

Industrial Perspective on the Usefulness of Design Rationale for Software Maintenance: A Survey

Muhammad Ali Babar¹ Antony Tang² Ian Gorton¹ Jun Han²

¹*National ICT Australia Ltd. and
University of NSW
Australia*

{malibaba,ian.gorton}@nicta.com.au

²*Faculty of ICT*

*Swinburne University of Technology
Australia*

{atang,jhan}@ict.swin.edu.au

Abstract

Software maintenance is widely known as a problematic area that may consume up to 80% of a software project's resources. It has been claimed that providing an effective mechanism to access Design Rationale (DR) has great potential to improve software maintenance processes. However, we postulate that the first step towards exploring the potential of DR for improving software maintenance should be to gain a better understanding of what DR means to practitioners, how valuable they consider DR to be and how they use DR. To determine the perceived usefulness of DR, we surveyed a large number of software designers. This exploratory study has discovered that practitioners recognize the importance of DR to understand existing designs and frequently use it to reason about proposed modifications. The results of this study establish that DR is perceived by practitioners to be useful and the efforts required to capture DR for the purpose of maintenance are worthwhile. The findings allow us to identify areas of further research on DR support that have the potential to improve the maintenance process.

1. Introduction

Maintenance and enhancement of large-scale software systems is one of the most difficult and expensive activities in the software development lifecycle. Software maintenance costs can be more than 40 percent of the total cost of developing the software [1, 2]. Some studies conclude that maintenance effort can consume up to 80 percent of the system and programming resources [3]. Given such a high cost activity, any effort aimed at lowering the maintenance cost by improving the quality of

maintenance practices is considered a valuable objective [4]. One of the means of improving maintenance processes is the ease of access to the information required to support program comprehension task [4, 5], which may consume a large amount of the total time spent on software maintenance [2]. There have been several research efforts to understand the information needs, use and search strategies of software maintainers in order to provide appropriate support mechanisms [4, 6, 7]. However, these studies are primarily focused on lower level maintenance tasks like source code maintenance and modifications and hence do not consider the information requirements of those maintenance tasks that have architectural implications.

Like others [8-11], we believe that design rationale can be a vital source of information to support Software Architecture (SA) sensitive modifications and enhancements. Design Rationale¹ (DR) represent the reasoning for designs including how functional and quality requirements are satisfied, why certain structures are selected over alternatives and what type of system's behavior is expected under different environmental conditions [12, 13].

There is growing emphasis on the importance of capturing and managing architecture design rationale [5, 14-16]. However, capturing DR poses many challenging problems. For example, it can increase the design efforts by 40 percent [8], which may affect project schedule and cost [9]. Designers usually do not find immediate benefit in the extra work required to capture and maintain DR. That is why a common perception is that designers generally do not explicitly document the contextual knowledge about their design

¹ Wherever we use DR, we mean rationale underpinning the architecturally sensitive design decisions. We call this architecture design rationale.

decisions [14, 17]. But there is little empirical research on determining ways to use DR in performing maintenance related tasks. We believe that a key to making the DR capture worthwhile is to demonstrate its use and utility [9]. The resources required to capture DR can only be perceived as useful if the information gathered is valuable for supporting a particular task. We assert that perhaps the best evidence to demonstrate the use of DR can be found among those who make architectural design decisions on a regular basis for new or existing systems.

The purpose of this paper is to report the results of a survey of a large number of practitioners who had experience in architecture design. The results should shed light on potential usefulness of DR for improving software maintenance processes and on the designers' perceptions of the value of documenting and using DR. The goals of the work described in this paper are:

- To gain an understanding of designers' perception of importance of architecture design rationale use and documentation.
- To gather empirical evidence for the value of capturing DR to support maintenance tasks.

