Addressing Challenges to Open Source Collaboration With the Semantic Web

Anupriya Ankolekar
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213
anupriya@cs.cmu.edu

James D. Herbsleb
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213
jdh@cs.cmu.edu

Katia Sycara
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213
katia@cs.cmu.edu

Abstract

Despite the remarkable success of open source software, there are a number of challenges to collaboration in open source software development, in particular, with respect to supporting collaboration among developers, supporting potential contributors, and in bringing users and developers together. In this paper, we examine some of the possible enhancements of open source development environments, and consider the application of Semantic Web technology to address these.

1. Introduction

Because it has proven remarkably successful in circumstances that are extremely challenging for traditional development methods and environments, open source software development has received the attention of many researchers within the software engineering community [4]. With almost no face to face communication, and very little use of industry-style project management and coordination, open source developers have built a variety of widely-used, reliable, well-known software systems, e.g. the Apache web server and the Mozilla browser [14].

Open source software development has been described as "extreme distributed software development" [14]. The community around an open source software project is usually located around the globe and interacts primarily through asynchronous textual modes of communication, such as email and discussion boards, that are logged in publicly browsable archives. Although open source software projects can vary considerably in their particulars, they do possess a few typical features. Every successful open source software project has a community of people involved with the project at various levels. The largest group within the community is usually the user community, which is primarily interested in using the software. Some users report bugs, but that is more commonly the domain of a smaller group of 'contributors'. Contributors are not only users of the software, but are also interested in the general development of the project. They are likely to download the most recent (possibly unstable) versions of the software, actively report bugs, and submit code, either to fix bugs, provide further enhancements to the software or to contribute patches. At the centre of the community lies a small select group, sometimes even a single person, of 'core' developers, who not only contribute code, but also guide the project by reviewing contributed code and selecting a subset to be committed to an 'official' release of the software.

There are also a number of tools and practices that are typically found in open source software projects, although each project customizes these to fit its own requirements and culture. A version control system, such as CVS, is used to maintain code, through which anyone can browse, however, usually only core developers can commit code [7]. Bugs and feature requests are tracked by means of an issue tracking system such as Bugzilla. Besides these code management tools, open source software projects use a number of tools for collaboration and coordination. The primary ones are group mailing lists, asynchronous discussion forums and more recently, chat facilities. Despite the considerable success of open source software projects, there are a number of ways in which open source development environments could be improved, particularly with respect to supporting collaboration among developers, supporting new and potential contributors, and in bringing users and developers together.

In this paper, we examine some of the possible enhancements of open source development environments, and consider the application of Semantic Web technology to address these. The rest of the paper is organized as follows: Section 2 examines literature on geographically distributed software development in commercial environments. We speculate that open source development, as one type of geographically distributed development, suffers some of the same limitations on collaboration, and might benefit from tools that address these problems. Section 3 discusses the Semantic Web approach, a promising set of technologies to improve the processing of information on the Web. We then discuss, in Section 4, the use of semantic annotations and their processing in constructing tools that address the

challenges of collaboration in open source software environments.

2. Open source challenges

While open source practices and tools have been remarkably successful, we believe there are several areas where there are opportunities for improvement: supporting collaboration among subsets of developers, supporting new developers, and supporting the broader user community by heightening awareness of the needs of non-developer users.

2.1. Collaboration

Software development is a closely-coupled activity, often with tight integration and interdependencies between modules, and therefore requires a substantial amount of coordination and communication between developers [10] if they are to collaborate on features. Geographically distributed collaborative work, on the other hand, tends to result in significantly reduced communication between team members [1, 12]. One of the biggest needs in distance collaboration is for more awareness of the activities of developers at remote sites. The lack of awareness stems both from reduced communication and from missing contextual information, which is naturally and implicitly shared in co-located contexts, but is difficult to obtain in distributed work. In general, unlike their counterparts in commercial environments, most open source software developers work relatively independently of one another, which makes it difficult to add substantial new features requiring close coordination among a team. This tension between the needs and the capabilities of distributed software development environments leads, in commercial development, to misunderstanding, miscommunication and coordination problems during integration [9]. Improved collaborative capabilities might allow developers to work together more effectively.

Another problem in geographically distributed developments is the difficulty in finding and consulting experts at remote locations. For ‘uncertain’ projects, informal communication has been found to be particularly important [13]. Open source software projects almost epitomize uncertain projects, since the objectives and trajectory of the project are often ill-specified. However, informal communication is nearly absent in open source communities. In commercial environments, this appears to cause development to take much longer when it is split across sites [10]. Fostering social interaction and supporting the social structures within the open source software community would encourage informal communication and provide a context for community members to interact and share information [6].

