Developing Scientific Software: The Role of the Internet

Aleksandra Pawlik, Judith Segal, Helen Sharp, and Marian Petre

The Computing Department, The Open University

In this article, we describe how scientists use the Internet when they develop scientific software. We present the findings from 27 interviews with scientists-developers and professional software developers who develop scientific software. Based on the analysis of the empirical data, we discuss three aspects of the role of the Internet in scientific software development practice: 1) the use of the Internet in addressing the gaps in scientists' software development knowledge, 2) the use of network-based tools and methods to manage the software development process, and 3) communication between scientists-developers via the Internet.

Introduction

Software development is an inseparable part of research in many scientific domains. As research progresses, it raises new questions and challenges that the existing software may not be able to address. At the same time, advanced domain knowledge is necessary to understand what the software is supposed to do. For these two reasons, in many cases scientists develop scientific software themselves. The software that they develop is not their primary goal but rather the means to an end (Basili et al., 2008; Kelly, 2007; Segal, 2009).

That ultimate end is to progress scientists' own research (Segal, 2007). Typically, these scientists do not have the same background in software development that professional software developers would have (Sanders & Kelly, 2008; Segal, 2008). Understanding the software development practices of scientists may reveal ways in which software engineers and scientists themselves can better support development of scientific software. As the Internet is more and more present in the research world, the focus of this paper is on the role of the Internet in scientists' software development practice. Since the study had an exploratory character, we did not design it on the basis of any pre-existing theoretical assumptions. The aim was to provide a broad understanding of how the Internet fits into scientific software development.

A brief scan of related publications reveals that there is very little literature that discusses the role of the Internet in the software development practices of scientists. Research about the Internet and science mainly discusses the ways the Internet facilitates collaboration between scientists and scientific institutions (e.g., Olson et al., 2008). When it comes to software development practices, it has been noted that scientists advance their knowledge about software development in various ways, including self-study resources available on the Internet (Sanders & Kelly, 2008). In 2006, Wilson noted that scientists shared and discussed their source code primarily via e-mails. However, more recent studies (Hannay et al., 2009; Nguyen-Hoan et al., 2010) indicate that other tools dedicated to supporting software development, such as version control systems, are in use by scientists.

Our study provides insight into and information about the role of the Internet in scientific software development practice. We believe that our work will address the dearth of literature on scientific development practices and the Internet.

Methodology

Since the aim of our study was to explore and understand real-life practices of scientists developing software, we used a qualitative approach (Robson, 2002). The methodology of our study follows the guidance for using interviews in software engineering research as described by Seaman (1999). We used semi-structured interviews to obtain the data that would answer our research questions and to explore the research area, allowing the information to emerge from the data. The interviewer always had an interview guide to make sure that no topic that we wanted to cover was omitted. All interviews were recorded and transcribed. We interviewed 27 participants, out of whom:

- 20 were scientists who were developing software that was also used by other scientists,
- two were scientists who developed software for their own use (however they were happy to share their code if they were asked),
- two were scientists with a degree in computer science developing software used by other scientists,
- one was a professional software developer engaged in a scientific software development project,
- two were scientists who mainly used scientific software developed by other scientists.

The last two participants' views and opinions were informative in our study as these scientists-users often had to refer to the same resources as the scientists who developed the software. The opinions of these two participants helped us to form a better overview of the role of the Internet in scientific software development.

The data collected during the interviews were then analyzed using thematic analysis (Boyatzis, 1998); that is, the data were coded using codes formed during a bottom-up analysis, and then these codes were grouped to form themes. The themes then helped to articulate the findings of this study.

Findings

In this section, we will present the findings from our study. First, we will discuss the role of the Internet in progressing software development skills among the scientists. Then we present the findings on how the Internet facilitates a scientific software development process and how it supports communication between scientists-developers.

Addressing Gaps in Software Development Knowledge

Our study showed that the Internet was one of the main resources used for addressing gaps in the scientists' software development knowledge. Out of 27 interviewed scientists, 24 were either almost exclusively self-taught when it comes to software development skills or combined self-teaching with a one- or two-semester course in programming. These courses were usually a part of their undergraduate degree. The self-teaching was based on multiple materials, and among these materials, the Internet appeared to be one of the main sources of information. The way these online resources were used varied. It could be regular checking up on whether there are any new solutions in the existing technologies that the scientists used. As one of the participants reported:

> Every now and then I go through it [Python doc] and I go, "Hey, there's a new trick [that] has [been] developed in Python that I want to be using"—mainly the online resources.

