



## A survey of architecture design rationale

Antony Tang<sup>a,\*</sup>, Muhammad Ali Babar<sup>b</sup>, Ian Gorton<sup>b</sup>, Jun Han<sup>a</sup>

<sup>a</sup> Faculty of ICT, Swinburne University of Technology, John Street, Hawthorn, Melbourne, Vic. 3122, Australia

<sup>b</sup> National ICT Australia Ltd. and University of NSW, Australia

Received 17 January 2006; received in revised form 25 April 2006; accepted 26 April 2006

### Abstract

Many claims have been made about the consequences of not documenting design rationale. The general perception is that designers and architects usually do not fully understand the critical role of systematic use and capture of design rationale. However, there is to date little empirical evidence available on what design rationale mean to practitioners, how valuable they consider it, and how they use and document it during the design process. This paper reports a survey of practitioners to probe their perception of the value of design rationale and how they use and document the background knowledge related to their design decisions. Based on 81 valid responses, this study has discovered that practitioners recognize the importance of documenting design rationale and frequently use them to reason about their design choices. However, they have indicated barriers to the use and documentation of design rationale. Based on the findings, we conclude that further research is needed to develop methodology and tool support for design rationale capture and usage. Furthermore, we put forward some specific research questions about design rationale that could be further investigated to benefit industry practice.

© 2006 Elsevier Inc. All rights reserved.

*Keywords:* Design rationale; Software architecture; Survey

### 1. Introduction

Design rationale captures the knowledge and reasoning that justify the resulting design. This includes how the design satisfies functional and quality requirements, why certain design choices are selected over alternatives and what type of system behaviour is expected under different environmental conditions (Gruber and Russell, 1991; Lee, 1997). Despite the growing recognition of the need for documenting and using architecture design rationale by researchers and practitioners (Bass et al., 2003; Bosch, 2004; Curtis et al., 1988), there is a lack of appropriate support mechanisms and guidelines on what are the essential elements of design rationale, and how to document and reason with

design rationale in making architecture design decisions. The recently adopted IEEE standards (1471-2000) for describing architecture (IEEE, 2000) and the architecture documentation methods like Views and Beyond (V&B) (Clements et al., 2002) raise the awareness of and provide some guidance on documenting design rationale; however, for the reasons discussed in Section 2, each has its limitations.

This article describes our initial investigations into the use and documentation of design rationale. In the long term, we aim to develop a conceptual framework and associated tools to facilitate the capture and use of design rationale. We believe that understanding the current industry practice of design rationale is one of the most important steps towards that goal. However, there is little empirical research that studies what practitioners think about design rationale, how they document and reason with design rationale, and what factors prevent them from documenting design rationale.

\* Corresponding author. Tel.: +61 3 92145198; fax: +61 3 98190823.

E-mail addresses: [atang@ict.swin.edu.au](mailto:atang@ict.swin.edu.au) (A. Tang), [malibaba@nicta.com.au](mailto:malibaba@nicta.com.au) (M.A. Babar), [ian.gorton@nicta.com.au](mailto:ian.gorton@nicta.com.au) (I. Gorton), [jhan@ict.swin.edu.au](mailto:jhan@ict.swin.edu.au) (J. Han).

The need to improve the capture and the use of design rationale in system design and maintenance has been reported by several researchers (Bosch, 2004; Tyree and Akerman, 2005; Burge, 2005). They allude to a perception that architects generally do not realize the critical role of explicitly documenting the contextual knowledge about their design decisions. Lack of empirical evidence makes it difficult to support or refute these claims. Hence we set out to gather evidence from those who design architectures on a regular basis, in order to examine the attitudes of practitioners who have the most impact on the immediate and future use of design rationale approaches.

This article reports the findings of a survey of practitioners who have had experience in architecture design in the Asia Pacific region. The findings of this survey shed light on how design rationale are used, documented, and perceived by designers and architects working in this region. The objectives of the study are:

- To understand the architects' perceptions about architecture design rationale and the importance of the different elements of design rationale (such as design constraints, design strengths and weaknesses).
- To determine the frequency of documenting and reasoning with different elements of design rationale, the main reasons for not documenting design rationale, and the common methods, techniques, and tools used to document design rationale.
- To identify the potential challenges and opportunities for improving the use and documentation of design rationale in practice.

During this study, we have encountered several interesting findings which enabled us to identify a set of research questions for further investigations. Since a theory explaining the attitude and behaviour toward the use of design rationale does not exist, this study employs an inductive approach (i.e., using facts to develop general conclusions) as an attempt to move toward such a theory.

The article makes three significant contributions to the Software Architecture (SA) discipline:

- It presents the design and results of the first survey-based empirical study in architecture design rationale practices.
- It provides information about how practitioners think about, reason with, document and use design rationale.
- It identifies the problems and contradictions of current design rationale practices. As a result, we propose a research agenda that aims to explore and enhance current architecture design rationale practices.

We discuss the current approaches to using design rationale in Section 2. We present our research approach in Section 3. Section 4 presents the results of the survey. A discussion of our findings and their limitations are in Sections 5 and 6 respectively. We identify areas for future work and make some concluding remarks in Section 7.

## 2. Background and related work

### 2.1. Design rationale approaches in software engineering

Early work emphasizing the importance of design rationale in software design can be found in Parnas and Clements (1985) and Potts and Burns (1988). Since then, the software engineering community has experimented with several design rationale approaches such as Issue Based Information Systems (IBIS) (Kunz and Rittel, 1970), Questions, Options, and Criteria (QOC) (Maclean et al., 1996), Procedural Hierarchy of Issues (PHI) (McCall, 1987), and Design Rationale Language (DRL) (Lee and Lai, 1996). Most of these methods have been adopted or modified to capture the rationale for software design decisions (Potts and Burns, 1988) and requirements specifications (Dutoit and Paech, 2002; Lee, 1991; Ramesh and Dhar, 1992). Other approaches (e.g. Potts, 1999; Potts et al., 1994) combine rationale and scenarios to elicit and refine requirements. While there are claims of several benefits of using these to capture design rationale, it is not clear how much or how far these techniques have been adopted by practitioners.

Design rationale have been considered an important part of software architecture since (Perry and Wolf, 1992) laid the foundation for the evolving community of software architecture. In the following years, researchers have emphasized the need for documenting design rationale to maintain and evolve architectural artifacts and to avoid violating design rules that underpin the original architecture (Bass et al., 2003; Bosch, 2004). The growing recognition of the vital role of documenting and maintaining rationale for architectural decisions has resulted in several efforts to provide guidance for capturing and using design rationale such as the IEEE 1471-2000 standard (IEEE, 2000) and the Views and Beyond (V&B) approach to document software architecture (Clements et al., 2002).

