

What Matters to Students – A Rationale Management Case Study in Agile Software Development

Mathias Schubanz

Software and Systems Research Group
Brandenburg University of Technology
Cottbus, Germany
M.Schubanz@b-tu.de

Claus Lewerentz

Software and Systems Research Group
Brandenburg University of Technology
Cottbus, Germany
Claus.Lewerentz@b-tu.de

Abstract—Documenting design decisions and their rationale (Design Rationale, DR) in software development projects is vital for supporting the comprehension of the product, product quality, and future maintenance. Although an increasing number of research publications address this topic, systematic approaches and supporting DR tools are found very rarely in practice. In software engineering education, DR is usually not well covered in teaching. The lack of suitable decision documentation is mainly an issue in agile software development. In agile approaches, documentation is regarded as less important than working products. To explore possibilities for integrating decision documentation into Scrum processes for educational software development projects, we conducted a series of eight case studies. These were part of software lab courses in three universities, i.e., BTU Cottbus, PUT Poznan, University of Stuttgart, with about 400 participants in 82 project teams. We introduced additional process elements in Scrum and developed a lightweight capture technique to support the decision capture.

This paper describes the case study setup and corresponding implementation and, thus, an example approach of managing rationale in Scrum. Additionally, it presents a data analysis of the students' most relevant decisions documented throughout the case studies. We conclude the paper with a discussion on the observations we made during the case study executions and the applicability of the approach in educational software projects.

Index Terms—rationale management, agile software development, scrum, teaching, case study, decision types, design decision

I. MOTIVATION

During a software development project, the members of the project team make many decisions. These decisions concern technical issues on all levels of detail as the software architecture or the selection of particular implementation platform technology as well as organizational aspects as the prioritization of requirements or the use of specific development tools.

The overall set of decisions profoundly influences the quality properties of the developed product as well as the efficiency and effectiveness of the development process itself. Explicit and conscious handling of decisions together with their rationale (i.e., decision alternatives, selection criteria, and reasoning) and observed consequences is vital for the comprehension and sustainable maintenance processes for software products [7] [14]. The management of decisions and

so-called *design rationale* (DR) allow for traceability and comprehension of typical “why”-questions on the products and processes. Furthermore, decision management creates an essential opportunity for reflection and learning.

The relevance of DR for software developers was examined and confirmed several times, for example, by Tang et al. [46]. In their study, software engineers confirmed that they capture DR and consider this to be relevant in their work processes. Despite the many supporting arguments and the advocates for the documentation and use of DR, sustaining and managing decisions and their rationale is said to be applied seldom in practice [2], [6]. Its intangible nature is just one of many driving factors behind this contradiction. Another is the lack of adequate process integration and tool support for handling rationale information [30]. Furthermore, the structured and systematic handling of decisions is only dealt with very selectively in teaching, as described in Kleebaum et al. [23].

The explicit management and documentation of decisions seem particularly crucial in *agile software development* (ASD). The very concept of agility promotes frequent independent decision making by the single developer or the development team. At the same time, ASD deprioritizes documentation in favor of working software (cf. Agile Manifesto [3]).

Moreover, ASD promotes efficiency in knowledge transfer through direct informal face-to-face communication [5], [10], [15], [35]. In addition to the fact that developers, in general, are reluctant to document decisions, especially when it is unclear which ones are to be documented [1], they understand ASD in particular as the liberation from recording any information at all [44]. Sometimes developers also tend to refer to source code as the only real documentation artifact [38], [39], [44]. However, even well-structured and readable code only covers the “What?”, not the “Why?” and especially not the “Why not?”.

As part of an ongoing research project [42] with a focus that is now tailored to ASD, we try to empirically analyze the described area of conflict with the help of *grounded theory* [9] and propose corresponding solutions. In order to do so, we have observed the work of students and conducted eight case

studies, where they were required to document their most important decisions actively. These case studies were carried out within the scope of software engineering lab projects in the computer science courses of three universities, i.e., *Brandenburg University of Technology Cottbus - Senftenberg* (BTU), *Poznan University of Technology* (PUT), and the *University of Stuttgart* (US). We aimed to use the experience and data from the case studies to answer the following three research questions (RQ) and at the same time, achieve the following learning objective (LO):

- RQ1** What are suitable ways to introduce decision capturing and reflection in an agile development process?
 - RQ2** Which types of decisions are crucial in agile educational software development projects?
 - RQ3** To what extent do students capture decision alternatives and their rationale?
-
- LO1** Raise the awareness of students for decision making and rationale management in software engineering.

The remainder of the paper is structured as follows: [Section II](#) discusses related work in the field of rationale management in general and in an educational context. [Section III](#) presents the case study setup and details on its execution. [Section IV](#) elaborates on our results with respect to our research questions and the learning objective. Subsequently, we discuss these findings and some conclusions in [Section V](#). We conclude by describing potential future work in [Section VI](#).

II. RELATED WORK

In this section, we discuss related work. [Section II-A](#) elaborates on the topic of rationale management in the context of software engineering. [Section II-B](#) focusses on rationale management in ASD. Finally, [Section II-C](#) examines related work on teaching rationale management.

A. Rationale Management in Software Engineering

Early research on rationale management dates back to the 1970s and aimed at the use in politics (cf. Kunz and Rittel [28]). The ideas were quickly translated into first practical projects. Conklin and Yakemovic [11] reported on the systematic capture and use in various engineering disciplines which already took place in the 1980s. A few years later, research in the field of *Human-Computer-Interaction* (HCI) recognized the relevance of managing rationale, and various contributions appeared (cf. Fischer et al. [16], Lee and Lai [32], MacLean et al. [34]). These then gradually served as a basis for research on rationale management in software engineering. Much of this research identified the decision making process as a central point for improvement. Accordingly, many approaches propose models, tools, and methods for documenting decisions. To name just a few, these include the *Architecture Rationalization Method* (ARM) by Tang and Han [47] which attaches rationale to software architecture with the help of fine-grained model elements, an extendable meta-model facilitating the RUSE model by Wolf [52] which integrates system models,

collaboration models and organizational models with the help of a tool called Sysiphus, or the *Architecture Tradeoff Analysis Method* (ATAM) by Kazman et al. [22] which evaluates an architecture against a set of defined quality goals to derive analyses and rationale.

