

Design Rationale for Software Maintenance

Janet E. Burge and David C. Brown
jburge@cs.wpi.edu, dcb@cs.wpi.edu

<http://www.cs.wpi.edu/Research/aidg/DesignRationale.html>

*Department of Computer Science, Worcester Polytechnic Institute
100 Institute Road Worcester, MA 01609, USA*

1. The Problem

For a number of years, members of the Artificial Intelligence (AI) in Design community have studied *Design Rationale* (DR), the reasons behind decisions made while designing. A record of what decisions were made, and why, is especially valuable for software maintenance. One reason for this is that the software lifecycle is a long one. Large projects may take years to complete and spend even more time out in the field being used (and maintained). The combination of a long lifecycle and the typically high personnel turnover in the software industry increases the probability that the original designer is unlikely to be available for consultation when problems arise.

Lee's survey [1] presents an excellent overview of DR research. There has also been work specific to software design, such as Boehm's WinWin [2].

2. The Approach

Design rationale has a number of potential uses for documentation, evaluation, and assistance during all types of software maintenance.

To drive and evaluate this research, we will develop a system that supports the maintainer. This system will present the relevant DR when required and allow entry of new rationale for the modifications. It will also perform inferencing over the rationale to ensure the rationale is consistent and complete. Existing rationale will be used to guide the designer in determining rationale for new changes. When changes are made, the system will propagate any necessary changes to the existing DR as well as alerting the maintainer if the code modifications are the same as those made earlier and then rejected.

3. Current Status and Expected Results

Some preliminary work has been done in determining ways to inference over the rationale. This resulted in a

prototype system, InfoRat [3]. InfoRat supports validation of the rationale and evaluation of the design. It demonstrates that intelligent reasoning over DR can provide more beneficial use for the collected DR than just its retrieval and presentation. DR inferences can guide the design process and check for design quality. Poor rationale may lead to weaker designs.

A study is being performed to investigate rationale for different phases of software development and how it is used and modified during maintenance. Through this study, we hope to arrive at a better picture of what software rationale is and how it can be represented to support software maintenance.

In our future research, we plan to develop a representation for the rationale that occurs at multiple levels in the development process, from requirements through maintenance. This will include a design rationale ontology that supports inferencing by indicating the relationships between arguments at different levels of abstraction. We will also categorize different uses of DR during maintenance and design the DR representation to support them. We will develop a system to support DR use that propagates changes through the rationale to ensure that it is kept current and that captures how the rationale evolves over time. We will also develop a way to attach the rationale to the development artifacts (diagrams and code) so that it can be presented to and modified by the user.

References

- [1] J. Lee, "Design Rationale Systems: Understanding the Issues", *IEEE Expert*, Vol. 12, No. 3, 1997, pp. 78-85.
- [2] B. Boehm, and P. Bose, "A Collaborative Spiral Software Process Model Based on Theory W", *3rd International Conf. on the Software Process*, IEEE Computer Society Press, CA, 1994, pp. 59-68.
- [3] J. Burge, and D. C. Brown, "Inferencing Over Design Rationale", *Artificial Intelligence in Design '00*, J. Gero (ed.), Kluwer Academic Publishers, Netherlands, 2000, pp. 611-629.