

READING THE FREE MANUAL: FOREIGN KNOWLEDGE IN THE WORK OF BRAZILIAN SOFTWARE DEVELOPERS

Yuri Takhteyev

yuri@sims.berkeley.edu

UC Berkeley School of Information

Abstract: The paper describes the use of free online technical documentation by Brazilian software development based on a 5 month interview study in Rio de Janeiro, Brazil. I argue that use of such technical documentation plays an integral part in modern software development and changes the way technical knowledge is shared both locally and globally. I compare free technical documentation with free/open source software, concluding that the two phenomena share a number of similarities and can be seen as two sides of the continuum of free “functional works.” At the same time, free code and free documentation differ in specific ways that help explain interviewees’ lack of concern with the fact that the documentation they use is not “free as in ‘freedom.’”

Free / open source software has recently attracted a lot of attention both in popular media and academic discussion (see Ducheneaut 2005 for a review). In this paper I will discuss a closely related phenomenon that has similarly transformed the way software is developed – the availability of massive amounts of free documentation on the Internet.¹ While both free software and free documentation are widely used by software engineers in the United States, their use in other countries is particularly interesting as it spans a number of boundaries: geographic, cultural and linguistic. It is also of practical importance since it may help software industries in those countries to catch up with the American software industry.²

The paper discusses some aspects of the use of online documentation and, to a lesser extent, of free code by software developers³ in Rio de Janeiro, Brazil, based on a 5 month interview study. I try

-
- 1 Throughout this paper I will use the term “documentation” to refer to all types of technical content represented in textual form or combination of text and images. This would include: books, magazines, and various types of online documents - FAQs, How Tos, tutorials, forums, articles, technical blogs, PDF books, etc.
 - 2 I say this with caution: after all, as I will show in the paper, free foreign knowledge is fundamentally knowledge about *foreign* technology, relevant only in a context where a foreign technological system has been adopted. It can thus be interpreted as yet another form of colonialism (see Coleman and Hill 2004).
 - 3 People who write software programs are often called “programmers” by outsiders. People who do such work in the United States, however, rarely use this term, and instead call themselves “software developers” or “coders.” Brazilian programmers similarly

to show how the use of textual resources online is integrated into the daily work of software developers, including the ways in which it complements local processes of learning from colleagues. In doing so I want to show the importance of such resources and the extent to which they form an integral part of software development and their role in “importing” knowledge from abroad. In the second part I present a comparison between “free” online documentation and free/open source software. I argue that the two are in an important sense part of a continuum of free “functional works” (Stallman 2001/2002), yet differ in interesting ways. In particular, I address the question of whether “free” documentation as described in this paper is “free” in the same sense as “free software.” I hope that such a comparison would provide insights into both phenomena.

1. THE STUDY

From July to December 2005 I conducted open-ended interviews with 50 software professionals working in Rio de Janeiro to understand what knowledge they need to do their work and how they acquire such knowledge. The interviews lasted between 45 minutes and 2½ hours and included discussion of education and career history, current job functions, and detailed examples of some of the things that the participants learned more recently. The interviewees were recruited using a theory-driven snowball sample, which aimed to include a wide range of work environments and levels of expertise. The sample thus included developers and system analysts working for small and large private software firms, public enterprises and universities, as well as a number of people writing software for academic research. I interviewed people with a wide range of expertise, from self-educated novices to computer science professors with doctoral degrees from outside Brazil. Most of the interviewees, however, either had higher

rarely use the Portuguese term “programador” and more often refer to themselves as “desenvolvedores” (“developers”). (To the best of my knowledge, the term “coder” has no Portuguese equivalent.) I thus use the term “software developers” or “developers” to refer to people who write software. Brazilians often distinguish between “developers” proper and “system analysts” (“analistas de sistemas”), the latter approaching what would be called “software architects” in the US. In this paper I group those together, using the term “developer” for both.

education or were nearing the end of a university program while working full-time. They attended a range of local universities, including PUC-Rio (which has Brazil's top ranked computer science program), local public universities, and lower-ranked for-profit private schools⁴, though with a skew towards higher-ranked schools. Finally, while most of the interviewees were currently directly involved with writing software code, I interviewed a small number of people who have done development in the past but are currently working in managerial roles. In addition to more formal interviews I've conducted less formal conversation with about 20 people involved with Rio software industry in other roles (e.g., funding agencies) to get a wider view of the industry.

The interviews were conducted in English or Portuguese depending on interviewees' level of comfort with spoken English.⁵ Translated quotes from interviewees in Portuguese are accompanied by the original text in a footnote. Quotes without such footnotes come from interviewees done in English.⁶ All Portuguese quotes used in this paper were verified by a native speaker of Brazilian Portuguese. The names used in the paper are pseudonyms. Each interviewee, however, is always identified by the same pseudonym. The pseudonyms are selected from actual names used in Brazil today.⁷

4 As a general rule, public universities in Brazil are seen as providing better education, with private schools taking those students who either could not pass exams into public university or need to work full-time while studying. (Free public education still requires funds for living expenses, transportation and books and is thus off-limits to many lower class youths regardless of their qualifications.) Several prestigious private catholic universities like PUC-Rio provide a notable exception from this rule.

5 As my Portuguese improved over the 5 months of the study, my interviewees grew more reluctant to speak English to me. Thus, some of the interviews that were conducted in English in August probably would have been conducted in Portuguese if they were to happen in November. More generally, interviewees' level of comfort did not always correspond to their English proficiency – some were eager to speak English despite limited skills, others preferred Portuguese despite seemingly speaking fluent English.

6 When transcribing English interviews, I corrected minor grammatical errors, but always kept interviewees choice of vocabulary.

7 I avoided, however, using any of the common pseudo-North European names ending in -son (e.g., “Edmílson”), since such names have a strong lower-class connotation to many Brazilians. (Several of my interviewees had such names, all coming from working class families. Other interviewees of working class origin, however, had traditional Portuguese names.)


2. LEARNING ON THE INTERNET

The interviewed software professionals are nearly unanimous in attributing significant importance to Internet resources in their learning. Many stress this importance quite emphatically. To give just one example out of many, a developer says:


Alvaro: I think that today the Internet is a great vehicle and I use it to learn anything. If you mention something today, I don't know what, I go straight to my micro, Google – enter Google – and read.⁸

Many others describe it just as emphatically. Another developer calls the Internet “the world’s greatest library” containing “all the imaginable and unimaginable resources.”⁹ “The Internet has everything you want,” says yet another interviewee, “You just need to know how to find it.”

Interviewees over 30 often specifically highlight the change that the Internet has brought, contrasting it to their earlier experience. Comparing his current experience to mid 1980s, an interviewee says:

Vinicius: Then if you knew that the person knew about it, you would spend more time trying to talk to him. It’s not necessary anymore. You don’t need to, actually... And again, this is primarily due to the Internet. You can get *any* kind of information you want on the Internet.

In contrast to his current reliance on the Internet, Vinicius made extensive use of situated learning,¹⁰ afforded by the knowledge-rich environment of Brazil’s best computer science department. Those who lacked such opportunities had to get by with quite a bit less. Another interviewee talks about learning to program in Clipper in 1988 using a one page reference card:


Yuri: how did you learn Clipper?

8 Alvaro: Eu acho hoje que a Internet é um grande veículo e eu hoje uso ela, por exemplo, para saber alguma coisa. Voce falar hoje alguma coisa, não sei o que é, ja vou para meu micro, google, tá, entra no Google, e leio.

9 Jamie: Hoje em dia a maior biblioteca mundial é a Internet. A Internet hoje, ela tem todos recusos imaginaveis e imaginaveis.

10 Vinicius’ account of his learning in 1980s fits perfectly with the classical accounts of situated learning, such as Lave and Wenger (1991), Brown and Duguid (1991), or Orr (1996).

Otavio: A reference card.
Yuri: Huh?
Otavio: A reference card.
Yuri (unsure if I understood what he said): What do you mean? A book?
Otavio: No, a little card. A little reference card, you know?
Yuri: Just a card?
Otavio: (Unclear) reference card. There was no material available.
Yuri: But that was sufficient to learn to program Clipper?
Otavio: It would have been easier with better materials, but that's what I had.¹¹

When asked how he learned Perl in late 1990s, Otavio explains that this was quite different – he had the Internet  while no other interviewees talk about getting by with *just a reference card*, many early memories of learning to program involve resources only marginally better: magazine articles, occasional manuals, poorly translated books.¹² While the Internet hardly solved all problems of technical learning, in view of most interviewed developers it clearly made an immense difference.

Easy access to online documentation affects not just the initial learning process early in developers' careers but also their later practice, which involves continued learning as a necessary component. Virtually all developers stress the changing nature of software technology and the need to “keep oneself up-to-date” (“se manter atualizado”). Some make this statement with a caution, clarifying that one needs to distinguish between true changes and mere fads and that they want to see a technology prove itself before investing their time into it; others opt to “ride

11 Yuri: E como você aprendeu o Clipper?

Otavio: Cartão de referencia.

