Towards Enabling Design Rationale Capture by Asking the Right Questions

Mathias Schubanz

Institut für Informatik, Informations- und Medientechnik
BTU Cottbus - Senftenberg
Postfach 10 13 44
03013 Cottbus
M.Schubanz@b-tu.de

Abstract: Over the last thirty years many research has been conducted to capture the ”how” and ”why” behind design decisions. This information is known as design rationales (DR). Approaches to capture, store, preserve, and use DR have emerged from research activities. However, as of today they only found exceptional application within industrial practice. Rationales have been analysed in respect to its nature, its structure, and its quality. Additionally, some researchers performed analyses on them. They found out that DR have the potential to sustainably contribute to design, re-design, testing, maintenance, and improving product quality over the whole product-life cycle. Within this paper a research proposal is presented striving to tackle the exceptional use of DR in software documentation. We want to promote the capture of DR by paying more attention on the questions ”What information has to be captured by rationales?”, ”How detailed should the captured information be?”, and ”How should rationale capture be integrated into the development process?”. It is the goal to promote the use of DR within the whole software development process. In short, within this paper we introduce the topic of DR, related research and elaborate on the intended research on the topic of DR capture.

1 Introduction

Design rationales document the reasoning behind given actions and decisions. They provide an answer for the ”how” and ”why” which is often asked in relation to decisions made. Therefore, they play an important role in decision-making. For instance, when one needs to decide on one out of a set of available alternatives, they provide insight to the arguments, the criteria considered, and potential consequences. Rationales also document the assumptions behind decisions which often, if not explicitly asked for, stay implicit in the heads of those involved. They are substantially connected to argumentation and sometimes even are used as a synonym for argumentation [FLMM91]. Hence, capturing DR avoids tacit knowledge to remain in the stakeholders’ heads.

The concept of documenting the decision-making process and its connected rationales did not originate in the software engineering domain. It has initially been applied in urban architecture and the policy domain [BCMM08]. In the late 80’s several approaches to document group meetings and argumentation within the field of human-computer interaction design were published [CMW96, CR91, CY91, MYM89]. Based on the corresponding focus of capturing rationales of software and interface designs, the term *design rationale* (*DR*) got well established. Thus, *DR* and *rationale* are treated as a synonym from here on.

Kunz and Rittel [KR70] were one of the first to approach the field of DR. They shaped rationales as *the reasons behind existing decisions*. According to MacLean et al. [MYBM91] proper rationales also include the design space. In other words, design rationales also consider how a given artefact is located in the space of existing design options. Rationales also play an important role in software architecture documentation. Correspondingly, Tang and Han [TH05] defined *Architecture Rationale* to record the reasons of requirements enhancements and its design rationales. However, considering all available definitions of DR would go beyond the scope of this paper. Following, we refer to the definition of MacLean et al. [MYBM91] when talking about DR.

DR are said to support multiple software engineering activities as, for instance, design, re-design, test, and maintenance [DMMP06]. Nevertheless, DR has been employed only exceptionally in industrial practice yet [Bur08]. There is a clear disparity between the its potential and the use of DR. Therefore, this paper presents a research proposal which is intended to result in a DR capture framework providing concrete guidance to developers when capturing rationales. This framework is expected to contain a set of reference questions grouped into templates. These questions shall be answered to capture rationales during software development. The framework is further expected to contain guidelines supporting the integration of rationale capture into existing software development processes. It shall be rounded out by a reference implementation of DR capture in a development process.

The goal of the dissertation is to resolve the uncertainty connected to DR capture (see Section 4) by providing a reference framework which guides developers when capturing DR. The work shall provide answers for the following research questions:

RQ-1: What information has to be captured by rationales?

RQ-2: How detailed should the captured information be?

RQ-3: How should rationale capture be integrated into the development process?

The remainder of the paper is structured as follows: Section 2 motivates the usage of DR. Within Section 3 related work with a focus on software engineering is presented. Section 4 discusses research gaps related to the successful application of DR. Finally, a research outline including previous work is presented in Section 5.