However, both of these are deficient in several ways. For example, the former provides a definition of design rationale without further elaborating on their nature and how they might be captured. The latter method provides a list of design rationales without justifying why they are important and how the information captured is beneficial in different design context. Moreover, it is unclear if the list of design rationales are complete.

Different approaches tend to characterize design rationale with different information. For example, Tyree and Akerman (2005) provides a template that captures certain types of information as design rationale; the V&B (Clements et al., 2002) approach considers some other types of information (e.g. information cross-cutting different views) as design rationale; and the Architecture Rationalization Method (ARM) uses qualitative and quantitative rationale in design reasoning (Tang and Han, 2005). Thus, there is clearly a need for a common vocabulary or standard guidance so that practitioners understand the issues in reasoning with and consistently documenting design rationale.

## 2.2. Generic design rationale

According to the Cambridge dictionary, a rationale is a reason or intention for a particular set of thoughts or actions. When architects and designers make design decisions based on their reasoning, what do they consider as a reason or an intention? Is a requirement or a constraint an intention or reason enough for a design choice? Or is it some generic justification that allows designers to judge that a design choice is better than its alternatives?

The nature of design rationale is different between different design activities. The architecture decision process is different to the detailed design decision process. Bratthall et al. (2000) suggested that architectural level design addresses decisions with system-wide implications. Lassing et al. (2001) suggested that architecture decisions have a large influence on the quality of the resulting system. Eden and Kazman (2003) suggested that what separate architecture design activities from other design activities are: (a) architecture is concerned with the selection of architecture elements, their interactions and the constraints where design is concerned with the modularization and detailed interfaces of design element; (b) architecture is concerned with issues beyond algorithms and data structures of the computations; and (c) architecture focuses on the externally visible properties of the software components. Since architecture design is at a higher level of abstraction and considers an array of different inputs, the complexity of the design reasoning is generally higher than detailed design of a localized software component. Therefore, design rationale for the two levels of design would be quite different. Given the complexity of architecture design, the types of design rationale and the way they influence architecture decisions are not very well understood. Hence we are motivated to investigate into this area.

In this survey, we used nine types of generic design rationales selected from various references to test if and how our respondents perceive and use them. This set of generic rationale characterizes different aspects in which reasons can be portrayed and compared. Their selection is based on the templates or methods proposed by researchers to capture design rationale (Clements et al., 2002; Tyree and Akerman, 2005; Bass et al., 2003; Tang and Han, 2005). A generic design rationale is an abstract grouping of reasons for justifying decisions that are made in the design process. We used common terminologies so that practitioners could relate to them. Since this is an exploratory study, the list of generic design rationales (below) is comprehensive but not exhaustive.

- (1) *Design constraints* are the limitations placed on the design. They can be business or technical in nature (Clements et al., 2002).
- (2) *Design assumptions* are used to describe what are otherwise unknown factors that affect the design (Tyree and Akerman, 2005).

- (3) *Weakness* of a design describes what the design cannot achieve, which may be functional or technical in nature (Bass et al., 2003).
- (4) *Benefit* of a design describes what benefits the design can deliver to satisfy the technical or functional requirements (Tang and Han, 2005).
- (5) *Cost* of a design describes the explicit and implicit costs to the system and the business (Tang and Han, 2005).
- (6) *Complexity* of a design is a relative measure of the complexity of the design in terms of implementation and maintenance (Tang and Han, 2005).
- (7) *Certainty of design*, i.e. the design would work, is a measurement of risk that the design would meet its requirements (Tang and Han, 2005).
- (8) *Certainty of implementation*, i.e. the design is implementable, is a measurement of risk that the development team has the skill and the resources, in terms of schedule and cost, to implement the design (Tang and Han, 2005).
- (9) *Tradeoffs* between alternative designs is a mechanism to weigh and compare alternatives given each alternative design has its supporting design rationale and priorities (Bass et al., 2003).

Although the above list of design rationales have been suggested by various researchers and common sense tells us that they are useful, no empirical studies have been carried out to prove that they are actually used in the software industry. In this survey, we asked our respondents to rank these rationales according to their usefulness and how often they are being used and documented. The results are reported in Section 4.

## 3. Research approach

### 3.1. Research method

Considering the objectives of our research and available resources, we decided to use the survey research method to understand architects' perception of, and current practices in architecture design rationale. A survey research method is considered suitable for gathering self-reported quantitative and qualitative data from a large number of respondents (Kitchenham and Pflieger, 2001–2002). Our survey design was a cross-sectional, case control study. Survey research can use one or a combination of several data gathering techniques such as interviews, self-administered questionnaires and others (Lethbridge, 2005). We decided to use a questionnaire as a data collection instrument because we wanted to obtain the information from a relatively large number of practitioners, many of whom we would not be able to contact personally.

### 3.2. Survey instrument construction

Having reviewed the published literature on design rationale, we developed a survey instrument consisting of 30

questions on the understanding and practice of design rationale and 10 questions on demographics (e.g. age, experience, gender, education and others) of the respondents. Some of the demographic questions were designed for screening the respondents and identifying the datasets to be excluded from the final analysis. The questions were placed in different sections, namely design rationale importance, design rationale documentation, design rationale usage, and demographic information. Questions on demographics were put in the last section as it is considered good practice (Kitchenham and Pfleeger, 2001–2002). In addition, a coversheet explaining the purpose of the study and defining various terms was attached to the questionnaire.

### 3.3. Instrument evaluation

The questionnaire underwent a rigorous review process for the format of the questions, suitability of the scales, understandability of the wording, number of questions, and length of time required to complete by experienced researchers and practitioners in software architecture domain. We ran a formal pilot study to further test and refine the survey instrument. The pilot study was conducted with eight people who were considered strongly representative of the potential participants of our survey research (i.e. practitioners with more than 3 years software design experience). Data from the pilot study was not included in the analysis of the main survey. The feedback from the pilot study helped us refine the questionnaire, which was submitted for ethical committee approval.

### 3.4. Instrument deployment

We decided to use an online web-based survey, as this is usually less expensive and more efficient in data collection (Simsek and Veiga, 2001). In order to implement the survey, we used an online web-based tool Surveyor (ObjectPlanet Inc., 2002). Participants accessed the survey through a URL.

### 3.5. Target population

The inclusion criteria was a software engineer with three or more years of experience in software development and who has worked or is working in a design or architect role. We did not have a formal justification for the amount of experience required of valid respondents. We based this on our extensive experience in designing architectures for large systems that made us believe that people with three or more years of experience in software development would be able to give reliable answers to the type of questions we had.

