Informing Design of A Search Tool for Bioinformatics

Medha Umarji

Carolyn Seaman

Dept. of Information Systems,

Univ. of Maryland, Baltimore County

Overview

- Background and prior work
- Results from survey of bioinformatics professionals
- Current challenges in bioinformatics software development
- Design of a search and indexing mechanism for bioinformatics software
- Conclusions

Background

- Our prior work in Bioinformatics
  - Exploring and characterizing bioinformatics professionals
  - Quality assurance practices in bioinformatics projects
  - Teaching software engineering to end-users
- Current work
  - Contributing to bioinformatics research, education and practice from a software engineering perspective

Survey of bioinformatics professionals

- Online survey posted on mailing lists from the open-bio foundation
- Software development paradigm
  - Rapid prototyping, iterative
  - Selected agile practices adopted widely
  - Heavy involvement in open source
- Characteristics of people
  - Highly educated
  - Even mix of computer science and biology-related majors
  - Self taught
- High use of CVS/SVN repositories

Current challenges in bioinformatics

- Redundancy
  - Different scripts written to solve similar problems [1]
  - Low reuse
- Users
  - End-users (self-taught programmers)
  - Professional programmers (no domain knowledge)
- Quality
  - Is lower priority than getting the algorithm or tool to work [2]
  - Reliability and accuracy are still important in computational life-sciences
- Integration
  - Extremely difficult problem [3]
  - Highly related to the reuse problem

1. Barker, J. and Thornton, J. Software Engineering Challenges in Bioinformatics. In Proceedings of the International Conference on Software Engineering (Keynote address), Edinburgh, Scotland, UK, 2004
2. Stein, L. Bioinformatics: Gone in 2012. In Proceedings of the O'Reilly Bioinformatics Technology Conference (Keynote Address), San Diego CA, 2003
3. M. Burnett, C. Cook, and G. Rothermel, "End-user software engineering," *Commun. ACM, vol. 47, pp. 53-58, 2004*

Current trends

- With the open source movement, reuse should no longer be an elusive goal
- Massive repositories of source code are available on the web
- Project hosting sites such as Sourceforge.net
- Code-specific search engines are indexing these repositories (Koders, Krugle and Google Code Search)
- Open source enables opportunistic development strategies

Addressing the challenges in bioinformatics software

- Reuse in this field is low, despite emphasis on open source
- Existing tools do not provide adequate support
  - BioWareDB – Excellent database but poor search capability
  - Gonzui – Only prototype in 2004
- Agile nature of bioinformatics should promote reuse

➔**We propose a tool for supporting reuse**

- Indexing all available code would improve reuse and subsequently improve quality
- Professional programmers could also learn from existing artifacts

Search and indexing tool

- The tool could be a plug-in or a stand-alone implementation or an addition to existing functionality
- Code search engine functionality
- Would operate on an ontology of biology-related keywords and topics
- Search on source code from a variety of different sources such as
  - project hosting sites
  - code repositories of journals
  - open source project websites
  - lab websites

Search and indexing tool (Contd.)

- Built-in feature for annotations and recommendations
- Would enable social network analysis of CVS data leading to studies of collaboration
- This tool is still in its conceptual phase and has to be prototyped
- We hypothesize that such a tool would support reuse
  - But this idea needs confirmation from bioinformaticians

Tool development strategy: Contextual inquiry

- A design technique for creating tools by working closely with users
- User is a partner in the design process
- In-depth understanding of the user context
- A focused process
- Starts with structured interviews and observations of users working with existing code search engines

Conclusions

- Next step is to engage bioinformatics researchers and programmers to validate the feasibility and utility of such a tool
- An example of exploratory work leading to domain understanding leading to an idea for a tool and its design
- As software engineering becomes more domain-specific, tools need to evolve
- Our findings reveal that a large proportion of bioinformatics software development is opportunistic and tools that support the same should be created

Discussion

- Feasibility?
- From a methodology standpoint, how can we use our studies of programmers to create solutions for them?