Later work focused primarily on overcoming the obstacle of capturing decisions. According to Burge and Brown [7], the problem to be solved is that many possible applications of rationale do not come into effect because the methods and tools are not in place to support these opportunities. Approaches that try to mitigate this obstacle include Ishino and Jin [21] or Myers et al. [36] that integrated decision capture directly into CAD tools and thus into the working environment of the engineers. The same idea with a closer relationship to software development was implemented in *DecDec* tool by Hesse et al. [17], which is directly integrated into the Eclipse IDE [13] and thus into the toolchain of a developer. As a documentation tool for decision knowledge, it is geared towards collaborative and incremental decision making processes. Similarly, Helaba [33] offers a joint workspace to support communication around design artifacts and activities in multidisciplinary teams. There are many other tools available to assist in capturing decisions. An overview of the area of software architecture documentation in this context is provided, for example, by Tang et al. [45].

B. Rationale Management in Agile Software Development

Based on previous work, research presented many contributions on the topic of documenting decisions and their rationale in the context of the increasingly popular ASD. Among other aspects, they present new decision modeling approaches for ASD in general and Scrum in particular. For instance, Waagenaar et al. [50] propose a concrete Scrum artifact model or Wang et al. [51] elaborate on safety-related documentation by introducing *safety-epics* and *safety-stories*.

Other papers also focus on identifying best practices for documentation in ASD. For instance, Hoda et al. [19] present a structured approach to experience-based patterns in agile documentation. With five concrete patterns, they strive to enable developers to create exactly the right amount of documentation. Rüping presents another much more comprehensive pattern approach in his book on agile documentation [40]. Based on experience and concrete cases from previous projects, Rüping derives documentation patterns tailored to several problems in ASD.

Researchers also developed ASD-specific tools to capture rationale. *Echo* [31], for instance, facilitates agile requirements gathering under consideration of the relationship to decisions and their rationale. Voigt et al. [49] also proposed *sprintDoc*, which introduces developing documentation artifacts (wiki pages). Due to the existing history, changes to issues are directly linked to changes in the documentation and thus traceable. Other approaches even employ machine-learning techniques to extract undocumented design decisions (cf. Bhat et al. [4]).

C. Rationale Management in Teaching

Despite the many contributions from research addressing the trade-off between agile principles and rationale management, there are so far only a few contributions in the area of integrating decision documentation into agile process models. In practice, developers seem not to have a high awareness of the importance of DR and DR is yet very rarely included in software engineering education. The literature describes only a few approaches. For instance, Kleebaum et al. [23] integrate rationale management in the context of a software development project both in the lecture and in practical development tasks. Lago et al. [29] go a similar way here. In the context of a software architecture lecture, the topic of software architecture decision making is dealt with and then deepened by a specially designed card game, called *DecidArch*. Other approaches include a card game developed by Schriek et al. [41] who intend to prompt the students to consider design elements more intensively or De Boer et al. [12] who teach students how to elicit, communicate, and document architecture design decisions.

III. CASE STUDY DESIGN

In this section, we present the case study design. Initially, Section III-A gives a general overview of how the case study was carried out. We then describe the pilot study that was conducted in advance and its findings (Section III-B). Section III-C explains how we have incorporated these findings into the case study setup. Subsequently, Section III-D describes the data collection procedure. A description of the analysis procedure is provided in Section III-E, followed by elaborations on the validation procedure in Section III-F.

A. Case Study Overview

Between 2015 and 2019, we conducted eight case studies with computer science students from the *Brandenburg University of Technology Cottbus - Senftenberg* (BTU), *Technical University Poznan* (PUT), and the *University of Stuttgart* (US). We integrated the case studies into mandatory software development projects as part of the Bachelor's or Master's study programs.

The tasks of small student teams were to develop comprehensive software products during the lecturing period of one semester, i.e., about 15 weeks. The products were mostly interactive web or desktop applications, e.g., online versions of popular board games. The expected work effort for a project was between 500 h (PUT) and 1.000 h (BTU, US), thus 100-200 h per person.









In addition to the development task, the students attended complimentary lectures that discussed software engineering techniques. The lectures did neither specifically address decision making techniques nor the topic of rationale management.

All projects used the Scrum process, which served as the common agile organizational structure for the case study. The typical project team consisted of four to five undergraduate students (BA program) working as developers. The roles of *Product Owner* (PO) and *Scrum Master* (SM) were typically

taken over by graduate students (MA program). Occasionally not enough graduate students were available. In these cases, the SM was taken over by the undergraduate students or by the study organizers. In the latter case, the study organizers, however, did not get involved in the selection of important decisions.

After an initial setup and exploration phase, the teams worked in two-week (BTU) or four-week (PUT, US) development sprints. In total, the case studies included 82 development teams with about 350 undergraduate developers and 50 graduate students (cf. Table I). In this context, it must be noted that the graduate students often worked with several teams simultaneously, for example, as PO for one team and as SM for another team.

TABLE I
OVERVIEW OF CONDUCTED CASTE STUDIES.

University	Semester	Teams
 Brandenburgische Technische Universität Cottbus - Senftenberg	BTU 2015 Winter	2
 PUT	PUT 2016 Summer	27
 Brandenburgische Technische Universität Cottbus - Senftenberg	BTU 2016 Winter	5
 PUT	PUT 2017 Summer	31
 University of Stuttgart Germany	US 2017 Summer	1
 Brandenburgische Technische Universität Cottbus - Senftenberg	BTU 2017 Winter	4
 Brandenburgische Technische Universität Cottbus - Senftenberg	BTU 2018 Winter	6
 Brandenburgische Technische Universität Cottbus - Senftenberg	BTU 2019 Winter	6 ¹
		82

B. Pilot Case Study

According to research question *RQ1*, the main objective was to integrate the capture and documentation of decisions into a Scrum process. It was one of our main goals to change the textbook version of the Scrum process only as much as necessary because we had to assume that most students in that early phase of their studies used the Scrum process for the first time. Furthermore, too many additional process elements contradict the first agile principle (cf. agile manifesto [3]).

To meet these requirements, we conducted a pilot case study over one semester with two teams at the BTU in advance of the case studies listed in Table I. Thus, in the first two sprints, we conducted workshops together with the students. During these workshops, we discussed issues being significant to the students, elicited alternatives, made decisions, and documented them.

Based on the experiences from the interaction among the students, the interaction between the students and the teaching staff as well as the dynamics of the decision making, we identified the need for the following extensions to the Scrum process, called *Extension Requirements* (ER) in the following:

- ER1** The team has to have a clear vision of who will execute tasks related to the case study. Thus, one team member is responsible for recording the decisions.
- ER2** The case study setup has to embed this responsibility in clearly defined process elements that provide a context and time frame within the sprint.

¹At the time of publication, the case study was still ongoing.