2 Motivation

The essential capability of DR is to reflect the intention behind a decision. One of the most important application scenarios in this respect is the usage of DR as a means of communication. As such they have initially been used in urban architecture and in the political domain [BWMK09]. In the latter, the *IBIS* approach [KR70] by Kunz and Rittel played an important role. According to Carrol and Rosson [CR91] DR *build a pertinent understanding* of the context, the users, the tasks, the technologies, and the situations considered. This is of particular interest in the software engineering domain as there is a variety of stakeholders participating in, for instance, requirements engineering and design. This ranges from developers, who intend to understand the work of their colleagues, up to the technical sales and distribution department. They use DR to reduce the complexity contained in a software design. In other words, DR essentially *serve the perception and understanding of a given design* [BCMM08, Cas96, CY91, DMMP06, KLVV06, MYBM91, Sag12, SH94]. Conklin and Yakemovic [CY91] underpin these assumptions by a case study. At the NCR corporation they introduced a DR management system in an industrial setting for the scope of a particular project. The authors report several positive effects which can be attributed to captured rationales. At the same time they also came across several typical rationale management problems. For instance, participants did from time to time not capture DR properly. This could be attributed to a diverse perception of *what rationales are* and *when they need to be captured*. Correspondingly, Conklin and Yakemovic report in their case-study description that at one point in the project a particular feature was planned to be realised by the development team without a concrete reason attached to it. The responsible software engineer expected everybody else in the team to be familiar with the reasons for the feature request. In fact, nobody else but him knew, that this particular feature had to be introduced due to a number of customer complaints.

A positive conclusion from the case study is that DR support *knowledge preservation*. They contribute to preserve information in relation to an existing software artefact, like a component. Imagine the following: the lack of information in the previous example would not have been discovered. Even worse, the responsible software engineer would have left the company. At that point the corresponding design knowledge would have been lost. Documenting DR could prevent this. Within the same case study [CY91] the authors report a successful case of knowledge preservation. Prior to the completion of a detailed software design one of the centrally involved software engineers left the company. Nevertheless, on the basis of the DR documentation the remaining team did manage to finish the detailed design without difficulty.

Tang and Han [TH05] claim that DR also play a central role for *architecture verification*. Hence, DR support software designers when they check whether the architecture model is complete, the rationale of the design decisions is sound, and the design can satisfy the given system requirements. This way errors could be revealed. Analogously, during the mentioned case study [CY91] the authors manually converted the documentation repository including its DR from the *itIBIS* to the *gIBIS* format. This conversion revealed *11* additional design problems, whereof *7* most likely would not have been discovered until code would have been written.

DR can also be applied to support several other capabilities. For instance, DR could *support reasoning* on a given set of alternatives [MYBM91]. Hence, DR also *facilitate release planning* [Ruh11]. Alternatively, DR also can *facilitate decisions within a variant space* [TB12]. A detailed overview on capabilities of DR outlined in the literature within the last 25 years is provided by Sagoo [Sag12].

Despite of the various application scenarios in which DR could contribute, they have not found much application in industrial practice yet. Known applications in the domain of software engineering are mainly based on research activities performing case studies or concept evaluations. They have found more application in the field of aviation industries [BWMK09]. There are several barriers to a successful DR capture as, for instance, the high effort to capture DR, maintain the information, as well as no clear understanding of how they could be handled. Within the set of problems a central role could be attributed to the uncertainty in relation to benefits of DR and a lack of certainty on what information is valuable up to which granularity. More discussion on this topic can be found in Section 4.

3 Related Work

This section will present related work in the field of DR management. The first part discusses existing approaches to represent DR. Section 3.2 provides a brief overview on DR in relation to software architectures. Finally, Section 3.3 will briefly review approaches focussing on the DR capture process.

3.1 Design Rationale Models

Probably the most important approach in the field of rationale management dates back to 1970. The IBIS approach [KR70] defined a structure to be used for documenting issues discussed in, for instance, group meetings. It has been widely used and often adopted within proceeding research activities. Several other also similar approaches have been proposed in the late 80's and early 90's. One of these is the *Questions-Options-Criteria (QOC)* approach by MacLean et al. [MYBM91]. They altered the IBIS approach by adding relevant criteria as well as the opportunity to evaluate arguments. This is why the authors claim to support a design space exploration. Lee [Lee89] proposed another important approach providing an expressive grammar to capture DR in structured way. With its seven node types and fifteen link types it contains a high complexity [BD04].