This study was designed to explore the need for using DR during software maintenance. We have discovered several interesting findings which support the notion of DR's importance in software maintenance. These results prompt us to identify a set of research questions to guide our future efforts in this line of research. Moreover, since an established theory explaining attitude and behaviour toward, and use of architecture design rationale does not exist, this study is using an inductive approach (i.e. using facts to develop general conclusions) as an attempt to move toward such a theory. The main contributions of this paper are:

- Based on empirical evidence gathered through a survey, we have established that architecture design rationale is perceived by practitioners to be important in software maintenance.
- It identifies further research areas that need to be explored to support and improve current architecture design rationale practices.

2. Background

2.1 DR Approaches in Software Engineering

Researchers and practitioners in different design engineering disciplines have developed several DR methods since their origin from argumentation formalization theory developed in social science discipline in late 50s [18]. Early work that emphasizes the importance of design rationale in software design

can be found in [19, 20]. The SA community is increasingly emphasizing the importance of documenting and using DR to maintain and evolve architectural artifacts and to avoid violating design rules that underpin the original architecture [14-16]. The growing recognition of the importance of documenting and maintaining DR has resulted in several efforts to provide guidance for capturing and using DR such as the IEEE 1471-2000 standard [21] and the Views and Beyond (V&B) approach [22]. However, the former provides a definition of design rationale without further elaboration, while the latter provides a list of elements of DR without justifying why these elements are important and how the information captured is beneficial in different contexts.

Hence, there is a need for guidelines on how to document and use DR to understand the architectural ramifications of any modification or enhancement effort. Our research efforts are focused on improving the use and documentation of DR. We aim to achieve this by developing and assessing a conceptual framework and support structure to facilitate the capture and use of DR for improving software maintenance processes. We postulate that understanding the current industrial practices related to DR use and practitioners' perception of the usefulness of DR is one of the most important steps towards that goal.

2.2 Software Maintenance and Architecture Design Rationale

Software maintenance has been defined as the changes performed on operational application systems to correct faults, to adapt to a changed environment or to improve performance or other attributes [23]. Several researchers advocate that DR can be a source of invaluable knowledge to support maintenance or enhancement design decisions [24, 25]. They argue that if the original designer is not available during software maintenance, DR documentation can provide answers to many of the questions regarding the "Why" of certain design choices [11].

Moreover, any change in one component may have potential "ripple effects", which need to be identified and evaluated. This is called impact analysis, which is an important activity of software maintenance process. Impact analysis requires the knowledge of the dependencies between different components and also involves backward (source code to design) and forward (design to source code) dependency traceability [26]. Availability of architecture design rationale can greatly facilitate each of these tasks [27].

3. Research Approach and Findings

Considering the objectives of our research and available resources, we decided to use a survey research method to understand architects' perceptions and their current practices in architecture design rationale. Having reviewed the published literature on design rationale, we developed a survey consisting of 30 questions on design rationale understanding and practices and 10 questions on the demographics of the respondents. Demographic questions were designed to screen respondents and help identify data sets to be excluded from the final analysis. We ran a formal pilot study to test and refine the survey instrument.

We used an online web-based tool, Surveyor [28] to implement the survey questionnaire. The target population for the survey consisted of people with three or more years of experience in software development and who work as a software designer or architect. Considering the fact that software designers usually have major time constraints, it was not feasible to attempt random sampling because the response rate could be low. Consequently, we used availability and snowballing sampling techniques. The major drawback of these sampling techniques is that the results are statistically generalizable only to the population with the same characteristics as the samples. Being an exploratory study, we believe our sampling techniques are appropriate.

We invited a pool of designers and architects drawn from the industry contacts of the four investigators, and past and current postgraduate students of Swinburne University of Technology and the University of New South Wales. We also requested the invitees to forward the invitation to others who were eligible for participation.

Apart from the questions on demographical characteristics, we grouped the questions into different sections depending on the topic of research, namely

- architecture evaluation in organization;
- role of DR in maintenance and enhancements;
- considerations of architectural risks;
- the level of importance, use and documentation of different types of rationales (such as constraints, strengths, weaknesses and others).

