Teaching Rationale Management in Agile Project Courses

Anja Kleebaum[1], Jan Ole Johanssen[2], Barbara Paech[1], and Bernd Bruegge[2]

[1]Heidelberg University, Heidelberg, Germany
[2]Technical University of Munich, Munich, Germany

kleebaum@informatik.uni-heidelberg.de, jan.johanssen@tum.de,
paech@informatik.uni-heidelberg.de, bruegge@in.tum.de

Abstract

Rationale management is beneficial since it supports decision-making and prevents knowledge vaporization. To apply rationale management, developers need to know how to systematically capture rationale and how to exploit the documentation. We believe that teaching these skills to students further integrates rationale management into the daily work of developers and has positive effects on both the software development process and on the quality of the software. In this paper, we report on a lecture on teaching rationale management to students. In this lecture, students are introduced to a rationale model, to capture and exploitation methods, and tool support for rationale management. The goal is to motivate them to apply rationale management. We present the students' results as well as their attitude and feedback towards the applied methods. Further, we sketch how rationale management will be applied during the semester.

1 Introduction

Developing software means to continuously solve issues and to make decisions. Developers possess knowledge about decisions, the issues they solve, alternatives, and their justifications. This knowledge is called decision knowledge or rationale. The success of a development project strongly depends on the decision-making abilities of the developers and other relevant stakeholders. Rationale management supports explicit decision-making by capturing rationale and by using the documentation [7]. Developers are said to be reluctant to capture rationale systematically [1, 14, 18]. We believe that this can be alleviated by teaching developers how to capture rationale and by raising their awareness for the benefits of rationale management.

As part of our previous work, we developed methods and tool support to integrate rationale management into agile development processes, in particular into continuous software engineering [11, 12]. In this paper, we report on a lecture on teaching rationale management to university students. The goal

of this lecture is to teach a rationale model to students, to introduce them to methods and tool support for rationale management, and to motivate them to apply rationale management in the future. We gave this lecture as part of an agile multi-project course at the Technical University of Munich, in which student teams work on different software projects over the period of one semester. The projects are initiated by industrial customers [4]. As a default set of tools, the students use the issue tracking system JIRA and the wiki system Confluence. During the lecture, we presented our methods on rationale management to the students and they applied parts of our tool support. We asked students for their feedback on the presented tools and to perform exercises.

This paper is structured as follows. In Section 2, we outline background knowledge on rationale management and the agile multi-project course. In Section 3, we present our teaching material including six exercises, and the course of the lecture. Then, we summarize the students' feedback, present the exercise results, and discuss lessons learned. In Section 4, we describe plans on how to evaluate rationale management during the agile project course. Section 5 lists related work and Section 6 concludes the paper.

2 Background

This section introduces the rationale model we use, basics about issue tracking and wiki systems, our previous work on integrating rationale management into agile software development, and a description of the agile multi-project course that the lecture is part of.

2.1 Rationale Models

A rationale model represents knowledge in the same way a system model represents a system. Rationale can be modeled as a graph of rationale elements and edges that represent the elements' relationships. There are various models for building such graphs of rationale. In general, these models differ in the types of elements and edges that they allow. For example, an advanced model is the decision documentation model,

Emoji	Name	Indicating Phrases
	Issue	I have a question ... How should ... ..., any suggestions? We need to discuss how ...
	Alternative	I { suggest | propose } ... One { option | proposal } is ... What { about | do you think } ...
	Pro	The { advantages | pros } are ... I { like | prefer } it because ... I agree with user ...
	Con	The { disadvantages | cons } are ... I don't like it because ... I disagree with user ...
	Decision	Let's do ... We decided ... The best option is ...

Table 1: Types of rationale elements, their representing emoji, and indicating phrases adapted from [5].

which makes fine-grained differences in element types such as implications resulting from a decision or constraints that need to be considered [9]. There could be various types of relationships between rationale elements, for example, a decision can *solve* an issue or it can also *lead to* a new issue.

In order to teach rationale management, we use an easy-to-learn model to represent rationale. This model covers five types of rationale elements: *issue*, *alternative*, *pro-* and *con-argument*, and *decision* (Table 1). In our model, we distinguish three types of relationships, i.e., edge types. The *relates to* relationship is the default edge type. Only for arguments different types are used: A pro-argument *supports* an alternative or the decision, whereas a con-argument *attacks* an alternative or the decision.

2.2 Issue Tracking and Wiki System

Developers can capture rationale in various documentation locations, for example, in the issue tracking [3, 10, 18] or the wiki system. Issue tracking systems are widely applied to store requirements, bug reports, as well as development tasks. They contain various information types such as functionality or quality requests and *as-is* descriptions [17]. Wiki systems are collaborative writing tools that can be applied for many purposes, for example, for meeting management and requirements elicitation [19]. For a lecture on rationale management, we assume that students use the issue tracking system JIRA and the wiki system Confluence.[1] JIRA and Confluence are commercial systems that provide free licenses to universities. Both systems can be extended with plugins.[2]

---

[1] https://atlassian.com/software
[2] https://developer.atlassian.com

2.3 Continuous Rationale Management

Agile processes support lightweight, flexible, and continuous software development. Rationale management integrates well into such processes, yet, it should be easy to apply, i.e., developers should need as little effort for it as possible. Capturing and exploring rationale should be non-intrusive, which means that developers should be able to perform rationale management as part of their daily practices rather than having to change their development context. For example, developers should be able to capture and explore rationale simultaneously with performing development tasks, implementing requirements, or committing code.

