it stayed this way one year and we got one contact. [Laughs.] Without success. Just a contact for "Maybe we could use..." for commercial use. So, we decided we were not going to sell it. But after that we noticed that there were people using it and people were liking it, and we were liking that idea of other people using Lua.

Lua 2.5, released in the early 1995, included a license written in proper English "legalese" and allowing almost unrestricted use of Lua for both academic and commercial purposes.²⁹²

The change of license not only gave additional freedom to Lua's potential users, but also signified the authors' changing perspective on what they could and could not achieve with Lua. Having started with a rather vague idea of what Lua could lead to and originally holding open the possibility of selling it for money, they move to "liking the idea of other people using it." Roberto then continues:

Roberto: That kind of... touched... satisfying, a kind of gratification for us, gratifying, whatever. And so we started to feel good about that. [...] And then we published the article: "Let see, let's try to get more users, to promote Lua." So we put up... And then the reaction was very strong, and then it started to be really important—the outside users.

Consistent with Becker's theory of motivation (see chapters 1.1 and 2.1), while Lua's authors do not start off with an intention of distributing free software, they develop the appropriate perceptions and judgments as they engage in the activity. As members of an academic community, however, and in particular being fairly fluent in the culture of Anglo-American computer science, from which the free software movement was born, they were of course well

292 "Naively, we wrote our own license text as a slight collage and rewording of existing licenses," later wrote Lua's authors (Ierusalimsky et al. 2007). While the authors say that they do not remember from what sources they borrowed the text of the license, the first part of the Lua 2.1 license is identical to the license of Tcl 7.3 (released in 1993), while the rest generally corresponds to the X11 license.

prepared for developing such perceptions and judgments.

In 1996, the team wrote an article for *Software: Practice & Experience* (Ierusalimschy et al. 1996) and another one for *Dr. Dobb's Journal* (Figueiredo et al. 1996). Articles in those widely-read publications (the latter targeting software professionals rather than academics), introduced Lua to a larger audience and led to a stream of questions from abroad, and the decision to set up a mailing list.

Luiz Henrique: Around that time, I remember now, we wrote this article in *Dr. Dobb's Journal* and from then on we started to get messages from abroad, people asking questions about Lua. So, we thought, well, maybe we are going to get too many questions and won't have time to answer them all. So we created the mailing list for that, so that other people could answer our questions. [...] Maybe Lua is going to get some interest, and how about creating a community? [...] If we were going to get a community, maybe we should have a mailing list so that they could talk among themselves? To not have to answer everyone individually.

The list (Lua-L) was setup in February 1997.²⁹³

Prior to the creation of Lua-L, the project had no dedicated mailing list. (The need for the list was largely obviated by the close proximity of the people using it—see previous section.) The English Lua-L thus became the central forum for the Lua community, attracting many Lua users from PUC, as well as a smaller number from other Brazilian research institutions. In 1997 the Brazilians (people with .br email addresses) comprised a little under a quarter of the list's participants, constituting the list's largest "minority." As the list grew, however, the percentage of the Brazilian participants started to decline, eventually getting overtaken by Germans in 2007.²⁹⁴

More Exciting Users

A month before the mailing list was set up, the Lua authors received a message from a programmer working for Lucas Arts, which read:

To: "lua@icad.puc-rio.br" <lua@icad.puc-rio.br>

293 Each of the three members of the team spent some time abroad (in different places) between 1995 and 1997, though this fact did not come up in any of my interviews. While their separation had roughly preceded the setup of the mailing list, the list did not become the locus of Lua development, even to the limited extent as happened with Kepler's list after Alan's departure (see chapter 3.4).

294 In 2007, the list received messages from 38 .de addresses (6.1% of the total) and 26 .br addresses (4.2%). This most likely undercounts the actual number of Germans and Brazilians because of the wide use of gmail.com addresses (28.2% of the addresses in 2007). This most likely undercounts the actual number of Germans and Brazilians because of the wide use of gmail.com addresses (28.2% of the addresses in 2007). Note that .br addresses account for a much larger proportion of messages (12.7% vs. 3.7% for .de), because of a small number of very active Brazilian participants. See appendix J for details.

Subject: LUA rocks! Question, too.
Date: Thu, 9 Jan 1997 13:21:41 -0800

Hi there...

After reading the Dr. Dobb's article on Lua I was very eager to check it out, and so far it has exceeded my expectations in every way! It's elegance and simplicity astound me. Congratulations on developing such a well-thought out language.

Some background: I am working on an adventure game for the LucasArts Entertainment Co., and I want to try replacing our older adventure game scripting language, SCUMM, with Lua. (Ierusalimschy et al. 2001, p. 8).