On the basis of the previously mentioned approaches several similar models and corresponding tools have been developed. For instance, the IBIS derivatives *itIBIS* [CY91], *gIBIS* [CY91], *rIBIS* [RE91], *PHI* [FLMM91], *DRed* [BWMK09], and *QAR* [HOK14] were proposed. Some derivatives of QOC are *DREAM+TEAM* [LP07], *IVMM* [TB12], *EvoPL* [SPBL12], and *RUSE* [Wol08]. A derivative of DRL is *RATspeak* [BB08].

3.2 Software Architecture Documentation

Among the numerous approaches to DR some had a dedicated focus on software architecture. For instance, Tang and Han [TH05] proposed the *Architecture Rationalization Method (ARM)* which is based on *Architecture Rationale*. The *Architecture Tradeoff Analysis Method (ATAM)* by Kazman et al. [KKC00] evaluates multiple software architectures against a set of defined quality goals. Its output is intended to develop a set of analyses, rationales, and guidelines. Asundi et al. [AKK01] extended *ATAM* by the *Cost Benefit Analysis Method (CBAM)* which links costs and business goals to architecture decision-making. Bass et al. [BCNS06] propose to capture DR for a software architecture differently. They use to two distinct graphs, a *structural graph* and a *causal graph*.

Another aspect which relates to DR is the documentation of software architectures and the connected reasoning. Here, several models have been proposed. *Tyree's Template* by Tyree and Akerman [TA05], *Archium* by Jansen and Bosch [JB05], the *Architecture Design Decision Support System Model* by Capilla et al. [CND07], or *AREL* by Tang et al. [THV09] are just some to name in this respect.

3.3 Design Rationale Capture

For a long time *DR capture* did not receive much attention. The DR schema was the point research focussed on. Over time it became clear that the schema is not the only key to success. Hence, corresponding tools and methodologies emerged from research activities. One of the first approaches was the *graphical IBIS (gIBIS)* approach [CY91]. It tried to mitigate existing barriers to DR capture by graphical tool support. A similar approach called *MIKROPLIS* and later *PHIDIAS* was presented by McCall et al. [MBd$^+$90]. They used the PHI approach which is based on IBIS [FLMM91]. Its essential advantage is that DR could be created by reusing already captured information from previous projects.

Shipman et al. [SIM99] went another way by proposing to capture DR without a schema. The approach captures all information on a 2D space and structures the results according to its spatial arrangement. Similarly, Reeves et al. [RS92] annotate artefact designs with textual notes within a CAD system. Thus, DR are annotated directly to the artefact design. Myers [MZG99] modified a CAD system by augmenting the symbol library with semantic information. Hence, DR is captured based on the symbol usage. The main goal of the previous approaches is to reduce the process disruptiveness by avoiding a DR schema.

On a rather abstract level Schneider [Sch06] approached the problem by proposing guidelines to DR capture which need to be turned into concrete operational practices, techniques, and/or rules. Apart from the underlying DR schema Bass et al. [BCNS06] also propose guidelines to DR capture. Their goal is to guide architects during DR capture. In contrast thereto, Casaday [Cas96] provides a set of eight templates residing on a more detailed abstraction level. In a more recent work Durdik and Reussner [DR13] capture design decisions related to the use of patterns by using a question catalogue.

4 Research Gaps

As outlined in Section 3 a considerable set of research has been conducted on the topic of DR. However, there still remains much uncertainty [Bur08] on DR usage. DR capture is accompanied by a contradiction. While there is no shortage of advocates for the value of

DR, there also is no shortage of reasons why DR are unlikely to be captured [Bur08]. There is a big uncertainty whether the payoff will justify the costs, how DR should be maintained, how it will we used, and most important, what needs to be captured. This uncertainty hindered the successful integration of DR into industrial practice. For now, there are two known cases to be mentioned when talking about successful DR capture. As outlined in Section 2 the first one is the successful application of DR during the KDS project in the NCR corporation [CY91] on the basis of the *gIBIS tool*. The second case is the successful application of the *Design Rationale Editor (DRed)* within Rolls Royce [BWMK09]. What initially started as a research project ended with a company-wide rollout of the *DRed* approach.