### 3.6. Sampling technique

The study needed responses from likely time-constrained software engineers, who we expected were less likely to respond to an invitation from unfamiliar sources. This

made it hard to apply random sampling. Consequently, we decided to use non-probabilistic sampling techniques, availability sampling and snowball sampling. Availability sampling operates by seeking responses from those people who meet the inclusion criteria and are available and willing to participate in the research. Snowballing requires asking the participants of the study to nominate other people who would be willing to participate. The major drawback of non-probabilistic sampling techniques is that the results cannot be considered statistically generalizable to the target population (Kitchenham and Pfleeger, 2001–2002), in this case software designers/architects. However, considering the exploratory nature of our research, we believe that our sampling techniques were reasonable.

### 3.7. Invitation mechanics

We used two means of contacting potential respondents: personalized contact and professional referrals. The invitation letters were sent to a pool of software designers/architects drawn from the industry contacts of the four investigators and past and current students of the post-graduate information technology courses offered by the Swinburne University of Technology and the University of New South Wales. We requested the invitees to forward the invitation to others who were eligible for participation and provide us the contact for the forwarded invitation.

### 3.8. Data validation

For access control and data validation purposes, the survey URL was sent via email. Moreover, the responses gathered in the survey provided another mechanism of checking the validity of the respondents as genuine software engineering practitioners. For example, only one of the 81 respondents did not provide a job title and all other respondents had relevant job titles. A large number of respondents (55%) provided quite insightful and detailed comments to several open-ended optional questions.

## 4. Survey findings

The survey questionnaire was divided into seven main parts. The perception of the importance of design rationale, the use of design rationale and the documentation of design rationale are discussed and analyzed in this article together with the profile of the respondents. Architecture evaluation in organizations, architecture enhancements and risk undertakings in architecture design are the other three parts which will be reported separately. Readers who are interested in the statistics and the questionnaire are referred to Tang et al. (2005).

### 4.1. Demographic data

We directly sent survey invitations to 171 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 23% response rate. Anonymity and lack of resources did not allow us to contact non-respondents. Out of the total responses, we decided to exclude 46 responses from the analysis as they were incomplete or the respondents did not meet the work experience criteria (minimum 3 years software development experience).

In summary, 80.2% of our respondents were male and 19.8% are female. The Office of Technology Policy uses Census figures to estimate that women represent 26.9% of computer systems analysts in the United States (Meares and Sargent, 1999). An Australian report found that 24% of undergraduate students in Information Technology in 2003 are women (Carrington and Pratt, 2003). Although there is no statistics for the gender distribution of architects and designers for the Asia Pacific region, the reference information suggests that the ratio of gender distribution in this survey is reasonable. 67.9% of the respondents live in Australasia, 28.4% reside in Asia and 3.7% did not specify the region of their residence.

The respondents' experience in the information technology industry varies between 4 years and 37 years with an average of 17.12 years and a median of 15. The respondents have worked as a designer or architect for 9.75 years on average and a median of 8 years. These results show that the average respondents are experienced in design and architecture.

The average length of time an architect work with an organization (current or previous) is 7.65 years (a median of 6 years). This profile indicates that the respondents have had relatively stable jobs and they are likely to be familiar with the development standards within their organizations.

The average number of co-workers on the current (or last) project is 25 people (a median of 15 people). Although we cannot directly identify the size of the projects our respondents are involved in by the amount of software development such as source line of codes or development effort such as man months, this profile gives an indication of the relative size of the projects. 85.2% of the respondents have received an IT related tertiary qualification. This question is aimed to clarify the level of IT training received by the respondents.

The demographic data gives us confidence that our respondents are practitioners who are experienced in software architecture and design. Despite not being able to apply systematic random sampling because of the reasons described in Section 3, the results are representative of architects and designers with similar characteristics.

#### 4.2. Job nature of architects/designers

In the survey, we asked respondents to tell us the primary tasks they perform as a designer/architect. A primary task is a task in which they spend at least 10% of their time on. The objective is to find out the scope of their job role. A summary of the percentages of respondents who perform those primary tasks are listed below:

- overall system design (86.4%),
- requirements or tender analysis (81.5%),
- non-functional requirements design (64.2%),
- software design and specification (58%),
- project management tasks (50.6%),
- IT planning and proposal preparation (49.4%),
- data modelling (44.4%),
- implementation design (42%) ,
- program design and specification (35.8%),
- test planning and design (29.6%),
- training (19.8%).

Our typical respondent's main efforts are spent in the early project phases including requirements and tender analysis, overall design, high level design, non-functional design and software design. Most of them also have management responsibilities such as project management and IT planning. To a lesser extent, they perform detailed design and implementation activities.

We asked our respondents if their projects or organizations recognize software architect roles. 43.2% of respondents said software architects are formally recognized across all projects in the organization, while 48.1% said only some projects in the organization recognized the use of architects. This may be due to the organization structure or the nature of the projects. As indicated in the findings, respondents told us that projects under circumstances such as new projects (23.5%), mission critical projects (25.8%), high risk projects (27.2%) and high cost projects (18.5%), would involve software architects.

#### 4.3. Designer's perception of the importance of design rationale

As there is little empirical evidence on how important design rationale are considered by designers, we posed a number of questions to this end. Respondents were asked to indicate how often they reason about their design choices and whether they think that design rationale are important to justify their design choices.

The responses to those questions revealed (Tables 1 and 2) that the majority of designers frequently apply reasoning to

Table 1  
Frequency of reasoning about design choices

	Never				Always
	1	2	3	4	5
No. of respondents	0	1	8	34	38
Percentages	0	1.2	9.9	42	46.9

Table 2  
Importance of design rationale in design justification

	Not important			Very important	
	1	2	3	4	5
No. of respondents	0	1	11	30	39
Percentages	0	1.2	13.6	37	48.1

Table 3  
Frequency of considering alternative designs

	Never				Always
	1	2	3	4	5
No. of respondents	0	1	15	31	34
Percentages	0	1.2	18.5	38.3	42

Table 4  
Importance of each generic rationale

	Not important				Very important
	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)
Design constraints	0.0	1.2	11.1	38.3	49.4
Design assumptions	3.7	7.4	14.8	44.4	29.6
Weakness	2.5	7.4	28.4	43.2	18.5
Costs	0.0	7.4	14.8	43.2	34.6
Benefits	1.2	1.2	7.4	54.3	35.8
Complexity	0.0	2.5	25.9	46.9	24.7
Certainty of design	0.0	3.7	11.1	29.6	55.6
Certainty of implementation	2.5	4.9	16.1	32.1	44.4
Tradeoffs	0.0	4.9	30.9	44.4	19.8

justify their architectural choices and they also consider that design rationale are important to justify their design choices.