Once the list was created, he became an active participant and in April revealed to the list that he was working on a “scripted adventure game engine.” Soon, other users started to discuss their use of Lua for scripting video games. A year later, Lucas Arts released *Grim Fandango*, which became Lua's first international success case—not quite yet “used by Microsoft” but a major step towards it.

Grim Fandango also gave it a new “place.” Lua now had a new, international “origin” being associated with a community that was not narrowly localized. *People ask where Lua is from*, says Steve, the interviewee introduced earlier. *But they do not usually mean location, but rather which industry or context. For example, JavaScript comes out of the Web. So, I give two answers: “PUC-Rio” and “the games industry.”* Steve himself understands quite well that Lua does not really “come out of” the game industry. The game industry, however, provides a context in which Lua can be understood. *People don't care who wrote it*, he says. *They want to know how it fits into the world.* The game industry thus provides Lua with a pedigree that makes sense in a foreign context.

Lua's success at LucasArts also gave Lua new advocates, located in about the best place for promoting Lua. As the LucasArts programmer explained on the Lua mailing list in 2001, face to face interactions in California were instrumental in helping Lua gain popularity in the games industry. In 1998, some of LucasArts programmers attended the Game Developer's Conference, the largest trade event for computer games developers, held annually around San Francisco Bay Area. One of them made a presentation about implementing scripting languages for computer games.

It was quite heavily attended, probably 200 or 300 people in the room. Rob talked at length about the benefits but also all the complexity of writing your own language from scratch, and went through lexing, parsing, analysis, compiling, etc. etc. in good detail. [...] Near the end of his talk many game programmers looked quite discouraged, realizing that a decent scripting language was not something you could whip up overnight, especially your first time out. As the questions started, he pointed out that you can take pre-existing language interpreters and get much quicker results, specifically mentioning Lua. He talked about Grim and pointed me out in the audience, and I stood up and

gave a brief blurb about Lua. Between us we said that it only took a day to embed, code base was clear and easy to modify, was small and fast, extensible, easy to pick up for designers, etc. People lit up and furiously started scribbling notes and looked really excited. I got a few inquiries afterwards, but game developers being who they are, most of them just went out and checked it out on their own. Soon enough the list was overflowing with game programmer inquiries...

This “brief blurb” about Lua delivered at a conference in the very center of the software world, brought Lua to the attention of the larger gaming industry. This attention eventually led to the use of Lua in a game released by Microsoft Games, giving Lua the coveted “used by Microsoft” status.

The growing list increasingly became an important source of influence on Lua. “Tecgraf was kind of stable and was not demanding that much,” says Roberto. While having a non-demanding employer may be a blessing in many lines of work, this is not necessarily the case in software development. (See chapter 2.1.) Finding that Tecgraf was no longer presenting serious changes, the Lua team increasingly turned their attention to the outside. “So I think it was also kind of like: ‘Let’s try to find more... more exciting users,’” says Roberto. The list members supplied the desired challenge, by applying Lua to new domains and running it on new platforms:

Roberto: Specific things that emerged, more than people asked. Problems that emerged, in different environments. One instance, one example is embedded devices. Tecgraf never used Lua in embedded devices. Now they use, but I am not sure if they would try to use it, if they would push Lua in this direction by themselves. That was something that from... people outside started to use Lua on those very different devices, on very small computers.

Lua’s gradual “porting” to foreign contexts went hand in hand with porting in the more narrow technical sense: Lua was increasingly used on computing platforms that were never used by Tecgraf:

Roberto: That put more pressure to make Lua really portable. In the beginning our goal of portability was Tecgraf’s set of computers. So, that was our goal, must run on that. It was a very large variety of computers that Tecgraf had, so from the beginning it was very portable. So it must run on DEC, on VAX, on ta ta ta. And then later when people... I remember in 98 someone wrote and said they ported Lua to Cray, the supercomputer Cray 1. That was very exciting: “Wow, Lua is running on Cray.” And then these things started for instance to show us that we must really think about ANSI C and about real standards. And not about “It runs on those machines and that’s good enough.” For instance this was something that came from outside.

Paradoxically, Lua’s origin in Brazil offered it an early start on portability. Due to the restrictions imposed by the Market Reserve, Petrobras had limited choice of what computers it used, having

to buy from different manufactures depending on who had was allowed to bring computers into Brazil in a particular year. Over the years, it had accumulated a rather diverse collection of computers, requiring that Lua could be run on all of them (Ierusalimschy et al. 2007). This collection did not include things like the Cray super-computer, however.