When comparing the lack of industrial DR usage to its occasional success the following question arises: *Why did DR not find industrial application on a broad scale yet?* Kruchten et al. [KLVV06] simply answer this by stating the following: *This is mostly because the burden to capture assumptions and decisions outweighs largely the immediate benefits that the architect may draw.* Correspondingly, one of the main reasons is the uncertainty on the usage of DR. Burge [Bur08] outlined this in three questions:

- How useful are the potential benefits of DR?
- How insurmountable are barriers to DR usage?
- Will the value of DR justify the cost?

Within future research these questions need to be answered to overcome the *barriers to DR usage*. A first contribution in this area has been proposed by Horner and Atwood [HA06]. They categorised the *barriers to DR usage* into *cognitive*, *capture*, *retrieval*, and *usage limitations*. *Cognitive limitations* consider limitations connected to human information processing. *Capture limitations* address the capture of information required to place DR in the right context, incentives to DR capture, the elicitation of tacit knowledge, political factors (Does the captured DR pose a risk to the developer or the company?), as well as the trade-off between cost and benefits of DR capture. *Retrieval limitations* are concerned about the information which needs to be captured and which is expected to be useful. Finally, the DR capture might constrained by *usage limitations* as, for instance, the uniqueness of design problems. Considering a development scenario where the design problems always differ substantially, DR re-use is an undesirable application scenario.

Another frequently discussed issue in connection to the uncertainty of DR capture is the process intrusiveness [DMMP06, Sch06]. Correspondingly, DR capture is considered to break the workflow, slow down the work progress and reduce the developers' motivation. A reduced motivation could also be caused once architects need to capture DR which reveal that they did not act as they were supposed to [DMMP06]. For instance, they decided to prefer a solution alternative which is connected to less development effort but contains a higher risk of failure. Another reason which often hinders DR capture is a lack of project resources and available time [DMMP06]. When things need to be cut due to a lack of time DR does not have the appropriate priority to be maintained. This is also due to a lack of immediate pay-off compared to the required effort [FLMM91]. Usually, the point in time when DR unfold their benefit is posterior to the time of capture [CY91].

Apart from the contribution proposed by Horner and Atwood, there is little discussion on the problems outlined above. For now, DR research mainly focussed on DR schemes and corresponding tooling. At least as important as the schema / tool combination is an investigation on the expectations and needs of practitioners. These are a key factor to overcome the *barriers to DR usage*. Thus, this area should gain more focus in DR research. Otherwise it will remain as it has done over the years: nice ideas, but not practical [KLVV06]. First contributions have been made here by Burge [Bur08] and Tang et al. [TBGH06]. Both conducted a survey on the needs and expectations of practitioners to DR usage. They found out that practitioners consider DR as important. Additionally, practitioners criticise a lack of methodology and tool support. However, there barely is empirical work in DR research [Bur08, TBGH06]. More effort needs to be invested here.

5 Project Outline

Considering the elaborations in the previous chapters *Design Rationale (DR)* do seem to be an important means of software documentation. They capture the knowledge behind design decisions and, therefore, capture a key asset of a company's value [Lie02]. However, as of today to the best of our knowledge DR are barely used in industry in general and in software engineering in particular. Potential reasons to that have been outlined in Section 4. To overcome exactly these research gaps an essential starting point is to provide practitioners with concrete guidance on what information is relevant to be captured. Accordingly, the research questions *RQ1* to *RQ3* have been set up (cf. Section 1). To answer these questions it is intended to develop a DR capture reference framework. It is expected to contain (1) a set of concrete questions to be answered by DR, (2) guidelines to support the integration of DR capture into existing software development processes, and (3) a reference implementation of a process for a DR capture system. The questions and guidelines are intended to be grouped together into a set of reference templates. Based on a development domain, the granularity level of documentation, and the development conditions (development team, development process, development resources) a developer will need to tailor the resulting DR capture reference templates to his/her concrete development scenario. Additional guidance will be provided by the reference implementation.