Yuri: Hã?

Otavio: Cartão de referencia.

Yuri: Que significa? Um livro?

Otavio: Não, um cartãozinho. Aquele cartãozinho de referencia, sabe?

Yuri: So uma cartão?

Otavio: (Unclear) cartao de referencia. Não havia material disponivel.

Yuri: Mas isso era bastante para aprender a programar em Clipper?


Otavio: Teria sido mais facil com melhor material, mas era o que eu tinha a mao.

12 Brazilian bookstores rarely sell American books in the original English. Most interviewed developers now “import” the original books through Amazon.com, but doing so was a lot harder in 1980s.

the technology wave,” trying to learn the latest technology and reap the rewards of being among the few people with such expertise. Both “laggards” and “early adopters” (Rogers 1995), however, agree that continued learning is a necessary part of staying in the industry. A developer who combines his day job as a Delphi freelancer with a post-graduate course focused on Java says:

Jaime: The fact that I am out there as a freelancer requires that I always learn new things. That is, I start learning a new thing, start to practice it, to develop and to try to make a living with it. But when I see that this is starting to exhaust itself, I will I try to learn something else. So, why am I here in this post-graduate program? Exactly because of this. Because around more or less five years ago it was one reality. Now the reality is different. Understand? So, what happens is that it leads me to try to learn new things. Hence the fact that I have to be here studying again, so that I could get new knowledge. [...] Because the companies are always asking for new things. At times I come to a company, there the manager of the company asks me: “Do you know how to work with X?” I say “Yes, I can.” Great, so I can develop. Now I come to another place: “Do you work with Y?” “No.” So you are out. Then I have to stop, learn Y.¹³

Compared to other interviewed developers, Jaime may be described as a “late adopter,” as he is still working with Delphi – a technology which, in Jaime’s terms, “is starting to exhaust itself.” Feeling that moving from Delphi to the more popular Java platform might be overdue, Jaime opts for a post-graduate course, rather than attempting to learn Java on his own, as is more commonly the case for early adopters. Even with a formal course, however, much of the learning occurs independently (again, using resources on the Internet), the value of the course

13 Jaime:  fato de eu ‘tar aqui fora, como freelancer, me leva a ‘tar sempre aprendendo coisas novas. Ou, seja, eu começo estudar uma coisa, começo a praticar aquilo, a desenvolver e tentar me sobreviver com aquilo. Mas se eu ‘tou vendo que aquilo esta começando a se esgotar eu vou tentar uma outra coisa diferente. Então, por que estou fazendo o curso aqui no pos-graduação? Exatamente por causa disso. Por que? Porque há coisa de mais ou menos cinco anos atrás a realidade era uma. Hoje a realidade já é outra. Entendeu? Então, está, o que é que acontece, ‘tá me levando a ter que aprender coisas novas. Então daí o fato de eu ter que ‘tar aqui novamente estudando, para poder adquirir novos conhecimentos. [...] Porque as empresas estão sempre pedindo coisas novas. As vezes chego numa empresa, aí o gerente da empresa me pergunta: “Voce sabe trabalhar com a Ferramenta X?” Eu digo: “Sim, consigo.” Beleza, então da para eu poder, da pra eu desenvolver. Agora chego a outra lugar: “Voce trabalha com a Ferramenta Y?” “Não.” Então você está fora. Aí eu tenho que parar, aprender Ferramenta Y.

most often seen as being in the direction that it provides for the students.¹⁴

3. DEVELOPMENT AND RESEARCH

In addition to the learning effort required to master each new technological paradigm (occasional for some, a continuous process for others), the developers face daily “research” (“pesquisa”) while making use of the technology they already know in general, as they try to solve specific problems. When asked what he does on his typical day, a developer explains:

Flavio: Most of the time I am developing. But to develop is not just to write code, right? There is also a large part of searching, studying, looking at specific topics, finding answers to questions, looking for references. There is also this set of things.¹⁵

When I later ask Flavio for a recent example, he says:

Flavio: I’ll talk about something specific, about a problem. I was getting an error while doing a database search using Hibernate. And I couldn’t find where the error was. [...] So, to resolve this, the first thing is to look at the actual product reference documentation, right?

Yuri: Which product?

Flavio: Hibernate itself.

Yuri: On the internet? In a book? What kind of documentation?

Flavio: A document on their own website. An API for the actual Hibernate. Hibernate’s own documentation. On their website. When I didn’t manage to resolve it, I asked the colleague sitting next to me.¹⁶

14 The relationship between formal instruction and self-learning with the help of the Internet warrants an extensive discussion that I do to attempt to provide in this paper. For now I will limit myself to a simpler assertion that materials found on the Internet appear to play an important – if undefined – role even in the context of formal instruction.

15 Flavio: A maior parte do tempo desenvolvendo. Mas desenvolver não é só escrever o código, né? Também tem uma boa parte também de procurar, estudar, ver determinado assunto, tirar dúvidas, procurar referências, né? Também tem um conjunto

16 Flavio: Vou falar concretamente mesmo, por algum problema. Eu estava tendo um erro, quando eu estava fazendo uma busca no banco de dados usando Hibernate. E eu não conseguia identificar onde estava o erro. [...] Então para resolver isso, primeira coisa é tentar achar na própria referência de produto, né?

Yuri: Que produto?

Flavio: Do próprio Hibernate.

Yuri: Na página da Internet? Num livro? Qual tipo de referência?

Flavio: Um documento existente na página deles mesmo. Um API do próprio Hibernate. Uma referência própria de Hibernate. Na página deles mesmo. Como não consegui resolver, perguntei pra colega do lado.

Flavio ends up talking to several colleagues, but none know the answer. He comes back to his seat:

Flavio: So I came back and tried modifying my program in such a way as to avoid this potential error. Then, however, we¹⁷ ended up finding out that it was a problem of incorrect usage, I was actually using a wrong parameter. A problem with [unclear].

Yuri: And you found this by...

Flavio: Going back to the documentation, reading a discussion forum, reading the actual forum of Hibernate.

Yuri: So, you tried to change something in your code, didn't manage, then went to the Internet and started looking...

Flavio: No, I was stubborn. I returned to try to change something and go around this problem, bypass it. Except that I was stubborn and didn't just bypass it. I said: "No, let me look at it again." I wasted a lot of time looking for this this information until I found it.¹⁸

Thus, after failing to resolve the problem based on his own or his colleagues knowledge, Flavio looks on the web, where he finds a satisfactory answer. While he points out that he "wasted a lot of time" looking for the information, the method he uses to find it is so mundane to him that it hardly requires an explanation. Like most interviewees, Flavio talks about web search without actually referring to it explicitly. Web search has become an invisible piece of "infrastructure"¹⁹ of Flavio's work, and the term "looking for information" is thus used as unproblematic. I have to ask him to tell me more specifically where and how he found the solution.

17 While Flavio says "we," it appears from what he says later that he found the solution on his own. Many other developers similarly talk in collective terms about work done individually. This usage seems to express commonality of goals and importance of helping each other, despite the fact that the developers spend most of their time working individually.

18 Flavio: Quando aí eu voltei e tentei modificar a minha programação de forma a evitar esse possível erro. Mas aí depois a gente acabou identificando que era problema de utilização errada, eu tava usando um parametro errado mesmo. Uma problema de [unclear].

Yuri: Você descobri isso por...

Flavio: Voltando a documentação, lendo forum de discussão, lendo forum do proprio Hibernate.

Yuri Então você tentou mudar alguma coisa no seu código, não conseguiu, depois foi a Internet e começou procurar...

Flavio: Não, fui teimoso. Voltei para mudar alguma coisa e passar esse problema, bypassar esse problema. Só que fui teimoso, não bypasssei. Falei: "Não. Deixa eu dar uma olhada de novo." Perdi mais tempo procurando esse informação ate que encontrei.

19 Bowker and Star (1999) provide a discussion of the difficulty of doing an ethnography of "infrastructure."

Flavio: On the actual forum, using Google. A great source of information is always Google.

Yuri: You started with Google and searched...

Flavio: ...for specific information that I would need there. Some phrase that would tell me what was the solution. A parameter (unclear).

Yuri: And this took you to the forum that explained...

Flavio: Yes, inside the actual forum someone had already had this problem and had given a solution which... more or less what I ended up using. And that was it. In fact, I just needed to reformat a parameter, readjust it to resolve this problem. The problem wasn't as complicated as it seemed. It was more of a problem of...²⁰

This part of the story brings out the often unspoken but crucial piece of web infrastructure: the Internet search through Google (yes, it's "always Google"). Web search stitches together into a navigable whole myriad documents authored by various people. In Flavio's case, the solution is found in an Internet forum – a fossil of a conversation that had once occurred in a different part of the world.