- ER3** The quality of recorded decisions needs to be assured through corresponding measures to enable rationale reuse, as discussed by Thurimella et al. [48].
- ER4** Capturing decisions must be regulated in a way that certain documentation templates and tools are to be used. Additionally, the case that a team might not agree on a set of decisions to capture has to be addressed as well.

C. Case Study Setup

To address the ERs listed above and enable a systematic approach to capture decisions, we decided to extend the textbook Scrum by two process elements, as shown in Figure 1.

Decision Capture At the end of each sprint planning (ER2) the Scrum Master (ER1) is responsible for identifying and documenting the three most relevant decisions for the current sprint together with the team. Once recorded, these have to be uploaded to the project repository.

Decision Review At the end of the sprint, during the retrospective (ER1), the Scrum Master (ER2) is responsible for reviewing the documented decisions together with the team (ER3). In case of changes, inconsistencies, or recently emerged important decisions, the Scrum Master has to revise the documentation. (ER3).

In addition to these extensions, a case study guideline was provided to students. It contained additional instructions on how to carry out the identification and selection of the most relevant decisions if there is no immediate consent (ER4). The guidelines also contained the requirements for the use of documentation templates and the corresponding tools (ER4). For more information, please refer to Section III-D.

Furthermore, a *git* [8] repository was made available to the students, which provided the documentation templates (ER4), as well as an extensive collection of sample documentation (ER4). However, there were no instructions on what types of decisions should be documented in order not to restrict them in their thinking and decision making.

D. Data Collection – From Mindmaps to Markdown Records

The data collection process varied throughout the different parts of the case studies. Apart from the pilot study, it was carried out by the students over the entire period. The capture technique, as well as the associated tools and templates for recording decisions, were repeatedly examined for their usability and improved throughout the pilot case study and the case studies.

Initially, during the pilot case study, we have chosen lightweight tool support, i.e., *Text2Mindmap* [20], to record decisions. For this purpose, we used the *Questions Options Criteria* (QOC) model [34] model and mapped it using Text-ToMindmap. By this means, the case study organizers captured the decisions made during the architecture workshops of the pilot study. After the workshops, the generated mind maps

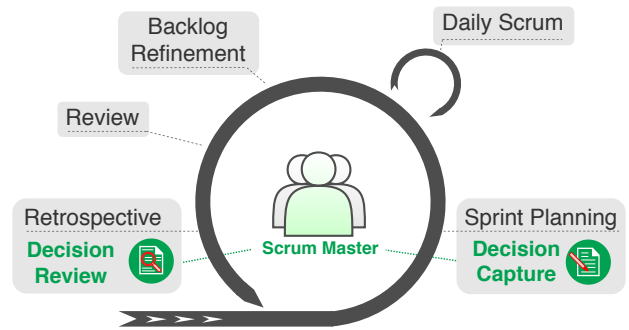


Fig. 1. Altered Sprint Procedure.

were converted into images using screenshots and uploaded to the students' project repositories.

This approach showed insufficient usability and transferred too much responsibility to the case study organizers. Additionally, the generated mindmaps did not show optimal readability. Subsequent changes due to, for instance, changed requirements or feedback could not be incorporated. We needed to either recreate the documentation entirely or save the raw data along with the results to regenerate the mindmap. We also found that students rarely documented their decisions when none of the case study organizers were attending the sprint planning. All of these issues were reasons for the authors to refine the capture technique for the planned case studies.

Autonomy in Decision Capture:

With the goal of more active engagement of the students, the authors decided to give the students more responsibility and thus to increase the autonomy in the documentation of decisions. It was necessary to choose another handling of the templates. For the first three case studies in 2015 (BTU) and 2016 (BTU, PUT), Excel templates were compiled based on the previous QOC-based [34] modeling and distributed to the students. This way, the responsibility lay solely with the team, respectively, the Scrum Master, as envisioned in the case study setup in Section III-C. Additionally, sprint planning meetings no longer had to be accompanied by the case study organizers for decision documentation. This change required the students to manage decisions consciously and independently and not to let this happen as a passive process triggered by a third party. According to the case study setup (cf. Section III-C) the Scrum Master was responsible for uploading the completed documentation to the project repository at the end of the sprint.

MADR – Markdown Architecture Decision Records:

Based on the feedback and experience of the 2016 case studies as well as a simultaneously conducted industry case study, we refined the handling of templates in the following case studies. A crucial point of criticism was the manageability. Working with Excel files proved to be impractical and cumbersome. Thus, it was perceived as too time-consuming to perform reviews of the captured decisions by the team or revise the documented decisions. Accordingly, the case study organizers sought alternatives in dialogue with the students and the industry partner. The choice fell on the use of Markdown (MD) files. MD is a lightweight markup syntax, which makes it easy

to build documents [37]. Since both students and industry partners version their development projects with *git* [8] and its management frontend *GitLab* [18], the use of MD also offered direct integration into the tool environment, since *GitLab* renders MD files in the web interface nicely.

Besides, the management of MD files in *git*, combined with a corresponding branching strategy, enables collaborative work with short feedback cycles. Since we already used *git* in the case study, it seemed reasonable to make the MD templates available in a public *GitHub* repository (cf. [43]).

The revised methodology, including the new templates, was supposed to be implemented in the first case study of 2017 (PUT), but could not be implemented for organizational reasons. The recording using Excel templates was applied once again. During the second case study in 2017 with the University of Stuttgart (US), the students used the new methodology. Recording vital decisions and their rationale with the help of MD files attracted such interest that our partners from the University of Stuttgart have forked and further refined the original *GitHub* project. The result is a new *open-source* (OS) project that was not only used in the second case study in 2017, which is an OS project itself (cf. [26]) but has reached into the OS community. Currently, more than ten completely independent OSS projects employ *Markdown Architecture Decision Records* (MADR) [25] to capture their decisions. Besides, other developers from the community have worked on the MADR *GitHub* project (cf. [24]).

Subsequently, we used MADR for the following case studies in 2017, 2018, and 2019 (BTU). Thanks, among other things, to the comprehensive instructions, there was hardly any negative feedback from the students.

E. Data Analysis – Coding and Classification

To answer research question RQ2, we searched the literature for a comprehensive classification of decision types, e.g., Kruchten [27]. However, in our opinion, the classifications found did not provide structures clear enough to be intuitive and intelligible for a data analysis like this. Accordingly, we used an inductive approach and coded the data. Each of the authors did this individually. Subsequently, we tried to derive a consistent categorization from the codes, which resulted in several discussions and involved several iterations. Discrepancies in the two result sets had to be identified and also resolved on a case-by-case basis. After further iterations, we refined the codes into a simple and straightforward classification. The results can be found in Section IV-C.

