Informing Design of a Search Tool for Bioinformatics

Medha Umarji
Dept. of Information Systems
University of Maryland Baltimore County
Baltimore, MD, USA
+1 (410) 455 3956
medha1@umbc.edu

Carolyn Seaman
Dept. of Information Systems
University of Maryland Baltimore County
Baltimore, MD, USA
+1 (410) 455 3937
cseaman@umbc.edu

ABSTRACT
We conducted an exploratory survey of software developers involved in creating, testing and maintaining bioinformatics software. In this paper we discuss how our study led us to conceptualize a tool that would index bioinformatics source code on the Web and create a searchable code repository. Such a tool would promote code reuse, facilitate software development for end-user programmers in bioinformatics, and eventually enhance the quality of commonly accessed source code. We propose the technique of contextual inquiry through interviews for detailed tool design and development. During the survey and surrounding investigation, we observed that domain-specific communities of practice provide useful domain knowledge that can facilitate tool design and development. This study is an example of how empirical studies can inform the design of tools and techniques to support scientific software development.

1. INTRODUCTION
As the ubiquity of software increases, software functionality is becoming more specific to the domain in which it will be used. Although there will always be a significant role for general-purpose software (e.g. general spreadsheet applications, word processors, operating systems), the segment of the software market devoted to domain-specific software is growing.

Developers and users in such domains as bioinformatics, space exploration, and mathematical modeling have evolved into communities of practice with their own domain-specific approaches to and philosophies about software development. The domain knowledge needed to effectively produce scientific software eclipses the computing knowledge needed in these domains. Hence scientific software professionals are expected to have a very diverse knowledge base spanning specific scientific disciplines, data visualization, database design and management and technology, among others. Also, several domains such as bioinformatics, present an exemplary usage of the open source philosophy for collaborative research and software development.

This combination of extreme domain-specificity, the unique background of many developers, and the extensive collaboration and formation of communities of practice make scientific software development a lens into the future of software development in general. This paper is a part of our ongoing work in the field of bioinformatics.

In an effort to better understand, and thus to study, the bioinformatics software development community, we conducted a survey of bioinformatics developers, asking questions about their background, the tools and techniques they employ, and their perceptions of quality and maintainability among other things. The intent of the survey was to describe bioinformatics software development, in order to provide software engineering researchers with the necessary background information, to address relevant and important problems in this domain.

From the survey we learned that almost half of bioinformatics professionals are end-user programmers while the other half are professional programmers. Many survey respondents stated that the reliability, maintainability and overall quality of software were unsatisfactory. We learned that open source involvement was a major part of bioinformatics software development. Several different tools were used for defect tracking and configuration management. There was no unified process for developing and maintaining applications that were actually very similar to each other.

Based on our study of this domain through the survey and surrounding artifacts, we concluded that a tool that would support search and reuse of bioinformatics software would be a good contribution to this community. This tool would not only search for bioinformatics software components but also for information contained in bug reports, defect databases and CVS/SVN repositories. The design of this tool was informed not only by our survey findings but also by existing tools, ubiquity of bioinformatics open source software and communities of practice.

Empirical studies are important in the process of learning and characterizing phenomena such as domain-specific software development. In this paper we propose contextual inquiry as a technique for specifying tool features so that they would be based on actual user input and areas where users need support. This study is an example of how empirical studies can guide creation of tools to support scientific software development.

In the following section we discuss some of the problems facing bioinformatics software development. Section 3 highlights some of the findings from our survey. Section 4 discusses current trends and the design of a search tool. Section 5 is conclusions and implications.

2. CHALLENGES IN BIOINFORMATICS SOFTWARE DEVELOPMENT
Prior literature on this topic points to some specific challenges facing the bioinformatics domain.

Redundancy: Reuse of software and programs was emphasized by Lincoln Stein, a proponent of open source in bioinformatics and a thought leader in this domain [1]. Stein compared bioinformatics with Italy in the Middle Ages i.e., a land of city states, with each state having its own government, its own currency and its own problems. Similarly, in bioinformatics, different scientists may be working on different (or similar)

aspects of the same problem. Stein urged bioinformatics researchers to join open source communities, share their programs and data, so that as a result of these collaborations, complex problems can be broken down and solved more easily.