Another strategy to progress one's knowledge related to software development is to go straight to a search engine when, for example, an issue with source code appears. Google seems to serve well as the first reference point that can be used to find the answer. In fact, as one of the participants reported, Google was probably the only place he could seek support:

> What I would do, I would write some code, run it, [and] when I get an error I would then take the error and put that error into Google and then see what the error was. Then if I got anything that was reasonable as an answer back, I would try to figure that out. Try the solutions that were on the Internet. Other than that, there wasn't really much more I could do or many more resources I could use.

This strategy of pasting in the code of the error or the error message itself (whether it was an output from a compiler or a runtime error) was common for scientists-developers with varying levels of experience in software development. Two participants said that they would rather check the details of programming language syntax or a particular compiler output on Google than use a textbook on their desk.

The participants did not explicitly discuss the ways they evaluate the reliability of the sources they use. Only two interviewees explained to us their methods for assessing the correctness of the implementation solution or particular syntax details that they found online. As one of them put it:

> If it comes down to coding, resources for C++ or something like that, either it works or it doesn't. Whether it's trustworthy or not doesn't matter to me because I will run a test that will test the program and check a bit and get some output additional information to check if the command works as they claim probably double check against other sites [the interviewee means other online resources], if it agrees and gives the same information.

Our findings show that the Internet is used on a regular basis by scientists in order to address the gaps in their software development knowledge. The scientists appear to simply choose solutions that are "good enough" to address an issue with which they are dealing at a given moment. This finding is consistent with the model of scientific software development proposed by Segal (2008).

Use of Network-Based Tools and Methods Facilitating the Software Development Process

The Internet provides a number of tools and methods to help manage scientific software development projects as well as to support the work coordination within the development team. Sixteen of our participants explicitly discussed how they use the Internet in coordinating their work. The ability of managing tasks via the Internet seems to be an ideal fit for scientific software development projects, which often represent collaborations between groups or individuals at different physical locations. Twenty-four of our participants were involved in projects in which the development team was distributed. Even if the core development team was co-located and the developers could discuss everything in person, there were usually external source code contributors or people writing documentation who did not even work in the same country.

The approaches to Internet-supported management ranged from using advanced tools dedicated to managing projects to less formal management via e-mails. One of the participants described in detail how Trac (i.e., a project management system) was adapted for one of the software development projects in which he was engaged:

> We described what the project was trying to achieve. We divided that into tasks for each task. We put the name of the person who was by principle the leader of the task. They could nominate other people to actually share the work and then associate tickets with tasks. And we had milestones, so for a demonstration or deliverable we had a milestone. The milestone then was based on completing the tickets.

Another participant described how his team tried out different tools to address the needs of their project. These systems were used both for managing the project and also for managing the project Web site content.

> We started off using media-wiki. […] for the project website we used the system [the user could not remember the exact name], which is fine, but it's more for managing small projects and labs. It's great for secretaries to use. But to interact with it problematically is a little bit more tricky, and its authentication realm is different. This was OK, but we wanted something better. We tried media-wiki and then went to Google-wiki. I think now we're going to change to the system called Drupal [www.drupal.org].

Summarizing, the scientists use a variety of network-based tools and methods to facilitate the software development process. These tools and methods seem to be suitable for scientific software development projects in which scientists-developers are often not co-located. Additionally, these tools and methods encourage scientists-developers to prioritize their tasks, assign roles and responsibilities, and set up goals and milestones.

Communication via the Internet between Scientists-Developers

Out of 25 of our participants who were developing scientific software, only one was exclusively a solo developer at the time of the interview. All others were engaged in projects that included other developers. In all cases, communication among the developers of scientific software was carried out via the Internet.

E-mails were the most commonly used means of communication. All participants who were developers either explicitly mentioned using e-mails while working on the software, or we found implicit evidence for that (for example, during the interview, the scientists showed us an example of a design document circulated via e-mail).

E-mails were used for a number of purposes: for sending bits of source code, for circulating minutes of meetings, for discussing details of implementation, and so on. If a given project had other tools supporting development management and collaboration, these tools and e-mails tended to be used for different purposes. As one of the participants described it:

> When something needs to be acted upon, we all have an e-mail dialog if it's not supposed to be permanent, if it's just the decisions required. But if it's more permanent, then it will all go on a Web site and we like these content management systems.