We also asked the respondents about the frequency of considering alternative architecture designs (explanation for alternative architecture designs was provided) during their design process, as this is another indicator of the awareness of reasoning about design choices and the rigour that needs to be employed during this process. The responses to this question are provided in Table 3. The result indicates that the majority of respondents compare between alternative designs before selecting a particular architectural design among available alternatives.

We asked the respondents to rank the importance of each of the nine generic design rationales listed in the survey. This ranking reflects the perception of respondents towards how useful a given design rationale is in design. Since decision making is something our respondents do on a regular basis, their perception of design rationale's importance should reflect the reasoning process that is usually done intuitively. Table 4 presents the responses to this question. The majority of respondents considered that all nine design rationales are important.

The responses for all rationales are skewed towards the very important end. *Benefits of design*, *design constraints* and *certainty of design* receive the highest support with combined percentages (levels 4 and 5) of 90.12%, 87.65% and 85.19%, respectively. All other rationales are also considered important with the majority of respondents selecting level 4 or 5. This shows that most designers perceived that these rationales are important in reasoning about design decisions.

Apart from the above-mentioned nine generic rationales, we also asked the respondents to add other ratio-

nales that they use for making architectural design choices. A significant number of the respondents (28), mentioned additional types of factors that influence their design choices. We have classified those factors into three broad categories. These are:

#### Business Goals Oriented

1. Enterprise strategies, technical directions and organizational standards.
2. Management preferences and acceptance.
3. Adherence to industry standards.
4. Vendors relationship.

#### Requirements Oriented (functional/non-functional)

5. Fulfill functional and non-functional requirements.
6. Satisfy client business motivations.
7. Buy vs. build decisions.
8. Maintenance and expected life-cycle of products.

#### Constraints and Concerns

9. Viability of solutions.
10. Consideration of existing architecture constraints.
11. Current IT architecture and capabilities.
12. Compatibility with existing systems.
13. Prior use of the design and how successful.
14. Availability of technology and tools.
15. Prototype and staged delivery.
16. Time to market.
17. Available time.
18. Risk.

These rationales show a variety of factors that influence the design decision making process. They also provide the context to enable architects and designers to trade-off conflicting goals by argumentation.

#### 4.4. Using design rationale

Another important area of the survey was how frequently design rationale are used. An objective of the study is to discover whether respondents' perceived importance of design rationale (i.e. what they think) and their behaviour (i.e. what they do) are consistent. Therefore, the same set of design rationale we presented and discussed in the previous sections were used to query our respondents. In this section, we present the results of a multi-item question on how often they use the generic rationales to reason about architectural decisions. Most respondents say that they frequently or always use the nine generic design rationales listed in the questionnaire. Table 5 summarizes the frequency of how respondents think they use the different types of rationales.

The results show that the *design constraint* rationale is used most frequently. The reason for the high usage of this could be that designers are usually expected to explore the solution space within certain business and technical constraints. These constraints are consequently prominent in their minds and must be taken into account from the beginning of a project.

Table 5  
Design rationale frequency of use

	Never		Always		
	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)
Design constraints	0.0	0.0	12.3	42.0	45.7
Design assumptions	2.5	2.5	30.9	33.3	30.8
Weakness	1.2	8.6	34.6	37.0	18.6
Costs	1.2	9.9	19.8	38.3	30.8
Benefits	1.2	1.2	12.3	49.4	35.9
Complexity	0.0	2.5	27.2	34.6	35.7
Certainty of design	2.5	1.2	11.1	32.1	53.1
Certainty of implementation	3.7	3.7	16.0	33.3	43.3
Tradeoffs	0.0	6.2	29.6	42.0	22.2

Other more frequently used rationales are *benefits of design*, *certainty of design* and *certainty of implementation*. The combined usage frequencies (level 4 and 5) for these rationales are 85.3%, 85.2% and 76.6%, respectively. We suspect that designers frequently use these types of rationales as they have to make a business case for their architectural choices to the management. They also have to justify their design choices using technical arguments to architecture reviewers and technical stakeholders such as programmers, implementers and maintainers. So they use more often those rationales that can help them to justify their architectural decisions.

On the other hand, respondents are less likely to use those rationales that can highlight the weaknesses of their design decisions. So the combined usage frequencies (level 4 and 5) reported by respondents are relatively lower: *design weakness* (55.6%), *costs* (69.1%) and *complexity* (70.3%). This tendency of designers to pay relatively less attention to the weaknesses of their design decisions can also be explained by the Lassing et al.'s warning against gathering scenarios to evaluate an architecture by the designers themselves, as it is highly likely they would come up with the scenarios that have already been addressed by the proposed architecture (Lassing et al., 2002). Thus, we hypothesize that designers unknowingly look for those positive rationales to support the design decisions and pay less attention to those negative rationales.

#### 4.5. Documenting design rationale

Several arguments have been made about the importance of documenting key architecture decisions along with the contextual information (Parnas and Clements, 1985; Tyree and Akerman, 2005). It is important that design rationale are documented to a sufficient extent in order to support the subsequent implementation and maintenance of systems. With regards to design rationale documentation attitude and practice, we paid special attention to the frequency of documenting discarded design decisions, frequency of documenting each of the generic rationales, the reasons for not documenting design decisions (barriers to design rationale documentation), and methods and tools

Table 6  
Frequency of documenting discarded decisions

	Never				Always
	1	2	3	4	5
No. of respondents	11	18	17	19	16
Percentages	13.5	22.2	21	23.5	19.8

Table 7  
Frequency of documenting generic design rationale

	Never		Always		
	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)
Design constraints	1.2	2.5	13.6	19.7	63.0
Design assumptions	3.7	3.7	13.6	25.9	53.1
Weakness	3.7	23.5	37.0	14.8	21.0
Costs	7.4	16.0	30.9	21.0	24.7
Benefits	2.5	9.9	18.5	32.1	37.0
Complexity	3.7	9.9	35.8	30.9	19.7
Certainty of design	18.5	14.8	19.8	24.7	22.2
Certainty of implementation	18.5	17.3	24.7	22.2	17.3
Tradeoffs	6.2	18.5	25.9	32.1	17.3

used for documenting design rationale. Table 6 presents the breakdown of the responses to the question on documenting discarded design decision.

