Improving Design Intent Research for Software Maintenance

Paul S Grisham[1], Hajimu Iida[2], and Dewayne E. Perry[3]

[1,3]Empirical Software Engineering Laboratory
Electrical and Computer Engineering
The University of Texas at Austin
{grisham, perry}@ece.utexas.edu

[2]Software Design Laboratory
Graduate School of Information Science
Nara Institute of Science and Technology
iida@itc.naist.jp

Abstract

Design intent is a collection of decision-making factors that explain a design. Annotating software architecture models with design knowledge such as design intent may benefit maintenance activities. Unfortunately, researchers do not understand how software maintainers conduct design activities and use design documentation. This position paper presents a summary of design activities and design knowledge, research ideas on its use in software maintenance and design intent documentation in global developments.

1. Software Architecture and Intent

Software engineering relies on the expertise and judgment of designers to evaluate early designs. For this reason, understanding the *intent* of designers is critical when adapting or evolving software designs. As much as 80% of time spent in software development is spent in discovery or rediscovery of legacy systems [1]. Much of this time is spent trying to determine the original intent of a legacy design [2].

The Perry/Wolf model of software architecture explicitly includes *rationale* [3], although we now prefer to use the term *design intent* to describe a collection of decision-making factors that explain the design—including, but not limited to, rationale [2, 4]. Architectural drift occurs when changes to the implementation are not reflected in the design. Erosion is caused by changes that violate implicit or explicit design constraints. Although work in static analysis of architectures has provided some metrics to measure drift and erosion [5, 6], implied design rules remain difficult to infer from the final system design [7].

2. Research Trends

Current research focuses on schemas and tools for modeling knowledge about the design instead of how that knowledge is used in a maintenance context. This focus may reflect software researchers' natural bias toward technology and systems building and away from the social, behavioral and cognitive sciences.

There are also practical reasons for a lack of studies into design knowledge and maintenance. A clear and immediate benefit is necessary to inject a modeling technique into an initial design process. Also, the practicability of obtaining realistic design knowledge makes maintenance documentation studies difficult.

3. Initial Design Activities

Design rationale (DR) modeling [8] was proposed as a means of capturing knowledge about designs, but software design does not follow a rational process [9].

Modern software design is mainly a problem structuring activity producing satisficing solutions [10]. Early studies showed that software designers rely on emergent knowledge and drift between problems at different levels of abstraction [11]. Experienced designers do not spend much time on problem analysis, but structure the problem to fit known solutions and to scope subsequent problem-solving [12]. Moreover, designers may not even be aware they are making critical decisions at the time, only discovering the impact later in the design process. As a consequence, structuring the design process around rationale tends to impede problem solving for initial development [13].

For these reasons we suggest that research deemphasize DR modeling and study how data is being consumed during maintenance.

4. Maintenance Design Activities

In a survey of software professionals, 80% claimed that DR is necessary to understand a design [14], but the study does not identify specific maintenance benefits. In a software change analysis study, there was a positive benefit in cost and accuracy of changes with DR documentation in a simple program, but the impact was inconclusive on a more complex program [15].

In a study of design maintenance in the aerospace domain, participants used rationale documentation opportunistically (apply specific DR when questions could not be answered from the design) and extensively (scan through DR to identify design issues and use the design to understand how design solutions fit together) [16]. In an exploratory study, we found that participants did not know how to terminate searches over unstructured design documentation [4].

These results suggest that more research is required on how software maintainers solve problems and interact with documentation.

5. Design Intent Research Questions

We are developing new studies for understanding maintenance designers and their work processes. We believe the results from these studies will help researchers improve tools and notations for modeling and reusing design knowledge.

- Given that maintenance design is more constrained than initial design, is there a cognitive and behavioral difference in design activities?
- Senior designers create the initial system design of one project and then move to a new project. Maintenance is often left to novice architects [17]. How does expertise affect maintenance design?
- How do consumers of documentation identify information relevant to their needs and terminate searches? How can we structure intent knowledge to provide clear separation of concerns?
- What modeling principles should be used? Initial designers tend to favor modeling positive knowledge [14], but since maintenance design is more constrained, recording mitigating actions [18] and anticrises [19] may be more useful.

6. Impact on Global Development

Software companies may outsource their maintenance activities to other companies and countries. There is no agreed-upon set of notations or tools for modeling meta-knowledge about designs. Software designs are often documented using word processing and spreadsheet tools, rather than specialized design environments [17]. Design knowledge is often only available as natural language documentation. We have not yet explored how language and cultural differences may prove to be a barrier to the adoption of design intent.