This paper only discusses the questions related to DR use and usefulness for architecture sensitive modification and enhancements. The findings related to other questions have been reported in [29]. We also provide an extensive analysis of the data for all the sections of the study instrument in [30].

4.1. Respondent Demographics

We directly sent survey invitation to 171 potential practitioners. Our invitation was forwarded to 376 more people by the original invitees, meaning 547 invitations were sent. We received a total of 127 responses, which corresponds to an uptake rate of 23%. Out of the total responses, we excluded 46 responses from the analysis as they were incomplete or the respondents did not meet the 3 years IT work experience criteria.

80% of our respondents were male and 20% are female. 68% of respondents live in Australia and New Zealand, 28% reside in Asia. The respondents' experience in the information technology industry varies between 4 years and 37 years with a median of 15 years. A median score for working as a designer or architect is 8 years and a median score for working with one organization (current or previous) is 6 years. An average number of co-workers on the current (or last) project is 25 people. 85% of the respondents have received an IT related tertiary qualification. These demographics give us confidence that we have gathered data from practitioners who are experienced in software architecture. We assert that the findings of this survey can be generalized to designers with similar characteristics.

4.2. Perceived Importance of DR

We postulated that a positive attitude towards DR would indicate that practitioners would be more likely to use and document DR. This is an indication that DR can be useful, at least for those who use it as a support tool when designing or maintaining a software architecture [11]. Moreover, several arguments have been made about the importance of documenting key architecture decisions along with the contextual information [17, 19]. A lack of, or low quality documentation has been consistently quoted as one of the problems in maintenance process [3, 31]. Our questionnaire included questions aimed at gaining some understanding of the designer's perception of the importance of DR use and documentation. These questions concerned the frequency of reasoning about design decisions, importance of DR for justifying design decisions and the level of DR documentation.

	Never to Always				
	1	2	3	4	5
No of Resp. (%)	0 (0)	1 (1.2)	8 (9.9)	34 (42)	38 (46.9)

Table 1: Reasoning about Design Choices

Table 1 presents the responses to the question on the frequency of reasoning about design choices. The results show when making design decisions, large majority of the respondents reason about their design decisions on a frequent basis.

	Strongly disagree to Strongly agree				
	1	2	3	4	5
No of resp. (%)	0 (0.0)	1 (1.2)	11 (13.6)	30 (37.0)	39 (48.1)

Table 2: Importance of DR for Justification

We asked a direct question on the perceived importance of DR for justifying design choices. Table 2 shows that an overwhelming majority of the respondents think that DR is important to justify design decisions, which indicates a positive attitude towards the value of DR. These results are similar to [10], where participants in controlled experiments found that DR were important in coming up with correct solutions to enhancement problems.

We asked the respondents to indicate the overall level of documenting DR. 62.9% of the respondents replied that they completely document DR, while 24.7% described their DR documentation as partial. This indicates that DR documentation is seen as an important activity by the majority of respondents. We will further discuss this issue in Section 5.

4.3 Usefulness of DR for Software Maintenance

The next section of the survey attempted to examine the usefulness of DR by identifying various uses of DR during software maintenance processes². We also were interested in the kind of help DR can provide for architectural modification and evolution activities.

We asked respondents how often they revisit design documentation and specifications to help them understand the system before performing enhancements. Table 3 presents their responses, which indicate that the majority of the respondents consult design documentation frequently while performing maintenance or enhancement tasks. These results show that if DR are sufficiently documented and provided to the software maintainers, they are more likely to frequently use DR. These results are also consistent with the findings reported in [4], which conclude that system documentation, if available, is frequently used when performing maintenance tasks. It is also found that if the knowledge leading to design is available, a

² Only 69 respondents decided to respond to the questions in this section. Thus, results in section 4.3 are based on maximum of 69 responses.

maintainer can effectively perform modification tasks by consulting that knowledge [8].