Demanding and attentive users are often considered in open source communities to be a valuable resource *per se*, as they help to “push” the project forward, and provide feedback and a source of gratification to the authors. In Lua’s case, however, foreign list members were not merely asking for features. Many became active members who helped answer questions and contributed ideas and resources. In 2000, one of the list members organized a wiki, which has become an important resource for Lua users. In 2001, one member’s “misunderstanding” of Roberto’s explanation of a future feature contributed a major improvement (Ierusalimschy 2007, page 2-15). The list members also offered substantial help with the first edition of *Programming in Lua* (see chapter 1.2).

Also in 2001, an email was sent to the Lua list asking: “Just out of curiosity, why aren’t you guys using Lua.org for your main URL?” Other members quickly pointed out that lua.org, lua.com and lua.net domains were all taken, perhaps by domain speculators. (Short domain names tend to be quite popular.) The discussion moved to comparing the advantages of “lualanguage.org” versus “lua.org.br.” The .br option was described by Roberto as complicated for bureaucratic reasons, while other users expressed concerns about the hassle of country-specific domains.²⁹⁵ In the end, however, the difficult decision between the “clumsy” lualanguage.org and the “country-specific” lua.org.br could be avoided: the original poster approached the owner of “lua.org,” purchased the domain and donated it to the Lua authors. (From 2004, another user has volunteered to host lua.org website on his company’s servers in the United Kingdom.)

Breaking from Tecgraf

During the first decade of its existence, Lua was a “Tecgraf project” and Tecgraf served as “a good home” for it, according to Roberto. This continued through the late 1990s, even as Lua was increasingly looking outwards. Around 2003, however, Tecgraf stopped paying for Lua development.

While this separation was to some extent expected, some of my interviewees attribute the ultimate break to the transition from Lua 3.2 to Lua 4. Released in November 2000, Lua 4 introduced substantial changes in the way Lua connected to code written in C. The change was originally motivated by the desire to allow a program written in C to run multiple Lua programs

295 “I agree.. though lua.org.br would be shorter, I for one never think of adding country specific things to the end of a .org domain (well, I guess that isn't a .org, but still). I think one of the hallmarks of a radical domain name is that you can think of the name of the project and add .org to the end and you go there.. that's just me though.”

at the same time.²⁹⁶ This ability was requested by some of the users as early as 1998 and also turned out to be necessary for one of Tecgraf’s own projects, called “CGILua,” which aimed to make it possible to use Lua for developing web applications. (CGILua later became the basis of Kepler—the project described in chapter 3.4.) The team originally introduced the feature by making the smallest possible modifications to Lua, trying to make sure that the new version (“Lua 3.3”) would require almost no changes to the existing software. However, the limitations of this approach soon became clear, and the team proceeded to make more and more serious changes to how Lua connects to C code. Eventually the interface between Lua and C was changed to a point where many existing programs would have to undergo serious modifications. The team then decided to use the opportunity to completely redesign the interface, thus changing it even further. When the new version was released in November 2000, it was deemed sufficiently different to be called “Lua 4.0” rather than “Lua 3.3.”

While offering substantial improvements, the new API made obsolete all old C code interfacing with Lua. Roberto offered suggestions on how to fix the old code to make it work with Lua 4, but few Tecgraf projects undertook such migration.

Roberto: Then there was this big problem of compatibility. I think maybe this was the main breaking point. [...] The change from 3.2 to 4.0. That was a big change in the API so for people that only used Lua as a language, it was not that big, but for people that integrated Lua into other tools, the C API changed a lot and all applications in Tecgraf were in that kind of API stuff.

Yuri: Was that something you foresaw?

Roberto: The break or their reaction?

Yuri: Well, either.

Roberto: The break [in compatibility] for sure we foresaw, but their reaction I think... We wrote some compatibility code and some things to help, but people mainly didn’t use it, at all. [...] They never changed to Lua 4. So they started to drift apart from the Lua community. I mean, because everything was written in new manuals, and new discussions and new tricks and everything was evolving around Lua 4.0 and they were...

Even CGILua, the project that motivated the changes that eventually led to Lua 4.0, never released a version that worked with Lua 4.0. (This was to a large extent caused by the fact that two of the key people working on CGILua left Tecgraf before Lua 4.0 was released. A version of CGILua for Lua 4 was done by one of its users, but was never released—see chapter 3.4.) Unwilling to make the transition, Tecgraf’s projects got “stuck” with Lua 3.2, a version that soon started to lose the interest for the Lua community. Lua’s and Tecgraf’s paths started to diverge.