5.1 Previous Work

In our previous work we augmented an approach to model *software product line evolution (SPLE)*, called *EvoPL* [PBD$^+$11], with the basic concepts from decision-making and rationale modelling. Based on the *QOC* model ([MYBM91] , see Section 3) we support the capture of goals, requirements, decision alternatives, relevant criteria, as well as explicit DR (see [SPBL12]). Further, we provided a first IDE-integrated tool support on prototype level. In [SPP$^+$13] we evaluated our model from [SPBL12]. To do so, we analysed the *SWT* project, a sub-project of the *Eclipse Project* [DRW04]. We retrospectively analysed the documentation of their coarse grained plans called *Themes and Priorities* tracing actions down to fine-grained commit comments. The results showed, that the introduced modelling concepts (cf. [SPBL12]) haven reflected by analysed data. However, we were not able to systematically reconstruct explicit DR information.

Based on experiences from this research and the research gaps discussed in Section 4 the research focus has been slightly adjusted. One of the central problems which we intend to address now are the *barriers of DR usage*. As a first step to approach this problem in [SPJB14] we conducted a literature survey looking for questions to be answered by DR. Additionally, we started to work on a combination of rationale capture guidelines and the questions from [SPJB14] for the requirements engineering domain. It is intended to propose a first approach on questions-based rationale capture guidance.

5.2 Research Outline

The results from the literature survey in [SPJB14] and current research will serve as a first step to set up an initial set of DR questions and guidelines. However, the work needs to be extended. A research outline continuing this work is listed in the following steps:

1. The literature survey on questions to be answered by DR (see Section 4) needs to be enlarged to a *Systematic Literature Review* [KC07]. The goal is to gain a better understanding on the expectations outlined in rationale research.
2. The information found in the literature needs to be related to information on the expectations and needs to DR usage from the practitioner's point of view.
3. Based on the results from *step one* and *two* an initial set of question templates has to be set up. These shall serve as the basis for first guidelines to capture rationales.
4. The results from *step three* need to be tailored to a subset of domains and developments processes. Moreover, a reference implementation has to be developed.
5. The results from *step four* need to be refined in cooperation with practitioners. Finally, the usability of the defined of the DR capture framework needs to be empirically evaluated in an industrial project.

In the *first step* we want to focus on one of the central capabilities awarded to DR. They are expected to answer a given set of questions [BCNS06, BCMM08, DMMP06]. This concept can also be found in the QOC [MYBM91] and the QAR [HOK14] approach. "Why was this decision made?", "How is requirement X satisfied?", or "What are the implications of making modification Y?" are just some examples. Some of the central questions, as those just mentioned, are outlined in the literature again and again. Others, only appear seldom. Hence, an essential starting point to focus the research on is to determine the information which is required to be captured by DR. To do so, it is intended to perform a *Structured Literature Review* [KC07] on the availability of questions to be answered by DR in the corresponding literature. As outlined in Section 5.1 a first literature review has been performed already. However, this has several limitations. It needs to be enlarged to a *Structured Literature Review* considering more background information on the questions found. Thus, it is important to consider the context the questions were used in. They might originate from a theoretical background used for, e.g., motivation purposes in research papers. Alternatively, they could originate from a practical background as, e.g., they emerged during a case study. The results of the literature review need to be analysed in detail. Potential research questions for the literature review could be the following:

RQ-A What questions to be answered by DR are outlined in the literature?

RQ-B Which background do the questions found originate from?

RQ-C What differences can be found based on the origin of the questions?

RQ-D How have the questions found been evaluated?

RQ-E What kind of question-based guidance for DR capture has been proposed yet?

