Literature Study on Design Rationale and Design Decision Documentation for Architecture Descriptions

Matthias Biehl
Embedded Control Systems
Royal Institute of Technology
Stockholm, Sweden
biehl@md.kth.se

July 2010

Version 20101101204500
TRITA-MMK 2010:06
ISSN 1400-1179
ISRN/KTH/MMK/R-10/06-SE

Abstract

In this document we provide an overview of the state of the art in documentation of design rationale and design decisions for architecture descriptions. We define the terminology of the area and compare the concept of rationale to similar concepts. We provide an overview of areas of contemporary research in design rationale. For each of the identified areas, we describe both the challenge and proposed solutions. Based on the findings from the literature we present evaluations of rationale and design decision documentation. The findings from this survey are promising and many open research questions with respect to rationale and design decision documentation need to be addressed. In the literature reviewed in this document it is for example not addressed how design decisions and design rationale should be represented, captured and used in model-based and model-driven development approaches. A need for further research in this area exists.

Contents

1 Introduction **6**
- 1.1 Other surveys . . . 6
- 1.2 Overview over this Document . . . 6

2 Terminology **7**
- 2.1 Architecture . . . 7
  - 2.1.1 Architectural Views . . . 7
  - 2.1.2 Architectural Patterns . . . 7
  - 2.1.3 Architectural Knowledge . . . 8
  - 2.1.4 Classification of Architectural Knowledge . . . 8
- 2.2 Design Rationale . . . 9
  - 2.2.1 Metamodel for Rationale . . . 9
  - 2.2.2 Characterization Scheme for Design Rationale . . . 10
- 2.3 Design Decision . . . 11

3 Comparison of Rationale with Similar Concepts **12**
- 3.1 Versions . . . 12
- 3.2 Variability and Variants . . . 12
- 3.3 Safety Case . . . 13
- 3.4 Traces . . . 13

4 Overview of Research on Rationale **13**
- 4.1 Research Communities . . . 14
- 4.2 Classification of Research on Design Rationale . . . 15

5 Identifying and Representing Rationale **16**
- 5.1 Metamodels for Rationale . . . 16
- 5.2 Ordering Rationale Elements . . . 18

6 Capturing Rationale **19**
- 6.1 Point of Time for Capturing . . . 19
- 6.2 The Capture Problem . . . 19
- 6.3 Reducing the Capture Problem . . . 20

7 Tools for Rationale Management **20**

8 Using Rationale **21**
- 8.1 Memory Aid and Communication Support . . . 21
- 8.2 Traceability and Consistency Checking . . . 22
- 8.3 Design Space Analysis, Exploration and Trade-off Analysis . . . 22
- 8.4 Change Management and Change Impact Analysis . . . 22
- 8.5 Evolution and Maintenance . . . 23
- 8.6 Teaching . . . 23
- 8.7 Reuse . . . 23

9 Benefits and Inhibitors **24**
- 9.1 Benefits of Using Rationale . . . 24
- 9.2 Inhibitors for Using Rationale . . . 25

10 Conclusion **26**

References **27**

1 Introduction

"When car developers at Ford Motor Company wanted to learn why the original Taurus design team was so successful, no one could tell them. No one remembered or had recorded what made that effort so special; the knowledge gained in the Taurus project was lost forever. The most valuable asset in any company is probably also its most elusive and difficult to manage: knowledge." [31]

The knowledge gained throughout a design process is a valuable asset and is important during the design itself, during the subsequent evolution and maintenance phase and for similar, new products. It is good practice to manage the knowledge of all kinds of engineered products, including the knowledge of intangible products, such as software. The knowledge can be classified into knowledge about the product (the "what") and knowledge about its justification, rationale and decision (the "why"). The "what" is usually documented in the form of models and source code. The "why" is usually not documented or represented explicitly, it is tacit and shared between individuals.

Individuals might forget this knowledge or leave the organization and take the knowledge with them. This phenomenon of disappearing architectural knowledge is called *architectural knowledge vaporization* [12]. Typical problems resulting from architectural knowledge vaporization are unawareness of previous design decisions and a lack of understanding of the artifacts such as the architecture descriptions, models and source code [36].

In this document we describe approaches for documenting the "why" in the form of design rationale, design decisions and architectural knowledge.

1.1 Other surveys

Early work on design rationale is summarized in [95, 82, 108]. The use of rationale in different software engineering activities is surveyed by Burge et al. [16] and by Dutoit et al. [38]. Barbar et al. [4] provide a survey focusing on the use of knowledge management for software architecture. Kruchten et al. [77] provide a survey with a historical perspective on architectural descriptions and design decisions.

1.2 Overview over this Document

The rest of this document is structured as follows. Section 2 introduces the terminology used in this document. In section 3 we compare the concept of rationale to similar concepts for modeling and managing metadata. In section 4 we give a brief overview of the different research communities involved and categorize contemporary research issues. The different research issues are presented in a section of their own: representing rationale in section 5, capturing rationale in section 6, managing rationale in section 7 and an overview of how rationale information can be used in section 8. In section 9 we present how the use of design rationale is evaluated in the literature. We summarize our findings in section 10.

2 Terminology

In this chapter the terminology is introduced, taking into account the terms related to rationale.

2.1 Architecture

The architecture of a software-intensive system is defined as "the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution" [65]. The architecture is a means for communication, analysis and synthesis. The terminology and best practices regarding software and systems architecture are specified in the two standards for architectural description of software-intensive systems (IEEE/ANSI 1471-2000) [65] and ISO/IEC 42010:2007 [66].

2.1.1 Architectural Views

An architectural *view* is a representation of the system from the perspective of a related set of concerns [26, 65]. The goal of views is to reduce the perceived complexity for a specific stakeholder. An architectural view provides a coherent selection, projection, abstraction or simplification of the architecture according to the interests of a specific stakeholder. A selection of several views and concerns forms an architecture framework. In [10] a set of default viewpoints is proposed, such as a module viewpoint, a component&connector viewpoint and an allocation viewpoint. A set of predefined views is proposed by Kruchten as the 4+1 model [74]. The 4+1 model comprises a logical view describing the functionality of the end-user, an implementation view for programmers, a process view for system integrators focusing on quality attributes, a deployment view for delivery and a use-case view connecting all of them. A *viewpoint* establishes the purpose for a view and the techniques or methods used for constructing the view [110, 65]. The viewpoint also specifies the conventions for interpreting, constructing and using an architectural view.

2.1.2 Architectural Patterns

Patterns offer reusable knowledge for recurring problems that is verified and proven through multiple uses in practice. *Architectural patterns* are also called *architectural styles*. Examples of patterns are the *pipes and filters pattern* and the *model view controller pattern* [50].

Pattern languages and pattern catalogs document common patterns that provide means for systematic reuse. In order to reuse architectural knowledge it is desirable to store, categorize and catalog it. Once this catalog is built and mined from the experience of several architects, it can be taught to novice architects. Architects can use the patterns from the catalog as verified building blocks. General pattern catalog are design patterns for object oriented programming [50] and the series of pattern-oriented software architecture by Buschmann et al. [19, 112, 92, 17, 18].

2.1.3 Architectural Knowledge

The development of an architecture is knowledge-intensive work, as it requires a deep technical understanding of several specific domains, human processes and how the product is used. Architecture knowledge (AK) comprises the knowledge that is essential to the construction of the architecture. The boundaries of architectural knowledge are not clearly defined. Several proposals have been made:

- AK = design decisions and design [78, 68]
- AK = drivers, decisions, analysis [56]
- AK = process, products, people, tools [32]

In this document we use the first definition of architectural knowledge as design decisions and design. In addition we provide a scheme for the characterization of architectural knowledge.

2.1.4 Classification of Architectural Knowledge

Architectural knowledge can be classified by four criteria [68, 46]. The criteria are the generality of the knowledge, the degree of formalization, the roles of the involved actors and the knowledge management strategy.

- **Generality of Architectural Knowledge:** Architectural knowledge can be specific to an application or it can be generic. Generic architectural knowledge can be applied in several applications [80]. *Application-specific knowledge* is also called *episodic knowledge*, *application-generic knowledge* is also called *library knowledge* [109]. An example of application-generic knowledge are design patterns (see section 2.1.2).
- **Architectural Knowledge Type:** Knowledge can be formalized to different degrees from *implicit* on the one end of the spectrum to *explicit* on the other end [46]. An example for implicit knowledge is the experience of the architect. Explicit knowledge on the other hand is written down and codified and thus independent of the person originally holding that knowledge. Explicit knowledge can be further divided into textual and formal knowledge.
- **Architectural Knowledge Management Roles:** The people interacting with architectural knowledge can take different roles, such as the role of a *producer* or the role of a *consumer* of architectural knowledge [46]. The producer captures and shares AK, the consumer assesses AK, learns from it and reuses it.
- **Architectural Knowledge Management Strategies:** To manage architectural knowledge, different strategies can be employed, such as personalization, codification, or a mix of the two [59]. There is a relation between the architectural knowledge management strategy and the preferred architectural knowledge type [68]: Using the *personalization strategy*, knowledge is shared from person to person through socialization. The focus is on knowing "who knows what". This strategy depends largely on

implicit knowledge, which can be shared effectively through personal interaction with the right person. The *codification strategy* on the other hand relies on sharing explicit knowledge. Once the knowledge is codified, it can be communicated and shared independently of the persons involved. A hybrid knowledge management strategy shares defined parts of architectural knowledge through the codification strategy, other parts through the personalization strategy.

2.2 Design Rationale

The dictionary defines the term *rationale* as the *fundamental reason* or the *underlying principle*. In engineering the *design rationale* captures the reasoning underlying the creation and use of artifacts [16], it subsumes the design decision, alternatives and the design decision rationale. Tang et al. [128] characterize design rationale as the reasons for making a decision and choosing a solution and explaining the relationships between the solution and the context of the solution. The *design decision rationale* is the reasoning attached to a particular design decision and captures the justification for a particular design decision.

The term rationale is linked to the architecture of a system. According to IEEE 1471 [65] an architectural description needs to provide the rationale for the choice of architectural concepts, including the alternative concepts that have been considered. While architecture is concerned with the finished artifact (the "what"), rationale is the description/documentation of the relevant decisions that lead to the architecture (the "why").