End-users: There is a wide range of people who use and produce bioinformatics software, from purely research-oriented biologists with no programming background, to professional programmers with no domain knowledge[2, 3]. Many scientific software developers have backgrounds in areas other than technology or computing, but they do produce effective software [4]. Thus, studying the strategies employed by these software developers will inform future tools and best practices to support domain-specific software developers in other areas. In a recent study, Barker and Thornton [5] stated that software engineers (professional programmers) should be involved in development of bioinformatics software, and mature development practices should be adopted due to the significant complexity of the tasks involved. Wang et al. made similar observations and added that there was a need for effective development of preferably web-based tools [6].

Quality assurance practices: Software failures can be costly, and therefore effective quality assurance activities should be adopted in bioinformatics projects. Heusden suggested that the open source development model used for many bioinformatics products can have weaknesses in terms of quality assurance [7]. He stated that relying on Raymond's famous phrase, "given enough eyeballs, all bugs are shallow"[8], for open source projects is not enough because it is often the case that only some portions of bioinformatics source code are reviewed frequently. A contributing factor that exacerbates this problem is insufficient support for the reuse process [9].

3. SURVEY OF BIOINFORMATICS DEVELOPERS

We conducted an online survey of bioinformatics and biomedical professionals subscribed to OSS project mailing lists listed at the open-bio foundation. The survey addressed characteristics of the population and the domain, such as academic background, involvement in open source, and perceptions of product quality. After blank responses were deleted, we had a valid sample of 126 respondents. The response rate was 27.9%.

The results of the survey showed that close to half of the software developers in this domain have degrees in computer science (50%) and related disciplines while close to 40% of respondents were from a Biology-related discipline. Therefore, we conclude that this domain has quite a few professional programmers who do not have adequate domain knowledge and a sizable proportion of users not formally trained in computer science. Forty-seven percent of the respondents reported having worked on open source projects, 21.6% reported working on private lab-based software or proprietary software, and 31.2% did not divulge names of their projects. Thus open source involvement is prevalent in this domain.

Practices such as extreme programming and prototyping are frequently used, especially for developers who work on larger projects. Unix-based platforms such as Linux are the main programming platforms. Bioinformatics developers perceive the quality of their software to be high in general, but there is room for improvement, particularly in the testing and maintenance processes. Configuration management tools were very popular (70% of respondents used them) but defect tracking tools were not used as often (15% reported using Bugzilla). Documentation and comments were mentioned very often as techniques for enhancing maintainability.

From the survey we learned that there is a high proportion of end-users as well as professional programmers in this domain, therefore any tool should cater to the demands of both these communities. We also learned that the current reliability, quality and maintainability of bioinformatics software is not satisfactory, and this software is not tested adequately. However, there is potentially a lot of information (meta-data) in CVS/SVN repositories about the source code, embedded in comments and documentation.

4. DESIGN OF A SEARCH TOOL

Given the ubiquity of open source in the bioinformatics domain, and the findings from our survey one solution that would tackle the increasing redundancy, help end-users to develop better software and enhance quality, is to create and promote an effective mechanism for indexing bioinformatics open source code available in different repositories on the Web, and use algorithms specifically designed for internet-scale source code searching to find and retrieve code snippets, components and applications.

The suggestion for such a search tool comes from current trends in source code search and reuse on the Internet[10]. In the next section we discuss the presence of massive repositories of source code, and search engines that index and retrieve software components from these repositories. The difference between the existing search mechanisms and our proposed search engine is incorporating search by domain-specific parameters such as vocabularies of research terms, current research problems and social networks and collaborations. Therefore in Section 4.2 we discuss learning about the domain from communities of practice and other social networks.

4.1 Source code on the Internet

There is a massive amount of open source code available on the Web and it comes from open source projects, websites that support communities of practice, and language-specific archives. Among these are Koders.com with over 226 million lines of code (MLOC), Krugle.com with over 2 billion lines of code (TLOC), csourcesearch.net over 283 MLOC, and Google Code Search with over 1TLOC. These are repositories of source code for all types of software that may or may not include scientific, domain-specific software. In the recent past search engines that index and retrieve source code (such as Google Code Search) have been created, to take advantage of the availability of open source code repositories.

An increasing proportion of this open source software is domain-specific. For example, bioinformatics open source project sites like BioJava (http://biojava.org) and BioPerl (http://www.bioperl.org/) contain huge compilations of source code. Additionally, some open publishing portals like BioMedCentral (http://www.biomedcentral.com/) make software corresponding to publications available to all members. BioWareDB [11], a hyperlinked database of freely available and commercial bioinformatics and biocomputing software, had grown to 2800 validated entries by 2003. There is also a large number of bioinformatics projects hosted at project hosting websites, SourceForge.net, and FreshMeat.net.