Apart from exchanging e-mails, the scientists also intensively used other tools such as Skype, Internet messengers, mailing lists, or even a forum to communicate with others engaged in the same project. For projects in which scientists-developers were not co-located, communication via the Internet was very frequent. In cases when the core development team was co-located, communicating via the Internet was not that often; however, it did take place, such as when one of the core developers was temporarily at a different location or an "occasional" developer needed to consult the main team.

Discussion

In this section, we will discuss the benefits and challenges of the use of the Internet in scientists' software development practices. We will also provide some suggestions on the ways some of these challenges could be addressed.

Addressing Gaps in Software Development Knowledge

The immediate availability of information on the Internet addressing the gaps in scientists' software development knowledge may be very useful for them. There may be two potential reasons for this usefulness. First, for scientists-developers, software is usually only a means to an end rather than the main goal itself (Segal, 2007). This means that as soon as they find a solution for their software development problem, they can get back to their main task, which is advancing their research. Hence, scientists prefer a solution that is simply "good enough" over a more sophisticated and flexible one. Second, scientists typically lack formal education and experience in software development that professional software developers have. The Internet may possibly help them to find a solution, even if they are not sure what type of solution they want and where to look for it. When they get stuck with a compiler or runtime error, they can search for a solution via an Internet search engine (most often using Google); instead of spending hours trying different fixes and consulting textbooks or colleagues, a scientist may go online to find a solution to a bug.

However, there may be certain challenges related to using the Internet for addressing the gaps in the scientists' knowledge about software development. First, there is the question of the appropriateness of the solutions that the scientists-developers find online. When it comes to fixing problems with implementation, one of the reported strategies was to "see if it works and (sometimes) compare the solution with other resources." Assuming that this strategy actually gives some confidence in the particular solution, there is still uncertainty on how a quick fix may affect other parts of the software or how it may affect further software development and maintenance. A quick fix that deals with a given bug may generate problems in the future. Second, there is the question of trustworthiness and reliability of the sources from which the scientists learn about various aspects of software development. In our research, the scientists did not discuss to any extent which sources were, according to them, trustworthy and reliable. The participants generally did not describe how they assess if the Internet resources that they use suit their needs best. We cannot be sure whether this is due to the fact that the verification process was so obvious for them that they did not even think about mentioning it or whether they simply accepted a solution that was "good enough" (Segal, 2007). Summarizing, the Internet can certainly be a great help for progressing one's skills in scientific software development, but more research about how to mitigate the risks discussed above is needed. The first step could be finding ways to raise scientists' awareness of these risks. Another approach could be finding an effective way of creating and running "knowledge centers" that could gather information about reliable resources useful for scientists developing software. Such knowledge centers could also be platforms for exchanging ideas and experiences among scientists-developers. Some actions to gather the information useful for scientists-developers and to assist them in communication between each other have been already taken by the Software Sustainability Institute (SSI) in the United Kingdom. This institute organizes workshops and trainings for scientists developing software as well as for professional software developers who are involved in scientific software projects. SSI also collects information about sources used and recommended by the scientific software community. These sources are entered into a freely available "knowledge base."

Use of Network-Based Tools and Methods Facilitating the Software Development Process

Many tools and methods that support the software development process may help the scientists-developers to organize and coordinate their work in scientific software projects. The participants in our research mentioned that they were engaged in some projects in which the software development process itself was not clearly laid out and organized. In our study, we observed that introducing tools such as a version control or Trac imposes the assigning of roles and responsibilities and requires setting up milestones and goals. These changes, if they took place, were perceived as advantageous by our participants.

The challenge related to the use of network-based tools and methods facilitating software development process is that their variety may become overwhelming. Setting up, for example, a version control, an issue tracker, and a wiki will be meaningless if the developers involved in the project do not use them. In fact, as one of our participants commented, if a scientist-developer is involved in five projects at a time and all five of them have a wiki or an issue tracker, it is highly likely that he will not contribute to any of them. And scientists-developers may sometimes be involved in multiple projects at the same time. We think that more research is needed to establish a balance between using various tools that support the software development process and the actual needs and character of a given scientific software development project. Maybe it could be possible to create a kind of step-by-step evaluation of project's needs that could help scientists-developers to identify which tools would be useful for them.

Communication via the Internet among Scientists-Developers

The main benefit of communication via the Internet is the fact that it may accommodate the distributed nature of most scientific software development projects. Scientists who are physically based at different institutions in different locations can communicate every day, not only verbally discussing various matters (as it could be done over the phone) but also sharing source code, demonstrating program behavior using a shared desktop, or having a team teleconference.