There have been contradictory reports about whether the designers need to know the system's DR to understand an unfamiliar design. Herbsleb and Kuwana reported that during a design meeting designers rarely asked questions about DR [32]. While based on an empirical study, Karsenty found that access to DR is important, especially when a designer work on someone else's design [11]. Moreover, Seaman reports that human sources are usually desired to help explain different aspects of the system to be maintained [4].

	Never to Always				
	1	2	3	4	5
No of resp. (%)	0 (0.0)	8 (11.9)	9 (13.4)	20 (29.9)	30 (44.8)

Table 3: Frequency of Revisiting Design Documentation before Making Changes.

	Strongly Disagree to Strongly Agree				
	1	2	3	4	5
No of Resp. (%)	1 (1.5)	3 (4.6)	9 (13.8)	23 (35.4)	29 (44.6)

Table 4: Fail to Understand the Reasons for Existing Design without DR.

The participants of our study were asked if they think that they may fail to understand the reasons behind a design without the relevant DR. Table 4 presents the responses to this question. The results show that a large number of respondents think that DR can help comprehend existing design decisions. This help may come in two forms:

- if not captured, DR knowledge is lost making it unavailable for evaluating past design decisions without having access to the decision maker.
- expedite the reasoning process by minimizing the need of deducing DR from the design specifications or accessing the human sources.

Thus, we can say that there is some evidence that the availability of DR is useful to improve software maintenance.

Another important use of the rationale underpinning the design decisions is to help assess the available alternatives for modification or enhancement. Constraints such as real-time considerations, resource limitation, logistic knowledge and others help to capture the limitations to the design. The knowledge provided by DR assist in understanding past decisions for future modification [33].

	Strongly disagree to Strongly agree				
	1	2	3	4	5
No of Resp. (%)	0 (0.0)	3 (4.5)	10 (14.9)	21 (31.3)	33 (49.3)

Table 5: DR Helps Evaluate Previous Design Decision for Future Modifications

When asked about the usefulness of DR to help understand past design decisions to assess potential modification, a majority of the respondents find DR helpful in this regards (Table 5). Hence, we concluded that there is evidence to support the above-mentioned researchers' claims that DR are an important source of effective reasoning during architectural modification.

It is also stressed that if the knowledge concerning the domain analysis, patterns used, design options evaluated, and decisions made is not documented, it is quickly lost and hence unavailable to support subsequent decisions in the development lifecycle [14, 17]. Software maintenance experts also agree that many facts may be lost to the project, either because the developers may no longer be available to the project or the limitations on a human's ability to memorize detailed facts [31]. In order to empirically assess these claims, we asked our respondents how often they think they forget the reasons underpinning their design decision after some time.

	Never to Always				
	1	2	3	4	5
No of resp. (%)	2 (3.0)	15 (22.7)	22 (33.3)	22 (33.3)	5 (7.6)

Table 6: Tendency of Forgetting the Reasons for Justifying Design Decisions

Table 6 shows the results to this question. The responses show that there are about 74% of respondents who say they forget reasons about design decisions either half the time or more. This finding should provide a strong reason for regularly documenting and maintaining DR to support architecture maintenance and evolution. As mentioned earlier, some of the reasons could be deduced through inspecting available design specifications but some reasons will inevitably be lost if the design is complex and the system was developed a long time ago.

DR have been considered vital in performing impact analysis as a result of any change request. Moreover, if the changes are architecturally sensitive, it becomes more important to support the impact analysis with DR [8, 10]. In order to determine the value of using DR during impact analysis, we want to determine how often the respondents perform impact analysis. The reason is that some claim that

maintenance is usually performed on an ad hoc and quick fix basis [34], which may not involve a systematic reasoning like impact analysis. A high frequency of impact analysis should indicate the need for knowledge underpinning the existing design.

	Never to Always				
	1	2	3	4	5
No of resp. (%)	0 (0.0)	4 (6.2)	13 (20.0)	29 (44.6)	19 (29.2)

Table 7: Frequency of Performing Impact Analysis.

