Comparing Bioinformatics Software Development by Computer Scientists and Biologists: An Exploratory Study

Parmit K. Chilana
The Information School
DUB Group
University of Washington
pchilana@u.washington.edu

Carole L. Palmer
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
clpalmer@uiuc.edu

Amy J. Ko
The Information School
DUB Group
University of Washington
ajko@u.washington.edu

Abstract

We present the results of a study designed to better understand information-seeking activities in bioinformatics software development by computer scientists and biologists. We collected data through semi-structured interviews with eight participants from four different bioinformatics labs in North America. The research focus within these labs ranged from computational biology to applied molecular biology and biochemistry. The findings indicate that colleagues play a significant role in information seeking activities, but there is need for better methods of capturing and retaining information from them during development. Also, in terms of online information sources, there is need for more centralization, improved access and organization of resources, and more consistency among formats. More broadly, the findings illustrate a variety of information problems that end-user biologists and professional software developers face in developing bioinformatics software and how they are influenced by the level of domain knowledge and technical expertise.

1. Introduction

Since the completion of the human genome project, computational tools have become indispensable in supporting analysis, integration, modeling, and visualization of large amounts of molecular data and advancing a major core of biological research [10]. Bioinformatics has become one of the fastest growing interdisciplinary scientific fields, bringing together Molecular Biology and Biochemistry (MBB) and Computer Science (CS), among other disciplines. A plethora of commercial and open-source bioinformatics tools are emerging, but they often lack transparency in that researchers end up dealing more with the complexity of the tools, rather than the scientific problems at hand [3].

Although research in bioinformatics has soared in the last decade, much of the focus has been on high-performance computing, such as optimizing algorithms and large-scale data storage techniques. Only recently have studies on end-user programming [5, 8] and information activities in bioinformatics [1, 6] started to emerge. Despite these efforts, there still is a gap in our understanding of problems in bioinformatics software development, how domain knowledge among MBB and CS experts is exchanged, and how the software development process in this domain can be improved.

In our exploratory study, we used an information use perspective to begin understanding issues in bioinformatics software development. We conducted in-depth interviews with 8 participants working in 4 bioinformatics labs in North America. These included software and database developers, and other professional staff, such as systems analysts and network administrators. Half of the participants in these practitioner roles had CS degrees, while the other half had primary training in MBB or a related biological field. We also compared the roles of CS researchers, who were more focused on the application of theoretical concepts in computational biology with MBB researchers, who were concerned with the acquisition of CS-related skills to solve MBB problems.

We used Robert Taylor's concept of *information use environments* as the conceptual framework for this study [14]. Furthermore, we also applied MacMullin & Taylor's *problem dimensions* [7] in the comparative analysis of the researcher and practitioner roles that emerged.

The results of this study shed light on how information is used to understand the biological problem, to translate the problem into code, to interpret the output, and to resolve related technical issues. The results show interesting similarities and differences among the information activities of practitioners versus researchers and computer scientists versus biologists.

SECSE'09, May 23, 2009, Vancouver, Canada
978-1-4244-3737-5/09/$25.00 © 2009 IEEE

Based on these preliminary results, we believe there is merit in further study to better understand how software developers with no scientific domain knowledge participate in not only bioinformatics, but also other sciences. Also, there is need to investigate how specific methods or tools can be developed to facilitate their participation. We are continuing to explore such issues in our current research.

2. Related Work

Although bioinformatics is a relatively new area of scientific research, we are beginning to understand the underlying information tasks and end-user programming activities of biologists.

For example, MacMullen & Denn's [6] research into information tasks of biologists revealed three categories: sequence alignment (i.e. sequences of DNA), structure prediction (i.e. protein folding) and function prediction (i.e. gene function). Tran et al. [15] developed a cross-sectional study with six different bioinformatics research labs and found four common themes with the way bioinformatics tasks were carried out. These included a lack of procedural documentation of high level tasks, use of home-grown strategies, lack of awareness of existing bioinformatics tools, and variations in individual needs and preferences.

Bartlett & Toms [1] also worked with biologists to understand how they conduct functional analyses of gene sequences using tools such as, GenBank, the genetic sequence database, and BLAST, an analytical tool for calculating similarities between sequences [1]. They discovered that these analyses served as an 'extension' of traditional techniques and a means of 'data reduction', but the scientists still had to verify results through experimentation. Furthermore, they suggested that each bioinformatics expert followed a 'unique process' in using these resources and that there were a number of technical and interface inconsistencies among these tools.