The main deficit of communication via the Internet is that it cannot replace face-to-face communication. It may be tempting to assume that in the Internet era, scientists-developers' meetings in person are unnecessary and since they are often time- and finance-consuming, such meetings should be cancelled. But this assumption may be wrong. One of our participants who worked in a project engaging scientists from not only different countries but also different continents said that he found Internet communication only effective with people whom he knew in person. Hence, as he reported, he spent about half of his time traveling between different locations to discuss the developed software and, what was equally important for him, to establish working relationships with other developers.

From what the participant said, it was clear that these relationships could be built and strengthened thanks to the informal interaction and the possibility to talk in person with other developers. This finding is consistent with Olson and Olson (2000), who found that some elements that are very important in collaborative software projects such as "informal hall time before and after [the formal meeting]," "implicit cues," and "spatiality of reference" are not and cannot be well supported by technology. The authors pointed out that in projects in which the elements of this collaboration are present (that is, in face-to-face interaction), there are fewer misunderstandings and productivity tends to be better. Improved data flow, whether it is sending a massive piece of source code, raw data, or a team videoconference stream, cannot entirely replace face-to-face communication. It is important to raise awareness among scientists-developers that despite having more and more powerful collaboration and communication tools, they still need to plan for meetings in person.

Conclusion

Our study provides evidence that the Internet can bring many benefits to scientific software development practices. It may help scientists-developers to keep focus on their main aim, which is advancing their research by easing the process of software development. The Internet is a vast source of knowledge that is easily accessible. This may help scientists to speed up the progress with software development, and the saved time may be allocated to advancing their research. The Internet supports collaboration and communication in scientific software development projects that tend to be of distributed nature. At the same time, our findings indicate that there are some risks involved when it comes to using the Internet in scientific software development. These risks may not seem very apparent and obvious at first glance, but they may in fact have a negative long-term impact on scientific software.

References

BASILI, V. R., CARVER, J. C., CRUZES, D., HOCHSTEIN, L. M., HOLLINGSWORTH, J. K., SHULL, F., & ZELKOWITZ, M. V. (2008). Understanding the high-performance-computing community: A software engineer's perspective. *IEEE Software*, *25*(4), 29-36.

BOYATZIS, R. E. (1998). *Thematic analysis: Coding as a process for transforming qualitative information*. Thousand Oaks, California: Sage Publications.

HANNAY, J. E., MACLEOD, C., SINGER, J., LANGTANGEN, H. P., PFAHL, D., & WILSON, G. (2009). How do scientists develop and use scientific software? In *Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science, Vancouver* (pp. 1-8). Washington, DC: IEEE Computer Society. Retrieved November 11, 2012 from http://tinyurl.com/atqnr5l

KELLY, D. (2007). A software chasm: Software engineering and scientific computing. *IEEE Software*, *24*(6), 120-119.

NGUYEN-HOAN, L., FLINT, S., & SANKARANARAYANA, R. (2010). A survey of scientific software development. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (Art. 10). New York: ACM Press.

OLSON, G. M., & OLSON, J. S. (2000). Distance matters. *Human-Computer Interactions*, *15*(2), 139-178.

OLSON, G. M., ZIMMERMAN, A., & BOS, N. (Eds.). (2008). *Scientific collaboration on the Internet*. Cambridge, MA: MIT Press.

ROBSON, C. (2002). *Real world research: A resource for social scientists and practitioner-researchers*. Malden: Blackwell.

SANDERS, R., & KELLY, D. (2008). Dealing with risk in scientific software development. *IEEE Software*, *25*(4), 21-28.

SEAMAN, C. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, *25*(4), 557-572.

SEGAL, J. (2007, September 23-27). Some problems of professional end user developers. Paper presented at IEEE Symposium on Visual Languages and Human-Centric Computing, Idaho. Retrieved October 3, 2012 from http://oro.open.ac.uk/17674/1/segal-ProfessionalEndUserDevelopers.pdf

SEGAL, J. (2008, May 13). Models of scientific software development. Paper presented at First International Workshop on Software Engineering in Computational Science and Engineering, Leipzig. Retrieved October 3, 2012 from http://oro.open.ac.uk/17673/1/SegalICSE08R.pdf

SEGAL, J. (2009). Software development cultures and cooperation problems: A field study of the early stages of development of software for a scientific community. *Computer Supported Cooperative Work (CSCW)*, *18*(5-6), 581-606.

WILSON, G. (2006). Where's the real bottleneck in scientific computing? Scientists would do well to pick up some tools widely used in the software industry. *American Scientist*, *94*(1), 5.