Flavio's story presents a typical scenario. In the course of their daily work, engineers face problems that they cannot solve using just their current knowledge. Depending on how much "social capital" they have, they may try make use of expertise of their local colleagues. Not infrequently, however, the colleagues do not know the necessary answer either. A common solution, is then to turn to the textual resources available on the Internet. Importantly, such a solution invariably involves bringing in technical knowledge from abroad, crossing a number of boundaries in the process.

 20 Flavio: No proprio forum mesmo, utilizando o Google. Uma fonte de boas consultas sempre é o Google.

Yuri: Voce comecou com o Google e buscou...

Flavio: Determinadas informações que eu precisaria ali. Alguma frase que me dissesse o que era solução ali. Parametro (unclear).

Yuri: E isso te levou ao forum que explicou que...

Flavio: É, dentro do proprio forum alguem ja tinha tido esse problema e tinha dado uma solução que, proxima à que eu usei. E pronto. Na verdade era um parametro só pra reformatar, pra reformatar e resolveu o problema. Não era nem tão complicado quanto parecia o problema. Era um problema mais de erro de... 

4. CROSSING BOUNDARIES

The sheer distance that separates Brazilians from major centers of software development is striking when one looks at a map. Some of the interviewees who travel frequently, complain about the difficulty involved in going to Europe or United States. Most, interviewed developers, however, have never left Latin America, except perhaps for a trip to Disney World in Florida or sometimes a tourist trip to New York. The physical crossing of space is thus a non-issue as it rarely occurs.²¹ The “virtual” crossing of space via telecommunication lines, on the other hand, happens daily, but usually proceeds unproblematically and developers seem to rarely concern themselves with the geo-spatial origin of the networked resources they use. While many interviewees say they “import” books through Amazon.com and typically refer to those books as “American,” no geographic labels are typically applied to online resources. Such resources are merely “used” or “read” with little attention being paid to the border-crossing involved in accessing them.

The linguistic boundary is crossed in a more noticeable way. Most of the technical documentation is available in English, a foreign language for all interviewed developers. (Materials in Portuguese are typically talked about as either non-existent or inferior .) Ability to read English thus becomes essential for software work and some developers prefix the usual “you can find anything on the Internet” with “if you know English.” All interviewed developers say they read English regularly and do so without difficulty. In fact, they rarely mention the language of the content they read, e.g., Flavio’s story in the previous section, told in Portuguese, makes no mention of the fact that all Hibernate documentation and forums are in English. While it is possible that many Brazilian developers who cannot *speak* English nonetheless read it without difficulties, I suspect that many of my interviewees understate their troubles with English texts, presumably as part of their “impression management” (Goffman 1959) in front of

21 By contrast, many developers travel often (sometimes weekly) to São Paulo, Brazil’s business capital.

a foreigner, since English proficiency is often all but equated with technical competence.²² In some of the more frank conversations, however, even interviewees who spoke good English sometimes reported difficulties with English. Two problems seem to be common. First, many report being able to read the “more technical” texts with ease, but experience difficulty with the more “conceptual” ones. Second, it appears that many interviewees have difficulty *expressing* themselves in English (even in writing), even if they can understand it well. Such asymmetric proficiency perhaps encourages the use of texts already available online instead of more active engagement, e.g., through mailing lists.

Usage of Internet resources also involves crossing a *social* boundary by making use of resources created by “outsiders.” Most interviewees have no direct interaction with non-Brazilians, their only foreign contacts being with friends and relatives who have moved abroad. Personal and professional social ties thus largely stay within the national/cultural community. This is similarly true for *virtual* interactions. Many Java developers talk with enthusiasm about local Java group (“RioJUG”) that hosts both real events and virtual discussion, often using the word “we” (“a gente”) – “we discuss,” “we bring” – and other words evoking a sense of community. Some talk about the community aspects of RioJUG explicitly:

Yuri: And do you know many of the people from the community²³?

Rafael: From the group? Yes. Because when there are big events like Sun Tech Days, everyone gets together, drives in a “caravan,” travel together, stay at the same hotel, understand? It really is a community, understand? Those are people who work in different companies here in Rio de Janeiro, companies that work with Java


22 The idea that good engineers must be able to read English was expressed quite commonly. Similarly, there was no doubt for me that most of my interviewees wanted to make it clear to me that they (and other Brazilians) are technically competent and fully “up to date,” suspecting that the purpose of my study was perhaps to evaluate whether they were behind in technical knowledge. Many also explicitly asserted their ability to speak English. Interestingly, while most developers argued for technical competence of “Brazilians” (in general), those who claimed mastery of English did so just for themselves. In other words, nobody ever told me that “Brazilians” in general read English with ease.

23 In this particular passage I make a mistake of inadvertently introducing the new term “community” into the conversation (which at this point I use just as a synonym for Rafael’s term “group”). Rafael, immediately takes up the term and expands on its meaning (“It really is a community”). His enthusiastic use of the term as well his prior discussion of RioJUG leave me confident, however, that while “community” is my term, it describes his experience quite well,


technology, right?²⁴

Participation in *virtual* discussions on the RioJUG forum extends this social interaction; the “exchange of ideas” in the forum is described with words like “solidarity” and “cooperation.” While getting a beer with a group of Perl programmers, I was told with similar enthusiasm about Cascavel-PM – a Perl mailing list started in a small town in the south of Brazil, that ended up attracting members from many parts of the country. Unlike RioJUG, Cascavel-PM brought together programmers from across Brazil, yet it was also described as a community. In contrast, foreign forums are more often talked about in pragmatic terms as “sources” of knowledge.

Those who take too active of a part in foreign communities may get censured. An interviewee who speaks fluent English and shows a clear preference for foreign forums and IRC channels tells me about a “funny situation” that happened to him:

Antônio  was in a Posgres chat channel and I was asking for some help, it was actually a Gforge issue. Then a guy helped me out with some stuff and then I don't actually recall why we went to this, but we asked “Where are you from?” and then [it turned out that] the guy was Brazilian and he said “Well, why aren't you at the Posgres-Br chat channel?” He said it to me. And I said, “I wasn't even aware there was a Posgres-Br.” It's not that... I don't have anything against it, you know, and I do think I could get very... quality answers from Brazilian forums. It's just that it's not a problem for me to go directly to the source because of the language. So, I tend to go directly to the developers and they tend to be American, or English is the language. But I don't really have anything against it.

Antônio gets censured for not participating in a Brazilian forum and his defensive position (“It's not that I...”) suggests that he has heard this accusation more than once. Importantly, Antônio

24 Yuri  você conhece muitos pessoas da comunidade?

Rafael: Do grupo? Sim, conheço. Porque quando acontece um evento maior como Sun Tech Days o pessoal se organiza, faz uma caravana, vai todo mundo junto, se hospeda no mesmo hotel, entendeu? Realmente é uma comunidade, entendeu? São pessoas que trabalham em empresas diversas aqui no Rio de Janeiro, são empresas que atuam com a tecnologia Java, né? [I am translating Rafael's “o pessoal” as “everybody.” This term, which literally means “the people” or “the presonel,” is most typically used in colloquial Brazilian Portuguese to refer to a group that the speaker associates with and in many contexts may be translated as “we.” This usage parallels the use of “o gente” (“the people”) as the predominant colloquial term for “we.” (The word “nós” is rarely used in informal contexts, at least in Rio de Janeiro.)

justifies his behavior in pragmatic terms: it's not that he has "anything against" the Brazilian community, it's just that he feels he will get better information by going to "the source." The same is true of many other interviewees: those who make use of foreign forums stress their informational utility, while participants in Brazilian forums stress the value of community.

Use of online resources thus involves crossing boundaries, but those boundaries are crossed in a limited way. Brazilian developers regularly venture into the foreign territory to retrieve the missing pieces of a technological system they have imported and adopted. They do not, however, typically engage *as members* in the English-speaking technical community. Instead, they make use of the textual "spillovers"²⁵ of a technological conversation occurring without their participation.²⁶ It is remarkable, however, how far such use of spillover knowledge can go.

5. THE TYPES OF RESOURCES USED

The English-speaking Internet presents the developers with several types of resources, which they often see as useful for different purposes, and below I discuss some of the most commonly cited ones.

5.1 Reference documentation

For many developers like Flavio, reference documentation that "comes with" the product of a framework is a natural place to check when facing a problem. While desktop products often come with reference documentation built into the software, various libraries and frameworks

25 I am deliberately taking the term "spillover" out of its usual context in the diffusion literature.

26 I need to point out, that this is true of most, but not all interviewees. Several of the interviewees worked did academic research and their work seemed well-integrated into international academic discussion (thanks to an academic funding system that rewards foreign publications and, critics say, encourages Brazilian researchers to work on solving American and European problems, rather than focusing on what Brazil's own needs). Additionally, one non-academic interviewee was involved in a leading role with an open source project with substantial non-Brazilian participation and another one was working (alone) on his own operating system which was available on the web together with documentation in English.

employed by developers more often have documentation provided on the web, as is the case with Hibernate – a Java framework used by Flavio. Typical of this genre of documents, Hibernate documentation²⁷ aims to provide a comprehensive overview of the technology. Reading such documentation is thus a frequently mentioned way to learn about a new technology.