About 44% of the respondents (level 4 and 5) document discarded decision very often. About 36% of the respondents (level 1 and 2) do not document discarded decisions. This is likely because designers are under pressure to produce design specifications on schedule. At this stage, we are not aware of any software development or project management methodology that mandate the documentation of discarded decisions or methodically schedule time for such activities to take place. However, documenting the discarded decisions can help newcomers to the project understand the reasons for discarding design alternatives and expedite such understanding during the maintenance phase of the project.

Respondents were also asked to indicate the overall frequency of documenting design rationale. 62.9% of the respondents replied that they completely document design rationale, which is an encouraging finding considering the common perception of design rationale not being widely documented.

We also investigated the frequency of documenting each of the generic rationales. Table 7 summarizes the frequency of documentation for each of the nine generic design rationales used in this research. The results show that *design constraints* and *design assumptions* are documented very frequently but the level of documentation is relatively lower for other types of rationale. 27.2% of the respondents replied that they never or seldom document *design weakness*. Similarly, 33.3% of respondents said they never or seldom document *certainty of design*. 35.8% of them said they never or seldom document *certainty of implementation*. These findings appear to agree with our previous asser-

tion that negative rationales receive relatively less attention.

Based on these results, it appears that design rationale are commonly documented by software designers and architects. However, it also appears that the reasons about why a design alternative is chosen and why it is better than other alternatives are usually not documented. We do not have any theoretical grounds for explaining this phenomenon.

While the level of documentation is relatively high, the survey results give us no insight as to whether the rationales are sufficiently documented so that other designers can understand the architecture design without additional assistance. This raises two issues worthy of further investigation, namely:

- identify the rationales documented by architects and evaluate their effectiveness in explaining the design;
- identify how the documented rationales are used in the development life-cycle.

#### 4.5.1. Barriers to documenting design rationale

We were also interested in identifying and understanding the reasons for not documenting design rationale. We believe that it is important to identify those factors that undermine efforts in documenting and maintaining design rationale. The respondents were given a list of reasons that are thought to be common causes of non-documentation in software engineering such as perceived usefulness, project budget and lack of time. The respondents also had a text box to provide other reasons.

Table 8 summarizes the responses to the reasons for not documenting design rationale. These results reveal that lack of time/budget (60.5%) is considered the most common cause of not documenting design rationale. There is also a lack of appropriate standards and tools to support the documentation process. Only 4.9% of the respondents were not aware of the need to document design rationale, while 9.9% of the respondents said that documenting design rationale is not useful. A few respondents also provided several other reasons for not documenting design rationale. These reasons are:

- Lack of formal review process.
- Not required for non-complex solutions.
- Afraid of getting into a long cycle of design review.
- Not required for low impact solution.

Table 8  
Reasons for not documenting design rationale

Topic of questions	Percent of respondents	Number of respondents
No standards	42	34
Not aware of	4.9	4
Not useful	9.9	8
No time/budget	60.5	49
No suitable tool	29.6	24

- The dynamic nature of technology and solutions make it useless to document design rationale.
- It is not required for high level decision making.

In summary, the reasons for not documenting design rationale can be classified into these groups: (a) the lack of standards and processes to guide why, how, what and when design rationale should be documented; (b) the time and budget constraints of projects; (c) the question of whether the cost and benefit of rationale documentation can be justified. These reasons are analogous to those concerning requirements traceability documentation in immature software development organizations (Ramesh and Jarke, 2001). Since the sample population is not specific to an industry or capability maturity level, the results may indeed reflect the general architecture design practice.

#### 4.5.2. Methods and tools for documenting design rationale

An important part of any task in the software development life-cycle is the availability of process support and suitable tools to enhance productivity. It is important to identify what type of support is available to designers to improve design rationale practices. Hence the survey included a question on the methods and tools used for documenting design rationale. Twenty respondents provided comments to this question. We list the methods and tools used by the respondents to document design rationale below:

- Apply organization standards and templates to document using Word/Visio/Excel/Powerpoint.
- UML tools.
- IBM GS Methodology.
- Document architecture decisions using formal method and notation.
- Internally developed tools.
- QMS Design Template document.
- Requirements Traceability Matrix.
- Architecture tool CORE.

Our respondents used proprietary tools, proprietary templates, the Microsoft Office suite or UML design tools to document design rationale. As we suggested earlier, there is little awareness about the standards like IEEE 1417-2000 and methodologies like V&B. Design rationale tools like gIBIS (Conklin and Begeman, 1988) are not used. Although these results are anecdotal evidence, they point to the lack of industry standards as well as proper tools to capture, maintain and trace design rationale during the development life-cycle.

#### 4.6. Comparing usage and documentation of design rationale

Given that design rationale are recognized by our respondents as important, it is revealing to compare the survey results concerning importance, use and documentation of each of the nine generic rationales. Table 9 presents

Table 9  
Design rationale usage

	Level of importance (%)	Frequency of use (%)	Frequency to document (%)
Benefits of design	90.1	85.3	69.1
Design constraints	87.6	87.6	82.7
Certainty that design would work	85.2	85.2	46.9
Cost of design	77.7	69.1	45.7
Certainty that design is implementable	76.5	76.5	39.5
Design assumptions	74.0	64.1	79.0
Complexity of design	71.6	70.3	50.6
Tradeoffs between alternatives	64.2	64.2	49.4
Weakness	61.7	55.6	35.8

the combined results from the last three sections. The scale is condensed by combining level 4 and level 5 (see the scale in the previous sections to interpret the results).

We used Spearman's Rank Order Correlation ( $\rho$ ) to test the correlations between the *Level of Importance* and the *Frequency of Use* for the nine generic design rationale. This revealed that they are all correlated with  $r$  values all above 0.5 with the exception of *design complexity*, and all of them tested significant with  $p < 0.01$ . This indicates that there is a strong relationship between what respondents believe and what they practice. We also observe that across most design rationale, the usage frequency is less than the perception of importance, and the documentation frequency is less than the usage frequency. This can be considered a strong indicator that our respondents are convinced of the importance of design rationale and use them more frequently than they document them. Lack of documentation may be caused by the reasons put forwarded by the respondents (Section 4.5.1). This may provide an explanation for the claims of design knowledge vaporization (Bosch, 2004; Tyree and Akerman, 2005).