As discussed briefly in Section 2, bioinformatics code is not contained in a single repository but is dispersed in different repositories all over the Web, and there is no search mechanism for identifying and locating relevant bioinformatics software objects when needed. The only known search solution is a code search engine called b-src (http://b-src.cbrc.jp), which is based on gonzui, a Japanese code search engine (http://gonzui.sourceforge.net/). However, b-src does not have the features to support a scalable implementation across all bioinformatics communities.

4.2 Learning Domain Characteristics

Learning about the domain is an important requirement prior to designing any tool. Domain-specific terminology, prominent research problems and the overall nature of a domain can be discerned from passive observations on mailing lists and communities of practice.

A community of practice is formed by a group of people united by a joint enterprise, who develop mutually beneficial social relationships in the process of working towards things that matter to them [12]. Artifacts, advice/tips and other relevant knowledge are contributed by members to provide a shared repertoire of resources for the community.

One example of such a community is the **Open Bioinformatics Foundation** (http://www.open-bio.org/). This community is formed by people who have been active in open source bioinformatics projects such as BioPerl and BioJava that provide frameworks for developing bioinformatics applications. This portal also links to artifacts such as wikis for definitions and terms used in bioinformatics, as well as news on recent developments such as licensing agreements.

Mailing lists and discussion forums also help to identify factors such as academic vs. research focus and commercial vs. government focus. We did some non-participant observation on these mailing lists and communities of practice, to better understand the needs and work practices of bioinformatics professionals.

4.3 Designing a Search Tool

In this section we outline the motivation for a search tool, possible features it would have, and propose a technique for further refinement and implementation of these features.

Motivation: This tool is based on the concept of pragmatic reuse, i.e., the reuse of source code components that were not designed to be easily reused [13]. Typically software components lack the meta-data and searchable keywords necessary for reuse. With the advent of open source the concept of code reuse has been revolutionized, it is now the norm rather than the exception. From our observation, the assimilation of pragmatic reuse has been slow in bioinformatics and other related communities because the software developed in these communities is difficult to find and not enough information is typically provided to facilitate reuse. Therefore there is a pressing need to index all the bioinformatics source code, and to create a searchable repository with relevant research information and meta-data about the code.

Such a tool would make it easy for an end-user programmer to identify and locate desired software components, thus avoiding the rework effort, and allowing him/her to focus on the research-related problem.

Features: The search engine could be tailored to index software that contains terms specific to bioinformatics and biology-related disciplines. These terms could be learned from online vocabularies or dictionaries. The indexing process can start by including common bioinformatics programming frameworks such as BioPython, BioJava and BioPerl.

The research laboratory from which the software application originated could be recorded as an indexing factor. Authors of a code snippet could be used in the indexing mechanism as well, so that users could retrieve code written by a specific programmer. Based on the code structure it might be possible to distinguish between academic and industrial open source projects.

Once an indexed repository is created, the search could be performed on meta-data such as author name, research problem and name of affiliated lab or company (if such information was available). Ideally, the repository could be classified into subjects such as genomics, proteomics etc.

If annotation capabilities and "tags" were built-in, such as in del.icio.us, we would be able to provide recommendations to users and relevant information about each code component. Typically, a keyword-based search could retrieve programs as well as artifacts surrounding source code, such as documentation. Search could also be performed based on the code structure – such as use of APIs and other dependencies.

Either this tool could be a stand-alone implementation based on algorithms specific to code search, or it could be designed as a plug-in to generic code search engines such as Google Code Search (http://www.google.com/codesearch).

There are several ways in which domain knowledge could be used to facilitate search. For example, the tool could have features that support social network analysis of communities of practice, and convert that information into a social dependency graph. Such a graph coupled with source code analysis would then enable researchers to learn about phenomena such as collaborative distributed scientific software development [14].

Specification and Evaluation: The next step would be to present this conceptual solution to bioinformatics professionals, and to get their opinion about the utility and feasibility of creating and deploying this tool.

"*Contextual inquiry is a way for users to participate in the design of general purpose systems. It is a technique for working with users to help them articulate their current work practices, system practices and associated experiences.*" [15]

Contextual inquiry has three core concepts: the user is a partner in the design process, in-depth understanding of the context is considered central to the solution, and the whole process is very focused. We believe that the rationale of a structured interviewing technique like contextual inquiry would be a good fit to the design of tools for scientific software development [16]. The exploratory nature of these interviews would also enable the discovery of knowledge that would benefit future tools for software engineering in the computational sciences.