Walter: Because there is one thing, one interesting fact: the of-the-shelf software, that software that is sold in a box, American, made in the United States and sold [unclear] – its documentation is very good. It's complete. So, reading software documentation I managed to find solutions for most things that I wanted. Even today it's like this.

Yuri: That was paper documentation?

Walter: No, normally it was documentation on the Internet or on the CD-ROM. What we call “digital media.”²⁸

As Flavio's example shows, however, official reference documentation is rarely “complete” as Walter suggests and often cannot provide answers to all questions faced by the developers (see also Orr 1996). This shortcoming is to a large degree compensated by online forums.

5.2 Forums

In their live form, forums are quite similar to mailing lists²⁹ in that they allow the user to ask a question of a community of people working with the same technology. In the example above, Flavio talks about a forum for users of Hibernate.³⁰ While many interviewed developers

27 <http://hibernate.org/5.html>

28 Walter: “...aquele que tem uma coisa, um fato interessante: o software de caixa, aquele software que vem em caixa, Americano, produzido nos Estados Unidos e vendido [unclear] - a documentação dele é muito boa. Ela é completa. Então, lendo a documentação do software eu consegui a maioria das soluções pro que eu queria. Ainda hoje é assim.

Yuri: Era a documentação em papel?

Walter: Não, geralmente era documentação em Internet ou o CD-ROM. O que nos chamamos “media digital.”

29 The line between “forums” and “mailing lists” is blurry and often depends on the point of view. One typically participates in a mailing list by sending and receiving messages as email; forums typically provide a web interface for both posting and reading. Many mailing lists, however, have archives on the web, and for someone reading the archives the difference between mailing lists and forums largely disappears. Some of the “forums” read by my interviewees thus might actually be “mailing lists” from the perspective of people contributing to them.

30 <http://forum.hibernate.org/>

participated in local mailing lists (in Portuguese), few report asking questions on forums outside Brazil (typically in English) such as the Hibernate forum mentioned by Flavio. (Those who do ask questions typically do so only infrequently and usually cannot recall the last time they actually did so.) While this may be partly due to the difficulty of formulating questions in English (which some interviewees openly admit), many stress that this is simply a matter of timing: asking a question on a forum means waiting for a few hours or even days for an answer.

The true value of the forums comes from their fossilized form: the archives. Searchable through Google, forum archives provide a vast store of quick solutions to a myriad of specific problems faced by software developers world-wide. A Google search on error codes, function names, or specific keywords thus often yields an answer. It is this relative ease of finding prior solutions that reduces the necessity of asking questions in forums. After mentioning that he learned a lot from Qualcomm forums, an interviewee clarifies that this did not involve any “direct” interaction:

Yuri: You never sent them questions?

Francisco: No, not directly. What I needed I found there already in the forums in the questions and answers that were there.

Yuri: And you didn't ask questions in the forums?

Francisco: No, never got to that. I never got to trying this. Because I never needed to. Because for everything that I needed at the time there already were answers.

Yuri: In general that's what you do?

Francisco: In general that's what I do. [...] In general there are... It's like... (Laughs.) I've never had a problem that nobody else had, understand? I've never had a problem that nobody else ever had or resolved.³¹

31 Yuri: Nunca mandou perguntas para eles?

Francisco: Não, diretamente não. O que precisei achei já em forums em perguntas e respostas que tinha.

Yuri: E você não fez perguntas em forums?

Francisco: Não, não cheguei a fazer isso. Não cheguei a tentar. Até porque não precisei, porque tudo que precisei na época já tinha respostas.

Yuri: Em geral você faz isso?

Francisco: Em geral faço isso. [...] Geralmente existe... Tipo assim... (Laughs.) Nunca tive um problema que ninguém nunca teve, entendeu? Eu nunca tive um problema que ninguém nunca teve ou resolveu.

5.3 Tutorials

Online reference documentation suffers from another shortcoming, however, which forums cannot resolve. As many authors have pointed out, “codified” or propositional knowledge contained in textual documents cannot be trivially turned into working knowledge. While some communities impart practical knowledge primarily through co-present participation in joint projects (Lave and Wenger 1991), interviewed software engineers generally stress the importance of individual practical experience - “doing” (“fazendo”) and “tinkering” (“mexendo”).


Alan: I believe it is by doing that you learn.

Yuri: What does that mean?

Alan: It means you have a problem and you are going to go and make a program. When you get a book and go read it – you are going to understand the book. Then when you start doing, start programming, you are going to see that what you read isn’t good enough. You need to learn more, to actually do it. Same with my project. I started reading, read a lot, and thought that already knew everything. And when I went to do it, I saw that I knew nothing. I had to learn much, much, much more. It’s by doing that you learn.³²

This recognition that just reading is not enough and the stress on practical experience is quite common. Contrary to the radical rejection of textual sources reported by some authors, however, the developers typically attribute significant importance to documentation. Reading is typically the first step – crucial but not sufficient for learning. “Reading and tinkering” (“lendo e mexendo”) is how some of the interviewees summarize their learning process.

A particular genre of technical documentation - a *tutorial* or *how-to* - explicitly aims to

32 Alan:  acredito que é fazendo que voce aprende.

Yuri: Que significa isso?

Alan: Significa você tem um problema e aí voce vai fazer um programa. Então se você pegar um livro e sair lendo – você vai entender o livro. Aí quando você começar a fazer, começar a programar você vai ver que na verdade o que você leu não é bem aquilo. Você precisa aprender mais, vai fazendo. Que nem no meu projeto. Eu comecei lendo, li muito, aí achava que já sabia tudo. E quando fui fazer vi que não sabia nada. Tive que aprender muito, muito, muito mais. É fazendo que você aprende.

instill *practical* knowledge by combining “reading and tinkering” and using the text to guide learner’s individual experience – an approach that seems to fit well with the idea (expressed by many developers) that while one can only get practical experience by actually *doing* it, one can do so more efficiently by “following the path” that has been walked by others before.³³ Looking at a tutorial like “The Road to Hibernate” by Michael Glögl,³⁴ we see that it consists of modules guiding the user through a sequence of steps. E.g. “The First Hibernate Application”³⁵ walks the user through the steps necessary to build a basic application. The user is expected to follow along, *actually constructing the application*, as the following quote from “The Road to Hibernate” shows:

Let's assume we want to have a small application where we can store events we want to attend and who hosts these events. We decide we want to use Hibernate for storage, because we heard it is the coolest thing in persistence :)

So the first thing we will do is set up our development directory and put all the jar-Files we need in it. We have to download the Hibernate distribution from the [Hibernate download page](#). Extract the needed jars from the hibernate archive. We will place them all in a lib directory under the development dir, so your devel directory should now look like this...

The tutorial assists the user in this process, however, by clearly laying out the steps, explaining important concepts and providing most of the code the learner will need:

So finally we can start building our first application. For convenience, we create a batch file in our development directory which contains all commands necessary for compilation. Under Windows, this would look like this:

```
javac -classpath .\lib\hibernate3.jar -d bin src\de\gloegl\road2hibernate\*.java
copy /Y src\hibernate.cfg.xml bin
copy /Y src\de\gloegl\road2hibernate\*.xml bin\de\gloegl\road2hibernate
```

We place this file called build.bat in our development directory. If you are using Linux, you can surely create an equivalent shell script :)

33 This link is not made explicitly by any of the interviewees. Many developers make references to tutorials and many talk – separately – about acquiring practical knowledge by “following the path.”

34 This tutorial, available at <http://www.gloegl.de/5.html> was not mentioned explicitly by any of the interviewees, but is simply the first hit on a Google search for “Hibernate tutorial,” making it a likely destination for anyone looking for a tutorial on Hibernate. I introduce it here in order to stick with Hibernate, which was mentioned as a new (and useful) framework by many Java developers I interviewed.

35 <http://www.gloegl.de/8.html>

The tutorial thus allows the learner to get “personal” experience doing work that they might have trouble doing on their own.³⁶

5.4 Books

The interviewees often refer to books as a source of “deeper” (“mais a fundo”), more foundational knowledge of a subject. This term is used to refer to both printed books and books online, including books available for download as PDF files as well as online books formatted as multi-page websites. What defines “books” is not the medium, but length and depth of treatment of the topic.

Yuri: Do you use books?

Jorge: I have tons of books.

Yuri: But do you use them much?