Within the *second step* we intend to prepare questionnaires and perform interviews with practitioners having different experiences and working in different domains. In detail, we want to communicate with practitioners (1) still being in education (e.g. students), (2) working in open source development projects, and (3) having industrial background. We intend to interview them using online questionnaires and/or live interviews. Therefore, we intend to find an answer to the following questions:

- What do practitioners expect of using DR? (in terms of effort, benefits, and the trade-off amongst them)
- What are the barriers to DR usage for practitioners?
- Which are the most important ones to overcome?
- What information do practitioners require when analysing, maintaining respectively evolving existing (legacy) software?
- What is the most relevant information for practitioners to be captured by DR?

In addition to the results from *step two*, we intend to analyse existing open source development projects for its documentation structure and DR usage. Based on the unstructured, decentralised development processes as well as the different information backgrounds of the developers, we expect that DR plays an important role to improve the communication within open source development projects. We want to exhibit if any DR related information is captured. If so, what information does the DR documentation reveal about DR capture, DR usage and the project itself? Furthermore, it is of interest to us, what information is often required by developers analysing existing software, performing maintenance operations, or evolving existing software.

In *step three* we want to produce initial set of question templates based on the results from *step one* and *two*. The resulting templates shall serve as the basis for a first version of guidelines to capture rationales. However, as we expect the that the questions found will reside on a rather general level we need to refine these.

The *fourth step* is intended to review the results from *step one* and *two* in respect to the domains the practitioners are working in. We think that there is no *one fits it all approach*. Based on the research proposed above, we will not be able to provide guidance for all development scenarios. Hence, the intended reference framework needs to be tailored to a subset of (1) domains, (2) team structures, and (3) development processes employed. This shall be done in collaboration with industry partners. If this is not done carefully, DR questions will remain on an abstract level. Hence, they would merely serve as a guideline for DR capture (similar to [BCNS06]) instead of concrete guidance to DR capture. To ensure the usability and the functional relevance of the intended outcome we intend to closely collaborate with practitioners from an industrial background. Software engineering in regulated domains, including certification-oriented processes seems to be a promising candidate. The need for careful documentation is already well established and first successful industry cases exist (e.g. [BWMK09]). We also got into contact with first practitioners from the automotive domain. Step *four* will be concluded by the creation of a reference implementation of a process for a DR capture system.

Within *step five* we refine the results from *steps one* to *four*. Again, we strive to have this activity to be supported by an industry collaboration from the automotive industry. An empirical evaluation within the automotive domain (see *step four*) on the usability of the outcomes shall be performed. The setting and scope of the evaluation is not clear yet.

Finally, within the thesis it is intended to provide tool-support for the ideas from *steps one* to *five*. The DR capture framework should be accompanied by an automated configuration wizard to increase its usability. Moreover, it is intended to develop an IDE-integrated tool support, similar to the one contained in SEURAT [BB08]. DR should be captured within the development environment depending on the concrete development process and usage scenario.

References

[AKK01] Jayatirtha Asundi, Rick Kazman, and Mark Klein. Using economic considerations to choose among architecture design alternatives. Technical report, DTIC Document, 2001.

[BB08] Janet E. Burge and David C. Brown. Software Engineering Using RATionale. *Journal of Systems and Software*, 81(3):395–413, 2008.

[BCMM08] Janet E Burge, John M Carroll, Raymond McCall, and Ivan Mistrk. *Rationale-Based Software Engineering*. Springer Publishing Company, Incorporated, 2008.

[BCNS06] Len Bass, Paul Clements, Robert L Nord, and Judith A Stafford. Capturing and Using Rationale for a Software Architecture. In *Rationale Management in Software Engineering*, pages 255–272. Springer, 2006.

[BD04] Bernd Bruegge and Allen H Dutoit. *Object-Oriented Software Engineering Using UML, Patterns and Java-(Required)*. Prentice Hall, 2004.

[Bur08] Janet E Burge. Design Rationale: Researching Under Uncertainty. *AI EDAM*, 22(4):311, 2008.

[BWMK09] Rob Bracewell, Ken Wallace, Michael Moss, and David Knott. Capturing design rationale. *Computer-Aided Design*, 41(3):173 – 186, 2009.

[Cas96] George Casaday. Rationale in Practice: Templates for Capturing and Applying Design Experience. In *Design Rationale: Concepts, Techniques, and Use*, pages 351–372. Lawrence Erlbaum Associates, 1996.