Umarji and Seaman [16] recently conducted a survey of software developers in bioinformatics and proposed the design of a search tool that would facilitate access to open-source bioinformatics software components. About half of their 126 respondents had CS degrees and most others had Biology-related training. Their findings indicated that open-source projects were the most common, configuration management tools were used, but there was room for improvement in testing and maintainability of software, and that comments and documentation could have potentially useful information that should be exploited.

In terms of end-user programming, Massar et al. [8] designed *BioLingua*, an interactive web-based programming environment to give biologists more control and flexibility in carrying out analyses with genomics, metabolic, and experimental data and higher-level representations. Their system made use of a symbolic programming language to provide a transparent, integrated interface to commonly used bioinformatics tools by hiding the implementation details.

Similarly, Letondal [5] developed *Biok* an integrated programmable application for analyzing DNA and protein sequences or multiple alignments. They used a novel approach in creating this tool by using a combination of participatory design and end-user programming techniques. Using a "participatory programming" approach, the goal was to allow biologists have more control and flexibility over tools by working collaboratively with software developers during design.

In summary, although these studies shed light on the general information and end-user programming activities of biologists, we know little about the information activities of computer scientists in bioinformatics and how they compare. Also, while there has been recent research in understanding the information activities of software developers [4], their activities in complex scientific domains are not well-understood. This study is a step towards filling this important gap by investigating the information activities that occur in developing and maintaining bioinformatics software by considering both biologists and computer scientists with no biological domain knowledge.

3. Method

3.1. Study Design

We used the semi-structured interview technique for data collection, since it allows for the kind of open-ended discussion that can capture the situational aspects of information use, while also providing a way to find consistencies among responses from different participants. We developed ten interview questions (see Appendix 1) based on the following themes:

- Nature of current software development work and description of technical environment
- Strategy used to overcome technical issues during development

- Strategy used to overcome domain-related challenges during development
- Recall of recently encountered problems and the steps used for resolution
- Awareness and choice of information resources and decisions made to determine relevance

We interviewed participants in their work settings to establish context and facilitate recall. Each interview lasted approximately 45 minutes. At the end, we asked the participants to complete a short questionnaire to provide basic demographic information, such as educational background, relevant coursework, and attendance at conferences. We analyzed transcripts of the interviews to identify recurring themes using approaches consistent with the between- and among-interview comparisons of Rubin and Rubin [13] and Miles and Huberman [9].

3.2. Participants

There were eight participants in this study, located at two CS and two MBB research labs in North America (see Table 1 for complete summary). The respondent pool was balanced, with two participants in each of the four categories: CS researchers, MBB researchers, CS practitioners, and MBB practitioners. The CS labs were focused on computational biology research, with an emphasis on the use of probability theory techniques, machine learning, and other aspects of CS theory. The MBB labs were involved in a variety of bioinformatics projects ranging from the analyses of prokaryotic genomes to high throughput DNA sequencing. The participants' average experience in this field was about 4.25 years, ranging from one year in the same lab to eight years in multiple labs.

3.3. Analysis Approach

We used *information use environments* [14] as the conceptual framework for the study. This model emphasizes the context or situational aspects of information needs and how they affect how information is used to solve problems by different groups. In addition, several of Taylor's categories of information for problem resolution were of interest for this study: "enlightenment" (i.e., to establish context); "problem understanding"; "instrumental" (i.e., to figure out how to do something); "factual"; and "confirmational." We also found MacMullin & Taylor's *problem dimensions* [7] relevant in the comparative analysis of information use in the researcher and practitioner roles. A problem dimension is defined as a characteristic for judging the relevance of information to a given problem [7]. We highlight some of these dimensions further in our discussion.

4. Results

The results suggest a linear progression of information activities starting from the inception of an idea to the software implementation; however, some activities recur throughout the whole process. The results also show the range of information sources of importance to the researchers and practitioners and the decentralized nature of the information use environment in bioinformatics.