Jorge: The books have a problem that they are outdated. But for the initial kick-off you need a book. For example, you’ve got a totally new technology, as in my time was J2EE. This technology is a huge, huge, huge thing. If you don’t read a book, no Internet will help you. Because in the book the information is organized to explain to you. The main goal of the book is to pass a message to you. So, you need a book. So, the information written in a book, understand, it is important because it will guide you through this thing that you don’t know. [...] The book has to engulf the topic. It can leave out no part of the topic.³⁷

Jorge argues that “[n]o Internet will help you,” until you have gotten the general understanding

36 Vygotsky (1930/2002) introduces the concept of “zone of proximal development” (“ZPD”) as a set of tasks that a child cannot do by herself but can successfully perform with a help from adult and argues that most learning happens precisely when the child engages in tasks in ZPD. Online tutorials similarly offer a set of training wheels to allow the learner to perform tasks that are beyond their current knowledge.

37 Yuri: Você usa livros?

Jorge: Livros, é. Tenho uma carrada de livros.

Yuri: Mas você usa eles muito?

Jorge: Os livros tem um problema de que eles são muito desatualizados, né? Mas o pontapé inicial tem que ser um livro. Você tem, por exemplo, uma tecnologia totalmente nova, por exemplo como foi no meu tempo J2EE. Essa tecnologia é um negocio enorme, enorme, enorme! Se você não pegar num livro, não tem Internet que dê jeito. Porque num livro a informação ‘tá organizada para te explicar. O objetivo principal do livro é te passar aquela mensagem. Então tem que ter um livro. Então a informação escrita num livro, entendeu, ela é importante, porque ela vai te guiar numa coisa que você não sabe. [...] Porque o livro ele foi feito a proposito de dar visão global do conhecimento que você precisa. [...] Um livro, ele tem que englobar um assunto. Ele não pode deixar nem um pedaço do assunto de fora.

of the territory that can be provided only by text that “leaves out no piece of the topic.” Many interviewees agree that a certain level of understanding cannot be achieved without a book, though most prefer to start with online documentation and only use books for technology that clearly merits such serious understanding.

While Jorge primarily talks about paper books, he points out that a book can be a book on the Internet – what matters is whether the text “engulfs” the topic. Other interviewees, in fact use the term “books” only to later clarify that they mean PDF files.


Jaime: But this [looking up things on the internet] happens very often, it’s not just... Sometimes I go: “How does this work? How to implement?” Some times the book, it only gives you some theoretical part, but from here you have to dig there, research on the Internet and then you find it.


Yuri (having misunderstood what Jaime just said): So, you never use books?

Jaime: No, I do use books, I do use books. But... The truth is I use both: the books and the... But mostly books they have on the Internet – I download a lot. There are those public books, you know?³⁸

Like Jorge, Jaime defines a “book” in terms of genre, rather than medium. (In fact, his preferred medium differs from Jorge’s.)

Still, even as a strong minority of interviewees make regular use of downloaded books, books are more commonly thought of as being on paper. The more comprehensive knowledge thus often needs to be purchased and shipped from abroad, and the questions of depth, cost and time thus get intertwined. Explaining how he learns new technology, a developer stresses the importance of learning *quickly* and, at the same time, avoiding unnecessary cost:

Hugo  would download the software, install it. And of course I would use extensively “how-tos” - as short as possible, the shortest tutorials I would be able to

38 Jaime:  isso é muito frequente, não é só... Às vezes: “Como funciona? Como implementar?” Às vezes o livro ele só dá aquela parte teórica, mas daí você futuca lá, pesquisa na Internet e consegue.

Yuri: Então, você nunca uso livros?

Jaime: Não, uso livros, uso livros. Mas... Na verdade uso as duas coisas: o livro e a... Mas principalmente os livros que tem na Internet - eu baixo muito, né? Tem os livros publicos, né?

achieve. If those are not enough, have not enough detail, I would go to a second kind of tutorial, a little more detailed. It's a more objective, it's more [unclear] kind of tutorial. I need to learn and to learn fast. I don't want to go to the bones of the thing.

Yuri: And you wouldn't want to go get a book?

Hugo: No, the book is my last resort. Because I don't see a point in paying for something you can have for free.

Hugo's sudden switch from "learning fast" to saving money highlights the similarity between the those two kinds of cost. While books are generally seen as a venue towards better understanding of the field, in view of most interviewees the pace of technological change ("books get outdated") and deadline pressures makes such deeper understanding not worth the time it takes to truly read the book and the money one must pay for it.

6. THE EFFECTS ON SOCIAL LEARNING

Like Vinicius, quoted earlier, many interviewees point out that easy access to material available on the Internet reduces the need to make use of knowledge of co-workers and friends. Most interviewees say they prefer to start with an Internet search, rather than asking a colleague or a friend. Three common reasons are often mentioned:³⁹


1. Most interviewees say they prefer not to "distract" or "bother" others with questions. "The worst thing is the person who asks questions all the time," says one of the interviewees, explaining that he tries hard to not be such a person and, like many others, reports keeping a mental counter of how many times he has approached others with questions. This sentiment can be understood as a desire to "conserve" social capital that one expends when asking others for help. It also arises from the fact that research is seen as a key part of engineering work (see Flavio's quote above), and there is thus a thin line between learning from colleagues and asking them to do your work for you.

³⁹ I discuss those reasons in somewhat more detail in a different draft paper (Takhteyev 2005).

2. Many of the questions they face are outside the expertise available locally. Those who start by asking their colleagues (like Flavio does in an earlier episode), often come back empty-handed.
3. Searching on the Internet often simply takes less time than asking a colleague. When asked whether he usually asks friends searches for a solution online, a developer replies with a rhetorical question: “What is easier? Search on the Internet!” Engineers who share an office with others sometimes say that turning around to ask the colleague might be easier than looking it up online, but find that getting up and walking to another room (or picking up a phone) already requires more work than a “quick” Internet search.

The use of the Internet, however, doesn't necessarily *replace* local contacts or make them irrelevant (though some engineers say so). The developers find interaction with friends and colleagues valuable for several reasons, including the help they can provide with ill-defined problems (which cannot be trivially turned into a search query) and their opinions when choosing among potential solutions (since colleagues are trusted more than anonymous web pages). In most cases, however, easy access to textual documentations *changes* how people use local resources, arguably allowing them to extract *more* value out of other people's help. Two most notable differences are the practice of researching the topic on the Internet before asking questions and the exchange of references as an alternative to an extended conversation.

One of the ways to get the most out of face-to-face interaction with colleagues is to do one's “homework” before asking questions:

Luciano:  if I had somebody to talk to, I would start with Google, then ask for help.

Author: Why?

Luciano: I think to get more prepared for receiving the information through other people.


Author: What do you mean by “more prepared”?

Luciano: To know what he is talking about. Some concepts he might talk... to get

acquainted with the concepts. For example, “frequency,” “sampling”, for example. And when he tries to explain something to me he might use concepts that I must be acquainted to, the technical terms he is talking about.

In a reversal of a common assumption in Communities of Practice literature that one acquires basic meanings through co-present understanding, which then allows one to interpret “codified” knowledge (e.g., Duguid 2005), Luciano talks about going to the Internet for the basic notions in order to “be prepared for receiving the information” in a face-to-face interaction.

Additionally, discussion with colleagues can become more condensed due to the possibility of referring to easily accessible resources. In fact, in some cases, a just a few words uttered by a colleague may be sufficient. One developer talks about struggling to find a way to test his web application by putting it under a heavy simulated load. He eventually ends up talking to some colleagues while getting coffee in the office:

Wanderley:  and then another colleague, one of those colleagues said: “Well, you can use JMeter, it's a project inside Apache Group, Jakarta Apache.” And then, well, I have already heard about JMeter. And then I put “JMeter” into Google, of course. One of the first items that occurred in Google was Jakarta Apache JMeter page and then I started to look for more information about JMeter.

Wanderley’s story seem to fit a common “water cooler conversation” formula, except for one striking divergence: Wanderley’s colleague’s help involves little more than uttering a few words.

There is shared understanding that a single keyword “JMeter” is sufficient for finding ample textual documentation for the details of the proposed solution and Wanderley’s friend can thus help him with almost no effort. Absent such documentation, he would need to spend substantial time telling Wanderley what to do – an effort he might have chosen not to undertake.

The engineers also often relied on friends and colleagues to learn about new technologies. Here again, easy access to public documentation removes the need to discuss details:


Vinicius: Eventually they would say, hey [Vinicius], have you heard the new, I don't know, language that is coming out. Or this new technology that Microsoft, or Intel, or Sun is releasing. And so if it is something that clicks, if it is something you have interest in, you go and try to find information.

“Finding information” of course means doing a Google search. The ease of sharing keywords and links means that developers can share quite a few of them.