Sixty five respondents answered this question. Table 7 shows that a large number of respondents perform impact analysis before making any changes. Since impact analysis may consume a significant amount of resources and 80.4% designers consider it to be useful in evaluating previous design decisions, it can be argued that the availability of DR can improve this process. DR is considered an important input to impact analysis because if the knowledge about the factors that may have influenced the original design decisions is available, designers do not have to spend a large amount of time to understand the existing design and the potential ramifications of any changes [7, 24].

We were also interested in determining the importance of the tasks in impacts analysis, some of which require DR support. The importance of such tasks during impact analysis can be considered as an indicator of the need or usefulness of DR to improve this activity and subsequently maintenance process. To investigate this issue, there was a multiple-items question on the importance of various tasks.

	Not Important to Very Important				
	1	2	3	4	5
a) Analyse & Trace Req.	1.5	0.0	10.3	39.7	48.5
b) Analyse Specs. of Previous Design	1.5	10.3	19.1	35.3	33.8
c) Analyse DR of Previous Design	1.5	11.9	32.8	32.8	20.9
d) Analyse Implementation Feasibility	1.5	1.5	8.8	41.2	47.1
e) Analyse Violation of Constraints & Assumptions	1.5	8.8	25.0	41.2	23.5
f) Analyse Scenarios	1.5	7.6	16.7	36.4	37.9
g) Analyse Cost of Implementation	0.0	10.3	13.2	30.9	45.6
h) Analyse Risk of Implementation	0.0	4.4	5.9	32.4	57.4

Table 8: Importance of Each Impact Analysis Task (results in percentage).

Table 8 presents the distribution of the 67 responses to different items of this question. The results show that our respondents consider that analysing requirements, feasibility and risk of implementation are the most important aspects of impact analysis. The respondents seem more concerned to ensure that any changes to the existing system should be feasible, risk free and consistent with the existing requirements. A number of these tasks would benefit from some DR support, namely, (a), (b), (c), (e), (g) and (h). Documentation that includes DR would provide an understanding of what is in existence and why. That means documented specifications and DR could expedite the understanding of the system that requires modifications [10]. During impact analysis for the new enhancements, reasoning with DR on issues such as (d), (g) and (h) is also important. The summary of the findings for this multi-items question are that:

- 54% of the respondents think that analysing previous DR is important,
- 65% of the participants find that analysing previous constraints are important,
- 77% of the respondents perceive that analysing cost of new implementation is important, and
- 90% of the respondents report that analysing risk of new implementation is important.

We have already mentioned 80.6% of the respondents said that DR is important in helping them to understand previous designs and assess options in system enhancement.

Impact Analysis Tasks	Use of DR	Correlations
a) Analyse & Trace Requirements	Level of Documentation	$[r(67) = +.297, p < 0.05]$
	Importance in using DR	$[r(67) = +.333, p < 0.01]$
b) Analyse Previous Spec.	Level of Documentation	$[r(67) = +.267, p < 0.05]$
	Importance in using DR	$[r(67) = +.279, p < 0.05]$
c) Analyse DR of Previous Design	Level of Documentation	$[r(67) = +.251, p < 0.05]$
	Importance in using DR	$[r(67) = +.351, p < 0.01]$
e) Analyse Violation of Constraints & Assumptions	Level of Documentation	$[r(67) = +.295, p < 0.05]$
	Importance in using DR	$[r(67) = +.306, p < 0.05]$

Table 9: Correlation between Use of DR and Impact Analysis Tasks

We performed Spearman's rho correlation tests between each of the impact analysis tasks and:

- the overall level of DR documentation (reported in [29])

- the importance of using DR for justification.

The correlation analysis found that the items (a), (b), (c) and (e) in Table 8 are positively correlated with the overall level of DR documentation and the importance of using DR to justify design decisions. A summary of the results is shown in Table 9.