2.2.1 Metamodel for Rationale

Using the wide definition of design rationale, a rationale description contains the information relevant for making design decisions [82]. Rationale usually comprises the following elements:

- **Issue:** the design issue to be solved
- **Alternatives:** possible solutions to the issue
- **Decision:** the selected alternative to resolve the issue
- **Affected Artifact:** the manifestation of the decision in the architecture
- **Design Decision Rationale:** providing justification, reasoning and arguments for the design decision and its alternatives
- **Architecturally Significant Requirement:** a requirement that influence the alternatives and issues, can serve as an argument

In addition to the above elements, rationale can also comprise architectural constraints and architectural rules, which constitute previous architectural decisions that limit the solution space of the following design decisions. Rationale also provides traceability to the requirements and architectural elements, which can range from a simple pointer to a requirement to a complex trade-off analysis. An important part of the rationale are the links between related design decisions, between design decisions and requirements, and the links between design decisions and architectural elements [76]. Another addition to the metamodel

for rationale is a documentation of the assumptions under which a design decision is made [136, 81]. Further discussions on the metamodel and an overview of metamodels reported in the literature can be found in section 5.

2.2.2 Characterization Scheme for Design Rationale

We have identified a scheme for characterizing design rationale and its relation to design decisions and architectural models. This characterization is independent of the representation of design rationale.

Depth of Justification: Justification provides reasons for the nature of an artifact and gives an answer to the question "Why is a certain artifact the way it is?". This question can be asked recursively, so the number of recursions defines the *depth of the justification*. We define *rationale levels* (R0, R1, R2, R3) to capture the depth of justification. In this context an *artifact* can be a model, a system, subsystem, an architecture or a model element.

- R0 - Artifact Level: On the lowest level, there is no (= zero) rationale, but just the artifact that the rationale is attached to. The artifact is the foundation for the other levels to relate to. The artifact describes the *"what"*.
- R1 - Decision Level: The information on R1 level provides an answer to the question, why the artifact is the way it is. R1 answers this question by listing the design decisions and possible alternatives. R1 information justifies R0 information by providing the decisions that resulted in the creation of the artifacts.
- R2 - Decision Rationale Level: Decision rationale provides an answer to the question, why the decision was made. Decision rationale justifies the decision (R1) by providing reasons for the decision. Decision rationale can be of quantitative or qualitative nature. Quantitative rationale is e.g. the result of one or more quality analyses. Qualitative rationale is made up of arguments.
- R3 - Rationale Rationale Level: Rationale rationale provides a justification for the quality analysis, quality models, metrics and trade-off analysis that have been selected as criteria for making a decision.

The levels R0-R3 for the depth of justification have been inspired by the metamodeling levels M0-M3 by the OMG [101].

Subject of Justification: The subject of the justification defines what the rationale justifies.

- Existence: Rationale may justify the existence of a certain architectural element.
- Non-existence: Rationale may justify the non-existence of a certain architectural element.
- Property: Rationale may justify an attribute or property of the architectural model or the system, such as modularity or performance.

Type of Justification: There are different types of reasons that can be conveyed by design rationale.

- Technical: Rationale may provide reasons that are rooted in the product, its requirements or the development process.
- Executive: Rationale may provide non-technical reasons, for example finances, management, politics or culture.

Granularity of the Affected Artifact: Rationale information can be attached to model artifacts of different granularity.

- Architecture/Model: Coarse grained rationale provides justification for the model as a whole.
- Model Element: Fine grained rationale provides justification for a set of model elements that are part of a model.

2.3 Design Decision

Design decisions are decisions that directly influence the design of a system. When a software artifact is created, design decisions need to be made. Ideally decisions are made by the process of deliberation, pertaining a careful weighing of the alternatives. An architecture is the result of many design decisions, some of which have a larger impact than others. Decisions can be narrow in their scope or far reaching and affecting the complete system. Far reaching decisions have an impact on the architecture and are called *architectural design decisions* [68]. It is difficult to determine a priori which of the many design decisions are most important. Often it is only possible to find out about the importance of specific design decisions in retrospect, when trying to change an existing architecture [49].

Kruchten proposes an ontology of architectural design decisions [75]. The components of a design decision are introduced as:

- **Design Decision** is a textual description of the decision.
- **Design Decision Rationale** provides a justification for the decision.
- **Scope** or **Context** of the design decision can be either local (affecting only a specific part of the model), or crosscutting (affecting several parts of the model). Design decisions are made in a specific context and their validity and relevance depends on this context. Thus modeling a design decision entails modeling a context as well.

Further there is information for managing the decision such as state (e.g. preliminary or decided), cost associated with it, risk it deals with, category, author and timestamp. Design decisions do not exist by themselves, but depend on previous decisions and affect later decisions. Thus there are relationships between design decisions such as *constrains*, *forbids*, *enables*, *subsumes* (a wider specified design decision subsumes a tighter design decision), *conflicts with*, *overrides*, *comprises*, *bound to*, *is an alternative to*, *is related to* and *dependencies*.

The traditional view of architecture as components and connectors does not provide a notation for decisions. As a consequence of not recording the design decisions the information of design decisions is lost. In his position paper Bosch

proposes to describe the architecture using an explicit record of design decisions [67, 12]. The motivation is that explicit records of design decisions will reduce the effort necessary for changing the architecture and reduce design erosion. According to Bosch an architectural design decision consists of a restructuring effect (add components, split components, merge components, remove components, restructure components, add requirements, impose functionality), design rules (defines a particular way of performing a task), design constraints (defines what the system may not do) and a design decision rationale [12].

3 Comparison of Rationale with Similar Concepts

In this section we compare the rationale concept to similar concepts for managing and modeling metadata. With this study we can find out if similar concepts already provide what rationale can offer, or if a separate concept for rationale is justified.

3.1 Versions

Version management captures the evolution of an artifact over time and in distributed settings [7]. It allows retrieving earlier versions and developing alternative versions in parallel, known as branching. Examples of SCM systems are the Concurrent Versioning System (CVS) [133] and Subversion (SVN) [106]. In addition, version management systems specifically for models exist [3]. Version management is a state-based approach, which is focused on describing different versions as new states of an artifact instead of explicitly documenting the decision or change operations applied on the model [62]. The focus of rationale management is keeping track of the changes and the reasons for the changes (the "why"), whereas version management keeps track of the changed artifacts (the "what").

3.2 Variability and Variants

Variability is the ability of a software system to be efficiently extended, changed, customized or configured in a particular context. Decisions about binding time are managed by *variability management* and *configuration management* which describe the commonalities and variabilities of products. Software product lines or product families exploit the commonalities and variabilities of products by modeling an architecture with optional, alternative and reusable parts [27, 99]. Alternatives are called variants in variability modeling. To derive a product, a selection between the variants have to be made.

Both rationale and variability capture alternative options and the selection that has been made between them. Sinnema et al. [122] use variability modeling to capture rationale. The approach is based on the observation that the *alternatives* and *choices* of rationale management are similar to *variants* and *variation points* of variability management. Thurimella et al. [129] identify similarities between rationale management and variability management and unify it to their approach on issue-based variability modeling. The provide a mapping between concepts of rationale management and variability management, similar

to the approach by Sinnema et al. Variability focuses on what can be varied, but not on the reasons and justification for the variation. Variability usually describes alternatives that are intended to be realized. In variability management each allowed configuration represents a product. This means that every valid configuration is reasonable to build.

On the other hand not every alternative design decision expressed by rationale concepts should be realized. There can be design decisions that are rejected and will not be built. Rationale is used for design space exploration, exploring different options and getting a feedback regarding the quality of the design alternatives.

3.3 Safety Case

Before safety critical systems can be delivered, authorities often require a certification. Certification is intended to show that the system is safe enough, e.g. by showing that the engineering was done according to current industry standards and practices. It might also be required to show the safety-related decisions and reasoning. The safety case concept [73] can be used to describe the reasoning.

A safety case can be considered to be a specialized rationale focusing on safety. A safety case is similar to a design rationale, as it focuses on capturing the reasoning of a design. However, a safety case is created in retrospect, after the design is finished [130]. A safety case focuses on legal aspects, its purpose is satisfying authorities, and thus the description of design decisions needs to be simplified.

The scope of rationale is wider, it does not just focus on safety, but on other technical aspects as well. The purpose of design rationale is supporting design and thus design rationale and design decisions are captured as they are actually made. The documentation of design decisions for safety critical systems has been investigated by Wu et al. [137] and by Leveson [86].

3.4 Traces

Traces provide the ability to determine the origin and relation between model elements [104]. Traces can be created manually, or automatically by model transformations. Traces link source elements to target elements and optionally the transformation rule that established the link. Traces can also be used to check the consistency of a solution and whether the solution is actually solving the problem [52]. The ability to trace the origin and relation between model elements is one aspect of design rationale. However, traces do not provide the possibility to express alternatives, decisions and their justification.

4 Overview of Research on Rationale

In this chapter we give a short overview of the work in rationale research. In section 4.1 we identify the different research communities working on rationale. In section 4.2 we group and organize the current research areas. Sections 5 – 8 are organized according to this classification and introduce the respective areas in detail.

4.1 Research Communities

Several communities work on the topic of design rationale, spanning software engineering, mechanical engineering, economics, philosophy and psychology. These communities use different terminologies, have a different perspective, focus on different aspects of the problem and propose their own set of solutions.

- **Rationale Management:** The goals of rationale management are the refinement and the externalization of architectural knowledge [95, 82]. The approaches differ in the way they define, structure and elicit the rationale information.

- **Knowledge Management:** The tools and mechanisms of general knowledge management can be applied to support software engineering and software architecture [4]. Knowledge management is defined as the process that deals with systematically eliciting, structuring and facilitating the efficient retrieval and effective use of knowledge [41]. The goal of architectural knowledge management is to improve the business processes and practices by utilizing individual and organizational business resources, including skills, capabilities, experiences, routines, cultural norms and technologies. It involves tacit knowledge of experts and explicit knowledge, codified in procedures, processes and tools. Three basic approaches for knowledge management exist: (1) codification, which aims at making all knowledge explicit, (2) personalization, which aims at promoting "who knows what" and (3) the hybrid approach, which realizes that different types of knowledge can be better represented using codification or personalization.