[CMW96] Tom Carey, Diane McKerlie, and James Wilson. HCI Design Rationale as a Learning Resource. In *Design Rationale: Concepts, Techniques, and Use*, pages 373–392. Lawrence Erlbaum Associates, 1996.

[CND07] Rafael Capilla, Francisco Nava, and Juan C Duenas. Modeling and documenting the evolution of architectural design decisions. In *Proceedings of the Second Workshop on*

SHAring and Reusing architectural Knowledge Architecture, Rationale, and Design Intent, page 9. IEEE Computer Society, 2007.

[CR91] John M Carroll and Mary Beth Rosson. Deliberated evolution: Stalking the View Matcher in design space. *Human–Computer Interaction*, 6(3-4):281–318, 1991.

[CY91] E. Jeffrey Conklin and K. C. Burgess Yakemovic. A Process-Oriented Approach to Design Rationale. *Human–Computer Interaction*, 6:357–391, Sept. 1991.

[DMMP06] Allen H. Dutoit, Raymond McCall, Ivan Mistrik, and Barbara Paech. *Rationale Management in Software Engineering*. Springer-Verlag, New York, 2006.

[DR13] Zoya Durdik and Ralf Reussner. On the Appropriate Rationale for Using Design Patterns and Pattern Documentation. In *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures*, QoSA '13, pages 107–116, New York, NY, USA, 2013. ACM.

[DRW04] J. Des Rivières and J. Wiegand. Eclipse: a platform for integrating development tools. *IBM Syst. J.*, 43:371–383, April 2004.

[FLMM91] Gerhard Fischer, Andreas C. Lemke, Raymond McCall, and Anders I. Morch. Making Argumentation Serve Design. *Hum.-Comput. Interact.*, 6:393–419, September 1991.

[HA06] John Horner and Michael E Atwood. Effective Design Rationale: Understanding the Barriers. In *Rationale Management in Software Engineering*, pages 73–90. Springer, 2006.

[HOK14] Alireza Haghighatkhah and Harri Oinas-Kukkonen. An Argumentation-Based Design Rationale Application For Reflective Practice. In *Proceedings of the 22nd European Conference on Information Systems (ECIS)*, 2014.

[JB05] Anton Jansen and Jan Bosch. Software architecture as a set of architectural design decisions. In *Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on*, pages 109–120. IEEE, 2005.

[KC07] Barbara A. Kitchenham and S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE-2007-01, Keele University, 2007.

[KKC00] Rick Kazman, Mark Klein, and Paul Clements. ATAM: Method for Architecture Evaluation. Technical report, CMU / Software Engineering Institute, 2000.

[KLVV06] Philippe Kruchten, Patricia Lago, and Hans Van Vliet. Building up and reasoning about architectural knowledge. In *Quality of Software Architectures*, pages 43–58. Springer, 2006.

[KR70] Werner Kunz and Horst W. J. Rittel. Issues as elements of information systems. Technical report, Systemforschung, Heidelberg, Germany Science Design, University of California, Berkeley, 1970.

[Lee89] Jintae Lee. Decision Representation Language (DRL) and Its Support Environment. MIT Artificial Intelligence Laboratory Working Papers WP-325, MIT Artificial Intelligence Laboratory, August 1989.

[Lie02] Jay Liebowitz. A look at NASA Goddard Space Flight Center's knowledge management initiatives. *IEEE software*, 19(3):40–42, 2002.

[LP07] Xavier Lacaze and Philippe Palanque. DREAM & TEAM: A Tool and a Notation Supporting Exploration of Options and Traceability of Choices for Safety Critical Interactive Systems. In *Human-Computer Interaction – INTERACT 2007*, volume 4663, pages 525–540. Springer Berlin Heidelberg, 2007.

[MBd$^+$90] Raymond McCall, Patrick R Bennett, Peter S d'Oronzio, Jonathan L Ostwald, Frank M Shipman III, and Nathan Wallace. PHIDIAS: Integrating CAD Graphics into Dynamic Hypertext. In *ECHT*, volume 90, pages 152–165, 1990.