7. FREE TEXTS AND FREE CODE

Use of textual resources online as a source of knowledge in a technical practice poses a number of interesting questions.⁴⁰ For the rest of this paper I want focus on a particular aspect of the use of such resources – the fact that in an overwhelming majority of cases they are “free” and discuss the “free” status of such documentation in comparison with the phenomenon of “free software.”

The study described in this paper did not aim to specifically look at the use of open source software. In the process of talking about learning, however, software developers routinely talked about their use of “free”⁴¹ code. Their search for such software is motivated by the recognition that one can find ready solutions to many problems and save oneself time:

Wanderley:  important to know about libraries that one finds there on the Internet. Because there are many free libraries that people use and that are very well developed and well used. It's important to have knowledge about these things because you gain time. It's not good if you try to re-invent the wheel. So I think it's important to have knowledge about what is available online, what can be used, and just to do what's necessary to glue all the parts and make them work together.

The desire to avoid “reinventing the wheel” is similarly stressed by quite a few developers and

40 Not the least of those question is why such texts are understandable (or, how they are made meaningful) in a context of a different culture and how they relate to developers “situated” use of such texts (Suchman 1987). I believe, that the answer to this question lies in the fact that while socially and geographically isolated, Brazilian software engineers are solidly integrated into the “hybrid” / “heterogenous” network of software technoscience (Latour 1987), through their use of foreign hardware and software and their employment by firms that have business relations with foreign actors. I do not, however, intend to treat this question at length in this paper.

41 Their notion of “free software” does not necessarily coincide with “open source.” This question will be addressed below.

appears to be a part of a more general programmer ethos.⁴²

It is my impression, however, that the developers do not search for software or code. They search for *solutions to problems*. What they expect to find is a special kind of texts – what Stallman calls “functional works – that is, works whose use is to get a job done” (2001/2002). Such solutions may come in the form of software, documentation, or – most often – a combination of the two. While end-user software with a graphical interface is typically designed to be usable without a manual, this is rarely the case for software libraries – the form of software most commonly used by developers. Undocumented libraries may be nearly useless, and developers often talk about choosing specific libraries because they are well documented. In an extreme case, however, the code itself may be used as a form of documentation. One interviewee talks about the difficulty of using some of the new features of the Java programming environment faced by his research group:

Almiro: At the moment they are having difficulty understanding what are “template tables.” They already went into the discussion forum but haven’t gotten any feedback on this. Because there is no literature. The guy from Sun⁴³ himself said: “Look, the best source of knowledge is the source code.⁴⁴ There is nothing else.”⁴⁵

Reading even well-commented source code, however, is usually an unrewarding task and is typically seen as a last resort.⁴⁶

42 Many interviewees say they are “lazy” (“preguiçoso”) and thus try to re-use existing solutions. The term “lazy” is used with pride in this case – a sign of being smart enough to avoid unnecessary work. The idea that programmers are “lazy” in this positive sense is similarly common among American engineers and was famously expressed by Eric Raymond, one of the main spokesmen of the “hacker” community: “Good programmers know what to write. Great ones know what to rewrite (and reuse). [...] An important trait of the great ones is constructive laziness. They know that you get an A not for effort but for results, and that it’s almost always easier to start from a good partial solution than from nothing at all” (Raymond 1999).

43 Sun Microsystems is the company that provides Java.

44 While Java is not open source, its source code is available to some organizations, conditional on signing a non-disclosure agreement. Almiro’s group thus has access to Java source code and was reading it to learn how to make proper use of “template tables.”

45 Almiro: “No momento eles estão com dificuldade de entender o que é o template tables. Já entraram no fórum de discussão mas não consegue ter feedback disso. Porque não há literatura. O próprio cara da Sun falou: “Olha, o melhor fonte de conhecimento é o source code. Não tem outro.”

46 Unrewarding as it may be, being able to use the source code in lieu of documentation is considered an important skill, famously captured in the English-speaking programmer community with the phrase “Use the source, Luke” (a pun on “Use

Technical documentation, on the other hand, often comes sprinkled with code. Programming books, forum posts, and tutorials all usually embed pieces of code, either as an illustration or as an actual solution to a specific problem. Tutorials, in fact, often provide *all* the code necessary to create sample applications, even as they space it out with discussion. The Hibernate Tutorial mentioned above contains plenty of code throughout the text, e.g.:

At first we modify the main-Method:

```
public static void main(String[] args) throws java.text.ParseException {
    EventManager instance = new EventManager();
    if (args[0].equals("store")) {
        String title = args[1];
        Date theDate = new Date();
        instance.store(title, theDate);
    }
    System.exit(0);
}
```

So what we do here is read some arguments from the command line, and if the first argument to our application is store, we take the second argument as the title, create a new Date and pass both to the `store` method, where it gets really interesting...

Additionally, the page provides a link to a zip file with all the code, so the user doesn't actually have to put the application together in the order prescribed by the tutorial.

The link between code and readable texts is also quite present in the rhetoric of the free software movement. The first link is metaphorical: readable texts are often used as a metaphor for code to explain why code needs to be free. E.g., Stallman frequently compares code to cooking recipes, suggesting that preventing people from sharing code is just as unnatural as preventing them from sharing recipes (e.g., Stallman 1992/2002 and 2001/2002). The same point is made in Stallman's "The Right to Read" (1997), an anti-utopian sketch describing a future where people go to jail for sharing books and presenting a clear, if unexpressed, comparison to the situation with software copyright. The use of books and recipes as a metaphor for software shows that at least in Stallman's view, the Free Software movement is primarily

the Force, Luke" in Star Wars), featured in "The Jargon File" (Raymond 2003). Additionally, while reading externally-written code may be a last resort, reading code written by co-workers is usually a fact of life – many engineers talk about "inheriting" source code that others have worked on before. Since each library written inside the company is usually tightly integrated with everything else, throwing it just because it isn't well documented is rarely an option.

about removing restrictions on sharing technical *knowledge* broadly, rather than sharing software specifically. At the same time, Stallman groups together “recipes, computer programs, manuals and textbooks, and reference works like dictionaries and encyclopedias” under the heading of “functional works” - works that can be used to solve technical problems (Stallman 2001/2002, p. 141). Documentation and code are thus both ambiguous between being knowledge-imparting works and “functional” works.

The second link is provided by the more narrow practical concern. In a short essay entitled “Free Software Needs Free Documentation” Stallman argues that “[d]ocumentation is an essential part of any software package” and that “when an important free software package does not come with a free manual, that is a major gap” (Stallman 2000/2002, p. 67). As I will discuss later, Stallman’s notion of what would constitute “free documentation” is different from the “free” resources used by my interviewees.

8. FREE CODE HOT AND COLD

Code and documentation are, in fact, best seen as two extremes of a continuum of codified solutions. The distinction that defines this continuum is the degree of completeness, which appears to often be inversely related to didactic usefulness: a library provides the developer with a relatively quick solution to the problem, but often gives little technical insight, documentation helps the developer solve the problem by themselves, often spending much more time, but sometimes gaining a generally useful knowledge.

In “Understanding Media,” Marshall McLuhan distinguishes between two kinds of media: “hot” and “cool” (McLuhan 1964). Hot media provide information in “high definition” leaving little to imagination. “Cool media,” on the other hand, provide “meager amount of information.” With cool media thus

...much has to be filled in by the listener. On the other hand, hot media do not leave so

much to be filled in or completed by the audience. Hot media are, therefore, low in participation, and cool media are high in participation.

McLuhan's distinction between "hot" and "cool" media is useful for understanding the code-documentation continuum. "Hot" code provides a complete solution, leaving little to be filled by the user. "Cool" documentation provides the user with a "meager amount of information," relying on the user to fill in the details.

As a clear example of "hot" piece of open code, consider the Firefox web browser. Most users will just download the binaries and use the software "as is." Few will tinker with it, since most will find no reason to do so – the browser works well immediately after installation. Of those who will try to tinker, many will discover that doing so is quite difficult. An example of "hot" code, the Firefox browser is characterized by two related features: completeness and complexity. The browser is complete and ready to be used. This completeness comes at the price of high complexity: with over 2 million lines of code,⁴⁷ the Firefox browser is hardly a project that a beginner can tinker with.⁴⁸ The Hibernate Tutorial, mentioned above, on the other hand, provides the developer with code for only a basic application – unlikely to directly solve any real problem. The tutorial attempts to give the reader enough understanding of how this simple application is built, however, to enable them to construct solutions to their own problems. Firefox is thus "hot" and "low on participation" while the Hibernate Tutorial is "cool" and "high on participation."

"Hot" code is of course not free of interpretation. As any technology, the Firefox browser can be shown to have "interpretive flexibility" (Bijker 1995) and designers' ability to predict or predetermine users activity has been legitimately questioned (Suchman 1987). To say that "hot" code is used "as is" is not to say that it is used "as intended," but rather that its use and interpretations do not require access to its *textual* form. Without questioning the value of

47 http://news.zdnet.com/2100-1009_22-5632148.html

48 Many aspects of browser's high level behavior can be modified through Javascript extensions. This, however, gives the tinkerer only a limited understanding of what happens inside the browser.