#### 4.7. Necessity for documenting design rationale

In the survey, we asked the respondents about the use of design rationale in relation to carrying out impact analysis during system maintenance. To this end, we asked how often they revisit design documentation and specifications to help them understand the system before performing enhancements. Table 10 presents their responses, which indicate that the majority of the respondents consult design

Table 10  
Frequency of revisiting design documentation before making changes

	Never				Always
	1	2	3	4	5
No. of respondents	0	8	9	20	30
Percentages	0.0	11.9	13.4	29.9	44.8

Table 11  
Tendency of forgetting the reasons for justifying design decisions

	Never				Always
	1	2	3	4	5
No. of respondents	2	15	22	22	5
Percentages	3.0	22.7	33.3	33.3	7.6

Table 12  
Do not understand design without design rationale if not original designer

	Strongly disagree			Strongly agree	
	1	2	3	4	5
No. of respondents	1	3	9	23	29
Percentages	1.5	4.6	13.8	35.4	44.6

documentation frequently while performing maintenance or enhancement tasks. These results show that if design rationale are sufficiently documented and provided to the software maintainers, they are more likely to use the design rationale. These results are also consistent with the findings reported in Seaman (2002), 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 maintainer can effectively perform modification tasks by consulting that knowledge (Hamada and Adachi, 1993).

We asked our respondents how often they think they forget the reasons underpinning their design decision after a period of time. Table 11 shows the results to this question. The responses show that 74% of respondents forget the reasons concerning design decisions. This finding should provide a strong reason for regularly documenting and maintaining design rationale to support architecture maintenance and evolution. As mentioned earlier, some of the reasons could be reconstructed through inspecting available design specifications, but some reasons will inevitably be lost if the design is complex and the system was developed some time ago.

Often the architect who is responsible for maintenance is not the same person who originally designed the system. In such circumstances, we asked respondents whether they know why existing designs were created without documented design rationale. The results are shown in Table 12. Most respondents (80%) either agree or strongly agree with the statement that without design rationale, they may not understand why certain decisions are made in the design.

Given the results, it can be concluded that design rationale have a role to play in supporting maintenance. Architects and designers often refer to design documentation for impact analysis. This is because they either forget the reasons behind the design or they are not the original designers. Without documented design rationale, comprehending the system design is likely problematic and inefficient.

#### 4.8. Risk as a design rationale

When an architecture design is committed, designers may not necessarily have full knowledge of whether the implementation could fully satisfy the requirements. For instance, the performance of a system cannot be easily determined. Additionally, it is difficult to estimate how the architecture would cope with future changing technologies and requirements, so architects would make assumptions and assessments based to support their decisions (Lago and van Vliet, 2005). Examples are “the likelihood of this technology not being supported in the next 5 years is low” or “the performance should be okay because we do not expect any database contention”. In such cases, the architecture design cannot be fully tested until much later in implementation or even after it has been deployed. This aspect of the architecture design is quite different to designing a software module to satisfy a set of functional requirements. Detailed design which deals with localized issues can be unit tested more easily.

Since there are uncertainties in architecture design, architects make assumptions and risk assessments either implicitly or explicitly based on their beliefs. As such, ATAM suggested that risks and non-risks should be documented (Bass et al., 2003). Two types of risk assessments are suggested for architecture design reasoning (Tang and Han, 2005): the architecture design can successfully deliver the required benefits; and the development team is capable of delivering the design.

This survey asked the respondents to indicate if they do risk assessments and if so, what is an acceptable risk level. 24.7% of the respondents said they always explicitly quantify risks. 35.8% of the respondents said they sometimes explicitly quantify risks. 42% and 49.4% of the respondents indicated that they are *quite certain* and *most certain*, respectively, that their design can deliver the results and their team can implement the design. These results indicated that risk assessment is an important part of architecture design reasoning.

Given that risk assessment is frequently used in architecture design, the respondents were asked to identify an acceptable level of risk. 23.5% said that a 30% risk level was acceptable, 18.5% said that a 40% risk level was acceptable. However, 14.8% of respondents said that a 70% risk level was acceptable and 13.6% said that a 80% risk level was acceptable. There was no consensus on what is an acceptable level.

## 5. Discussion of findings

Based on the survey, there is evidence to support that design rationale are an important part of the design documentation, and practitioners believe that design rationale should be documented. There is also a general perception that methodology and tool support for design rationale is lacking and there are barriers to design rationale documentation. These findings lead to a number of areas that require further investigation.

#### 5.1. Different forms of design rationale

In addition to the nine generic design rationales reported in the survey, respondents had indicated that they consider other types of design rationale during design. Examples of the other design rationales they consider are *management preferences*, *vendor relationship* and *maintenance and expected life-cycle* (see Section 4.3 for a complete list). This list of rationales appeared to be different in nature to the ones that were used in the survey.

Except for *constraints* and *assumptions*, seven of the nine generic design rationales we listed can be quantified in some ways. For example, costs (i.e. specific values) of design alternatives can be used to select one alternative over others.

Along with *constraints* and *assumptions*, the design rationales suggested by the respondents are contextual in nature. These rationales provide a context to justify and reason about their design decisions. For example, the flexibility and user-friendliness requirements for the design of a system’s user interface may present a conflict, i.e., a design cannot satisfy both simultaneously. As such, the architect might seek additional information and agreement on priorities of these requirements from stakeholders in order to make a compromised decision, e.g. first user friendliness and later flexibility. The argumentation which is used to arrive at a decision would rely on a clear understanding of the underlying reasons (i.e. the context) and the priorities of each of the requirements. This type of decision rationalization relies on the context of the influencing factors such as requirements and assumptions and it uses qualitative argumentation in the decision reasoning.

Quantitative methods such as Asundi et al. (2001) and Al-Naeem et al. (2005) and argumentation methods such as Lee and Lai (1996) and Maclean et al. (1996) have provided supporting technologies to capture design rationale in different ways. However, they have not addressed the fundamental issue of how design occurs and what the intrinsic reasoning process is. As such, further research to investigate how practitioners make use of various types of design rationales and the roles these rationales play in the decision making process would provide a foundation for design reasoning and design rationale capture.

#### 5.2. The role of an architect

The survey results reported in Section 4.2 indicate that architects work on a variety of tasks such as requirements analysis, tender analysis, architecture design and software design. They also have management responsibilities. Program design and test planning is a much smaller part of their job. We interpret this result as an indication of the activities involved in the architecture design. As well as designing software, architecture design would also deal with: (a) high level issues such as planning and management; (b) constraints and assumptions arising from the information technology environment and the organiza-

tional environment; and (c) the complexity of the decision making arising from the competing technical and non-technical aspects in a project. Of the architects surveyed, 48.1% said that not all projects require an architect's involvement. Projects that are either new, mission critical, high risk, high cost, complex or have high impacts to other systems or organizations require architects. The results clearly indicate that architects are involved in a broad array of tasks across the whole project scope. Given that the architecture decision making process is complex and there are many technical and non-technical considerations that influence architecture decision making, it is important to identify the design rationale or considerations that influence architecture decision making, assess how they inter-play in the decision process; and generalize their use into architecture design patterns (Ali-Babar et al., 2006).