2.2. Support for new developers

Open source software project websites are primarily tailored to meet the needs of core developers. New or 'peripheral' developers, who would like to contribute to a project, are in the unenviable position of having to understand a part of the project well enough to contribute, without the mentoring that would take place in co-located contexts and with little support in the form of documentation, tutorials or guides. Some of the larger projects, such as Mozilla and Apache Cocoon [18], do invest effort into providing documentation of code and community procedures to an extent. However, the creation and maintenance of this documentation requires more resources than most open source projects can afford.

Furthermore, often new developers have to not only understand the code, but also the community, practices, and culture of that particular open source software project, which can vary considerably from project to project. Making the social networks of the community apparent would help new developers understand the culture and community they want to contribute to and ease their initiation.

2.3. Feedback from users

Finally, project websites often have relatively little to offer users of the software. It has often been speculated that open source development is not closely attuned to the needs of the larger, non-developer community of users. Early versions of Linux were extremely difficult for non-technical users to install and use. Although there are many channels by means of which a community can influence the course of development [17], if it is the goal of an open source development effort to build software for general users, stimulating more interaction among user and developer communities would be quite useful. It would both help to increase the technical sophistication of the user community and heighten the sensitivity of the developers.

2.4. Common themes

Although these may seem to be a rather diverse set of challenges, they do have a common underlying issue. In each case, there is a need to get the right information to the right person for the current task, and to present it in an understandable, usable way. This suggests that better integration and presentation of information from the various tools in open source projects may address some of the limitations of open source software development. Furthermore, increased visibility of the social network within a project community and improved possibilities for social interaction between members of the community

would ease a new developer's initiation into the culture and the community of a project.

3. The semantic web

Semantic web [8] technologies can directly address the need for better integration and presentation of information in open source collaboration tools. The semantic web allows semi- or unstructured information to be processed based on some representation of its content. This requires ontologies, annotations, and software (agents) that use these to process the information.

Several semantic web standards [2, 20], such as XML, RDF, RDF-S and OWL, have been defined for annotations, definitions of ontologies and ontological inferencing. To understand how ontologies, annotations and agents can work together in a semantically annotated and linked web of documents, consider an example which demonstrates a possible (and nearly realised) application of semantic web technology. Many semantic web conferences now publish their technical program, with the schedule of talks, on the Web, annotated with respect to several different Calendar ontologies [19]. In the following, we visualize a scenario where data marked up in ontologies such as these can provide useful information.

Imagine that you are going to a conference that has published its semantically annotated schedule. Each talk in the schedule refers to an Event concept in the ontology. The Calendar ontology describes an Event concept as having a Name, Venue, Time, Duration etc. Furthermore, the concept PaperTalk, in a separate Conference Schedule ontology, is defined as a subclass of an Event. In addition to the attributes of an Event, PaperTalks are known to have an Abstract, Keywords, and a list of Authors, each of which is a Person. The schedule specifies the location of ontologies, whose concepts are used in the annotation tags. When you point your agent, similar to the Retsina Calendar Agent [16] to the URL of the schedule, it first downloads the ontologies referred to by the schedule. The agent then parses the schedule with respect to these ontologies and displays a list of talks with a summary of each talk and links to the venue and the authors of the corresponding papers.

Now say you have an interest in knowledge capture and representation and are keen to identify the conference talks that are in this area. You ask the agent to filter the list and only display the talks that have knowledge representation and knowledge capture as Keywords in their Abstract. Each of the talks displayed would have links to the Authors. You click on the name of one of the Authors and you can see their Status (Professor, Researcher, Student etc.), Affiliation and Contact Details along with a list of other attributes. Clicking on Affiliation shows you a list of people presenting at the conference who are also part of the same institution. At the conference you meet with a few authors you would like to keep in touch with. You pull up the annotated schedule once again, click on the name of the author and ask your agent to add the author to your contact list. Since the Author is known to be a Person and a Person can be added to a list of Contacts, your agent adds the contact details of the Author to your contact list.

Thus, explicit description of the semantic content allows information from different sources to be gathered and usefully processed by agents. In the following section, we explore how the Semantic Web approach can be used specifically to address the integration issues of open source collaboration.

4. The semantic web for integrating information in open source environments

Integrating information for an open source community can occur at several levels. A basic level of integration simply pulls together existing resources, such as information from different tools on the project site, and links them appropriately for presentation to the user. The next level of integration is at the `computed object' level, where data available on the web site is aggregated and analyzed to dynamically provide new information about the activities and changing state of the project and community. At the final level of integration, software agents go beyond the project website, to find, integrate, and present resources from the broader community elsewhere on the Web.

4.1. Integrating information across tools

There are numerous sources of information in an open source project, from discussions within mailing lists to bug reports within the issue tracking system to code maintained by the version control system. At the simplest level of integration, information from these various tools can be organized and presented together, such that they provide a useful summary of the current activities within the project and its associated community. Hipikat [3] is an example of a tool that assembles information from these sources specifically to help an newcomer make a change in the code. It consists of a plug-in to the open source IDE Eclipse that builds a group memory from the development artifacts of a project and recommends relevant portions for a piece of code.