296 When a program written in C loads and executes chunks of Lua code, this code modifies a set of data that represents the “state” of the Lua interpreter. Up until Lua 4.0, all such code would modify the same state. Lua 4.0 introduced the ability to maintain multiple Lua states within the same C program (a “reentrant API”).

In addition to the practical problem of backwards compatibility, Lua 4 set the precedent for introducing features that brought only cost and little benefit to Tecgraf's projects. Some users of Lua did not take this well:²⁹⁷

Roberto: I think that they got... with some reason I think they got a little offended with the change to 4.0. I think that's why it was kind of a break point. I think this was the first change that we saw that it could hurt Tecgraf but we are going to do it anyway. We thought that it was not going to hurt *that* much, we tried and thought... Not something like "Oh, we are going to do it *because* it's going to hurt Tecgraf." We tried to minimize that, as I say, we did a lot of stuff to try to do compatibility layers and things like that. But we knew that it was going to have some problems, was going to be a big incompatibility. And so I think. That's why I say it was a kind of break point.

Lua 4 was therefore not only introducing a technical break with existing Tecgraf software, but also demonstrating the new priorities of the Lua team.

The new version of Lua was an improvement—for the new users.

Yuri: Why was this change made?

Roberto: Because it was *really* much better. [Laughs.]

Yuri: But better for who, let's put it this way?

Roberto: For any new user of Lua.

It also worked quite well for the existing foreign community. Many of the list members were interested in Lua as a hobby and found the quick pace of change engaging intellectually. Others used it in games: software that is typically abandoned soon after it is released. (For example, two "versions" of Grim Fandango were released in 1998: 1.0 and 1.01. No other versions of the game were ever produced. Of course, some of the users may continue playing the game for years.) As far as foreign users of Lua are concerned Lua 4 was a clear step forward and not only because of the improved API. Lua 4 demonstrated the authors' commitment to building a good language and their willingness to leave behind earlier mistakes. As earlier quotes from Rich and Craig show, foreign users took note of this commitment.²⁹⁸

While some of Tecgraf's users of Lua complained about this transition, most of those to

297 Ironically, but not surprisingly, Roberto Ierusalimsky was one of the few people willing to talk extensively and on record about this conflict. Most people who were there at the time either asked to not be quoted or (more often) downplayed the complaints.

298 The discussion on the Lua list in the months leading to Lua 4.0 shows that the authors and the users were not entirely indifferent to backwards compatibility, and in fact saw it as quite important. Some members also disagreed with the specific changes introduced by the new version. In the end, however, the desire to make improvement won over the concerns about backwards compatibility and Lua 4.0 was received by the list with much enthusiasm.

whom I talked in 2007 seemed to consider this sacrifice worthwhile. One of the early users of Lua says:

Silvio: I think we skipped 4.0. I am not sure. I can send you this later, if it's important for you.

Yuri: Ok. I am asking because I've heard that people say that there were some difficulties between around Lua 3.2 and Lua 4 and that some people at that point gave up.

Silvio: True. Basically, because the communication API between Lua and C changed drastically. It was in 4, I think, that the stack was introduced in the communication between Lua and C. And so whoever had much C code calling Lua had to make lots of changes. And there were people who really complained: "Oh, darn, must change..." But I don't see it this way. I think we must keep moving ahead. Lua has to evolve. We are not going to stop and make Lua stagnate, or stay with an interface that we know is worse, just because there are people who use it who are lazy to change their applications. It doesn't make sense. To stagnate for the sake of stagnation... Those who don't want to evolve can stick with version 3.2 and use it for the rest of their lives. It'll keep working, thank you very much.²⁹⁹

In 2001, Lua was clearly showing global potential, and limiting it for the sake of Tecgraf's older projects did not necessarily make sense, even from the perspective of some of the Tecgraf engineers already invested in earlier versions of Lua. Lua's success abroad was starting to bring certain dividends to PUC and Tecgraf (in terms of prestige if not money), as well as individual people at Tecgraf who, like Silvio, were incorporating Lua into their academic research. Constraining Lua's growth was not necessarily in Tecgraf's best interest.

At the same time, Tecgraf itself was increasingly looking at other technologies. The rapid software innovation in 1990s meant that by 2001 Tecgraf was getting requests for new types of applications and could make use of new tools for implementing them. The most important of those was Java—a programming language released by Sun Microsystems in 1995 that had become the new standard by 2001. Even such committed supporters of Lua as Silvio saw those new technologies as better for some of their projects.