### 5.3. Designers' attitude

Respondents frequently use design rationale to justify design choices. When we examine the list of design rationales they use, it appears that those design rationales that positively justify the design receive more attention than those negative rationales that explain why the design may have issues. That leads us to suspect that there might be a tendency to present *good news* rather than *bad news* during the design process. An analogous finding (Keil et al., 2004) may give us some insights to this behaviour. In many industry scenarios that we have encountered, some architects have a tendency to promote a design based on the benefits of new technologies. However they often do not explain the potential negative impacts of the new approach. Establishing if such a bias is commonly exhibited in architecture would be useful, because awareness of this phenomenon would help architects to be more objective in the assessment and selection of design choices.

### 5.4. Necessity for design rationale documentation

The survey found that there is a strong justification to document design rationale due to the tendency of the architects' forgetting their own design or the need for architects to modify systems designed by other architects. However, not all systems require comprehensive design rationale documentation because certain design rationale can be reconstructed for non-complex systems. The extent to which design rationale are documented also depends on the relative benefits it might deliver.

### 5.5. Risk assessment in architecture design reasoning

Two major differences exist between architecture design and detailed software design. Firstly, architecture design has a global impact, in terms of modifiability and correctness, to the system whilst the software design of a local module has a lower impact. Secondly, there are more certainty and testability in designing a software module than

an architecture. The survey showed that architects often carry out risk assessments in architecture design. Over half of the respondents explicitly quantify risks. However, when they were asked what risk level would be acceptable, there was no consensus. This implies that there is no common understanding on how to measure risks. Is an 80% uncertainly in the architecture design acceptable? 13.6% of architects seemed to think so. Since architecture cannot be fully tested at the early stage of the development cycle, it is better to have a lower risk architecture design which is more likely to succeed in delivering the benefits and being implemented. However, the risk assessment process in architecture design is not well understood and so there is much room for improvements in this area.

### 5.6. Design rationale methodology support

Some of the reasons for not documenting design rationale are budget constraints and lack of methodology. Given that most respondents consider design rationale important and their documentation useful, there needs to be guidelines under which the use and documentation of design rationale will provide greater benefits than the costs involved. This means that the need for design rationale documentation should be context dependent. For instance, a non-complex system may require little design rationale documentation since it can be reconstructed easily (Curtis et al., 1988). Our literature review shows that there is no comprehensive methodology to guide how we should use rationale-based techniques to design systems. Therefore, further studies of the use and documentation of design rationale to provide a methodology would be most beneficial.

### 5.7. Design rationale tool support

At present, tool support for design rationale capture and retrieval is inadequate. The various tools that respondents reported using, including word processors and UML-based tools, do not have traceability features to support systematic design rationale description and retrieval. Therefore, it is important to understand how best to capture, represent and use design rationale and then develop such tools to provide a design rationale enabled development environment.

In summary, the survey has gathered invaluable information about how designers use design rationale. It has confirmed that the use of design rationale in the architecture design process continues to be challenging for practitioners.

## 6. Limitations

Our study has several shortcomings. Like most surveys in software engineering, our study faced reliability and validity threats. For example, contrary to the common perception that there is a lack of documentation of

architecture design rationale, our finding shows that a large number of respondents document their design rationale. Such high level of rationale documentation could be biased because there might be a tendency for the respondents to respond according to what they think is a good idea instead of what they actually practise. Following the guidelines provided in Kitchenham and Pfleeger (2001–2002), 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, 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.

Most software engineering surveys also suffer from the problem of non-response error when members of the targeted sample do not respond to a survey. In our case it is possible that those who do not believe in the value of documenting architecture design rationale may have opted not to respond to the survey, which would have biased the results. Unfortunately, given the anonymous nature of the responses, we are unable to identify those participants who did not respond, hence, it is difficult to assess the representativeness of the sample.

Another limitation of the study is the non-existence of a proven theory of designers/architects' attitude towards documenting design rationale to guide our research. That is why we considered our research in understanding the practitioners' attitude and practices of design rationale documentation and usage as exploratory. Through the survey, we aimed at gathering facts in the hopes of drawing some general conclusions to help us and the software architecture community to identify research directions, and to facilitate the development of a theory on design rationale documentation and usage.

Geographical location of the respondents, mainly the Asia Pacific region, is another limitation as the findings cannot be generalized globally. Though, Australian and Asian software engineering practitioner are internationally renowned for technical competency and high standards, the only way to test the findings for regional bias is to replicate the study in geographical regions, especially, US and Europe. We are happy to make our survey instrument available for such comparative study.

## 7. Future work and conclusion

Our long-term research objective is to improve the design reasoning process for software architects. We are approaching this by firstly understanding the key elements of the process, and then attempting to develop appropriate support mechanisms and tools to facilitate the design process. In this study, we have gained important insights into the issues of design rationale use and documentation in the software industry. We found that practitioners view design rationale as important but there is a lack of methodology and tool support.

To achieve these aims, we need to identify the technical and socio-technical factors that influence those design decisions that have architectural implications. Some of the significant issues that we plan to pursue in our continued research in architecture design rationale are:

- What are the different types of design rationale and how can they be explicitly used to reason and objectively measure the relative merits of a design decision to improve the decision making process?
- Whether there is a common tendency, intentionally or unintentionally, to focus on positive aspects of design decisions and ignore the negative aspects.
- What are the design or system circumstances that influence the use and documentation of design rationale?
- Under what circumstances would the use and documentation of design rationale provide a positive return on investment? Answers to this question will help software practitioners make decisions about the level of detail and circumstance under which to document design rationale.

We plan to design and execute a large scale field study consisting of multiple case studies, as described by Yin (1998), and successfully demonstrated by Curtis et al. (1988) into the software design process. Some of the techniques that we plan to use to study practitioners' attitudes towards design rationale use and documentation are in-depth interviews and examination of design specifications. We expect these experimental techniques will enable us to discover the answers to the questions above. The results will allow us to develop a design rationale methodology and associated tools to enhance the future use and documentation of design rationale during software design and maintenance.