In order to use the Semantic Web technologies to integrate information across tools, several ontologies need to be created. First, an ontology that describes the structure of the tools is needed. An ontology that describes Bugzilla, for example, would contain the concept of a Bug Report. Each BugReport would have several attributes, such as ID, Component (indicating

which part of the code contains the bug), Description, Severity and so on. In addition, an ontology that describes the domain will be required. Such an ontology would contain, for instance, that a Browser has several Components, such as the Layout, the Parser, the Rendering etc. Finally, one would need an ontology that describes the structure of the code. This ontology would describe the code in terms of its modules and each module as a collection of files.

Annotating information gathered from tools such as CVS and Bugzilla with respect to the tool ontologies is rather straightforward. These tools contain enough formatting and structure information to allow the annotation to take place automatically. Automatic annotation of more unstructured documents, such as mail exchanges, or the annotation of documents with respect to domain ontologies is difficult at such a fine-grained level. Instead, given a document to be classified, information retrieval techniques can be used to identify the keywords of the document. These keywords will then mapped onto the closest known ontological concepts and the document is classified as being in that category.

Consider, for instance, the maintenance of the issue-tracking system, such as Bugzilla. Public bug submission entails a number of challenges for maintainers of the bug database. Maintainers of the bug database must check to see whether the bug has been classified correctly, provide more specific classification if required, bring the bug report to the appropriate developers' notice and create dependency links between bugs. This is a difficult task and even partial support would be very useful here.

With the ontologies described above, when a new bug report comes in for the Browser, agents could analyze the document, looking for keywords. If the keywords they find are part of the domain ontology, say Color or Fonts, and Color and Fonts are both defined to be subtopics of Rendering, the agents could infer that this bug report should be classified with bugs in Rendering. Furthermore, it could infer that this bug may be related to other bugs in Rendering. With knowledge of the code structure of the browser, an agent could infer that this bug may be related to other bugs within the same file or module. Thus an agent can provide support to a bug database maintainer by providing a short list of possible classification and candidate dependent bugs. A new developer wanting to contribute to a project through a bug fix, could use such processing to locate similar bugs that were now resolved. He could then note how they were fixed, for example, which files and modules required alteration, and try to use this information to fix the bug at hand.

Another potential application of semantic web technologies is in dynamically generating different types of interfaces for different roles or tasks of the community members. Each interface would link information such that different subsets of the information on the project site is presented. As an early example of the use of Semantic Web technologies in organizing information, Haystack [11] provides a platform for managing the personal information associated with a user alongwith a toolkit for constructing end-user semantic web applications.

4.2. Aggregating integrated information

Beyond being searched and linked, existing information from the various tools on the web site can be used to derive new kinds of information that reflects the dynamic nature of the project and raises awareness about member activities. The Expertise Browser [15], for example, uses CVS change data to provide information on which people have worked extensively on a particular module of code and are therefore likely to have much experience and expertise in it. It uses existing data in a novel way to address the collaboration challenge of identifying domain experts.

This approach could be significantly facilitated and extended with the help of Semantic Web technology. Semantic annotations of information in CVS along with ontologies that describe the architecture of the project can be used to help visualize the levels of activity in various parts of the project and the general trajectory of the project. For a potential contributor, such information would be useful to understand how to contribute meaningfully to a project.

Many open source software projects are beginning to use chat facilities for discussion, coordination and presence awareness. During a chat session, agents can monitor topics under discussion and, with the help of ontologies and semantic annotation, display relevant information from the project documentation and from web sites. Since information is already annotated semantically, with basic mechanisms to gather information from various parts of the site, developers can easily write their own widgets to monitor various parts of the project and share these with others, much like the Sideshow system [5].

Monitors to track changes can also be linked from dynamically generated profile of people or topics. Thus, a user browsing a developer's page can see whether the person is online and check if they are ready to chat and give advice. Similarly, when browsing dynamically generated pages on topics or modules, the user can also see the channels where this topic or module is currently being discussed. Dynamic processing can also be used to visualize mail activity and display patterns of mail exchange, which would be useful for potential contributors seeking to understand the community and culture of the project group.

Other examples of useful agents could be agents that periodically analyze activity patterns, e.g., mailing activity following the submission of code or a bug report,

or files that tend to be changed at the same time, and attempt to infer links from the patterns.. Other agents could analyze activity traces to compute "contribution" or "browsing" profiles of people and indicate people with similar profiles. This would help community members relate to each other and provide opportunities for social interaction. Such agents would also help make the user community more visible to the developers and foster a sense of community.