- **Architecture Description:** Design rationale is an important part of an architecture description [65]. The architecture community is interested in describing rationale and design decisions and linking them to existing architectural concepts such as ADLs, views, patterns, frameworks, architecture quality analysis and trade-off analysis.

- **Pattern Languages:** The goals of the pattern community are the categorization of architectural knowledge into pattern languages and catalogs and the utilization of the patterns (cf. section 2.1.2). Rationale is used to provide reasoning and guidance for the selection between similar patterns [60].

- **Software Evolution and Maintenance:** Most problems of maintenance and evolution arise from the difficulties of understanding existing systems that were created by other people [42, 64, 40, 141]. Rationale is used to improve the understanding of the architecture, for example by change impact analysis. Rationale may thus improve the efficiency and accuracy of evolution and maintenance [21].

- **Engineering Design:** Engineering Design provides methods and tools to support the mechanical engineer. Design rationale is used as a tool for improving the understanding of the design and for the reuse of experience [1, 2].

- **Cognition and Psychology:** Design rationale is a means to enhance the understanding of artifacts. Thus design rationale is a support tool for the humans working with the artifacts. This is why psychologists have examined how rationale supports cognitive processes in design and understanding. The psychological aspects of design rationale have been studied by Shum [118].

- **Philosophy:** The field of rationale argumentation has been studied in philosophy, which provides the concepts used for representing rationale information [131].

- **Economics:** Decision-making has been studied extensively in the field of economic sciences. Simon [121] studied the behavioral and cognitive processes of making rational human choices. Simon identified three steps of decision making: (1) the identification and listing of all the alternatives; (2) the determination of all the consequences resulting from each of the alternatives; and (3) the comparison of the accuracy and efficiency of each of these sets of consequences. He primarily considers decision making under uncertainty, since in practice it is impossible to have perfect and complete information at any given time to make a decision.

4.2 Classification of Research on Design Rationale

In order to present an overview of the research done on rationale in a structured way, we propose a classification scheme that is based on the different tasks involved when working with design rationale.

The architectural knowledge management community has identified the tasks: (1) architectural knowledge identification, (2) architectural knowledge acquisition, (3) architectural knowledge development, (4) architectural knowledge distribution, (5) architectural knowledge preservation and (6) architectural knowledge use.

The rationale management community has identified similar tasks: (a) elicitation of rationale, (b) recording of rationale, (c) structuring and indexing of rationale, (d) retrieval of rationale, (e) delivery of rationale, (f) use of rationale.

We group the tasks identified by the architectural knowledge management community and the rationale management community and get four areas of research in rationale:

- **Identifying and Representing Rationale** is the topic of chapter 5
  In order to represent and structure rationale documentation, the aspects of architectural knowledge worth documenting need to be identified. (Tasks 1, c).

- **Capturing Rationale** is the topic of chapter 6
  Capturing of rationale subsumes the tasks elicitation, acquisition and recording of rationale (Tasks 2, a, b).

- **Managing Rationale** is the topic of chapter 7
  Rationale management uses methods and tools for developing, preserving, distributing, delivering and retrieving of rationale (Tasks 3, 4, 5, d, e).

- **Using Rationale** is the topic of chapter 8
  Rationale can be used to support a variety of engineering activities (Tasks 6, f).

The areas of research are discussed in the following sections.

5 Identifying and Representing Rationale

When design rationale and design decisions are documented, it needs to be identified which aspects of architectural knowledge are important and how they should be represented.

5.1 Metamodels for Rationale

Design decisions are usually not represented explicitly, even though a number of different approaches for representing design decisions exist. These different representations have varying degrees of formality and rigor. Informal representations document design decisions in the form of free text, use cases and videos [23]. Programming languages employ comments to explain the intention of several lines of code. UML [103] and SysML [102] also provide a notation for comments, depicted by a box and a dashed line. However, the comments are merely graphical elements and not connected to the identity of a model element.

Template-based approaches provide a guideline for textual descriptions of design decisions such as the template by Tyree and Akermann [132]. The majority of approaches represent design decisions that are modeled according to a metamodel. These approaches explicitly represent the design deliberation process, including alternatives and arguments, the earliest being IBIS. Approaches for representing design decisions that were developed in recent years focus on linking design decisions with the architecture [77]. Some approaches propose a separate view for design decisions [37], others present the architecture and its design decisions in the same environment, as for example the AREL approach. AREL is a UML profile that can be used to annotate architectural models with design decisions and rationale [124]. The SEURAT approach links rationale to specific lines in the source code [15]. These models provide descriptions, traces and links to the artifacts affected by rationale.

Many different metamodels for rationale and design decisions exist and in the following we present some of them.

- **IBIS** (Issue-Based Information Systems) [79] is the first metamodel for rationale support and most rationale tools are based on this approach. It models rationale using the following concepts:
  - Issue: a question that needs to be resolved
  - Position: alternatives
  - Argument: support or counter of a position or other arguments
- **gIBIS** [30] is a hypertext-based version of IBIS.
- **PHI** (Procedural Hierarchy of Issues) [93] is based on IBIS and adds relationships between issues, resulting in a hierarchical structure of issues.

- **Potts and Bruns** [107] is the first system to integrate both artifacts and their rationale. It is based on IBIS with a slightly changed metamodel, where all arguments for a decision are merged into one justification element.
- **DRL** (Decision Representation Language) [83] is based on the Potts and Burns model, arguments are called claims and are decomposed in a similar way as in IBIS. It uses the concepts of claims and arguments. DRL is mainly used for design space analysis, but is expressive enough to be applied in other contexts as well.
- **QOC** (Questions Options and Criteria) [91] is focused on retrospective rationale documentation of the design space. Its metamodel contains the following concepts:
  - Question: a design issue that needs to be resolved
  - Option: corresponds to a decision or position
  - Criterion: a metric used for the evaluation of an option
- **Scenario-Claims Analysis** (SCA) [24] represents system features, criteria and evaluation of the features against the criteria. It does not provide an explicit representation of the argumentation or of design decisions.
- **RATspeak** [15, 14] is based on DRL. In addition to the concepts of DRL, the metamodel comprises requirements, assumptions and dependencies between alternatives. RATspeak defines an ontology for software engineering rationale that allows automated reasoning with the rationale information, for example for change impact analysis.
- **AREL** (Architecture Rationale and Element Linkage) [124] focuses on representing the factors that influence the design decision and on linking rationale to the architecture.
  - Design Concerns: model the context of the decision
  - Environmental Factors: external constraints of the design
  - Functional and Non-functional Requirements: motivate the design
  - Purposes and Goals: provide a context to guide the design
  - Design Options: comprise both the chosen design and alternative design
- **Archium** [70] is capable of describing both an architecture and the design decisions inherent in the architecture. It interprets design decisions as functions that change the architecture. It is based on ArchJava and assumes that the architecture is implemented in Java.
- **Intent Specifications** [86, 87] by Leveson focus on the documentation of reasoning for safety critical software. The representation language attempts to balance formal and informal specification. The intent is documented according to several views: the management view, customer view, system engineering view, component designer view, component implementer view and operations view. Intent specifications have been used to document assumptions of software components to enable their safe reuse in spacecrafts [136] and mobile robots [98].

- The **Template by Tyree and Akermann** [132] is the basis for many metamodels for rationale. It has the major concepts decision, constraint, issue, solution and rationale. Minor elements are the category, status, dependency, associated artifact, consequence and the affected stakeholder.

- **Kruchten's Ontology** of architectural design decisions [75] includes four major classes of design decisions: existence decisions, non-existence decisions, property decisions and executive decisions. An existence decision states that an element exists in the architecture, whereas a non-existence decision states that a particular element is not in the architecture on purpose. Property decisions target a specific attribute or property of the system, thus they are cross-cutting and affect multiple elements. Executive decisions are motivated by finances, methodology, politics, culture, or the people. The components of a design decision are introduced as the decision itself, the rationale, the scope of the decision, the cost associated with it and the risk it deals with. Further there is information for managing the decision such as state (e.g. preliminary or decided), category, author and timestamp. Design decisions do not exist by themselves, but depend on previous decisions and affect later decisions. Thus there are relationships between design decisions such as *constrains*, *forbids*, *enables*, *subsumes* (a wider specified design decision subsumes a tighter design decision), *conflicts with*, *overrides*, *comprises*, *bound to*, *is an alternative to*, *is related to* and *dependencies*.

- **SOAD** [144, 143] aims to build a library of domain specific design patterns and their rationale for the domain of service oriented architectures. Based on this library a tool can create a todo-list to guide engineers in the selection of design patterns from the library.

- **Design Constraints** [127] are used to describe how design decisions are formed based on external constraints.

- **DPRG** (Design Pattern Rationale Graph) [9] uses graphs to link design rationale with a model of the source code.

- **ATRIUM** [97] metamodel describes rationale by a text attribute for design decision and design rationale that is defined for each metaclass. They describe a model-driven development approach creates trace links between different model elements.

The survey [115] reviews a number of metamodels for capturing design decisions. They find that most metamodels have a consensus on the major modeling concepts and there are small divergences on the minor modeling concepts. They provide a table listing similar modeling elements of different tools, by mapping them to the template by Tyree and Akermann. A similar approach is taken by Liang et al. [89] to map rationale captured using different metamodels.

5.2 Ordering Rationale Elements

The rationale of an architecture consists of several rationale elements. In this section we investigate how several rationale elements are structured and ordered. In the literature two criteria for ordering rationale information are discussed.

- Chronological approaches order rationale according to a timeline. Rationale thus represent the history of the architecture.
- Structure-oriented approaches order rationale according to the architectural model. Rationale information is linked to the artifact being designed. The main advantage of this ordering principle is the traceability between rationale and architecture or between rationale and source code [15, 113]. To be able to manage the complexity of this information, a specific architectural view for design decisions is proposed, the "decision view" [37].

6 Capturing Rationale

To be able to use rationale information, the rationale has to be first externalized and captured. In this section we show at which point in time rationale can be captured. We discuss the problems encountered when capturing design rationale as well as approaches aiming to reduce these problems.