“machine as text” metaphor (Woolgar 1991), I want to highlight the fact that Firefox is *literally* comprised of about 100,000 pages of human-readable text. That text can in principle be read as a detailed manual on how to build a web browser. Typical users, however, do not read that text directly but will only *run* it on their computer. Furthermore, those interested in web browser design may be better served by texts that were intended to be read by people rather than computers. At the same time, we can find a range of texts fitting in-between: texts that include software code that is meant to be both run *and* read. Depending on its context, Javascript code can be as much *about* Firefox as it can *a part* of it. E.g., a Firefox tutorial⁴⁹ includes code snippet that contains the following line:

```
var button = document.getElementById("okbutton");
```


When run as part of a script this line serves as an instruction to the Javascript engine to get a reference to an object (the “OK” button) from another object (the document) and assign it a symbolic name (“button”). In the context of the tutorial, however, the line can be read by a human developer as saying that a particular type of object (an “element” of a page) can be accessed by passing a particular type of message (“getElementById”) to another object, answering a question “How can I get access to a button?” In other words, the line becomes a statement *about* features provide by the Javascript engine rather than an instruction to be executed by the latter.

9. FREE SOFTWARE BEGETS FREE MANUAL

Stallman’s second link between code and documentation suggests that free code needs free documentation to be useful. Indeed, as was mentioned above, interviewed developers often use quality of documentation as important factor in deciding whether to make use of a new technology – whether open or proprietary. Five years after Stallman’s manifesto on the need of

49 <http://www.xulplanet.com/tutorials/xultu/events.html>


free documentation (Stallman 2000/2002), they hardly share his observation that free software is lacking documentation. In fact, free or at least “more open” software is frequently described as having *better* documentation. In contrast to developers using open source frameworks like Hibernate, interviewees working with closed technology complain about lack of documentation, which they typically link with the closed nature of the product:

Miguel  would search for] something like “one portal is blocking the other [platform name].” But most of the times it’s not very successful. It’s a very proprietary, very specific kind of platform. And that’s something that sucks, to be using something like this.

Another developer mentions having trouble finding support on one of the more “closed” development environments for mobile phones:

Francisco: No, it’s not easy. Today it is, but then [two years ago] it wasn’t easy. At the time, the best reference was Qualcomm’s own site. Because, unlike J2ME, BREW is a proprietary technology of Qualcomm. J2ME is a more open thing. There are many people, there is a huge community of Java programmers, which there isn’t for BREW. So, finding information wasn’t [unclear], but if you had a problem, you had to scavenge for information, had to search a lot.⁵⁰

Note that both BREW and J2ME are technically proprietary (neither is free software / open source). As many other developers, however, Francisco feels that J2ME is relatively open when compared to other alternatives.⁵¹ He suggests that this openness leads to a larger developer community, which then results in more resources available online.

50 Francisco , não é muito fácil. Hoje é, mas na época não era muito fácil não. Na época, a maior referencia que tinha era o site da Qualcomm. Porque diferente do J2ME, o BREW é uma tecnologia proprietaria da Qualcomm. J2ME é uma coisa mais aberta. É da Sun, mas existe várias pessoas, existe uma comunidade enorme de programadores de Java, que não existe do BREW. Então, conseguir informação não [unclear], mas se você tiver um problema, tinha que ralar a informação, tinha que procurar muito.

51 J2ME is based on Java, whose quasi-open status has been a source of contention within the free software / open source community. Stallman has argued that Sun’s implementation of Java is “non-free” and called for the developers to avoid “the Java Trap” (Stallman 2004). Stallman’s response itself, however, serves as clear evidence for the prevalence of the opinion that Java is “free” or at least “open enough” to provide an acceptable basis for free software development. This debate is not limited to United States – while attending a “software livre” conference in Rio de Janeiro, I witnessed a discussion provoked by the inclusion of a Java developer on a panel on “free programming languages.”

10. FREEDOM AND COST

When calling for free documentation, Stallman clarifies that “[f]ree documentation, like free software, is a matter of liberty, not price” (2000/2002, p. 67), referring to his own now near legendary quote: “‘Free software’ is a matter of liberty, not price. To understand the concept, you should think of ‘free’ as in ‘free speech,’ not as in ‘free beer’” (Stallman 1996/2002). Most of the “free documentation” discussed in this paper so far, however, is not always “free” in Stallman’s sense. The Hibernate Tutorial mentioned earlier is explicitly dedicated to the public domain, representing the success of Creative Commons⁵² in convincing web authors to explicitly wave some rights. (The hibernate tutorial above links the words “public domain” to the Creative Commons’ public domain dedication.⁵³) Such explicit wavers, however, are still rare. While many of the authors might not actually attempt (or even desire) to enforce their copyright, most of the technical content is protected by copyright and does not grant the reader freedoms that Stallman argued for. Interviewed developers, however, appear to have little concern for whether the documentation they use is free in Stallman’s sense. In this section I address the question of why this might be the case.

Even in case of software, interviewed developers rarely discuss the “free as in freedom” aspects of open source software. More often, they talk about the ability to make use of software and documentation free of charge and without having to obtain a special permission. The freedom to modify software and redistribute changes are rarely mentioned. This may reflect the fact that only a handful of interviewed developers have ever contributed to an open source project, while almost all were users of open source software.⁵⁴ (Similarly, while all developers

52 <http://creativecommons.org>

53 <http://creativecommons.org/licenses/publicdomain/>

54 I do not know whether the rate of participation in open source is different between Brazilian and American or European software developers. According to a recent Softex study, however, most of Brazilian open source contributors are network technicians and support managers rather than professional software developers (Softex 2005, p. 22).

make wide use of free documentation online, only a very small number ever contributed to it.⁵⁵) Thus, while the Four Freedoms outlines in the Free Software Foundation's definition of "free software" may be important in understanding *production* of such software, when we look at the experience of the majority of people who *use* this software, the word "free" doesn't mean the same thing.⁵⁶

The issue of price is brought up repeatedly by the developers, when talking about both code and documentation. This should not be surprising, since Brazilian developers and the companies they work for earn substantially less than their American counterparts, and buying American books or software – typically priced with the American market in mind – is an expensive proposition.⁵⁷ One might thus say, that Brazilian developers primarily think of "free" software "as in 'free beer.'" Such characterization would be unfair, however. Stallman deliberately chooses the term "free beer" to attach a negative connotation to the concern with cost (especially when compared with the more noble concern with liberty). Some of the developers I interviewed, however, argue for the importance of free software as an affordable foundation for software development in Brazil and, potentially, the future of the country. This is hardly the same concern with saving a few dollars that Stallman aims to capture with the term "free beer."

Direct cost is not the only concern, however. Free solution typically means that the developer's decision to use it requires little if any approval by the management.⁵⁸ For this

55 On the other hand, a surprising number of interviewees were involved in teaching (typically night classes in private universities). Some talked about teaching as a matter of "giving back" to their society.

56 This focus on cost leads the developers to include in their notion of "free" software that does not fit into FSF's definition of "free software" or OSI's definition of "open source." In particular, many talk about Java as "free," despite the fact that Java is distributed as binaries and the source code is only available with a non-disclosure agreement.

57 Almost all interviewees said they preferred to buy books through Amazon.com or other US online stores. Translated books available in most bookstores in Rio were generally dismissed as worthless. (Though, *someone* must buy them.) Books written by Brazilians typically didn't even deserve a mention, except for a few *published in English and outside Brazil*. (One of the interviewees was working on such a book.) "Importing" books through Amazon.com was also generally seen as easy to do, apart from cost. There was disagreement on whether such books were "expensive," however: some found the cost prohibitive, others weren't particularly concerned with it. The majority bought foreign books when necessary, but considered cost an important factor and thought twice before buying.

58 I am not sure if their use of free software necessarily fits with the requirements of GPL and other open source licenses, as they typically develop proprietary code. A lot of the software is distributed under licenses that allow one to build proprietary

reason, some developers feel more confident in having a chance to use the same free solutions again later, outside their current job. Their investment into learning the solution is thus more worthwhile in the long term:


Edmundo: Because if you build a product that is independent of [Edmundo's company] or of another company, you need to use some pieces to make this product go, make it work. If you have knowledge, background in those open source solutions, you have a greater capacity, greater possibility to use those products – open source – in your new enterprise.⁵⁹

While Edmundo's company – a well-known multi-national – gives Edmundo exposure to some of the most advanced technology in his field, he might not be able to make direct use of this proprietary technology when he leaves.

Lack of restrictions on who can access software and documentation also increases the ease with which it can be shared. When one can easily forward a link to a resource or even just the keywords necessary for locating it, it makes little practical difference if one has the legal right to distribute the resource itself. In other words, semi-open platforms like Java and most of the “free” documentation give the user enough freedom to “help their neighbor” (Stallman's “freedom 2”).