Table 1: Profile of Participants

Role	Type of work	# of years in field	Academic Training
CS Researcher	Using probability theory, machine learning, and energy distributions from statistical physics for genomic data	2	PhD in CS BS & MS in CS
CS Researcher	Devising algorithms for extracting hidden patterns in genomic data using statistical techniques	1.5	PhD in CS MS in CS BS in Biochemistry
MBB Researcher	Developing applications for supporting genomic data transformation and analysis	7	Post-doc in CS PhD in Cell Biology
MBB Researcher	Developing custom visualizations to understand features of genomic data	5	PhD in MBB BS in Microbiology
CS Practitioner	Writing scripts for processing genomic data, maintaining existing systems	1.5	BS in CS
CS Practitioner	Developing and maintaining large genomic databases, providing system administrative support	3.5	BS & MS in CS
MBB Practitioner	Developing and maintaining large databases, providing data transformation support to custom formats, data analysis	6	MS in Zoology MS in CS
MBB Practitioner	Maintaining genome databases with user-defined annotations, optimizing database performance	8	BS in Biology MS in Microbiology IT diploma

4.1. Understanding the Problem

The CS researchers identified their primary focus as the development of an algorithm or a computational technique. MBB problems were used as examples of application areas. Thus, these researchers first sought to understand the problem from a theoretical CS perspective and then focused on learning the relevant MBB concepts and terminologies. CS practitioners spent less time on theoretical constructs and focused more on developing a practical solution to the biological problem. The participants in this category mentioned that their initial information tasks were centered on understanding information such as valid cut-off values and parameters. As one participant explained:

> [biologists] play the role of 'translators' in helping us [programmers] interpret the data or to explain the requirements for developing an application to solve a biological problem…

In contrast, MBB researchers focused on the biological problem and how computational tools could be used to aid or enhance understanding of a concept. They appeared more interested in learning practical skills such as programming or database development in order to frame their biological problem as a computational one. Unlike the researchers, MBB practitioners primarily needed to know what is sufficient in terms of working on an application and did not need to develop deep, research-level biological understanding. Their main concern was whether their existing technical skills were adequate for implementing a solution for a new problem.

4.2. Translating a Problem into Code

CS researchers generally tested their high level mathematical or statistical framework by translating it into code. They often used command-line interfaces and programming languages that they were comfortable with and did not find it challenging to locate any related technical information. Similar to CS researchers, the applications developed by MBB researchers were simple at the beginning to match their research purposes. However, the participants in this category had primarily self-taught programming skills and often sought information to further develop more advanced skills or obtain additional help from colleagues in implementing a solution. For example, one MBB researcher who started using C++ for a problem realized:

> …[we] have to work with very large data sets...[and] have to be aware of memory allocation and de-allocation issues when programming [with these]...

Contrary to the researchers, participants in both of the practitioner categories worked on large-scale applications meant to be used beyond *proof-of-concept*. CS practitioners pointed out that while understanding the biological problem was challenging, once they had a good grasp, programming the solution was usually trivial. MBB practitioners, on the other hand, were more comfortable in implementing a solution in a familiar language, such as Perl, and were usually less comfortable in transferring their skills in a new development environment.

4.3. Interpreting Results

There was consensus among all the participants that interpreting the output of a bioinformatics application required substantial biological insight. The MBB researchers were most confident about what they expected to see and could determine whether a program had succeeded in yielding the correct results based on their knowledge. MBB practitioners were also reasonably confident about evaluating outputs. One participant mentioned that he could develop sufficient biological insight for unfamiliar problems over time through relevant readings.

For CS practitioners, however, interpreting the results was described as being even more challenging than understanding the initial problem. Since a lot of the application development was in small iterations, they were faced with numerous intermediate outputs. CS practitioners usually needed more than background readings or reference materials in order to make sense of these outputs and depended mostly on MBB experts in their labs. CS researchers had to first interpret whether their proposed algorithms or models were yielding reasonable outputs in terms of CS theory, and then verified whether the outputs were consistent with the original biological problem. The following narrative by a CS researcher best summarizes this issue:

> …the problems are essentially from molecular biology so the 'solutions' also have to make sense to the biologists, regardless of the computational techniques used...journal articles can be useful [for interpretation]...but, it's even more valuable to talk to biologists who have expertise in that particular area…

4.4. Resolving Technical Issues

CS researchers and CS practitioners further pointed out that their background and experience with

programming prepared them to efficiently perform debugging. This encompassed looking at errors reported by the compiler and going into the source code to make the necessary modifications. For any peculiar error messages, the common consensus was to cut and paste error messages in Google search, as illustrated by one participant:

> I use the C++ library sometimes to look up a function..but generally use Google to look up specific error messages..[it's] usually very good, but lots of extra stuff...