6.1 Point of Time for Capturing

Design rationale can be captured before the architecture is built or after the architecture has been built [20].

- Rationale captured after the architecture has been built is called *retrospective design rationale* [118]. The goal of retrospective rationale is supporting the maintainer during the evolution of the system. Rationale has an explanatory role, and does not need be historically complete. Information that is not relevant in retrospect can be simplified or left out.
- Rationale that is captured before the architecture is designed is called *prospective design rationale*. Its goal is to support problem formulation, design space analysis and decision making. Rationale focuses on ongoing issues, positions and arguments.

Most of the powerful approaches found in the literature employ a prospective design rationale. In this document we will assume a prospective design rationale, unless otherwise noted.

6.2 The Capture Problem

The different hindrances for capturing rationale are summarized as the capture problem. The main reason for not capturing rationale is the overhead it entails. Further reasons are the disruption of creativity caused by documentation and the loss of individual power.

Tang et al. [123] studied empirically how software professionals document and use design rationale. He found that software professionals recognize the importance of design rationale for their work, however, they often omit to provide the "why" in the form of rationale and justification and focus only on the "what" in the form of architecture, models and source code. In case studies it was found that designers usually do not want to capture rationale and there are only a few individuals in a large team that capture the rationale [29]. Rationale capture is time consuming and perceived as an overhead. It is also argued

that rationale capture is intrusive to the design activity [48], thus hindering and blocking creativity and flow, especially if it is captured in a formal way. In addition, there is a conflict of interests, since the individuals capturing the rationale are not the beneficiary [54]. The individuals capturing the rationale will not profit from codifying their knowledge. On the contrary, employers make themselves exchangeable by capturing rationale, since codified knowledge reduces the companies dependence on the individual designer.

6.3 Reducing the Capture Problem

The overall goal of current work in rationale capture is the reduction of the amount of work required for capturing design decisions. In the following we present some of these approaches.

- Goal-oriented approaches limit the amount of information that needs to be captured by design decisions and to capture only the design decisions that are most important for reaching a certain goal. However, it is difficult to know beforehand, which information will be important later [43].
- Methodological approaches devise a method for reducing the amount of captured design decisions. One such method is the 'flag, filter and form' method [85]. Each design decision is flagged by setting a marker to remember the location of the design decision, then design decisions are filtered, to find the ones that are worth documenting. Lastly the selected design decisions are documented and more descriptive details are added to them.
- Indirect capturing of design rationale as a by-product has been explored. Data mining algorithms have been used to to capture rationale from email traffic among designers [72].
- Pattern-based approaches promise the reuse of generic rationale parts from a rationale library [57, 94].
- Generative approaches separate capturing from formalization [53]. Rationale is captured in a raw format and later formalized when needed [116]. It has also been proposed to automate the collection of decisions and rationale [96] to solve the capture problem. For example, the algorithm of *latent semantic analysis* is used [33] to extract knowledge from existing documents, e.g. word documents. Other generative approaches use incremental formalization [117] or differential description [48] of rationale to minimize the impact of the capture problem.

7 Tools for Rationale Management

In this section we introduce tools that support the management of design rationale. Rationale management tools provide design support, maintenance support, learning support and documentation support.

- **ADDSS** (Architecture Design Decision Support System) [22] is a web-based tool for capturing design decisions. It focuses on capturing and visualizing the evolution of the architecture, thus supporting iterative development. It also supports capturing and reusing architectural patterns.

- **Archium** [70] is a tool for representing both the architecture and the architectural design decisions. The tool makes it easy to see the relation between the design decisions and their effect on the architecture.
- **AREL** (Architecture Rationale and Element Linkage) [124] focuses on connecting design concerns in the problem space to design outcomes in the solution space using design decisions as connection points. AREL is implemented as UML profile.
- **Knowledge Architect** is a repository for architectural knowledge [90]. The repository can be accessed using different clients such as Word and Excel and has facilities to translate the architectural knowledge. It is based on semantic web technology.
- **SEURAT** (Software Engineering using RATionale system) [15] is an Eclipse plug-in for documenting design decisions, alternatives, reasons, arguments and for tracing requirements to code. The tool supports the maintenance process of software.
- **EAGLE** (Environment for Architects to Gain and Leverage Expertise) [47] supports both codified and personalized knowledge and can be tailored to the needs of different stakeholders using notifications and search mechanisms.
- **PAKME** (Process-based Architecture Knowledge Management Environment) [6] is a web-based tool that supports both codified and personalized knowledge.
- **SIBYL** [84] has a special focus on managing group design rationale, where several architects can contribute to the knowledge of the group.
- **ADkwik** [114] is a wiki platform with an underlying data model for rationale.
- **SEI-ADWiki** [8] is a wiki platform with a focus on communication and collaboration of rationale. It allows free text comments without a specific metamodel.

Lian et al. have conducted a study to compare some existing tools for rationale management [88] based on the use-cases that the tools support.

8 Using Rationale

In this chapter we explore how rationale can support the different activities throughout the life cycle. Additional information on the usage of rationale with respect to the different phases of software engineering is provided by Burge et al. [16].

8.1 Memory Aid and Communication Support

Rationale provides information both to the rationale author and to other stakeholders.

- Rationale can be used as a memory aid for the rationale author, to remember what was decided and why.
- Rationale is also a means for communicating architectural knowledge to other stakeholders such as team members, other teams, clients or maintainers. Rationale has e.g. been used to capture discussions in meetings [29]. In global software development virtual teams develop software without the possibility for face-to-face meetings. They need to efficiently exchange externalized architectural knowledge. Documentation of design decisions and design rationale has been found useful in global software development [135, 28, 61].

8.2 Traceability and Consistency Checking

Traces can be used to check the consistency of a solution. We can for example check, whether the recorded rationale is consistent with the decision taken. The AREL [124] approach allows consistency checking between rationale and architecture elements.

Several approaches have explored using rationale information as the link between requirements and architecture, linking problem space and solution space [34, 46, 58, 55].

8.3 Design Space Analysis, Exploration and Trade-off Analysis

The activity of decision making comprises generating alternatives, evaluating them and selecting the best alternative, as it is recommended by the Capability Maturity Model Integration (CMMI) [25]. Which alternative is the best is measured by the criteria used. The criteria can be constraints, assumptions, risks, functional requirements and non-functional requirements.

If design decisions and their alternatives are documented, the alternatives can be evaluated. The evaluation can be automated to varying degrees [134, 111], from evaluation, to design space exploration and design optimization. While the alternatives (R1 level) are the input to the evaluation, the design decision rationale (R2 level) records the justification of the evaluation and trade-off decision [69, 5]. In this sense, design rationale is both a source of information used for analysis and a way to store analysis results.

8.4 Change Management and Change Impact Analysis

Changes to an architecture can be made at different times in the life cycle. Before the change is performed, it needs to be understood how a proposed change will impact the system. The impact of the change can be predicted by change analysis techniques [16]. Tang et al. [125] use design rationale as the basis for their automated tool support using Bayesian belief networks for change impact analysis. Bratthall et al. [13] examine in a controlled experiment if design rationale improves the correctness and the efficiency of change impact analysis. The participants were split in two groups, a study group that received a design rationale with the change requests and a control group that received the same change requests, but without design rationale. The results showed that

both efficiency and correctness of the change impact analysis were significantly improved when design rationale is available.

8.5 Evolution and Maintenance

Maintenance accounts for 67 - 90 percent of the total cost of the software system [42, 64, 40, 141]. Maintainers spend most of their time understanding the existing system before they are able to change it. Design rationale documentation has the potential to improve the maintainer's understanding of the system [21]. However, the documentation of design rationale is often kept separate from the architecture. Thus the architecture often evolves and the documentation is not updated appropriately. As a result, separate documentation is often an unreliable source of information [135]. Rationale description coupled to the architectural description can help to keep the architecture and its rationale synchronized and improve the trustworthiness of the documentation.

Karsenty [71] shows empirically that the time needed for maintenance tasks can be reduced considerably if design rationale and design decisions are documented.

8.6 Teaching

Rationale can provide a framework for articulating and teaching architectural knowledge [128], as it provides insight into how others have solved problems in similar situations [16]. We can differentiate between teaching application-specific knowledge and application-generic knowledge (cf. section 2.1.4).

- Application-generic knowledge, is independent of a particular application and can be taught to students, so they can reuse the knowledge in other applications.
- Application-specific knowledge is mainly to be shared among the team members working on the same project [1].

8.7 Reuse

Reuse promises that software systems can be built from tested and verified components, resulting in fewer defects and higher quality in less time. Reusable components make assumptions about the context they are used in. If a component is reused in a new context, it needs to be checked if the assumptions of the component can be met by the new context. The assumptions about the context can be stated as the design rationale.

Weiss et al. [87, 136] use *Intent Specifications* [86] for documenting assumptions of spacecraft components. The explicit documentation of assumptions enables the safe reuse of the components.

Habli and Kelly [56] experiment with an approach, in which design decisions are captured in one project and subsequently replayed in another project. Design decisions are made in a context, so the context needs to be captured in the form of assumptions and justifications, design dependencies and rationale for selection among alternatives. When replaying the design decisions in another project, the applicability of the design decision needs to be checked for the new context. Mismatches need to be detected and the design might need

to be adapted. Due to the possibility of such an approach to violate the design assumptions and dependencies, they conclude that reusing an architecture without rationale or reasoning is very risky.

Design patterns are generalized solutions and thus incorporate several design decisions. Patterns are independent of a particular context, they capture generic architectural knowledge. Patterns provide application-generic, explicit architectural knowledge and are technology independent. Design decisions on the other hand are specific to a particular context and application. Patterns need to be adapted to the context, when they are applied in a specific architecture. Harrison et al. [60] propose to use patterns to capture architectural design decisions and work out in detail the difference between architectural patterns and design decisions. Zdun works on the relationship between patterns and design decisions [60, 105]. He describes how design decisions can be derived from general pattern specification. He proposes a pattern-based architecture approach based on patterns and design decisions [140, 139]. Zimmermann et al. [144, 142] describe the concept of *reusable design decisions*. They combine ideas from design patterns with design decisions and build a domain specific catalog of design decisions for SOA systems.

