

The Expression of Design in Bug Reports

Research Issues for the Continuous Design Workshop

Robert J. Sandusky
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
sandusky@uiuc.edu

Software problem management is often neglected in discussion and research on software design and development. Software problems arise as soon as coding begins and continue to occur as long as the software is in use (design “bugs” arise even before coding begins). Traditional software design methods do account to some degree for the inevitable occurrence of software problems by encouraging the development of maintainable and understandable designs and systems.

Software problem management is also associated with the wider concerns regarding software quality. While bugs are expected, too many serious bugs can lead to the rejection or abandonment of software, damage to the reputation or viability of software-producing organizations and other financial, human or social costs.

Ongoing research is revealing the patterns of activity, socio-technical misalignments, compensatory strategies, and knowledge management strategies used by one free/open-source software community to manage software problems. One of the most common practices found in this community’s bug report repository is *negotiation*, a social process that is used whenever people organize in order to get things done (Strauss, 1978). Negotiation occurs in many contexts including defining an appropriate design to resolve a software problem. Thus, bug report repositories contain rationale, suggestions and descriptions of activity in support of continuous system design. These design negotiations frequently appear as responses to the (usually implicit) question, “What is the best design for the fix for this bug?”

In the context of distributed free/open source software communities, bug report repositories also play a role in constituting the community’s design practices: the continuous design practices of a given community would be affected by making changes to the bug report repository, its usage policies, etc.

See <http://www.lis.uiuc.edu/~sandusky/designInBugReports.pdf> for a more detailed discussion of these issues.

Some research questions include:

Conduct investigations into the software problem management processes of F/OSS communities

- Develop and refine research methods that can be applied to the study of software problem management
- Investigate a variety of projects varying in size; scope; project type; etc. (e.g., compare commercial / traditional software problem management with F/OSS software problem management) to allow comparisons between different kinds of organizations
- Develop an understanding of how software problem management knowledge is created and used

Develop a theory of design based in software problem management

- Improve understanding of relationships between “big” software design (i.e., design as expressed in formal design documents, roadmaps, etc.); software development; software maintenance; software problem management; continuous re-design (incremental design)

Develop tools for F/OSS development communities

- Software problem management tools might provide:
 - automatic identification of duplicate bug reports
 - automatic identification of bug report networks (e.g., automatic discovery of unexpressed, latent relationships – dependencies, for example – between bug reports)
 - summarization of bug report networks
 - visualization of bug report networks
 - recovery and correlation of design ideas, rationale, code from multiple sources (e.g., bug report repositories; code repositories; newsgroups; IRC/IM archives)