In fact, this approach was described by all the other participants in this study as well. The only difference for the MBB participants was that they were less likely to debug by inspection, and for more complicated problems they needed to consult a colleague with more programming experience to find resolution. The value of comments interweaved in the code was also highlighted by one MBB participant:

> ...[we like to] comment everything... it can make a big difference for other people wanting to use the software or to maintain it in the future…

One of the bigger technical hurdles faced by practitioners in both MBB and CS categories was dealing with the dynamic nature of bioinformatics research. As one participant described, maintenance was a significant issue:

> …the major challenge [is] constantly changing biological data…[we] have to constantly update...systems [we develop] cannot catch up with all the requirements of biologists...

4.5. Using Various Sources of Information

Online resources played a vital role in bioinformatics research and practice. MBB and CS researchers exhibited many of the same characteristics of scientific research found in previous studies of interdisciplinary scholars [11]. Journal articles, conference proceedings, and books are prominent sources. Consistent with other previous studies [1, 6, 15] both groups of researchers and practitioners relied heavily on non-bibliographic information sources. Colleagues also emerged as a significant source of information for both MBB and CS participants since much of bioinformatics software development is a collaborative effort.

In looking at the information tasks of computer scientists, there was variation among the researchers and the practitioners groups. Both were lacking a solid foundation in MBB, but CS practitioners were more hesitant to access journal articles or books related to MBB, compared to CS researchers. On the other hand, participants in both groups preferred talking to MBB experts or searching for examples or definitions online.

The MBB researchers and practitioners indicated that they regularly needed information on a programming language or technology. For example, one participant described a case where he had long been working with a system written in the *Perl* scripting language but experienced some limitations when he wanted to incorporate a particular functionality. He was aware that object-oriented languages such as *Java* would be more suitable for this type of implementation, but he did not know where to begin to learn about it or what changes to anticipate for the system as a whole if he were to switch languages. In this situation, he was dependent on his CS colleagues as the primary source of information. Another participant described needing to develop many skills related to a new programming paradigm and operating system when switching to a new platform. Even with several relevant books and online resources available, he relied on his colleagues' technical expertise to recommend useful examples and tutorials.

Interestingly, participants in both the CS and MBB practitioners categories pointed out that while they may not have been immersed in the actual research, they often had to look up recommended readings pertinent to the scientific problem to understand the inputs and outputs of their applications. Since the literature was highly scattered, they expressed a need for one-point access that would alleviate the current need to search different databases and portals, as well as an interface that would allow them to narrow down search options more effectively.

4.6. Coping with a Lack of Centralization

Bioinformatics researchers and practitioners who had been in the field for some time were generally aware of the tools and techniques that have emerged over the years. In particular, several mentioned that the renowned NCBI portal provides a categorized aggregation of many of the open source software tools and data sets. However, since there is no requirement for labs, researchers, or software developers to submit their information to services such as the NCBI, much of the data stored in protein databases and metabolic pathways databases is only accessible through a web search or by recommendation from a colleague. One participant described a specific case where he failed to locate data sets through the NCBI for a DNA sequence to test a regulatory module:

> ...used google as the starting point…it brought several results…went through each one to see which is

> relevant…couldn't tell by looking at the results alone…had to click on all the links [search results]…eventually found it by browsing through, but it was actually listed under some other name…

In fact, most participants highlighted the difficulty in locating specific data sets online using Google as the first step. Another participant also pointed out a case where he failed to locate a genome data set because he did not know all the variations of the genome name. A colleague had to send him the link to the data set via email. Lack of standard naming conventions was also noted as a problem for locating data sets. As found in previous studies of bioinformatics [2, 15] even when data sets or tools of interest are successfully located, they are often not consistent in format.