9 Benefits and Inhibitors

In this section we present, how the documentation of design decisions and design rationale is evaluated in the literature. In section 9.1 some of the benefits of using design rationale and in section 9.2 some of the inhibitors for using rationale are presented.

9.1 Benefits of Using Rationale

Falessi et al. [44, 45] investigate empirically, when rationale is deemed useful by developers. They designed a controlled experiment to test the perceived value of design decisions for performing different software engineering activities. One of their findings is that there are big differences in perceived value, depending on which element the rationale is associated to. To strengthen their results, they first performed this experiment at a university in Spain and later replicated that experiment in Rome with similar results. Empirical studies by Bratthall et al. [13]and Karsenty [71] showed that both change impact analysis and maintenance tasks can be improved if design rationale and design decisions are documented.

Shum [118, 119] has studied the people-related, psychological and cognitive effects of rationale documentation. Rationale can improve different aspects of communication:

- Rationale can support an individual engineer or a team of engineers
- Rationale can be a means to systematically explore the design space, analyze alternatives and trade-offs.
- Rationale can be the basis for a focused discussion on the design decisions, including their advantages, disadvantages and interrelations. Rationale is intended to support communication, reflection and analysis in design [63].

- During maintenance activities the existing system needs to be understood before it is changed. Rationale can reduce the time to change the system [13].
- Rationale can foster learning from the design of others and creates possibilities to learn from successes and failures from the past [1].

Rationale can improve design reuse, design communication and design verification [138]. Records of explicitly represented architectural knowledge are a support for the engineer when maintaining and evolving systems [78]. If the designers needs to justify their design decisions, the quality increases, which has been shown independently in several controlled experiments [126, 13].

For a company, rationale can be seen as the corporate technical knowledge of this company [11, 35]. The technical knowledge is independent of individual persons [138]. This 'corporate knowledge' is a strategic factor for any business [138]. *Learning organizations* [39] consciously manage their knowledge and treat it as a valuable asset. The effective application of knowledge management is claimed to be one of the important success factors of japanese companies [100].

9.2 Inhibitors for Using Rationale

Even though there are advantages of explicitly captured rationale, a number of inhibitors for the use of rationale exist.

- **Capture Problem:** Capturing rationale is expensive and time-consuming. Thus the documentation of rationale creates overhead for the decision maker [43, 120]. The benefits of rationale need to be worth the additional effort and overhead of capturing [63]. The capture problem and possible solutions are discussed in section 6.2.
- **Means - Ends Uncertainty:** Rationale documents important knowledge, a part of which will be potentially very useful during later evolution of the system. However, it is unknown which part of the knowledge will be needed in the future, since it is unknown which issues will come up and will need to be resolved in the future [39].
- **Trust Problem:** A general problem of knowledge management systems is the amount of trust that people have in the system. There may be a lack of trust both on the side of the provider and on the side of the consumer of the knowledge [51].
- **Power Issue:** The codification of rationale can expose the lack of knowledge [51, 43]. Some developers do not like to capture their architectural knowledge because they want to avoid that potential weaknesses of their design are exposed [123, 63]. If employees keep the rationale knowledge to themselves, they can benefit from it individually [43], as colleagues depend on their specific knowledge [1]. Engineers might not be willing to give up this power they hold through their internal knowledge.
- **Beneficiary Problem:** Capturing rationale is expensive and time-consuming for the rationale provider, the person capturing the rationale. However, the rationale provider does not benefit from captured rationale, as he already internalized this knowledge. Instead, other individuals are the beneficiaries.

10 Conclusion

In this document we provided an overview of the state of the art in the area of design rationale and design decision documentation. We defined the terminology and provided a classification scheme for design rationale. We compared the rationale concept to several similar concepts for modeling and managing meta data. The concept of rationale shares several aspects with a number of similar concepts, however these concepts provide a different scope, coverage or focus, so a separate concept for design rationale seems justified.

Research in rationale is multidisciplinary and contributions are made in different research communities. We provided an overview of the research communities and classified areas of contemporary research. For each of the identified areas, we gave an overview of problem descriptions and proposed solutions with references to publications. From the literature we collected both the benefits and inhibitors of using rationale and design decisions.

The findings from this survey are promising and many open research questions with respect to rationale and design decision documentation need to be addressed. In the literature reviewed in this document it is for example not addressed how design decisions and design rationale should be represented, captured and used in model-based and model-driven development approaches. A need for further research in this area exists.

Acknowledgements

This work has been partially funded by the ARTEMIS projects CESAR and iFEST. The author would like to thank Martin Törngren for reviewing this document.

References

[1] Ahmed, S., "Understanding the knowledge needs of novice designers in the aerospace industry," *Design Studies*, vol. 25, no. 2, pp. 155–173, March 2004. [Online]. Available: http://dx.doi.org/10.1016/j.destud.2003.10.006

[2] Ahmed, Saeema, "Understanding the use and reuse of experience in engineering design," Ph.D. dissertation, Cambridge University Engineering Department, 2001.

[3] Altmanninger, Kerstin, Seidl, Martina, and Wimmer, Manuel, "A Survey on Model Versioning Approaches," Johannes Kepler University Linz, Tech. Rep., 2009. [Online]. Available: http://smover.tk.uni-linz.ac.at/docs/IJWIS09_paper_Altmanninger.pdf

[4] Babar, Muhammad A., T. Dingsøyr, Lago, Patricia, and van Vliet, Hans, Eds., *Software Architecture Knowledge Management: Theory and Practice*, 1st ed. Springer, August 2009. [Online]. Available: http://www.worldcat.org/isbn/3642023738

[5] Babar, Muhammad A. and Gorton, Ian, "A Tool for Managing Software Architecture Knowledge," in *SHARK-ADI '07: Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 11+. [Online]. Available: http://dx.doi.org/10.1109/SHARK-ADI.2007.1

[6] Babar, Muhammad A., Northway, Andrew, Gorton, Ian, Heuer, Paul, and Nguyen, Thong, "Introducing Tool Support for Managing Architectural Knowledge: An Experience Report," in *IEEE International Conference on the Engineering of Computer-Based Systems*, vol. 1. Los Alamitos, CA, USA: IEEE, 2008, pp. 105–113. [Online]. Available: http://dx.doi.org/10.1109/ECBS.2008.27

[7] Babich, Wayne A., *Software Configuration Management: Coordination for Team Productivity*. Addison Wesley Longman, February 1986. [Online]. Available: http://www.worldcat.org/isbn/0201101610

[8] Bachmann, Felix and Merson, Paulo, "Experience Using the Web-Based Tool Wiki for Architecture Documentation," SEI, Tech. Rep., 1998.

[9] Baniassad, E. L. A., Murphy, G. C., and Schwanninger, C., "Design pattern rationale graphs: linking design to source," in *Proceedings of the 25th International Conference on Software Engineering, 2003*, 2003, pp. 352–362. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1201214

[10] Bass, Len, Clements, Paul, and Kazman, Rick, *Software Architecture in Practice (2nd Edition)*, 2nd ed. Addison-Wesley Professional, April 2003. [Online]. Available: http://www.worldcat.org/isbn/0321154959

[11] Bjornson, F. and Dingsoyr, T., "Knowledge management in software engineering: A systematic review of studied concepts, findings

and research methods used," *Information and Software Technology*, vol. 50, no. 11, pp. 1055–1068, October 2008. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2008.03.006

[12] Bosch, Jan, "Software Architecture: The Next Step," *Software Architecture*, vol. 3047, pp. 194–199–199, 2004. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24769-2_14

[13] Bratthall, Lars, Johansson, Enrico, and B. Regnell, "Is a Design Rationale Vital when Predicting Change Impact? A Controlled Experiment on Software Architecture Evolution," in *PROFES '00: Proceedings of the Second International Conference on Product Focused Software Process Improvement*. London, UK: Springer-Verlag, 2000, pp. 126–139. [Online]. Available: http://portal.acm.org/citation.cfm?id=713240

[14] Burge, J. and Brown, D. C., "Reasoning With Design Rationale," in *Artificial Intelligence in Design*, 2000, pp. 611–629. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.8817

[15] Burge, J. E. and Brown, D. C., "An Integrated Approach for Software Design Checking Using Rationale," in *Proc. Design Computing and Cognition Conference*, Cambridge, MA, 2004.

[16] Burge, Janet E., Carroll, John M., McCall, Raymond, and I. Mistrík, *Rationale-Based Software Engineering*, 1st ed. Springer, May 2008. [Online]. Available: http://www.worldcat.org/isbn/354077582X

[17] Buschmann, Frank, Henney, Kevlin, and Schmidt, Douglas C., *Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing*. Wiley, May 2007. [Online]. Available: http://www.worldcat.org/isbn/0470059028

[18] ——, *Pattern Oriented Software Architecture Volume 5: On Patterns and Pattern Languages*. Wiley, June 2007. [Online]. Available: http://www.worldcat.org/isbn/0471486485

[19] Buschmann, Frank, Meunier, Regine, Rohnert, Hans, Sommerlad, Peter, and Stal, Michael, *Pattern-Oriented Software Architecture Volume 1: A System of Patterns*, 1st ed. Wiley, August 1996. [Online]. Available: http://www.worldcat.org/isbn/0471958697

[20] Capilla, Rafael, "Embedded Design Rationale in Software Architecture," in *Proceedings of the WICSA 2009*, 2009. [Online]. Available: http://wwwp.dnsalias.org/w/images/8/84/EmbeddedDesignRationale100.pdf

[21] Capilla, Rafael, Nava, Francisco, and J. C. Dueñas, "Modeling and Documenting the Evolution of Architectural Design Decisions," in *SHARK-ADI '07: Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent*. Washington, DC, USA: IEEE Computer Society, 2007, p. 9. [Online]. Available: http://dx.doi.org/http://dx.doi.org/10.1109/SHARK-ADI.2007.9

[22] Capilla, Rafael, Nava, Francisco, S. Pérez, and J. C. Dueñas, "A web-based tool for managing architectural design decisions," *Sw. Eng. Notes*, vol. 31, no. 5, pp. 4+, 2006. [Online]. Available: http://dx.doi.org/10.1145/1163514.1178644