Apart from the reasons why Brazilian developers might not be interested in some of the more libertarian readings of either free software or free documentations, there are additional reasons that apply specifically to documentation. Code encountered by the developer can often be used “as is” - the developer can literally copy-and-paste the code into their application or link it with a downloaded library. The use of borrowed code is thus usually unambiguous. Code downloaded from the Internet literally becomes part of the developed system and any doubts

applications with it. Even for “copyleft” libraries or code distributed without clear (and thus copyrighted by default), license violations are less likely to be enforced, giving developers a certain level of comfort.

59 Edmundo:  que você quando bola um produto que é independente de [Edmundo's company] ou de outra empresa, você tem que usar algumas coisas para aquele produto sair, funcionar. Se você tiver um conhecimento, um background nesses open sources, você tem grande capacidade, uma grande possibilidade de usar esses produtos – open source – no teu novo empreendimento.

about its legal status affect the system as a whole. Clear licensing of code available on the web is thus quite important.⁶⁰ Documentation, on the other hand, cannot be inserted into a system “as is,” but necessarily needs to be understood and turned into working code. While one can envision (as Stallman does) documentation authors building directly on top of earlier documentation, reusing pieces of old texts just like developers reuse old code, this does not seem to be common, and probably not because of licensing restrictions. I would propose two reasons. First, documentation becomes obsolete quicker than code. Thus, while hundreds of billions of lines written in the COBOL programming language in 1960s and 1970s are still running in their original form, a young engineer who wanted to work with COBOL today might be better served by using a more recent COBOL manual, which would explain COBOL in modern terms and showing how COBOL might be different from the more recent programming environments. It can also avoid discussion of outdated technologies like punch-cards which no doubt were mentioned in the original COBOL manuals. Importantly, a COBOL manual *can* be substituted easily, while even small changes in the COBOL libraries may make thousands of systems stop working. Second, different parts of documentation can be “linked” together with much more flexibility than different code libraries. An author wanting to re-use such a foundational text as Kernighan and Ritchie's *The C Programming Language* (usually affectionately referred to as “K&R”), which is copyrighted and not available online, has a number of options: she can quote small passages (under “fair use”), summarize some topics, present some of the ideas in her own words, or suggest that the reader consult a copy of K&R. She can also collect errata in the book and publish them on a website⁶¹. This range of methods for linking new and old work is not available when linking code, where small inconsistencies would lead to failure of the system.

60 Most of the larger open source projects typically license their code clearly, typically choosing between GPL or BSD license. Smaller snippets of code, however, especially those embedded in documentation, are frequently lack explicit licenses.

61 See for an example <http://cm.bell-labs.com/cm/cs/cbook/2ediffs.html>.

11. CONCLUSION

Brazilian software developers make regular use of resources on the Internet when looking for solutions to technical problems. Such resources span a spectrum from complete software packages, to forums discussions, tutorials and books. All of those resources typically involve some combination of code and textual discussion. The use of software and documentation is thus intertwined in the experience of software developers. I argue that free software and documentation may be best seen as two extremes of a continuum of free “functional works.”

With both software and documentation, the developers display strong preference for “free” solutions. Their notion of “free,” however, does not necessarily correspond to the common definitions of “free software” and “open source,” especially in case of “free” documentation. Cost is a key consideration, as is the ability to make use of the solution both now and in the future. Free access to solutions is also key in supporting their dissemination, making it possible for engineers to advise each other to “just google” for this or that. Wider adoption of free software also leads to growth of communities around them and wider availability of free documentation around it. The relative lack of interest in documentation that is “free as in ‘freedom,’” however, is also explained by the different use of the “code” and “documentation” sides of the continuum of functional works.

12. BIBLIOGRAPHY

Becker, G. S. (1962) "Investment in human capital: a theoretical analysis," *The Journal of Political Economy*, 70(5).

Bijker, Wiebe (1995). *Of bicycles, bakelites and bulbs: towards a theory of sociotechnical change*, Cambridge, MA: The MIT Press.

- Bowker, Geoffrey C. and Susan Leigh Star (1999). *Sorting Things Out: Classification and Its Consequences*, Cambridge: MIT Press.
- Brown, John S., Allan Collins and Paul Duguid (1989). "Situated cognition and the culture of learning." *Educational Researcher*, 18(1), 32-42.
- Brown, John Seely and Paul Duguid (1991). "Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation," *Organization Science*, 2, 40-57.
- Coleman, Biella and Benjamin Hill (2004). "How Free Become Open and Everything Else Under the Sun," *M/C: A Journal of Media and Culture*, 7.
- Ducheneaut, Nicolas (2005). "Socializing in an open source software community: A socio-technical analysis," *Computer Supported Cooperative Work*, 14, pp. 323-368.
- Duguid, Paul (2005) "'The Art of Knowing': Social and tacit dimensions of knowledge and the limits of the community of practice," *The Information Society*, 21(2), pp. 109-118.
- Goffman, Erving (1959). *The Presentation of Self in Everyday Life*, New York: Anchor, Doubleday.
- Lave, Jean and Etienne Wenger (1991). *Situated Learning: Legitimate Peripheral Participation*, Cambridge: Cambridge University Press.
- Latour, Bruno (1987). *Science in Action: How to Follow Scientists and Engineers Through Society*, Boston: Harvard University Press.
- McLuhan, Marshall (1964). *Understanding media: the extensions of man*, The New American Library, New York.
- Orr, Julian (1996). *Talking About Machines: An Ethnography of a Modern Job*. Ithica, NY:

Cornell University Press, 1996.

Raymond, Eric S. (1999). "The Cathedral and Bazaar," in: *The Cathedral and The Bazaar*, Sebastopol, CA: O'Reilly and Associates.

Raymond, Eric S. (2003). "UTSL," in: *The Jargon File*, v. 4.4.7, <http://www.catb.org/~esr/jargon/html/U/UTSL.html>, last accessed Feb. 16, 2006.

Rogers, Everett M. (1995). *Diffusion of Innovations* (Fourth Edition), New York, Free Press.

Softex (Associação para Promoção da Excelência do Software Brasileiro). (2005). Impact of the Free Software and Open Source on the Software Industry in Brazil, Softex report, http://observatorio.softex.br/components/com_observatorio/arquivos/Softex%20ingles%20para%20site.pdf, last accessed Feb. 15, 2006.

Stallman, Richard M. (1992/2002). "Why Software Should Be Free," in: J. Gay, ed. *Free Software Free Society: Selected Essays of Richard M. Stallman*, Boston: GNU Press. Also at <http://www.gnu.org/philosophy/shouldbefree.html>, last accessed Feb. 20, 2006.

Stallman, Richard M. (1992/2002). "Why Software Should Be Free," in: J. Gay, ed. *Free Software Free Society: Selected Essays of Richard M. Stallman*, Boston: GNU Press. Also at <http://www.gnu.org/philosophy/shouldbefree.html>, last accessed Feb. 20, 2006.

Stallman, Richard M. (1996/2002). "Free Software Definition," in: J. Gay, ed. *Free Software Free Society: Selected Essays of Richard M. Stallman*, Boston: GNU Press. Also at <http://www.gnu.org/philosophy/free-sw.html>, last accessed Feb. 20, 2006.

Stallman, Richard M. (1997). "The Right to Read," *Communications of the ACM*, 40(2). Also at <http://www.gnu.org/philosophy/right-to-read.html>, last accessed Feb. 21, 2006.

Stallman, Richard M. (2000/2002). "Free Software Needs Free Documentation," in: J. Gay, ed. *Free Software Free Society: Selected Essays of Richard M. Stallman*, Boston: GNU

Press. The same text is also available at <http://www.gnu.org/philosophy/free-doc.html> (last accessed Feb. 20, 2006) under a title “Free Software and Free Manuals.”

Stallman, Richard M. (2004). “Free But Shackled - The Java Trap,” available at <http://www.gnu.org/philosophy/java-trap.html>, last accessed Feb. 22, 2006.

Suchman, Lucy (1987). *Plans and Situated Actions: The Problem of Human-Machine Communication*, Cambridge: Cambridge University Press.

Takhteyev, Yuri V. (2005). “Googling across the Equator” (draft) http://www.takhteyev.org/papers/googling_across/

Vygotsky, Lev S. (1930/2002). “Орудие и знак в развитии ребенка.” [Orudie i znak v razvitii rebenka - Tool and sign in child development] in Vygotsky, L.S., *Психология [Psikhologiya - Psychology]*. Moscow: EKSMO-Press.

Woolgar, Steve (1991). “Configuring the user: The case of usability trials.” In J. Law (ed.), *A Sociology of Monsters: Essays on Power, Technology and Domination*, London & New York: Routledge, pp. 58-99.