F. Validation Procedures

We have taken various measures to address potential threats to validity. Concerning internal validity, for example, we did not get involved in the selection of decisions to be documented. The authors explicitly pointed out to the students that the case study will not affect the evaluation of the lecture. Another aspect of internal validity was the integration of free-text answers into the follow-up survey. This way, students could provide individual answers if they considered the predefined

answers to be inappropriate. Furthermore, we tried to reduce the individual influence of the authors during data analysis by coding the data separately by each author. Only afterward, the results were compared and standardized to a uniform categorization.

Concerning threats to external validity, one needs to be aware that the software development labs and their associated lectures had an impact (as discussed in Section V). It was hardly possible to mitigate this influence. The limited project duration of only a few weeks and the steep learning curve also affected the external validity. Nevertheless, the free choice of development topics coupled with the high number of participating groups and the resulting large amount of documented decisions should be favorable to external validity.

IV. RESULT EVALUATION

In this section, we try to give answers to the research questions and the learning objective outlined in the Motivation section based on the results of the case studies. As a basis for this discussion, we surveyed the students after completing the case study. The survey comprised three predefined questions (survey questions *SQ1* to *SQ3*) and one open question for individual feedback. Participation was voluntary. The survey resulted in a response of 56 answers. Considering all of the students who already completed the case study by the time of publication, the response rate is about 18%.

Based on the results of this follow-up survey, Section IV-A elaborates on the achievement of learning objective *LO1*. Subsequently, Section IV-B answers research question *RQ1* as far as possible and reports on the students' perspective in which Scrum phase decisions have to be documented. In Section IV-C, we then report on our findings on the types of decisions captured by the students and thus try to answer *RQ2*. Section IV-D concludes with a discussion on the scope and nature of the information collected for a decision, i.e., research question *RQ3*.

A. *LO1 – Raise the Awareness for Rationale Management*

To check the achievement of our learning goal, we asked the students to rate the usefulness of the conducted case study. The question had a dedicated focus on the communication within the team, respectively, with stakeholders (survey question *SQ1*, cf. Figure 2): As can be seen, it turned out that roughly equal parts of the students *strongly agree* or *agree* on the usefulness of decision documentation ($\sim 37\%$), respectively *disagree* or *disagree strongly* ($\sim 43\%$).

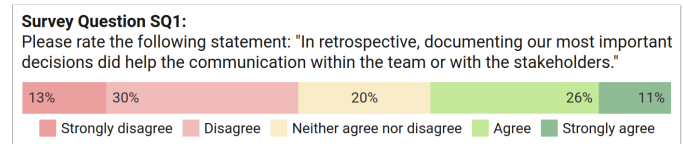


Fig. 2. Results for *SQ1*: usefulness of documented decisions, in percent.

Besides, we asked students whether they reused documented decisions (survey question *SQ2*, cf. Figure 3) and if so, for what purpose. Here about 38% denied, and a further 14%

replied that they used the decisions merely as a template for recording further decisions. The remaining 48% of the students opposed this by stating an explicit use case. These included

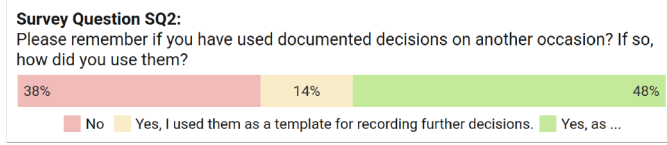


Fig. 3. Results for SQ2: reuse of captured decisions, in percent.

the following scenarios:

- A review was conducted.
- A wrong decision was made and had to be reconsidered.
- A bug had to be fixed.
- A similar problem had to be decided on.
- One team member needed advice.
- The requirements had changed.

The above answers indicate that the participating students have become aware of the topic and its importance. Several students also confirmed this impression in an open question on feedback on the case study. For example, one student wrote: “*captured decisions help you to understand the problem thoroughly and [...] have a better picture of the project and its structure so that you can develop better software.*”

B. RQ1 – Suitability of the Chosen Approach

As described earlier, building on initial ideas and the results of the pilot case study, we decided on an approach to integrate rationale management into Scrum (cf. Section III). Accordingly, for research question RQ1, we can only discuss the applicability of our specific approach. As the answers to questions SQ1 and SQ2 show, there is no uniform opinion among the participating students. However, a considerable share of the students considers the decision capture approach to be helpful and applicable. In this context, it was vital for us to find out how the participants in the case study assessed the timing of the decision capture. Accordingly, we have asked students to indicate phases in which they consider documentation of decisions to be most appropriate (*survey question SQ3*, cf. Figure 4). Several answers could be given.

Again, there is no uniform opinion on the part of the students. However, a clear majority (59%) voted for capturing decisions during the sprint planning. The second most votes came with 43% and 39% for review and retrospective, the phases at the end of a sprint. Still, just over a quarter of the respondents (27%) were in favor of capturing decisions while working on the backlog. The least preferred approach is decision capture during planning poker (21%). We explicitly neglect that 23% of the students who argue for capturing decisions during the Daily Scrum because it would conceptually contradict the Daily Scrum.

The three most frequently mentioned responses also reflect our observations during the execution of the case studies. Although the case study setup clearly states that the most important decisions should be captured during sprint planning, the students occasionally documented them at various other

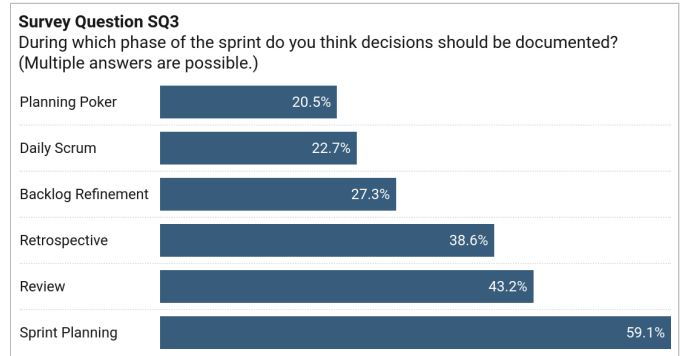


Fig. 4. Results for SQ3: phases to capture decision in, in percent.

points throughout the sprint. Reasons given by the students include the fact that it was just forgotten and was made up for at the end of the sprint. Sometimes it also happened that the vital decisions only emerged in the course of the sprint. Correspondingly, the integration of the decision capture into sprint planning is a possible and preferred option, although it is not the only feasible option for process integration.