[23] Carroll, John M., Alpert, Sherman R., Karat, John, Van Deusen, Mary, and Rosson, Mary B., "Raison d'Etre: capturing design history and rationale in multimedia narratives," in *CHI '94*, 1994, pp. 192–197. [Online]. Available: http://dx.doi.org/10.1145/191666.191741

[24] Carroll, John M. and Rosson, Mary B., "Deliberated evolution: stalking the view matcher in design space," *Hum.-Comput. Interact.*, vol. 6, no. 3, pp. 281–318, 1991. [Online]. Available: http://dx.doi.org/10.1207/s15327051hci0603%5C&4_4

[25] Chrissis, Mary B., Konrad, Mike, and Shrum, Sandy, *CMMI: Guidelines for Process Integration and Product Improvement (Sei Series in Software Engineering)*, 2nd ed. ed. Addison-Wesley Longman, Amsterdam, November 2006. [Online]. Available: http://www.worldcat.org/isbn/0321279670

[26] Clements, Paul, Bachmann, Felix, Bass, Len, Garlan, David, Ivers, James, Little, Reed, Nord, Robert, and Stafford, Judith, *Documenting Software Architectures: Views and Beyond.* Addison-Wesley Professional, September 2002. [Online]. Available: http://www.worldcat.org/isbn/0201703726

[27] Clements, Paul and Northrop, Linda, *Software Product Lines: Practices and Patterns*, 3rd ed. Addison-Wesley Professional, August 2001. [Online]. Available: http://www.worldcat.org/isbn/0201703327

[28] Clerc, Viktor, Lago, Patricia, and van Vliet, Hans, "Global Software Development: Are Architectural Rules the Answer?" *Global Software Engineering, International Conference on*, vol. 0, pp. 225–234, 2007. [Online]. Available: http://dx.doi.org/10.1109/ICGSE.2007.21

[29] Conklin, E. Jeffrey and Yakemovic, K. C. Burgess, "A process-oriented approach to design rationale," *Hum.-Comput. Interact.*, vol. 6, no. 3, pp. 357–391, 1991. [Online]. Available: http://dx.doi.org/10.1207/s15327051hci0603%5C&4_6

[30] Conklin, Jeff and Begeman, Michael L., "gIBIS: a hypertext tool for exploratory policy discussion," *ACM Trans. Inf. Syst.*, vol. 6, no. 4, pp. 303–331, October 1988. [Online]. Available: http://dx.doi.org/10.1145/58566.59297

[31] Davenport, Thomas H. and Prusak, Laurence, *Working Knowledge: How Organizations Manage What They Know.* Project Management Institute, December 1997. [Online]. Available: http://www.worldcat.org/isbn/0875846556

[32] de Boer, Remco, Farenhorst, Rik, Lago, Patricia, van Vliet, Hans, Clerc, Viktor, and Jansen, Anton, "Architectural Knowledge: Getting to the

Core," in *Software Architectures, Components, and Applications* , ser. Lecture Notes in Computer Science, Overhage, Sven, Szyperski, Clemens A., Reussner, Ralf, and Stafford, Judith A., Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 4880, ch. 12, pp. 197–214. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-77619-2_12

[33] de Boer, Remco C. and van Vliet, Hans, "Architectural knowledge discovery with latent semantic analysis: Constructing a reading guide for software product audits," *J. Syst. Softw.*, vol. 81, no. 9, pp. 1456–1469, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2007.12.815

[34] Dick, Jeremy, "Design Traceability," *IEEE Softw.*, vol. 22, no. 6, pp. 14–16, 2005. [Online]. Available: http://dx.doi.org/10.1109/MS.2005.150

[35] T. Dingsøyr and Conradi, Reidar, "A Survey of Case Studies of the Use of Knowledge Management in Software Engineering," *International Journal of Software Engineering and Knowledge Engineering*, vol. 12, no. 4, pp. 391–414, 2002.

[36] T. Dingsøyr and van Vliet, Hans, "Introduction to Software Architecture and Knowledge Management," in *Software Architecture Knowledge Management*, Ali Babar, Muhammad, T. Dingsøyr, Lago, Patricia, and Vliet, Hans, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, ch. 1, pp. 1–17. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02374-3_1

[37] J. C. Dueñas and Capilla, Rafael, "The Decision View of Software Architecture," *Software Architecture*, pp. 222–230, 2005. [Online]. Available: http://dx.doi.org/10.1007/11494713_15

[38] Dutoit, Allen H., McCall, Raymond, Mistrik, Ivan, and Paech, Barbara, Eds., *Rationale Management in Software Engineering.* Springer, 2006.

[39] Easterby-Smith, Mark and Lyles, Marjorie A., Eds., *The Blackwell Handbook of Organizational Learning and Knowledge Management.* Wiley-Blackwell, August 2005. [Online]. Available: http://www.worldcat.org/isbn/140513304X

[40] Eastwood, A., "Firm fires shots at legacy systems," *Computing Canada*, vol. 19, no. 2, p. 17, 1993.

[41] Ebert, Christof and De Man, Jozef, "Effectively utilizing project, product and process knowledge," *Inf. Softw. Technol.*, vol. 50, no. 6, pp. 579–594, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2007.06.007

[42] Erlikh, Len, "Leveraging Legacy System Dollars for E-Business," *IT Professional*, vol. 2, no. 3, pp. 17–23, 2000. [Online]. Available: http://dx.doi.org/http://dx.doi.org/10.1109/6294.846201

[43] Falessi, Davide, Becker, Martin, and Cantone, Giovanni, "Design decision rationale: experiences and steps ahead towards systematic use," *SIGSOFT Softw. Eng. Notes*, vol. 31, no. 5, p. 2, 2006. [Online]. Available: http://doi.acm.org/10.1145/1163514.1178642

[44] Falessi, Davide, Cantone, Giovanni, and Kruchten, Philippe, "Value-Based Design Decision Rationale Documentation: Principles and Empirical Feasibility Study," in *WICSA '08: Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 189–198. [Online]. Available: http://dx.doi.org/10.1109/WICSA.2008.8

[45] Falessi, Davide, Capilla, Rafael, and Cantone, Giovanni, "A value-based approach for documenting design decisions rationale: a replicated experiment," in *SHARK '08: Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge*. New York, NY, USA: ACM, 2008, pp. 63–70. [Online]. Available: http://doi.acm.org/10.1145/1370062.1370079

[46] Farenhorst, Rik and Boer, Remco C., "Knowledge Management in Software Architecture: State of the Art," in *Software Architecture Knowledge Management* , Ali Babar, Muhammad, T. Dingsøyr, Lago, Patricia, and Vliet, Hans, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, ch. 2, pp. 21–38. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02374-3_2

[47] Farenhorst, Rik, Lago, Patricia, and van Vliet, Hans, "Eagle: Effective Tool Support for Sharing Architectural Knowledge," *Int. J. Cooperative Inf. Syst.*, vol. 16, no. 3/4, pp. 413–437, 2007.

[48] Fischer, Gerhard, Lemke, Andreas C., McCall, Raymond, and Morch, Anders I., "Making argumentation serve design," *Human-Computer Interaction*, vol. 6, no. 3, pp. 267–293, 1996. [Online]. Available: http://portal.acm.org/citation.cfm?id=261723

[49] Fowler, M., "Design - Who needs an architect?" *Software, IEEE*, vol. 20, no. 5, pp. 11–13, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1231144

[50] Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns*. Addison-Wesley, 1996.

[51] Ghosh, T., "Creating incentives for knowledge sharing," MIT Sloan School, Tech. Rep., 2004.

[52] Goknil, Arda, Kurtev, Ivan, and van den Berg, Klaas, "Tool support for generation and validation of traces between requirements and architecture," in *ECMFA-TW '10: Proceedings of the 6th ECMFA Traceability Workshop*. New York, NY, USA: ACM, 2010, pp. 39–46. [Online]. Available: http://dx.doi.org/10.1145/1814392.1814398

[53] Gruber, Thomas R. and Russell, Daniel M., "Generative design rationale: Beyond the record and replay paradigm," in *Design Rationale*, Moran, T. and Carroll, J. H., Eds. Lawrence Erlbaum, 1992.

[54] Grudin, Jonathan, "Why CSCW applications fail: problems in the design and evaluationof organizational interfaces," in *CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work*.

New York, NY, USA: ACM, 1988, pp. 85–93. [Online]. Available: http://dx.doi.org/10.1145/62266.62273

[55] Grunbacher, P., Egyed, A., and Medvidovic, N., "Reconciling software requirements and architectures: the CBSP approach," in *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, 2001, pp. 202–211. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=948560

[56] Habli, Ibrahim and Kelly, Tim, "Capturing and Replaying Architectural Knowledge through Derivational Analogy," in *SHARK-ADI '07: Proceedings of the Second Workshop on SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 4+. [Online]. Available: http://dx.doi.org/10.1109/SHARK-ADI.2007.6

[57] Hagge, Lars and Lappe, Kathrin, "Sharing Requirements Engineering Experience Using Patterns," *IEEE Software*, vol. 22, no. 1, pp. 24–31, 2005. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MS.2005.17

[58] Hall, J. G., Jackson, M., Laney, R. C., Nuseibeh, B., and Rapanotti, L., "Relating software requirements and architectures using problem frames," in *RE '02: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 137–144. [Online]. Available: http://dx.doi.org/10.1109/ICRE.2002.1048516

[59] Hansen, M. T., Nohria, N., and Tierney, T., "What's your strategy for managing knowledge?" *Harvard Business Review*, vol. 77, no. 2, 1999.

[60] Harrison, Neil B., Avgeriou, Paris, and Zdun, Uwe, "Using Patterns to Capture Architectural Decisions," *IEEE Software*, vol. 24, no. 4, pp. 38–45, 2007. [Online]. Available: http://dx.doi.org/10.1109/MS.2007.124

[61] Herbsleb, J. D., Paulish, D. J., and Bass, M., "Global software development at Siemens: experience from nine projects," in *Proceedings of the 27th International Conference on Software Engineering*, 2005, pp. 524–533. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1553598

[62] Herrmannsdoerfer, Markus and Koegel, Maximilian, "Towards a generic operation recorder for model evolution," in *IWMCP '10: Proceedings of the 1st International Workshop on Model Comparison in Practice*. New York, NY, USA: ACM, 2010, pp. 76–81. [Online]. Available: http://dx.doi.org/10.1145/1826147.1826161

[63] Horner, John and Atwood, Michael E., "Design rationale: the rationale and the barriers," in *NordiCHI '06: Proceedings of the 4th Nordic conference on Human-computer interaction*. New York, NY, USA: ACM Press, 2006, pp. 341–350. [Online]. Available: http://dx.doi.org/10.1145/1182475.1182511

[64] Huff, S., "Information systems maintenance," *The Business Quarterly*, vol. 50, 1990.