The presence of correlation indicates that those respondents who analyse and trace requirements are also the people who prepare DR documentation and think that DR is important. Similarly, those who believe that design specifications, design rationale and constraints are important in impact analysis also value the importance of DR to justify their design decisions and document DR. That means for those designers who document DR, it seems likely that they will use this during impact analysis. Moreover, it also indicates that the designers who trace requirements and design specifications value the importance of DR. These findings provide evidence to support the usefulness of DR for system enhancements.

5. Discussions

DR are useful in supporting maintenance activities because they expose implicit assumptions, constraints and dependencies in an architecture design which might otherwise be unnoticed and cause design inconsistency. Our findings indicate that practitioners perceive DR as important to support software maintenance and enhancement tasks. The majority of the designers, who participated in our survey, frequently use design rationales to justify design choices. There is overwhelming support for consulting DR, in whatever form it is available, for evaluating available solutions for architectural modifications. Moreover, there seems to be a common agreement among the majority of the respondents that DR are helpful to understand past design decisions in order to make future modifications. Another interesting finding is that a large number of the respondents believe that they document the knowledge underpinning their design decisions. These findings are contrary to the claims made in the literature about the lack of DR documentation [14, 17]. One might say that people with positive attitude towards DR may have reported a spurious level of documentation. A counter argument is that the level of documentation reported by the respondents may not be inflated but the information captured in design specifications documents provides only indirect DR through deduction. However, both arguments lack empirical evidence.

Altogether, these findings express a positive perception of DR usefulness for supporting software

maintenance tasks. This bodes well for any future effort of introducing systematic ways of DR capture and use in industry. It has been demonstrated that a positive attitude towards a technology impacts one's intention to use the technology [35, 36]. Hence, a positively perceived importance and high frequency of use of DR by the respondents can be used to predict that the state-of-the-practice in the use and documentation of DR will improve with the increasing realization of the damaging effects that the failure to capture and maintain design rationale causes.

There also seems to be a common practice of performing systematic impact analysis that usually requires access to the knowledge leading to the design being analyzed. Such a high frequency of impact analysis enables us to conjecture that DR, an important input to the impact analysis [26], needs to be captured and maintained to improve the maintenance process.

The positive correlation between impact analysis tasks and the use and documentation of DR indicates that the respondents realize the connection between the two tasks at different stages of the development life-cycle. One interpretation of this finding can be that there is a vital need to capture and maintain DR as a first class entity to facilitate traceability during maintenance activities. Non-availability of DR may result in violating the constraints or assumptions underpinning the design decisions [14, 37]. That is why we argue that the availability of DR is likely to facilitate systematic reasoning to minimize the risks associated with system enhancements. Collectively these results can be used to conclude that there is a strong support for the use of DR in various aspects of impact analysis.

It is important to state that the findings discussed are based on subjective, perceptual data. Though, it is argued that the subjective perceptions usually lay a better foundation to develop a theory than objective data [38], we admit that the reliability of the perceptual retrospective data can be questionable. That is why we realize that there is a need for additional research, possibly a longitudinal one, that includes more objective means to study the usefulness of DR.

6. Limitations

Like most surveys in software engineering, our study faced reliability and validity threats. Following the guidelines provided in [39], we put certain measures in place to address validity and reliability issues. For example, the research instrument underwent rigorous evaluation by experienced researchers and practitioners [39], all the questions

were tested in a pilot study, and respondents were assured of anonymity and confidentiality. However, completely eliminating the possibility of bias error is difficult.

The results may suffer from non-response error. If only those with a positive opinion about the DR responded, the results would be biased. However, we are unable to identify non-respondents because the survey was anonymous. That respondents are mainly from the Asia Pacific region is another limitation as the findings may not be generalized globally. Thus, there is a need to replicate the study in other geographical regions. Our survey instrument is available in [30] for such a comparative study.

Our study also suffers from the non-existence of a proven theory of designers' attitude towards documenting DR. Hence we consider this research as an exploratory effort to draw some general conclusions that can help establish some empirical support for future research directions that can develop and validate such a theory.