## References

- Al-Naeem, T., Gorton, I., Babar, M.A., Rabhi, F.A., Benatallah, B., 2005. A quality-driven systematic approach for architecting distributed software applications. In: ICSE, pp. 244–253.
- Ali-Babar, M., Gorton, I., Kitchenham, B., 2006. A framework for supporting architecture knowledge and rationale management. In: Rationale Management in Software Engineering. Springer, pp. 237–254.
- Asundi, J., Kazman, R., Klein, M., 2001. Using Economic Considerations to Choose Amongst Architecture Design Alternatives. Tech. Rep. CMU/SEI-2001-TR-035, ESC-TR-2001-035, Carnegie Mellon University.
- Bass, L., Clements, P., Kazman, R., 2003. Software Architecture in Practice. Addison-Wesley, Boston.
- Bosch, J., 2004. Software architecture: the next step. In: Proceedings of the 1st European Workshop on Software Architecture (EWSA), St. Andrews, UK, pp. 194–199.
- Bratthall, L., Johansson, E., Regnell, B., 2000. Is a design rationale vital when predicting change impact? a controlled experiment on software architecture evolution. In: Second International Conference on Product Focused Software Process Improvement, pp. 126–139.
- Burge, J., 2005. Software engineering using design RATIONALE. Ph.D. thesis. Worcester Polytechnic Institute.

- Carrington, K., Pratt, A., 2003. How Far Have We Come? Gender Disparities in the Australian Higher Education System.
- Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., Stafford, J., 2002. Documenting Software Architectures: Views and Beyond, first ed. Addison Wesley.
- Conklin, J., Begeman, M., 1988. gIBIS: a hypertext tool for exploratory policy discussion. In: Proceedings of the ACM Conference on Computer-Supported Cooperative Work, pp. 140–152.
- Curtis, B., Krasner, H., Iscoe, N., 1988. A field study of the software design process for large systems. *Communications of ACM* 31 (11), 1268–1287.
- Dutoit, A.H., Paech, B., 2002. Rationale-based use case specification. *Requirements Engineering* 7 (1), 3–19.
- Eden, A., Kazman, R., 2003. Architecture, design, implementation. In: International Conference for Software Engineering, pp. 149–159.
- Gruber, T.R., Russell, D.M., 1991. Design Knowledge and Design Rationale: A Framework for Representing, Capture, and Use. Tech. rep., Knowledge Systems Laboratory, Stanford University, California, USA.
- Hamada, M., Adachi, H., 1993. Recording software design processes for maintaining the software. In: Proceedings of the 17th Computer Software and Applications Conference.
- IEEE, 2000. IEEE Recommended Practice for Architecture Description of Software-Intensive System (IEEE Std 1471-2000).
- Keil, M., Smith, H.J., Pawlowski, S., Jin, L., 2004. ‘Why didn’t somebody tell me?’: climate, information asymmetry, and bad news about troubled projects. *SIGMIS Database* 35 (2), 65–84.
- Kitchenham, B., Pfleeger, S.L., 2001–2002. Principles of Survey Research, Parts 1 to 6. *Software Engineering Notes*.
- Kunz, W., Rittel, H.W.J., 1970. Issues As Elements of Information Systems. Tech. Rep., Institute of Urban and Regional Development, University of California.
- Lago, P., van Vliet, H., 2005. Explicit assumptions enrich architectural models. In: ICSE, pp. 206–214.
- Lassing, N., Rijsenbrij, D., Vliet, H.V., 2001. Viewpoints on Modifiability. *International Journal of Software Engineering and Knowledge Engineering* 11 (4), 453–478.
- Lassing, N.H., Bengtsson, P., van Vliet, H., Bosch, J., 2002. Experiences with ALMA: architecture-level modifiability analysis. *Journal of Systems and Software* 61 (1), 47–57.
- Lee, J., 1991. Extending the potts and bruns model for recording design rationale. In: 13th International Conference on Software Engineering, pp. 114–125.
- Lee, J., 1997. Design rationale systems: understanding the issues. *IEEE Expert* 12 (3), 78–85.
- Lee, J., Lai, K., 1996. What’s in design rationale. In: Design Rationale: Concepts, Techniques and Use. Lawrence Erlbaum Associates, pp. 21–52, Chapter 2.
- Lethbridge, T.C., 2005. Studying software engineers: data collection techniques for software field studies. *Empirical Software Engineering* 10 (1), 311–341.
- Maclean, A., Young, R., Bellotti, V., Moran, T., 1996. Questions, options and criteria: elements of design space analysis. In: Design Rationale: Concepts, Techniques and Use. Lawrence Erlbaum Associates, pp. 53–106, Chapter 3.
- McCall, R., 1987. PHIBIS: procedural hierarchical issue-based information systems. In: Proceedings of the International Congress on Planning and Design Theory.
- Meares, C., Sargent, J., 1999. The digital work force: building infotech skills at the speed of innovation. Available from: <<http://www.tech-nology.gov/reports/techpolicy/digital.pdf>>.
- ObjectPlanet Inc., 2002. Surveyor: Web-based Survey Application. Available from: <<http://www.sharewareconnection.com/surveyor.htm>>.
- Parnas, D., Clements, P., 1985. A rational design process: how and why to fake it. *IEEE Transactions on Software Engineering* 12, 251–257.
- Perry, D.E., Wolf, A.L., 1992. Foundations for the study of software. *ACM SIGSOFT* 17 (4), 40–52.
- Potts, C., 1999. ScenIC: a strategy for inquiry-driven requirements determination. In: Proceedings: 4th IEEE International Symposium on Requirements Engineering. IEEE Computer Society Press, pp. 58–65.
- Potts, C., Burns, G., 1988. Recording the reasons for design decisions. In: 10th International Conference on Software Engineering, pp. 418–427.
- Potts, C., Takahashi, K., Antón, A.I., 1994. Inquiry-based requirements analysis. *IEEE Software* 11 (2), 21–32.
- Ramesh, B., Dhar, V., 1992. Supporting systems development by capturing deliberations during requirements engineering. *IEEE Transactions on Software Engineering* 18 (6), 498–510.
- Ramesh, B., Jarke, M., 2001. Towards reference models for requirements traceability. *IEEE Transactions on Software Engineering* 27 (1), 58–93.
- Seaman, C.B., 2002. The Information gathering strategies of software maintainers. In: Proceedings of the International Conference on Software Maintenance, p. 141.
- Simsek, Z., Veiga, J., 2001. A primer on internet organizational survey. *Organizational Research Methods* 4 (3), 218–235.
- Tang, A., Babar, M., Gorton, I., Han, J., 2005. A Survey on Architecture Design Rationale. Tech. Rep. SUTICT-TR2005.02, Swinburne University of Technology.
- Tang, A., Han, J., 2005. Architecture rationalization: a methodology for architecture verifiability, traceability and completeness. In: Proceedings of the 12th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS ’05). IEEE, USA, pp. 135–144.
- Tyree, J., Akerman, A., 2005. Architecture decisions: demystifying architecture. *IEEE Software* 22 (2), 19–27.
- Yin, R., 1998. The abridged version of case study research. In: Handbook of Applied Research Methods. Sage Publications, pp. 229–259, Chapter 8.