[65] IEEE, "Recommended practice for architectural description of software-intensive systems (IEEE Std 1471-2000)," IEEE, Tech. Rep., 2000.

[66] IEEE and ISO/IEC, "Systems and software engineering - Recommended practice for architectural description of software-intensive systems (ISO/IEC 42010 IEEE Std 1471-2000)," ISO, Tech. Rep., July 2007. [Online]. Available: http://dx.doi.org/10.1109/IEEESTD.2007.386501

[67] Jansen, A. and Bosch, J., "Software Architecture as a Set of Architectural Design Decisions," in *Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on*. Washington, DC, USA: IEEE, 2005, pp. 109–120. [Online]. Available: http://dx.doi.org/10.1109/WICSA.2005.61

[68] Jansen, Anton, "Architectural design decisions," Ph.D. dissertation, University of Groeningen, August 2008. [Online]. Available: http://gradius.home.fmf.nl/research/thesis/Thesis-internet.pdf

[69] Jansen, Anton, de Vries, Tjaard, Avgeriou, Paris, and van Veelen, Martijn, "Sharing the Architectural Knowledge of Quantitative Analysis," in *Quality of Software Architectures. Models and Architectures*, 2008, pp. 220–234. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-87879-7_14

[70] Jansen, Anton, van der Ven, Jan, Avgeriou, Paris, and Hammer, Dieter K., "Tool Support for Architectural Decisions," *Software Architecture, Working IEEE/IFIP Conference on*, vol. 0, pp. 4+, 2007. [Online]. Available: http://dx.doi.org/10.1109/WICSA.2007.47

[71] Karsenty, Laurent, "An empirical evaluation of design rationale documents," in *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 1996, pp. 150–156. [Online]. Available: http://doi.acm.org/10.1145/238386.238462

[72] Kato, Yoshikiyo, Hori, Koichi, and Taketa, Kohei, "Capturing Design Rationale by Annotating E-mails," 2002. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.3291

[73] Kelly, T. P., "Arguing safety – a systematic approach to managing safety cases," Ph.D. dissertation, Department of Computer Science, University of York, 1999.

[74] Kruchten, Philippe, "The 4+1 View Model of Architecture," *IEEE Softw.*, vol. 12, no. 6, pp. 42–50, November 1995. [Online]. Available: http://dx.doi.org/10.1109/52.469759

[75] ——, "An Ontology of Architectural Design Decisions in Software Intensive Systems," in *2nd Groningen Workshop Software Variability*, October 2004, pp. 54–61.

[76] ——, "Documentation of Software Architecture from a Knowledge Management Perspective," in *Software Architecture Knowledge Management*, Ali Babar, Muhammad, Dingsoyr, Torgeir, Lago, Patricia, and Vliet, Hans, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, ch. 3, pp. 39–57. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02374-3_3

[77] Kruchten, Philippe, Capilla, Rafael, and J. C. Dueñas, "The Decision View's Role in Software Architecture Practice," *IEEE Softw.*, vol. 26, no. 2, pp. 36–42, 2009. [Online]. Available: http://dx.doi.org/10.1109/MS.2009.52

[78] Kruchten, Philippe, Lago, Patricia, and van Vliet, Hans, "Building Up and Reasoning About Architectural Knowledge," in *Quality of Software Architectures*, 2006, pp. 43–58. [Online]. Available: http://dx.doi.org/10.1007/11921998_8

[79] Kunz, W. and Rittel, H. W. J., "Issues as Elements of Information Systems," Univ. Calif. at Berkeley, Tech. Rep., 1970.

[80] Lago, Patricia and Avgeriou, Paris, "First workshop on sharing and reusing architectural knowledge," *SIGSOFT Softw. Eng. Notes*, vol. 31, no. 5, pp. 32–36, 2006. [Online]. Available: http://dx.doi.org/10.1145/1163514.1163526

[81] Lago, Patricia and van Vliet, Hans, "Explicit assumptions enrich architectural models," in *ICSE '05: Proceedings of the 27th international conference on Software engineering*. New York, NY, USA: ACM, 2005, pp. 206–214. [Online]. Available: http://dx.doi.org/10.1145/1062455.1062503

[82] Lee, J., "Design rationale systems: understanding the issues," *Expert, IEEE [see also IEEE Intelligent Systems and Their Applications]*, vol. 12, no. 3, pp. 78–85, 1997. [Online]. Available: http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=592267

[83] Lee, Jintae, "Decision representation language (DRL) and its support environment," MIT Artificial Intelligence Laboratory, Tech. Rep., August 1989. [Online]. Available: http://hdl.handle.net/1721.1/41499

[84] ——, "SIBYL: a tool for managing group design rationale," in *CSCW '90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work*. New York, NY, USA: ACM Press, 1990, pp. 79–92. [Online]. Available: http://dx.doi.org/10.1145/99332.99344

[85] Lee, Larix and Kruchten, Philippe, "Capturing Software Architectural Design Decisions," in *Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on*, 2007, pp. 686–689. [Online]. Available: http://dx.doi.org/10.1109/CCECE.2007.176

[86] Leveson, Nancy G., "Intent Specifications: An Approach to Building Human-Centered Specifications," *IEEE Transactions on Sw. Eng.*, vol. 26, no. 1, pp. 15–35, January 2000.

[87] Leveson, Nancy G. and Weiss, Kathryn A., "Making embedded software reuse practical and safe," in *SIGSOFT '04/FSE-12: Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering*. New York, NY, USA: ACM Press, 2004, pp. 171–178. [Online]. Available: http://dx.doi.org/10.1145/1029894.1029897

[88] Liang, Peng and Avgeriou, Paris, "Tools and Technologies for Architecture Knowledge Management," in *Software Architecture Knowledge Management* , 2009, ch. 6, pp. 91–111. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02374-3_6

[89] Liang, Peng, Jansen, Anton, and Avgeriou, Paris, "Sharing architecture knowledge through models: Quality and cost," *Knowl. Eng. Rev.*, vol. 24, no. 3, pp. 225–244, 2009. [Online]. Available: http://dx.doi.org/10.1017/S0269888909990038

[90] ——, "Collaborative Software Architecting through Knowledge Sharing," in *Collaborative Software Engineering*. Springer, 2010. [Online]. Available: http://www.cs.rug.nl/~{}liangp/download/liang2009csa.pdf

[91] MacLean, Allan, Young, Richard M., Bellotti, Victoria M. E., and Moran, Thomas P., "Questions, options, and criteria: elements of design space analysis," *Human-Computer Interaction*, vol. 6, no. 3, pp. 53–105, 1996. [Online]. Available: http://portal.acm.org/citation.cfm?id=261707

[92] Martin, Robert C., Riehle, Dirk, and Buschmann, Frank, *Pattern Languages of Program Design 3 (Software Patterns Series)*. Addison-Wesley Professional, October 1997. [Online]. Available: http://www.worldcat.org/isbn/0201310112

[93] McCall, R., "PHI: a conceptual foundation for design hypermedia," *Design Studies*, vol. 12, no. 1, pp. 30–41, January 1991. [Online]. Available: http://dx.doi.org/10.1016/0142-694X(91)90006-I

[94] McCall, Raymond J., Bennett, Patrick R., D'Oronzio, Peter S., Oswald, Jonathan L., Shipman, Frank M., and Wallace, Nathan F., "PHIDIAS: integrating CAD graphics into dynamic hypertext," in *ECHT 90*. New York, NY, USA: Cambridge University Press, 1992, pp. 152–165. [Online]. Available: http://portal.acm.org/citation.cfm?id=129667

[95] Moran, Thomas P. and Carroll, John M., Eds., *Design Rationale: Concepts, Techniques, and Use (Computers, Cognition, and Work)*, 1st ed. CRC, January 1996. [Online]. Available: http://www.worldcat.org/isbn/0805815678

[96] Myers, Karen L., Zumel, Nina B., and Garcia, Pablo, "Acquiring design rationale automatically," *Artif. Intell. Eng. Des. Anal. Manuf.*, vol. 14, no. 2, pp. 115–135, 2000. [Online]. Available: http://dx.doi.org/10.1017/S0890060400142027

[97] Navarro, Elena and Cuesta, Carlos, "Automating the Trace of Architectural Design Decisions and Rationales Using a MDD Approach," in *Software Architecture*, ser. Lecture Notes in Computer Science,

Morrison, Ron, Balasubramaniam, Dharini, and Falkner, Katrina, Eds. Berlin, Heidelberg: Springer Berlin / Heidelberg, 2008, vol. 5292, ch. 10, pp. 114–130–130. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88030-1_10

[98] Navarro, Israel, Lundqvist, Kristina, and Leveson, Nancy. An Intent Specification Model for a Robotic Software Control System. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.70.7351

[99] Neighbors, James M., "Software construction using components," Ph.D. dissertation, University of California, Irvine, 1980. [Online]. Available: http://portal.acm.org/citation.cfm?id=909597

[100] Nonaka, Ikujiro and Takeuchi, Hirotaka, *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, USA, May 1995. [Online]. Available: http://www.worldcat.org/isbn/0195092694

[101] OMG, "Meta Object Facility (MOF), v2.0," OMG, Tech. Rep., January 2006. [Online]. Available: http://www.omg.org/spec/MOF/2.0/

[102] ——, "Systems Modeling Language (SysML)," OMG, Tech. Rep., 2010. [Online]. Available: http://www.omgsysml.org/

[103] ——. (2010) Unified Modeling Language (UML). [Online]. Available: http://www.omg.com/uml/

[104] Paige, Richard F., "Traceability in model-driven safety critical software engineering," in *ECMFA-TW '10: Proceedings of the 6th ECMFA Traceability Workshop*. New York, NY, USA: ACM, 2010, p. 5. [Online]. Available: http://dx.doi.org/10.1145/1814392.1814393

[105] Pena-Mora, F. and Vadhavkar, S., "Augmenting design patterns with design rationale," *Artificial Intelligence Eng Des*, vol. 11, no. 2, pp. 93–108, 1997.