Specific technologies could sometimes be identified by locating a conference proceeding or journal article where it was described. PubMed was regarded as a very efficient tool for searching the literature, but it was also criticized for insufficient ranking in its retrieval sets. In addition, Google was identified as the top choice for finding information related to an error code or an unfamiliar concept, but participants consistently remarked on searches producing "too much junk" or "irrelevant results." Some recent efforts in improving such search activities in bioinforamtics are underway and strive to alleviate such problems [16].

5. Discussion

MacMullin and Taylor's problem dimensions [7] provide a useful analytical perspective for studies of interdisciplinary scientific information work [12]. In addition, our previous studies of neuroscientists and biologists suggest that some aspects of the information use environment observed in this study extend beyond the practice of bioinformatics software development.

Four of MacMullin and Taylor's eleven dimensions were prominent in this study: "initial state understood / not understood", "well structured / ill-structured", "familiar / new pattern", and "internal / external imposition." Below we provide a synopsis of each dimension and illustrative examples from our data.

Initial state understood / not understood: In the initial problem state, some of the contributing factors and the interrelationships are not understood. This dimension was evident in our results for all categories of participants as they began work on a new problem. MBB practitioners and researchers understood the biological relevance, but sought information to understand the inter-relationship with the technical implementation. Conversely, CS researchers and practitioners had to ensure that the algorithms or code that they are developing are valid biologically.

Well-structured / ill-structured: Problems that can be solved using logic or algorithms are well-structured; problems that are more complex have variables that are not understood, or are influenced by random factors, are ill-structured. MBB and CS practitioners appeared to work with well-structured problems, whereas researchers in both categories frequently faced ill-structured problems. Whether a research problem originated in CS or MBB, the complexity was usually high and required understanding of all inter-related factors. As researchers tackled different aspects of the problem, they identified more well-structured sub-problems and relied on practitioners to develop solutions.

Familiar / new pattern: When a problem is familiar, information needs are resolved using established procedures, but for new problem patterns, more substantive and future-oriented information is needed. Practitioners' information activities strongly exhibited this dimension. For example, CS practitioners dealt with familiar software issues using their established debugging methods, but they relied heavily on MBB colleagues in selecting the input parameters or interpreting the output. Similarly, MBB practitioners were more likely to favor familiar programming languages and platforms, tending to only try different solutions through the help of CS colleagues.

Internal / external imposition: Problems can be imposed internally due to dysfunctions in operation or externally due to environmental factors. Our participants discussed numerous examples of working to resolve internally imposed technical issues, as is true with any software development activity. External imposition is more complex, and very important in the context of bioinformatics. For example, while a technical solution or an algorithm may be valid from the CS perspective, the externally imposed biological context can often invalidate it or result in the need for significant changes.

Using this framework of problem dimensions has allowed us to better understand and compare how computer scientists and biologists use information to resolve problems in different stages of software development. There clearly is a preference for informal exchange of information whether it is to solve a technical problem or to further understand domain-related details. More research is still needed to not only establish a better understanding of this issue, but also

to develop methods and tools that can facilitate this exchange and help make the software development process in bioinformatics more efficient.

This study also has several limitations. The sample size of this study was relatively small and there was an inherent selection bias in that potential candidates who responded within a given time frame were the only ones considered. Also, there are many large bioinformatics initiatives occurring globally, yet this study is limited to perspectives from North America. Some limitations are also inherent in the use of the semi-structured interview technique since no two interviews ended up being the same. Thus, we generalize our results across all participants with some level of caution.

6. Future Work

Based on the results of this study, our next step is to investigate the tasks of software developers in more depth by using observational techniques. In particular, we want to focus on developers who do not have any domain-specific training. We are planning to continue this work within the biomedical domain, eventually extending our investigations to other sciences.

7. Conclusion

Although the exploratory study presented here was limited in the number of participants and labs represented, it offered a balanced view of the different roles involved in bioinformatics software development. More importantly, it corroborated findings from a number of earlier studies and demonstrated the value of a "problem-oriented" analytical approach to studies of scientific information use and software development. Several areas for investigation also emerged, related to the need for more systematic organization and accessibility of bioinformatics resources. Research initiatives in software design, human-computer interaction, and library and information science disciplines are beginning to address such issues, as will our continued work in understanding software development in the sciences.

8. Acknowledgements

We thank Jennifer Hill and Phillip Edwards for valuable comments.