7. Conclusions and Future Work

The goal of this study was to establish empirical support for the importance and usefulness of architecture design rationale to support software maintenance. This study has gathered new information about how software designers perceive and use design rationale. The results shed light on the role and usefulness of DR to support architecture modification and evolution and we are able to conclude that the effort and resources required to capture DR can be worthwhile depending on the context. From the survey results, we have identified following goals that we plan to follow in the future:

- Collect additional data on the perceived utility of DR for software maintenance through interviews and case studies to achieve more generalizable findings
- Conduct empirical studies to determine what level of documentation is currently captured in design specifications. Based on any findings, we can start to assess the context under which rationale documentation is considered important and how its use is justified by a return on investment.
- It is also important to determine the mechanics of the costs and benefits of capturing and using DR to improve software maintenance.

Furthermore, this survey did not examine the types of system, such as complex / simple or enterprise / stand-alone, or the usefulness of DR to maintain them. Anecdotal evidence tells us that DR is more beneficial for large/complex systems than simple/small systems.

Hence, there is a need for identifying the types and amount of DR required for different types of systems and in various contexts.

8. References

- [1] Brooks, F.P., *The Mythical Man-Month: Essays on Software Engineering*. 1995: Addison-Wesley.
- [2] Pigoski, T.M., *Practical Software Maintenance: Best Practices for Managing Your Software Investment*. 1996: John Wiley & Sons, Inc. New York.
- [3] Lientz, B.P. and E.B. Swanson, *Characteristics of Application Software Maintenance*. Communication of the ACM, 1978. **21**(6): pp. 466-471.
- [4] Seaman, C.B. *The Information Gathering Strategies of Software Maintainers*. Proc. of the Int'l Conf. on Software Maintenance. 2002. Montreal, Canada: IEEE, Computer.
- [5] Curtis, B., et al., *A Field Study of the Software Design Process for Large Systems*. Communications of the ACM, 1988. **31**(11).
- [6] Sim, S.E., et al. *Archetypal Source Code Searches: A Survey of Software Developers and Maintainers*. Proc. 6th Int'l Workshop on Program Comprehension. 1998. Ischia, Italy.
- [7] Tjortjijis, C. and P. Layzell. *Expert Maintainers' Strategies and Needs When Understanding Software: A Case Study Approach*. Proc. of the 8th Asia-Pacific Software Engineering Conference. 2001.
- [8] Hamada, M. and H. Adachi. *Recording Software Design Processes for Maintaining the Software*. Proc. of the 17th Computer Software and Applications Conference. 1993. Arizona, USA.
- [9] Burge, J.E. and D.C. Brown. *Rationale Support for Maintenance of Large Scale Systems*. Workshop on Evolution of Large-Scale Industrial Software Applications, Collocated with ICSM. 2003.
- [10] Bratthall, L., et al., *Is a Design Rationale Vital when Predicting Change Impact? - A Controlled Experiment on Software Architecture Evolution*, in *Lecture Notes in Computer Science*, F. Bomarius and M. Oivo, Editors. 2000, Springer-Verlag. p. 126-139.
- [11] Karsenty, L., *An Empirical Evaluation of Design Rationale Documents*. CHI, 1996. April 1996: pp. 13-18.
- [12] Gruber, T.R. and D.M. Russell, *Design Knowledge and Design Rationale: A Framework for Representing, Capture, and Use*. Tech Report KSL 90-45 1991 Knowledge Systems Laboratory, Stanford University, California, USA
- [13] Lee, J., *Design Rationale Systems: Understand the Issues*. IEEE Expert, 1997. **12**(3): pp. 78-85.