[106] Pilato, C. Michael, Collins-Sussman, Ben, and Fitzpatrick, Brian W., *Version Control with Subversion*, 1st ed. O'Reilly Media, June 2004. [Online]. Available: http://www.worldcat.org/isbn/0596004486

[107] Potts, C. and Bruns, G., "Recording the reasons for design decisions," in *ICSE '88: Proceedings of the 10th international conference on Software engineering*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1988, pp. 418–427. [Online]. Available: http://portal.acm.org/citation.cfm?id=55863

[108] Regli, W. C., Hu, X., Atwood, M., and Sun, W., "A Survey of Design Rationale Systems: Approaches, Representation, Capture and Retrieval," *Engineering with Computers*, vol. 16, no. 3 - 4, pp. 209–235, December 2000. [Online]. Available: http://dx.doi.org/10.1007/PL00013715

[109] Robillard, Pierre N., "The role of knowledge in software development," *Communications of the ACM*, vol. 42, no. 1, pp. 87–92, January 1999. [Online]. Available: http://dx.doi.org/10.1145/291469.291476

[110] Rozanski, Nick and Woods, Eoin, *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*. Addison-Wesley Professional, April 2005. [Online]. Available: http://www.worldcat.org/isbn/0321112296

[111] Saaty, Thomas L., "Decision making with the analytic hierarchy process," *International Journal of Services Sciences*, vol. 1, no. 1, pp. 83–98, January 2008. [Online]. Available: http://www.metapress.com/content/02t637305v6g65n8

[112] Schmidt, Douglas, Stal, Michael, Rohnert, Hans, and Buschmann, Frank, *Pattern-Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects*, 1st ed. John Wiley & Sons, September 2000. [Online]. Available: http://www.worldcat.org/isbn/0471606952

[113] Schneider, Kurt, "Rationale as a By-Product," in *Rationale Management in Software Engineering* , 2006, pp. 91–109. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30998-7_4

[114] Schuster, N., Zimmermann, O., and Pautasso, C., "ADkwik: Web 2.0 Collaboration System for Architectural Decision Engineering," in *SEKE*, 2007, pp. 255–260.

[115] Shahin, Mojtaba, Liang, Peng, and Khayyambashi, Mohammad R., "Architectural Design Decision: Existing Models and Tools," in *Proceedings of WICSA 2009*, 2009, pp. 293–296.

[116] Shipman, Frank M. and Marshall, Catherine C., "Formality Considered Harmful: Experiences, Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems," *Computer Supported Cooperative Work (CSCW)*, vol. 8, no. 4, pp. 333–352, December 1999. [Online]. Available: http://dx.doi.org/10.1023/A:1008716330212

[117] Shipman, Frank M. and McCall, Raymond, "Supporting knowledge-base evolution with incremental formalization," in *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 1994, pp. 285–291. [Online]. Available: http://dx.doi.org/10.1145/191666.191768

[118] Shum, S. J., "A Cognitive Analysis of Design Rationale Representation," Ph.D. dissertation, University of York, 1991. [Online]. Available: http://people.kmi.open.ac.uk/sbs/research/phd/phd.html

[119] Shum, Simon B., "Analyzing the Usability of a Design Rationale Notation," in *in Design Rationale: Concepts, Techniques, and*, 1996, pp. 185–215. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.52.5956

[120] Shum, Simon B. and Hammond, Nick, "Argumentation-based design rationale: what use at what cost?" *International Journal of Human-Computer Studies*, vol. 40, no. 4, pp. 603–652, April 1994. [Online]. Available: http://dx.doi.org/10.1006/ijhc.1994.1029

[121] Simon, Herbert A., *Administrative Behavior, 4th Edition*, 4 Sub ed. Free Press, March 1997. [Online]. Available: http://www.worldcat.org/isbn/0684835827

[122] Sinnema, Marco, van der Ven, Jan S., and Deelstra, Sybren, "Using variability modeling principles to capture architectural knowledge," *SIGSOFT Softw. Eng. Notes*, vol. 31, no. 5, p. 5, 2006. [Online]. Available: http://doi.acm.org/10.1145/1163514.1178645

[123] Tang, Antony, Babar, Muhammad A., Gorton, Ian, and Han, Jun, "A survey of architecture design rationale," *J. Syst. Softw.*, vol. 79, no. 12, pp. 1792–1804, December 2006. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2006.04.029

[124] Tang, Antony, Jin, Yan, and Han, Jun, "A rationale-based architecture model for design traceability and reasoning," *Journal of Systems and Software*, vol. 80, no. 6, pp. 918–934, June 2007. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2006.08.040

[125] Tang, Antony, Nicholson, Ann, Jin, Yan, and Han, Jun, "Using Bayesian belief networks for change impact analysis in architecture design," *Journal of Systems and Software*, vol. 80, no. 1, pp. 127–148, January 2007. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2006.04.004

[126] Tang, Antony, Tran, Minh, Han, Jun, and van Vliet, Hans, "Design Reasoning Improves Software Design Quality," in *Quality of Software Architectures. Models and Architectures* , ser. Lecture Notes in Computer Science, 2008, ch. 2, pp. 28–42. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-87879-7_2

[127] Tang, Antony and van Vliet, Hans, "Modeling constraints improves software architecture design reasoning." in *WICSA/ECSA*. IEEE, 2009, pp. 253–256. [Online]. Available: http://dblp.uni-trier.de/db/conf/wicsa/wicsa2009.html#TangV09

[128] Tang, Antony and Vliet, Hans, "Software Architecture Design Reasoning," in *Software Architecture Knowledge Management* , 2009, ch. 9, pp. 155–174. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02374-3_9

[129] Thurimella, Anil K., Bruegge, Bernd, and Creighton, Oliver, "Identifying and Exploiting the Similarities between Rationale Management and Variability Management," in *SPLC '08: Proceedings of the 2008 12th International Software Product Line Conference*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 99–108. [Online]. Available: http://dx.doi.org/10.1109/SPLC.2008.14

[130] F. K. E. P. Törner, "On Hazard Identification and Safety Cases In the Automotive Domain," Ph.D. dissertation, Chalmers University of Technology, 2008.

[131] Toulmin, Stephen E., *The Uses of Argument*. Cambridge University Press, January 1958. [Online]. Available: http://www.worldcat.org/isbn/0521092302

[132] Tyree, Jeff and Akerman, Art, "Architecture Decisions: Demystifying Architecture," *IEEE Software*, vol. 22, no. 2, pp. 19–27, 2005. [Online]. Available: http://dx.doi.org/10.1109/MS.2005.27

[133] Vesperman, Jennifer, *Essential CVS*, 1st ed. O'Reilly Media, June 2003. [Online]. Available: http://www.worldcat.org/isbn/0596004591

[134] Vetschera, Rudolf, "Preference-Based Decision Support in Software Engineering," in *Value-Based Software Engineering* , 2006, ch. 4, pp. 67–89. [Online]. Available: http://dx.doi.org/10.1007/3-540-29263-2_4

[135] Vliet, Hans, Avgeriou, Paris, Boer, Remco C., Clerc, Viktor, Farenhorst, Rik, Jansen, Anton, and Lago, Patricia, "The GRIFFIN Project: Lessons Learned," in *Software Architecture Knowledge Management*, 2009, ch. 8, pp. 137–154. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02374-3_8

[136] Weiss, Kathryn A., Ong, Elwin C., and Leveson, Nancy G., "Reusable specification components for model-driven development," in *In Proceedings of the International Conference on System Engineering, INCOSE*, 2003. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.5.7488

[137] Wu, Weihang and Kelly, Tim, "Managing Architectural Design Decisions for Safety-Critical Software Systems," in *Quality of Software Architectures* , ser. Lecture Notes in Computer Science, Hofmeister, Christine, Crnkovic, Ivica, and Reussner, Ralf, Eds. Springer Berlin Heidelberg, 2006, vol. 4214, ch. 9, pp. 59–77. [Online]. Available: http://dx.doi.org/10.1007/11921998_9

[138] Xin, Wang and Guangleng, Xiong, "Design rationale as part of corporate technical memory," in *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, vol. 3, August 2002, pp. 1904–1908 vol.3. [Online]. Available: http://dx.doi.org/10.1109/ICSMC.2001.973625

[139] Zdun, Uwe and Avgeriou, Paris, "A catalog of architectural primitives for modeling architectural patterns," *Inf. Softw. Technol.*, vol. 50, no. 9-10, pp. 1003–1034, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2007.09.003

[140] Zdun, Uwe, Avgeriou, Paris, Hentrich, Carsten, and Dustdar, Schahram, "Architecting as decision making with patterns and primitives," in *SHARK '08: Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge.* New York, NY, USA: ACM, 2008, pp. 11–18. [Online]. Available: http://dx.doi.org/10.1145/1370062.1370066

[141] Zelkowitz, Marvin, *Principles of Software Engineering and Design (Prentice-Hall software series).* Prentice Hall, 1979. [Online]. Available: http://www.worldcat.org/isbn/013710202X

[142] Zimmermann, Olaf, "An Architectural Decision Modeling Framework for Service-Oriented Architecture Design," Ph.D. dissertation, Universität Stuttgart, March 2009.

[143] Zimmermann, Olaf, Gschwind, Thomas, J. Küster, Leymann, Frank, and Schuster, Nelly, "Reusable Architectural Decision Models for Enterprise Application Development," in *Software Architectures, Components, and Applications* , 2007, pp. 15–32. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-77619-2_2

[144] Zimmermann, Olaf, Zdun, Uwe, Gschwind, Thomas, and Leymann, Frank, "Combining Pattern Languages and Reusable Architectural Decision Models into a Comprehensive and Comprehensible Design Method," *Software Architecture, Working IEEE/IFIP Conference on*, vol. 0, pp. 157–166, February 2008. [Online]. Available: http://dx.doi.org/10.1109/WICSA.2008.19