Still, the authors, like the students, consider the integration of the decision capture into the sprint planning to be the preferred way, since working with the recorded and thus communicated decisions during the sprint is thereby made possible.

C. RQ2 – What Matters to Students?

Based on the classification derived from the coding, Table II lists 702 currently recorded decisions grouped by the corresponding categories. Decisions that we could not attribute to any of the available classification types are listed under N/A. With about 2% and 12 decisions respectively, this includes decisions that were not clear enough as well as decisions in which the students deliberately wrote nonsense. Accordingly, a successful mapping of more than 98% of available decisions to the categorization appears sufficiently accurate for us and confirms the appropriateness of the chosen classification.

TABLE II
AGGREGATED NUMBER OF CAPTURED DECISIONS PER TYPE.

Decision type	Count	Percent
Definition / refinement of features / requirements	133	19
Development libraries / frameworks	100	14.3
Software architecture	97	13.8
Development process	89	12.7
Software quality measures (testing, etc.)	67	9.5
Prioritization of tasks / features	66	9.4
Development tools	62	8.8
Development platform	49	7
Deployment (Other in Fig. 5)	10	1.4
Communication within the team (Other in Fig. 5)	10	1.4
To-Do decisions (Other in Fig. 5)	7	1
N/A (Other in Fig. 5)	12	1.7
	702	

The resulting distribution of decision types within all captured decisions shows that there is not a single relevant

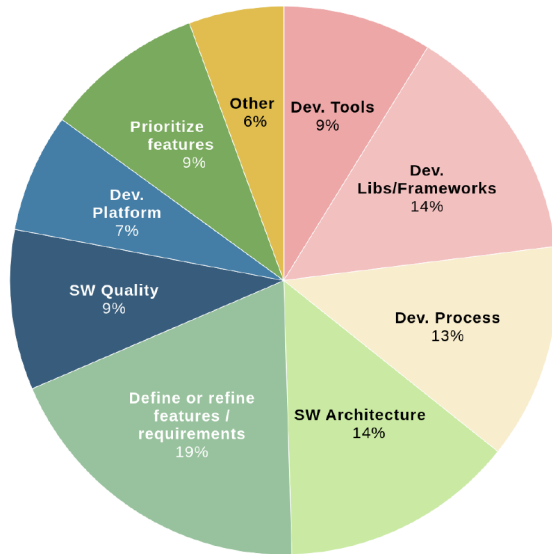


Fig. 5. Distribution of decision types over all case studies.²

decision type, but rather a group of important types. As can be seen in Figure 5, the five most frequently documented decision types account for more than two-thirds of all captured decisions ($\sim 69\%$). Combining the seven most documented decision types even accounts for nearly 88% of all decisions. On closer inspection of these seven, one can see that the most frequently documented decision type “*Definition/refinement of features/requirements*” is closely related to the “*Prioritization of tasks/features*.” Topics related to requirements analysis and its prioritization together represent a considerable part of all documented decisions with roughly 29%. A second thematic group also stands out on closer inspection, solving implementation problems relating to software architecture and those relating to the use of frameworks and libraries. Frequently, they go hand in hand anyway. These two together account for about 28% of all decisions and thus have a considerable influence on software development processes.

Even more striking is the evolution of documented decisions over time. In Figure 7, we have broken down the different types of decisions by the sprints from which they originate. Upon mere observation, considerable differences in the distribution of the types of documented decisions are noticeable. In the first sprints, for example, one can find an above-average number of decisions on the development tools, the libraries/frameworks used, and especially the development platform. These become considerably less over time. The decisions addressing the development platform, which in the beginning make up almost 20%, then disappear almost completely. Equally noteworthy is the progress of decisions on software quality. They play a subordinate role at the beginning and then gain considerably in importance throughout the sprints. Especially in the third sprint, the topic of software quality plays a vital role, with a share of 25% in all documented decisions. Subsequently, relevance decreases again. Just as important is the development of the decisions on features/requirements as well as on the software architecture.

These two together account for a little less than 20% in the first sprint, while it is about 40% and more from the fourth sprint onwards.

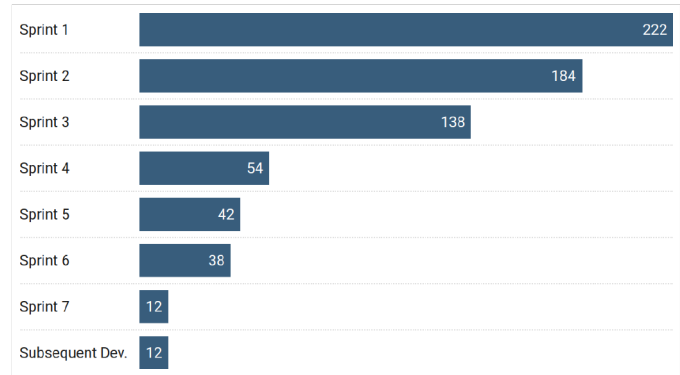


Fig. 6. Aggregated number of captured decisions per sprint, absolute.

For reading the graph, it is necessary to know that the projects have lasted a maximum of seven sprints. The development of the open-source project, to which the students of the University of Stuttgart contributed (cf. Section III-D), was continued afterward and resulted in further documented decisions. These are listed in the “Subsequent Dev.” column.

Furthermore, it is necessary to read the graph in the context of the total number of captured decisions per sprint. As one can see from Figure 6, this has changed permanently. Although the students were required to write down the three most important decisions, they sometimes recorded less or none at all. In individual cases, they even recorded more. In the first sprint, the number of documented decisions was still very high ($avg = 2.7$ decisions), but decreased afterward. During the third sprint, the students recorded significantly less ($avg = 1.7$ decisions). As the development projects at the PUT lasted only three sprints (cf. Section III), from the fourth sprint on only the teams of the US and the BTU documented their decisions (24 teams in total). In the following, the amount of documented decisions is somewhat more stable again.

It should be noted that for sprint five and six, there is no data from the 2019 case study of the BTU available at the time of publication. Accordingly, only 18 teams documented in sprints five and six. In particular, this means that in the fourth and fifth sprint, an average of about 2.3 decisions per team was documented, and in the sixth sprint, an average of 2.1 decisions per team was documented. For the seventh sprint (4 teams) as well as the subsequent development (1 team), the number of documented decisions increased again. However, each of these two, with 12 decisions each, accounted for just under two percent of the decisions recorded and thus have little significance.

In summary, the trend analysis also provides interesting insights for answering the research question RQ2. However, there is no simple answer to this question. Besides, the character of the projects, in which the case studies took place,

²All decision types with less than 2% share are subsumed under *Other*.