Evolving a set of specifications for such a tool would also be facilitated by discussions on mailing lists open source communities.

5. CONCLUSIONS

Currently there is a big gap in traditional software engineering research and scientific software development practice. Leveraging

open source as a medium of technology transfer and encouraging adoption of tools such as the one we have proposed here are worthy avenues to explore. Although in this paper we discuss the bioinformatics domain to a large extent, our proposed tool design would no doubt be applicable to other scientific domains. Communities of practice are prevalent in all scientific software development domains and are excellent pointers to domain knowledge.

Considering that domain-specific software development is such a specialized activity, any tools that are developed to support this activity should be based on a deep understanding of the work practices of developers and broader requirements of these communities. The tool design presented in this paper is just such an example of a study of work practices and an in-depth understanding of the domain. Moreover, we plan to use the technique of contextual inquiry to refine and arrive at accurate specifications for such a tool. We think that scientists and end-user programmers would be receptive to such interactions, and the end-result would be mutually beneficial.

This study has several components - an exploratory survey, non-participant observation on domain-specific communities of practice, and a proposed contextual inquiry method for tool design. It serves as an example of how empirical studies can be designed to advance software engineering for computational sciences.

6. REFERENCES

[1] L. Stein, "Bioinformatics: Gone in 2012.," in *O'Reilly Bioinformatics Technology Conference (Keynote Address)*. San Diego CA, 2003.

[2] M. Burnett, C. Cook, and G. Rothermel, "End-user software engineering," *Commun. ACM*, vol. 47, pp. 53-58, 2004.

[3] J. Carver, "Empirical studies in end-user software engineering and viewing scientific programmers as end-users: Position statement," presented at Dagstuhl Seminar Proceedings 07081: End-User Software Engineering, 2007.

[4] S. M. Baxter, S. W. Day, J. S. Fetrow, and S. J. Reisinger, "Scientific software development is not an oxymoron," *PloS Computational Biology*, vol. 2, September, 2006.

[5] J. Barker and J. Thornton, "Software engineering challenges in bioinformatics," in *International Conference on Software Engineering (Keynote address)*. Edinburgh, Scotland, UK, 2004.

[6] J. T. L. Wang, Q. Ma, and K. G. Herbert, "Software engineering and knowledge engineering issues in bioinformatics," *Handbook of Software Engineering and Knowledge Engineering*, vol. vol. 1., pp. 719-722, 2002.

[7] P. v. Heusden, "Applying software validation techniques to bioperl," in *Bioinformatics Open Source Conference*, 2004.

[8] E. S. Raymond, "The cathedral and the bazaar:Musings on linux and open source by an accidental revolutionary." Sebastopol, CA 95472, U.S.A.: O'Reilly and Associates, 1999.

[9] A. G. Koru, K. El-Emam, A. Neisa, and M. Umarji, "A survey of quality assurance practices in biomedical open source projects," *Journal of Medical Internet Research*, vol. 9, pp. e8, 2007.

[10] M. Umarji, S. E. Sim, and C. Lopes, "Archetypal internet-scale source code searching," in *Open Source Software (OSS 2008)*. Milan Italy, 2008.

[11] M. W. Matthiessen, "Biowaredb: The biomedical software and database search engine," *Bioinformatics*, vol. 19, pp. 2319-2320, 2003.

[12] J. L. E. Wenger, *Situated learning: Legitimate peripheral participation*. England: Cambridge University Press, 1991.

[13] R. Holmes and R. J. Walker, "Supporting the investigation and planning of pragmatic reuse tasks," in *Proceedings of the 29th International Conference on Software Engineering*: IEEE Computer Society, 2007.

[14] C. de Souza, J. Froehlich, and P. Dourish, "Seeking the source: Software source code as a social and technical artifact," in *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*. Sanibel Island, Florida, USA: ACM, 2005.

[15] H. Beyer and K. Holtzblatt, *Contextual design: Defining customer-centered systems*: Morgan Kaufmann, 1998.

[16] C. Letondal and W. Mackay, "Participatory programming and the scope of mutual responsibility: Balancing scientific, design and software commitment," in *Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices - Volume 1*. Toronto, Ontario, Canada: ACM, 2004.