4.3. Binding communities of practice

Each open source project defines its own community, which is a 'community of interest' or a 'community of practice', in that it brings together people who are interested in the same domain and issues and share similar practices. If individual projects use the Semantic Web to annotate, organize and integrate their information, different such communities can be easily located and linked, to share information, expertise and people.

For example, consider a developer involved in an open source project that develops software to analyze fMRI data may want to add functionality for a particular kind of statistical analysis of a particular type of fMRI data. Using ontologies about statistics and fMRI, agents could help indicate code that implements similar statistics or other communities that work on similar kinds of fMRI data. Using activity statistics and other such information computed by agents, he can judge the size of the community around the tool, its maturity, and the people who are actively involved and thus very knowledgeable about the domain.

Similarly, when a potential contributor raises an issue, submits a patch or even sends email to the developer mailing list, the developers can easily locate the past work of the developer and make a better judgment about his competency in writing code or raising issues.

5. Conclusion

We think that semantic web technologies can contribute to the various aspects of collaboration in open source environments such as supporting collaboration among developers, supporting potential contributors, and in bringing users and developers together. To work towards the potential we have outlined in this paper, a first step would be to build ontologies for collaboration tools and build increasingly sophisticated agents that can manipulate the data in those tools and present meaningful information.

6. References

[1] Allen, T. J. (1977). *Managing the Flow of Technology*. Cambridge, MA: MIT Press.

[2] Brickley, D. and Guha, R. V. "RDF Vocabulary Description Language 1.0: RDF Schema", *W3C Working Draft*, January 2003.

[3] Cubranic, D. and Murphy, G. "Hipikat: Recommending Pertinent Software Development Artifacts", *International Conference on Software Engineering*, Portland, OR, 2003.

[4] Feller, J. and Fitzgerald, B. *Understanding Open Source Software Development*, Harlow, Essex, UK: Pearson Education, 2001.

[5] Fussell, S. R., Kraut R. E., Lerch, F. J., Scherlis, W. L., McNally, M. M., and Cadiz, J. J. "Coordination, overload and team performance: effects of team communication strategies", *ACM Conference on Computer Supported Collaborative Work*, Seattle, WA, 1998.

[6] Girgensohn, A. and Lee, A. "Making Web Sites be Places for Social Interaction", *ACM Conference on Computer Supported Collaborative Work*, New Orleans, LA, 2002.

[7] Halloran, T. J., and Scherlis, W. L. (2002). *High Quality and Open Source Practices.* Presented at the 2nd Workshop on Open Source Software Engineering, Orlando, FL.

[8] Hendler, J., Berners-Lee, T., and Lassila, O. "The Semantic Web", *Scientific American*, May 2001.

[9] Herbsleb, J. D., and Grinter, R. E. (1999, May 16-22). *Splitting the Organization and Integrating the Code: Conway's Law Revisited.* Paper presented at the 21st International Conference on Software Engineering (ICSE 99), Los Angeles, CA.

[10] Herbsleb, J. D., and Mockus, A. (2003). An Empirical Study of Speed and Communication in Globally-Distributed Software Development. *IEEE Transactions on Software Engineering, To appear*.

[11] Huynh, David, Karger, David, and Quan, Dennis. "Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF", Semantic Web Workshop, 2002.

[12] Kraut, R. E., Egido, C., and Galegher, J. (1990). Patterns of Contact and Communication in Scientific Research Collaboration. In J. Galegher, R. E. Kraut & C. Egido (Eds.), *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work* (pp. 149-171). Hillsdale, NJ: Lawrence Erlbaum Associates.

[13] Kraut, R. E., and Streeter, L. A. (1995). Coordination in Software Development. *Communications of the ACM, 38*(3), 69-81.

[14] Mockus, A., Fielding, R. T., and Herbsleb, J. D. "Two Case Studies of Open Source Software Development: Apache and Mozilla", *ACM Transactions on Software Engineering and Methodology*, 11(3) 2002, pp. 309-346.

[15] Mockus, A. and Herbsleb, J. D. "Expertise Browser: A Quantitative Approach to Identifying Expertise", *International Conference on Software Engineering*, Orlando, FL, 2002.

[16] Payne, Terry R., Singh, Rahul, and Sycara, Katia. "Calendar Agents on the Semantic Web." *IEEE Intelligent Systems*, Vol. 17(3), pp. 84-86, May/June 2002.

[17] Scacchi, W. "Understanding the Requirements for developing open source software systems", IEEE Proceedings on Software, 149(1), 2002, pp. 24-39.

[18] The Apache Cocoon Project, http://xml.apache.org/cocoon/

[19] The First International Semantic Web Conference, 2002, http://iswc2002.semanticweb.org/overview.html

[20] W3C Semantic Web Activity, http://www.w3.org/2001/sw/