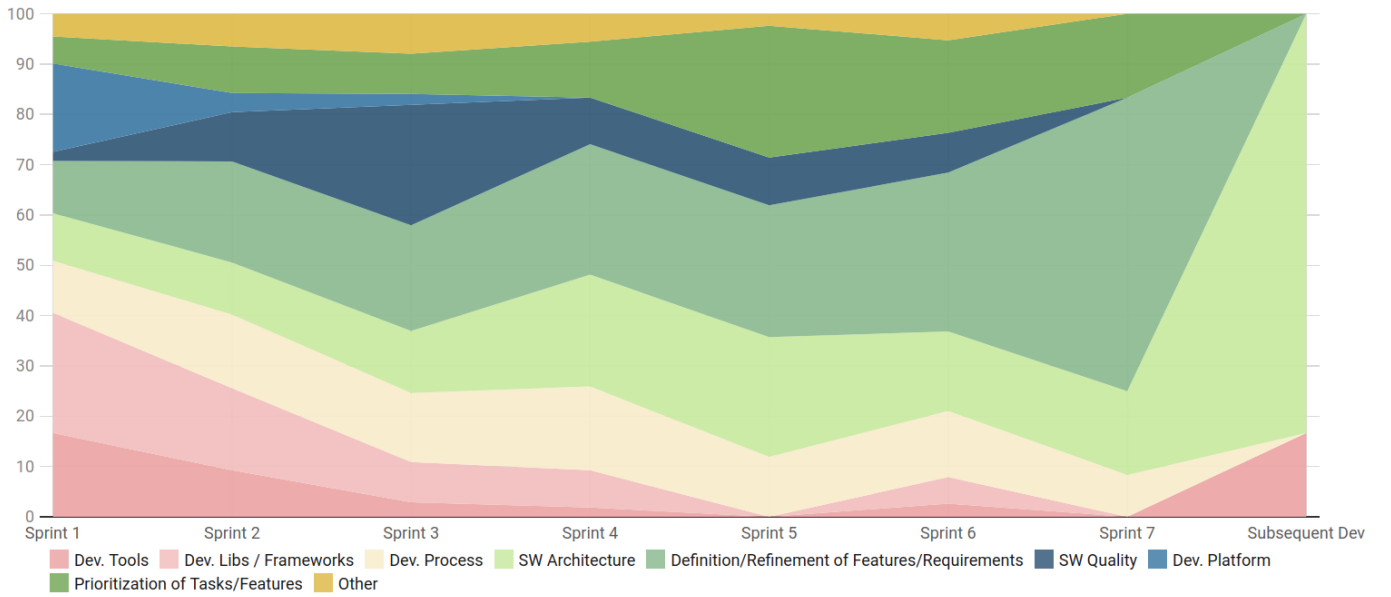


Fig. 7. Relative distribution of decision types over all case studies per sprint.

influences the results too. All projects started from scratch and did have a rather brief lifespan.

D. RQ3 – What Information has been Documented?

Another aspect that has been of interest to us while exercising the case studies is the quantity of information students capture for a decision. Specifically, we are interested in whether students only document the issue itself and the decision to solve or whether they also record additional information such as the alternatives considered. Also, this information, for example, can be accompanied by further rationales.

TABLE III
DECISIONS CLASSIFIED BY THE EXTENT OF DOCUMENTED INFORMATION.

Given answer type	Count	Percent
Alternatives & Rationale	273	38.9
Multiple alternatives	262	37.3
One alternative	123	17.5
No alternative (Other in Fig. 8)	20	2.9
Rationale only (Other in Fig. 8)	9	1.3
Explicit exclusion (Other in Fig. 8)	1	0.1
N/A (Other in Fig. 8)	14	2
	702	

To answer research question RQ3, we went through all documented decisions again and categorized them according to the degree to which the students completed the templates. Content-related considerations did not play a role.

In analogy to the left column from Table III, we distinguished between the simple naming of the problem, the documentation of the selected solution alternative, the naming of none, one, or more alternatives. We also classified the provision of further rationale according to the template.

The result was that about 94% of all cases, at least one considered solution alternative was specified for the documented decisions (cf. Figure 8). In nearly three-quarters of

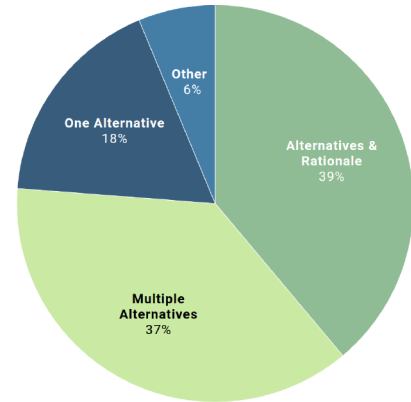


Fig. 8. Distribution of decisions classified by the amount of avail. information.

all cases ($\sim 76\%$), students recorded two or more solution alternatives for the documented decision. Additional rationales on the selected solution alternative and the reasoning behind were recorded in about 39%. This number includes cases with simple explanations formulated in one sentence as well as statements in which the rationale behind the decision was broken down into, e.g., positive and negative decision criteria.

V. DISCUSSION

The results presented in this paper suggest that the capture of decisions and their rationale is an issue that should have its place in software engineering education. Although the opinions from the follow-up surveys are not uniformly positive with the students, the early stage of their studies is affecting their judgment. Some students lack the experience that would change the assessment of such an approach. We observed that the more advanced the students were in their studies, the more positive was the feedback on the setup of our case study.

Furthermore, according to the opinion of the authors, the concrete context in which the respective case study took place

influenced on the quantity of the documented decisions, the amount of information recorded per decision as well as the type of documented decisions. For the authors, this impression is based on, for example, the complementary lectures that implicitly and unconsciously emphasize certain software engineering aspects. A repeatedly increased number of captured decisions on the topic of software quality aspects in the same sprints support this observation.

In the following, we briefly discuss the results of the research questions and the achievement of the learning goal:

RQ1 – Suitable Ways to Introduce Rationale Management: The follow-up surveys with the participating students showed that the case study setup provides a viable way to integrate the capture and management of decisions and their rationale into an agile process. The selected implementation also offers sufficient possibilities for adaptation to one's own needs. Feedback and experience have also shown, however, that it is not the only viable way to capture decisions. Adjustments are easy to implement, especially in the phases to which the decision recording is coupled.