7. Acknowledgements

This research was supported in part by the JSPS 2007 Summer Program, the NSF 2007 East Asia and Pacific Summer Institute (EAPSI) Program, and NSF CISE grant CCR–0306613 “Transforming requirement specifications into architectural prescriptions.”

8. References

[1] J. W. Davison, D. M. Mancl, and W. F. Opdyke, "Understanding and Addressing the Essential Costs of Evolving Systems," *Bell Labs Tech. Journ.*, vol. 5, pp. 44-54, Apr. 2000.

[2] D. E. Perry and P. S. Grisham, "Architecture and Design Intent in Component and Cots-Based Systems," in *Int'l Conf. on COTS-Based Software Systems*, Orlando, FL, 2006.

[3] D. E. Perry and A. L. Wolf, "Foundations for the Study of Software Architecture," *ACM SIGSOFT Software Engineering Notes*, vol. 17, pp. 40-52, October 1992.

[4] P. S. Grisham, M. J. Hawthorne, and D. E. Perry, "Architecture and Design Intent: An Experience Report," in *2nd Workshop on SHAring and Reusing architectural Knowledge - Architecture, rationale and Design Intent (SHARK/ADI)* Minneapolis, MN, 2007.

[5] E. Johansson and M. Höst, "Tracking Degradation in Software Product Lines through Measurement of Design Rule Violations," in *14th int'l conf. on Software Engineering and Knowledge Engineering (SEKE)* Ischia, Italy, 2002.

[6] S. Bhattacharya and D. E. Perry, "Architecture Assessment Model for System Evolution," in 8th *Working Int'l Conf. on Software Architecture (WICSA)* Mumbai, India, 2007.

[7] J. v. Gurp and J. Bosch, "Design Erosion: Problems and Causes," *Journ. of Sys. & Soft.*, vol. 61, pp. 105-119, Mar. 2002.

[8] J. Lee, "Design Rationale Systems: Understanding the Issues," *IEEE Expert*, vol. 12, pp. 78-85, May/Jun 1997.

[9] D. L. Parnas and P. C. Clements, "A Rational Design Process: How and Why to Fake It," *IEEE Transactions on Software Engineering*, vol. 12, pp. 251-258, February 1986.

[10] C. Zannier, M. Chiasson, and F. Maurer, "A Model of Design Decision Making Based on Empirical Results of Interviews with Software Designers," *Information and Software Technology*, vol. 49, pp. 637-653, June 2007.

[11] R. Guindon, "Designing the Design Process: Exploiting Opportunistic Thoughts," *Human-Computer Interaction*, vol. 5, pp. 305-344, 1990.

[12] N. Cross, "Expertise in Design: An Overview," *Design Studies*, vol. 25, pp. 427-441, September 2004.

[13] S. B. Shum and N. Hammond, "Argumentation-Based Design Rationale: What Use at What Cost?," *Int'l Journal of Human-Computer Studies*, vol. 40, pp. 603-652, April 1994.

[14] A. Tang, M. A. Babar, I. Gorton, and J. Han, "A Survey of Architecture Design Rationale," *Journal of Systems and Software*, vol. 79, pp. 1792-1804, December 2006.

[15] L. Bratthall, E. Johansson, and B. Regnell, "Is a Design Rationale Vital When Predicting Change Impact?" in *2nd International Conference on Product Focused Software Process Improvement (PROFES)* Oulu, Finland, 2000.

[16] L. Karsenty, "An Empirical Evaluation of Design Rationale Documents," in *SIGCHI Conf. on Human Factors in Computing Systems* Vancouver, British Columbia, Canada, 1996.

[17] P. Avgeriou, P. Kruchten, P. Lago, P. Grisham, and D. Perry, "Architectural Knowledge and Rationale – Issues, Trends, Challenges," *ACM SIGSOFT Software Engineering Notes*, vol. 32, p. 41-46, July 2007.

[18] W. Wu and T. Kelly, "Managing Architectural Design Decisions for Safety-Critical Software Systems," *2nd Int'l Conf. on the Quality of Soft. Arch. (QoSA)* Västerås, Sweden, 2006.

[19] P. Kruchten, "An Ontology of Architectural Design Decisions in Software Intensive Systems," in *2nd Groningen Workshop on Software Variability* Groningen, The Netherlands, 2004.