[MYBM91] Allan MacLean, Richard M. Young, Victoria M. E. Bellotti, and Thomas P. Moran. Questions, Options, and Criteria: Elements of Design Space Analysis. *Hum.-Comput. Interact.*, 6:201–250, September 1991.

[MYM89] Allan MacLean, Richard M Young, and Thomas P Moran. Design Rationale: the Argument behind the Artifact. In *ACM SIGCHI Bulletin*, volume 20, pages 247–252. ACM, 1989.

[MZG99] Karen L Myers, Nina B Zumel, and Pablo Garcia. Automated capture of rationale for the detailed design process. In *AAAI/IAAI*, pages 876–883, 1999.

[PBD$^+$11] Andreas Pleuss, Goetz Botterweck, Deepak Dhungana, Andreas Polzer, and Stefan Kowalewski. Model-driven Support for Product Line Evolution on Feature Level (accepted for publication). *Journal of Systems and Software (JSS) - Special Issue on "Automated Software Evolution"*, 2011. http://soft.vub.ac.be/iwpse-evol/specialissue.

[RE91] Gail L Rein and Clarence A Ellis. rIBIS: a real-time group hypertext system. *International Journal of Man-Machine Studies*, 34(3):349–367, 1991.

[RS92] Brent Reeves and Frank Shipman. Supporting communication between designers with artifact-centered evolving information spaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 394–401. ACM, 1992.

[Ruh11] Günther Ruhe. *Product Release Planning: Methods, Tools and Applications*. Auerbach Publications, 2011.

[Sag12] Jeevan Sagoo. *Design rationale for the regulatory approval of medical devices*. PhD thesis, Cranfield University, 2012.

[Sch06] Kurt Schneider. Rationale as a By-Product. In *Rationale Management in Software Engineering*, pages 91–109. Springer, 2006.

[SH94] Simon Buckingham Shum and Nick Hammond. Argumentation-Based Design Rationale: What Use at What Cost. *International Journal of Human-Computer Studies*, 40(4):603–652, 1994.

[SIM99] Frank M Shipman III and Catherine C Marshall. Spatial hypertext: an alternative to navigational and semantic links. *ACM Computing Surveys*, 31:14, 1999.

[SPBL12] Mathias Schubanz, Andreas Pleuss, Goetz Botterweck, and Claus Lewerentz. Modeling Rationale over Time to support Product Line Evolution Planning. In *Proceedings of VaMoS'12*, pages 193–199, Leipzig, Germany, 2012. ACM.

[SPJB14] Mathias Schubanz, Andreas Pleuss, Howell Jordan, and Goetz Botterweck. Guidance for Design Rationale Capture to Support Software Evolution. In *Workshop on Software-Reengineering & Evolution*, Bad Honnef, 2014.

[SPP$^+$13] Mathias Schubanz, Andreas Pleuss, Ligaj Pradhan, Goetz Botterweck, and Anil Kumar Thurimella. Model-driven planning and monitoring of long-term software product line evolution. In *Proceedings of VaMoS'13*, pages 103–107, Pisa, Italy, 2013. ACM.

[TA05] Jeff Tyree and Art Akerman. Architecture decisions: Demystifying architecture. *IEEE software*, 22(2):19–27, 2005.

[TB12] Anil Kumar Thurimella and Bernd Bruegge. Issue-based variability management. *Information and Software Technology*, 54(9):933–950, 2012.

[TBGH06] Antony Tang, Muhammad Ali Babar, Ian Gorton, and Jun Han. A Survey of Architecture Design Rationale. *Journal of Systems and Software*, 79(12):1792–1804, 2006.

[TH05] Antony Tang and Jun Han. Architecture Rationalization: A Methodology for Architecture Verifiability, Traceability and Completeness. In *ECBS*, pages 135–144, 2005.

[THV09] A. Tang, J. Han, and R. Vasa. Software Architecture Design Reasoning: A Case for Improved Methodology Support. *IEEE Software*, 2009(March/April):43–49, 2009.

[Wol08] Timo Wolf. *Rationale-based Unified Software Engineering Model.* VDM Verlag, Saarbrücken, Germany, 2008.