- [14] Bosch, J. *Software Architecture: The Next Step*. European Workshop on Software Architecture. 2004.
- [15] Bass, L., et al., *Software Architecture in Practice*. 2 ed. 2003: Addison-Wesley.
- [16] Perry, D.E. and A.L. Wolf, *Foundations for the Study of Software Architecture*. ACM SIGSOFT, Software Engineering Notes, 1992. **17**(4).
- [17] Tyree, J. and A. Akerman, *Architecture Decisions: Demystifying Architecture*. IEEE Software, 2005. **22**(2): pp. 19-27.
- [18] Shum, S.J., *A Cognitive Analysis of Design Rationale Representation*, in *Department of Psychology*. Dec. 1991, University of York, UK.
- [19] Parnas, D. and P. Clements, *A Rationale Design Process: How and Why to Fake It*. IEEE Transactions of Software Engineering, 1986. **12**(2): pp. 251-257.
- [20] Potts, C. and G. Bruns. *Recording the Reasons for Design Decisions*. Proc. of the 10th Int'l Conf. on Software Eng. 1988. Singapore.
- [21] *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*: IEEE Standard No. 1471-2000.
- [22] Clements, P., et al., *Documenting Software Architectures: Views and Beyond*. 2002: Addison-Wesley.
- [23] IEEE. *IEEE Standard for Software Maintenance (IEEE Std 1219-1993)*. 1993: Institute of Electrical and Electronic Engineers, Inc. New York NY.
- [24] Ramesh, B. and B. Dhar, *Supporting Systems Development by Capturing Deliberations During Requirements Engineering*. IEEE Transaction on Software Engineering, 1992. **18**(6): pp. 498-510.
- [25] Han, J. *Designing for Increased Software Maintainability*. Proc. of Int'l Conf. on Software Maintenance. 1997. Bari, Italy.
- [26] Fyson, M.J. and C. Boldyreff, *Using Application Understanding to Support Impact Analysis*. Journal of Software Maintenance: Research and Practice, 1998. **10**: pp. 93-110.
- [27] Tang, A. and J. Han. *Architecture Rationalization: a Methodology for Architecture Verifiability, Traceability and Completeness*. 12th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems ECBS 2005. 2005. U.S.A.: IEEE.
- [28] ObjectPlanet, *Surveyor, Web-based Survey Instrument*. 2002.
- [29] Tang, A., et al. *An Empirical Study of the Use and Documentation of Architecture Design Rationale*. Accepted in the 5th Working IEEE/IFIP Conference on Software Architecture. 2005.
- [30] Tang, A., et al., *A Survey of Architecture Design Rationale*. Tech Report SUTICT-TR2005.02 2005 Swinburne University of Technology, Available at <http://www.cse.unsw.edu.au/~malibaba/SUTICT-TR2005.02.pdf>
- [31] Kajko-Mattsson, M., *A Survey of Documentation Practice within Corrective Maintenance*. Empirical Software Engineering, 2005. **10**(1): pp. 31-55.
- [32] Herbsleb, J.D. and E. Kuwana, *Preserving Knowledge in Design Projects: What Designers Need to Know*. INTERCHI, 1993.
- [33] Selfridge, P.G., et al. *Managing Design Knowledge to Provide Assistance to Large-Scale Software Development*. Proc. of the 7th Knowledge-based Software Engineering. 1992.
- [34] Sousa, M.J.C. and H.M. Moreira. *A Survey on the Software Maintenance Process*. Proc. of the Int'l. Conf. on Software Maintenance. 1998. Maryland, USA.
- [35] Ajez, I., *The Theory of Planned Behavior*. Organizational Behavior and Human Decision Processes, 1991. **50**(2): pp. 179-211.
- [36] Davis, F.D., *Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology*. MIS, Quarterly, 1989. **13**(3): pp. 319-340.
- [37] Gotel, O.C.Z. and A.C.W. Finkelstein. *An Analysis of the Requirements Traceability Problem*. Proc. of Int'l Conf. on Requirements Eng. 1994.
- [38] Moore, G.C. and I. Benbasat, *Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation*. Information Systems Research, 1991. **2**(3): pp. 192-222.
- [39] Kitchenham, B. and S.L. Pfleeger, *Principles of Survey Research, Parts 1 to 6*. Software Engineering Notes, 2001-2002.