RQ2 – Crucial Types of Decisions: For research question RQ2, it has to be stressed that there is no simple answer. A simple analysis of the distribution of all documented decisions would be very short-sighted. A closer look at the available data reveals that the phase of development has a significant influence on which decisions are essential for students. In the specific case of the case studies, the organizational structure of the software development projects also has a notable influence here. Almost all projects started from scratch and only lasted for a limited time. Accordingly, the students needed to make technical decisions that would not occur at such frequency in projects running for a prolonged time. The credit points awarded in the respective courses are a limiting factor here.

All in all, it can be emphasized that two major topics account for a large proportion of the documented decisions. One relates to requirements analysis, features, their refinement as well as the sequence of realization. Decisions on how to implement these features, how are they mapped to the software architecture, which libraries and frameworks are to be employed, define the other one.

RQ3 – What Information do Students Capture?: With research question RQ3, the authors wanted to find out to what extent students document a decision. It turned out that they document more than expected. In the pilot case study, students sometimes only did what the guidelines specified. Throughout the case studies, however, the students provided several considered alternatives for a decision in the majority of the cases. More than that, the students added further rationale for the decisions made in more than a third of the cases.

LO1 – Raise the Awareness for Rationale Management: In the follow-up surveys conducted with the participating students, it has become apparent that a considerable number of students have become aware of the topic of decision

making and rationale management in software engineering. Moreover, the management of decisions and their rationale, as already mentioned in related works, can certainly be seen as a controversial issue.

VI. FUTURE WORK

Upon completion of the current case study, it is necessary to enter the new data and update the evaluation. Building on this, we see the necessity to integrate the lessons learned from the case studies into teaching in a more structured way. To this end, it is also necessary to increase the response rate to the follow-up surveys carried out subsequently with the participating students. In this way, we can incorporate more and more detailed feedback into further rationale management applications. Another aspect is addressing the topic of *software architecture decision making* in the context of the supplementary lectures to the software development project, as done by, e.g., Lago et al. [29].

The logical next step is to generalize the approach used here for the Scrum process in teaching. Thereby the experiences of the case studies, as well as the feedback of the students, play a crucial role. The goal is to make the approach applicable to other scenarios, if not to other agile process models.

Another strategy for future work is to conduct a similar case study setup in an industrial context. It has been done in a single case so far, however, for comparability with industrial projects, considerably more data needs to be collected. Another approach is to compare the findings with applications in the open-source community. To this end, it would be necessary to find projects that systematically record decisions and make them accessible. The decision data could then be compared with the data from the industry and from the case studies with the students. However, comparability from the perspective of the applied process model is questionable here. We hypothesize that only the type of essential decisions will be comparable.

ACKNOWLEDGMENT

The authors gratefully acknowledge the fruitful cooperation with our colleagues from PUT Poznan and the University of Stuttgart, namely Mirosław Ochodek, Bartosz Walter, and Oliver Kopp. Furthermore, we thank all students for participating in the case studies and taking some extra time and effort to support us. Without all of them, these case studies would not have been possible.

REFERENCES

- [1] ALEXEEVA, Z., PEREZ-PALACIN, D., AND MIRANDOLA, R. Design Decision Documentation: A Literature Overview. In *European Conference on Software Architecture* (2016), Springer, pp. 84–101.
- [2] BASS, L., CLEMENTS, P., AND KAZMAN, R. *Software Architecture in Practice*, 2 ed. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.
- [3] BECK, K., BEEDLE, M., VAN BENNEKUM, A., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., SCHWABER, K., SUTHERLAND, J., AND THOMAS, D. Manifesto for Agile Software Development. <http://agilemanifesto.org/>, February 2001.