9. References

[1] J. C. Bartlett and E. G. Toms, "Developing a protocol for bioinformatics analysis: An integrated information behavior and task analysis approach," *JASIST,* vol. 56, p. 469-482, 2005.

[2] A. Beveridge, "An object-oriented programming system for the integration of internet-based bioinformatics resources," *Applied bioinformatics,* vol. 5, p. 29-39, 2006.

[3] C. R. Johnson, R. MacLeod, S. G. Parker, and D. Weinstein, "Biomedical computing and visualization software environments," *Commun ACM,* vol. 47, p. 64-71, 2004.

[4] A. Ko, R. Deline, and G. Venolia, "Information needs in collocated software development teams," in *Proc ICSE, pp.344-353,* 2007.

[5] C. Letondal, "Participatory programming: Developing programmable bioinformatics tools for end-users," in H. Lieberman, F. Paterno, and V. Wulf (Eds.), *End-User Development*, Springer, 2005

[6] W. J. MacMullen and S. O. Denn, "Information problems in molecular biology and bioinformatics," *JASIST,* vol. 56, p. 447-456, 2005.

[7] S. E. MacMullin and R. S. Taylor, "Problem dimensions and information traits," *The Information Society,* vol. 3, p. 91-111, 1984.

[8] J. Massar, M. Travers, J. Elhai, and J. Shrager, "BioLingua: A programmable knowledge environment for biologists," *Bioinformatics,* vol. 21, pp. 199-207, 2005.

[9] M. B. Miles and A. M. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*: Sage, 1994.

[10] National Center for Biotechnology, "Bioinformatics Factsheet," http://www.ncbi.nlm.nih.gov/About/primer/bioinformatics.html, last accessed Jan 5, 2009.

[11] C. L. Palmer, *Work at the Boundaries of Science: Information and the Interdisciplinary Research Process*: Kluwer, 2001.

[12] C. L. Palmer, "Weak information work and "doable" problems in interdisciplinary science", in *Proc ASIS&T Annual Meeting*, vol. 43, p. 108-124, 2006.

[13] H. Rubin and I. Rubin, *Qualitative Interviewing: The Art of Hearing Data*: Sage, 2005.

[14] R. S. Taylor, "Information use environments," *Progress in Communication Sciences,* vol. 10, p. 217-255, 1991.

[15] D. Tran, C. Dubay, P. Gorman, and W. Hersh, "Applying task analysis to describe and facilitate bioinformatics tasks," *Medinfo,* vol. 11, p. 818-822, 2004.

[16] M. Umarji and C. Seaman, "Informing design of a search tool for bioinformatics," in *Proc ICSE Workshop on Software Engineering for Computational Science and Engineering*, 2008.

10. Appendix 1

Interview Questions

1. Can you briefly describe the kind of work you are currently doing and what kind of project(s) are you involved with? Also, tell us something about the technical environment of your work – what kind of operating system, software tools, programming languages, database systems and other technologies are your primarily working with?
2. How do you overcome technical problems that you encounter during the development process of a system? What's your general strategy in looking at different resources (ie. a book vs. code libraries vs. online forums) to resolve your problem? Normal debugging skills – very experienced now, repeating much of the earlier work, rarely using any external resources to debug
3. What kind of non-technical challenges do you encounter in developing applications (ie. related to an algorithmic concept or a molecular biology phenomenon)? How often do these occur compared to technical problems? What's your general strategy in resolving these?
4. During your process of resolving a problem, how do you assess whether a particular information resource is useful? What's your preferred method of saving a reference to that resource?
5. Please describe a problem which you encountered recently and demonstrate the steps which you roughly took to resolve it, including the sources you consulted along the way.
6. What kind of information do you share with your colleagues and how often? What about contacts outside your lab? How is this communication generally carried out?
7. Do you experience the need to look up resources for information on best-practices or standards in developing a system? Which resources do you find useful in this case?
8. Are there any particular print or electronic information resources that you access regularly to stay-up-to-date with the developments in your field? Is your choice of resources different if you are working on an existing system vs. developing a new system?
9. Are you aware of or do you use any resources available through your institution's library (could be print or electronic resources)? If yes, how often are these a part of your problem-solving strategy?
10. Do you have any suggestions on what can be done to make your access to information more efficient in your job?