Silvio: It was about six years ago [in 2001] that we started increasing significantly the number of projects in Java. [...] In 2001 we had a request from the client, like, "Oh, we want a system with such, such and such characteristics." And I thought it would be more interesting to use Java than... Because there is this thing... There is this saying: "For someone who has a hammer, everything looks like a nail." We have to have a tool box and to know when to use each tool, right? Lua is a great tool, but it's not the right tool for everything. Nobody would expect it to be. For the job that we had in front of us, the ideal solution was a

299 See appendix C, "Original Interview Quotes," (Silvio, May 2007, "Lua 3.2 e Lua 4").

mixture. That's what we did. We used Java to build the client, which we wanted to run over the intranet, et cetera, in the browser. At the time there was no JavaWebStart, but there already was an Applet plugin. So we managed, for example, to run over the intranet an application with a cool graphic interface, without having to worry about the platform of the client, the user. So, we chose to do the client and the server in Java, but the part that was distributed on the machines, which needed more of the operating system things we implemented in Lua. We did this shortcut. The only thing is that what happens is that the server and the client part requires a lot more implementation than the other, so today the project is mostly in Java.³⁰⁰

Ironically, despite Lua's oft-cited portability, Silvio's team found that using Java made it easier for them to run the application on their clients' computers without having to worry about the operating system that the clients were using. Additionally, while Lua still worked best for certain Petrobras-specific functionality, Java offered simpler solutions for the more generic problems such as the construction of user interface. Using Java also offered Tecgraf engineers an opportunity to gain experience with a new technology that was growing in popularity. For many, this dramatically broadened their options for employment if they were to ever leave academia and move into software industry.

While Silvio's project started as a mixture of Lua and Java, it has gravitated towards Java over time. Other projects started at the time were done in pure Java from the beginning. For those users of Lua less committed to the language than Silvio, the break introduced by Lua 4 served as a good opportunity to switch to Java.

Disembedding

This chapter has looked at the history of Lua, seeking to show step-by-step Lua's transition from a specific solution for a particular need of a Brazilian organization to an international programming language that in retrospect may appear to have been "born to fly." As we saw, several factors were important for this transition. Lua was designed by people who were competent in the "global" software practices and could explain their actions in the right language —both in the sense of literal fluency in English, but also in the sense of being fluent in the conceptual system of academic computer science. They had acquired this competence in part through physical travel to foreign centers of computer science research (such as Waterloo and Cambridge), as well as through their access to a local island of computer science research in Brazil (PUC's Department of Informatics), which had been constructed in Brazil through a complicated history (chapter 2.2).

Lua was developed by a group with simultaneous ties to Brazilian industry and Brazilian academia. This mixed origin was crucial for Lua's later success. The language was developed for

300 See appendix C, "Original Interview Quotes," (Silvio, May 2007, "Sem se incomodar com a plataforma").

a practical purpose, which from the beginning set it aside from programming languages designed purely in pursuit of programming language research. At the same time, Lua's origin in an academic institution made it portable. The Brazilian academic computer science community has developed bidirectional links with North American computer science of the kind that have no equivalent in industry. If Lua had been developed by a Brazilian software company without strong academic connections, its chances for success in the foreign industry would have been much reduced.

Understanding Lua's evolution requires paying attention to the gradual changes in the authors' goals for the language. Lua started as a practical project undertaken in an academic environment. This mixed origin introduced a crucial ambiguity to what the project could become. Had Lua been built specifically for the purpose of advancing computer science research, it would have suffered the fate of School—the language Roberto Ierusalimsky was developing as a part of his research, which has been all but forgotten. At the same time, stronger contacts with the local industry would have likely entangled the language in local relationship so closely that as to make its international success quite difficult. Unwilling to seriously pursue the option of commercializing Lua for the local market, the authors made early steps towards globalizing Lua and then made it a topic of academic papers, published both in Brazil and abroad. Foreign publications brought foreign users, helping the authors discover the satisfaction of interacting with a large number of users of software they wrote, who not only expressed gratitude but also had the sophistication to truly understand the virtues of the language—and to push its boundaries. Such satisfaction is of course similar to the one academics often seek when they publish their ideas with the hope that they would be valued by peers. The authors could thus acquire the “perceptions and judgments” necessary for development of free software by starting with a rather similar set of academic “perceptions and judgments,” and then developing such perceptions and judgments gradually while interacting with Lua users.

This chapter has focused on Lua's gradual disembedding from the local context and the increased success abroad. As we saw in the end, this success required breaking some of the local relationships. In the next chapter we turn more closely to this issue, looking at Lua's relation with the university, the city and the country where it remains to this day. We will also look at some of the complexities involved in managing a global project from a university in Rio de Janeiro.