- [4] BHAT, M., SHUMAIEV, K., BIESDORF, A., HOHENSTEIN, U., AND MATTHES, F. Automatic Extraction of Design Decisions From Issue Management Systems: A Machine Learning Based Approach. In *European Conference on Software Architecture* (2017), Springer, pp. 138–154.
- [5] BRIAND, L. C. Software Documentation: How Much is Enough? In *Proceedings of the 7th European Conference on Software Maintenance and Reengineering* (2003), IEEE, pp. 13–15.
- [6] BURGE, J. E. Design Rationale: Researching Under Uncertainty. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 22, 4 (2008), pp. 311–324.
- [7] BURGE, J. E., AND BROWN, D. C. Software Engineering Using Rationale. *Journal of Systems and Software* 81, 3 (2008), 395–413.
- [8] CHACON, S., AND STRAUB, B. *Pro Git*. Apress, 2014.
- [9] CHARMAZ, K., AND BELGRAVE, L. L. Grounded Theory. In *The Blackwell Encyclopedia of Sociology*. American Cancer Society, 2015.
- [10] CLEAR, T. Documentation and Agile Methods: Striking a Balance. *SIGCSE Bull.* 35, 2 (June 2003), 12–13.
- [11] CONKLIN, E. J., AND YAKEMOVIC, K. C. B. A Process-Oriented Approach to Design Rationale. *Human-Computer Interaction* 6 (Sept. 1991), 357–391.
- [12] DE BOER, R. C., FARENHORST, R., AND VAN VLIET, H. A Community of Learners Approach to Software Architecture Education. In *22nd Conference on Software Engineering Education and Training* (2009), IEEE, pp. 190–197.
- [13] DES RIVIÈRES, J., AND WIEGAND, J. Eclipse: A Platform for Integrating Development Tools. *IBM Systems Journal* 43 (April 2004), 371–383.
- [14] DUTOIT, A. H., MCCALL, R., MISTRÍK, I., AND PAECH, B. Rationale Management in Software Engineering: Concepts and Techniques. In *Rationale Management in Software Engineering*. Springer, 2006, pp. 1–48.
- [15] DYBÅ, T., AND DINGSØYR, T. Empirical Studies of Agile Software Development: A Systematic Review. *Information and software technology* 50, 9-10 (2008), 833–859.
- [16] FISCHER, G., LEMKE, A. C., MCCALL, R., AND MORCH, A. I. Making Argumentation Serve Design. *Hum.-Comput. Interact.* 6 (September 1991), 393–419.
- [17] HESSE, T.-M., KUEHLWEIN, A., AND ROEHM, T. DecDoc: A Tool for Documenting Design Decisions Collaboratively and Incrementally. In *1st International Workshop on Decision Making in Software ARCHitecture (MARCH)* (2016), IEEE, pp. 30–37.
- [18] HETHEY, J. M. *GitLab Repository Management*. Packt Publishing Ltd, 2013.
- [19] HODA, R., NOBLE, J., AND MARSHALL, S. How Much is Just Enough? Some Documentation Patterns on Agile Projects. *EuroPLoP2010; 15th European Pattern Languages of Programs* (2010).
- [20] IONA, J. Text 2 Mind Map. *The School Librarian* 65, 3 (2017), 150.
- [21] ISHINO, Y., AND JIN, Y. An Information Value Based Approach to Design Procedure Capture. *Advanced Engineering Informatics* 20, 1 (2006), 89–107.
- [22] KAZMAN, R., KLEIN, M., AND CLEMENTS, P. ATAM: Method for Architecture Evaluation. Tech. rep., CMU/Software Engineering Institute, 2000.
- [23] KLEEBAUM, A., JOHANSEN, J. O., PAECH, B., AND BRUEGGE, B. Teaching Rationale Management in Agile Project Courses. In *Tagungsband des 16. Workshops "Software Engineering im Unterricht der Hochschulen"* (2019).
- [24] KOPP, O. Markdown Architectural Decision Records. GitHub, Mar. 2017. <https://adr.github.io/madr/>.
- [25] KOPP, O., ARMBRUSTER, A., AND ZIMMERMANN, O. Markdown Architectural Decision Records: Format and Tool Support. In *ZEUS* (2018), pp. 55–62.
- [26] KOPP, O., BINZ, T., BREITENBÜCHER, U., AND LEYMAN, F. Winery—A Modeling Tool for TOSCA-Based Cloud Applications. In *International Conference on Service-Oriented Computing* (2013), Springer, pp. 700–704.
- [27] KRUCHTEN, P. An Ontology of Architectural Design Decisions in Software Intensive Systems. In *2nd Groningen Workshop on Software Variability* (2004), pp. 54–61.
- [28] KUNZ, W., AND RITTEL, H. W. J. Issues as Elements of Information Systems. Tech. rep., Systemforschung, Heidelberg, Germany Science Design, University of California, Berkeley, 1970.
- [29] LAGO, P., CAI, J. F., DE BOER, R. C., KRUCHTEN, P., AND VERDECCHIA, R. Decidarch: Playing Cards as Software Architects. In *Proceedings of the 52nd Hawaii International Conference on System Sciences* (2019).
- [30] LATOZA, T. D., AND MYERS, B. A. Hard-To-Answer Questions About Code. In *Evaluation and Usability of Programming Languages and Tools* (2010), ACM, p. 8.
- [31] LEE, C., GUADAGNO, L., AND JIA, X. An Agile Approach to Capturing Requirements and Traceability. In *Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)* (2003), vol. 20.
- [32] LEE, J., AND LAI, K.-Y. What's in Design Rationale? *Hum.-Comput. Interact.* 6 (September 1991), 251–280.
- [33] LOPEZ, M. G., HAESEN, M., LUYTEN, K., AND CONINX, K. Helaba: A System to Highlight Design Rationale in Collaborative Design Processes. In *Cooperative Design, Visualization, and Engineering*. Springer, 2015, pp. 175–184.
- [34] MACLEAN, A., YOUNG, R. M., BELLOTTI, V. M. E., AND MORAN, T. P. Questions, Options, and Criteria: Elements of Design Space Analysis. *Hum.-Comput. Interact.* 6 (September 1991), 201–250.
- [35] MELNIK, G., AND MAURER, F. Direct Verbal Communication as a Catalyst of Agile Knowledge Sharing. In *Agile Development Conference* (2004), IEEE, pp. 21–31.
- [36] MYERS, K. L., ZUMEL, N. B., AND GARCIA, P. Automated Capture of Rationale for the Detailed Design Process. In *AAAI/IAAI* (1999), pp. 876–883.
- [37] OVADIA, S. Markdown for Librarians and Academics. *Behavioral & Social Sciences Librarian* 33, 2 (2014), 120–124.
- [38] REEVES, J. W. What is Software Design. *C++ Journal* 2, 2 (1992), 14–12.
- [39] RUBIN, E., AND RUBIN, H. Supporting Agile Software Development Through Active Documentation. *Requirements Engineering* 16, 2 (2011), 117–132.
- [40] RÜPING, A. *Agile Documentation: a Pattern Guide to Producing Lightweight Documents for Software Projects*. John Wiley & Sons, 2005.
- [41] SCHRIEK, C., VAN DER WERF, J.-M. E., TANG, A., AND BEX, F. Software Architecture Design Reasoning: A Card Game to Help Novice Designers. In *European Conference on Software Architecture* (2016), Springer, pp. 22–38.
- [42] SCHUBANZ, M. Design Rationale Capture in Software Architecture: What has to be Captured? In *Proceedings of the 19th International Doctoral Symposium on Components and Architecture* (2014), ACM, pp. 31–36.
- [43] SCHUBANZ, M. Making Decisions Sustainable in Agile Software Development. GitHub, March 2017. <https://github.com/schubmat/DecisionCapture/>.
- [44] SELIC, B. Agile Documentation, Anyone? *IEEE Software* 26, 6 (2009), 11–12.
- [45] TANG, A., AVGERIOU, P., JANSEN, A., CAPILLA, R., AND BABAR, M. A. A Comparative Study of Architecture Knowledge Management Tools. *Journal of Systems and Software* 83, 3 (2010), 352–370.
- [46] TANG, A., BABAR, M. A., GORTON, I., AND HAN, J. A Survey of Architecture Design Rationale. *Journal of Systems and Software* 79, 12 (2006), 1792–1804.
- [47] TANG, A., AND HAN, J. Architecture Rationalization: A Methodology for Architecture Verifiability, Traceability and Completeness. In *ECBS* (2005), pp. 135–144.
- [48] THURIMELLA, A., SCHUBANZ, M., PLEUSS, A., AND BOTTERWECK, G. Guidelines for Managing Requirements Rationales. *Software, IEEE* 34, 1 (2017), 82 – 90.
- [49] VOIGT, S., HÜTTEMANN, D., GOHR, A., AND GROSSE, M. Agile Documentation Tool Concept. In *Developments and Advances in Intelligent Systems and Applications* (Cham, 2018), Á. Rocha and L. P. Reis, Eds., Springer International Publishing, pp. 67–79.
- [50] WAGENAAR, G., HELMS, R., DAMIAN, D., AND BRINKKEMPER, S. Artefacts in Agile Software Development. In *International Conference on Product-Focused Software Process Improvement* (2015), Springer, pp. 133–148.
- [51] WANG, Y., BOGICEVIC, I., AND WAGNER, S. A Study of Safety Documentation in a Scrum Development Process. In *Proceedings of the XP2017 Scientific Workshops* (2017), ACM, pp. 22:1–22:5.
- [52] WOLF, T. *Rationale-Based Unified Software Engineering Model*. VDM Verlag, Saarbrücken, Germany, 2008.