In order to fulfill this requirement of non-intrusiveness, we develop tool support that directly integrates into the development tools [12]. In particular, we integrate our tool support into the issue tracking system, version control system, wiki system, chat system, and the integrated development environment. We refer to our tool support as ConDec, standing for the continuous management of decision knowledge. The ConDec tool support is available online.[3]

While ConDec comprises features to trigger developers in capturing and exploring rationale [11], in this paper, we focus on the basic infrastructure necessary for capturing and visualizing rationale as a graph. Regarding the ConDec tool support, the students get to know and apply the ConDec JIRA plugin, i.e., the tool support for the issue tracking system JIRA.

The ConDec JIRA plugin supports different documentation locations of rationale, e.g., JIRA issues and comments. On the one hand, rationale elements can be captured as JIRA issues with special types for issue, alternative, argument, and decision. Note the difference between JIRA issue as an abstract container and the concrete issue as the rationale element. JIRA issue links are used to link the rationale elements with each other and also to JIRA issues of other types such as scenarios or tasks. On the other hand, rationale can be captured as part of the comments of JIRA issues.

2.4 The iPraktikum

The iPraktikum is a multi-project course in which up to 100 students work in eight to ten teams on real problems provided by an industry customer [4].

In particular in the first half of the semester, the practical character of the course is supported by theoretical, yet interactive lectures. During these lectures, the students learn the basic concepts of agile development, release and merge management, modeling, and usability engineering. Recently, we added a lecture on rationale management. During the iPraktikum, the students apply latest tools and frameworks that are used in real industry projects. Moreover, they get to use the results of latest research projects, which might

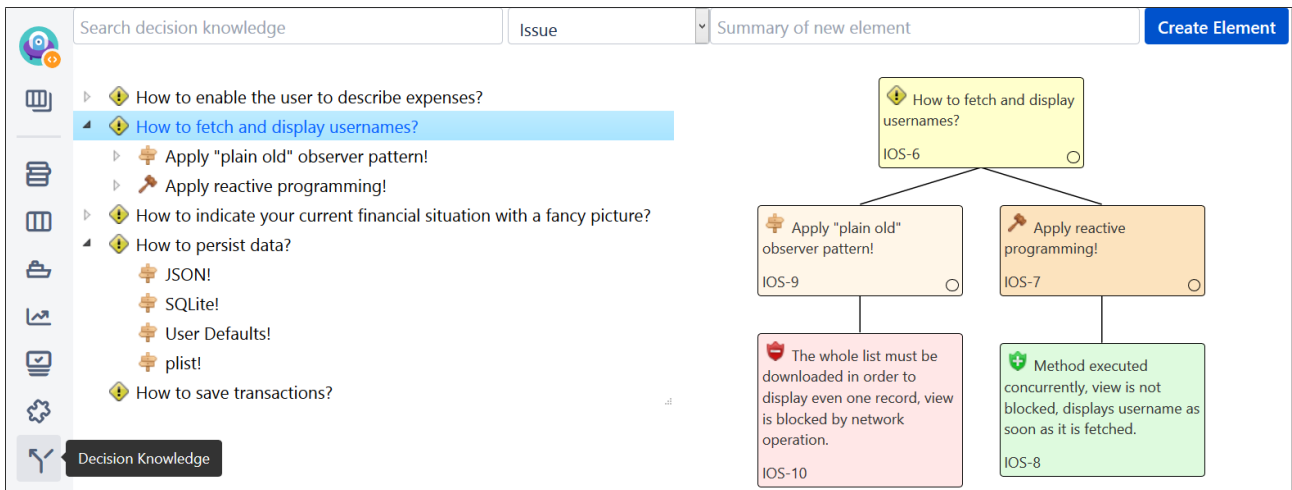---

[3] https://github.com/cures-hub

Figure 1: The decision knowledge view as a separate view for performing rationale management in JIRA.

find their way into the development process in the future. This prepares the students for their future use.

At the time of the rationale management lecture, they have already been introduced to JIRA and Confluence. Regarding JIRA, they have at least one month of experience. This knowledge is imparted through various channels: (a) a development introduction course in which the students are required to track the progress of their work via JIRA, (b) the work within their teams, and (c) a course-wide lecture on managing the backlog, on what JIRA issues are, on how to close, move, and work with them in sprints. Regarding Confluence, we provide the students with a meeting management introduction at the beginning of the course. This is handled by the coaches of a team, a special role within the team that is fulfilled by an experienced student and which is similar to a scrum master. The coach can decide on what to present, however, we provide a guideline that contains the major aspects of using Confluence. In particular, this relates to managing the team agenda, i.e., (a) how does a team meeting schedule look like?, (b) which roles are present during each meeting?, (c) what are guidelines, i.e., what to provide as the stand-up information and when to use it?

3 Lecture on Rationale Management

In this section, we describe the lecture, the results of its first instantiation, and then discuss these results and our lessons learned.

3.1 Preparation and Introduction

The lecture is designed to last 90 minutes. Students are grouped into teams and require a web-connected device with access to the internet.[4] During the lecture, three systems are needed: JIRA, Confluence,
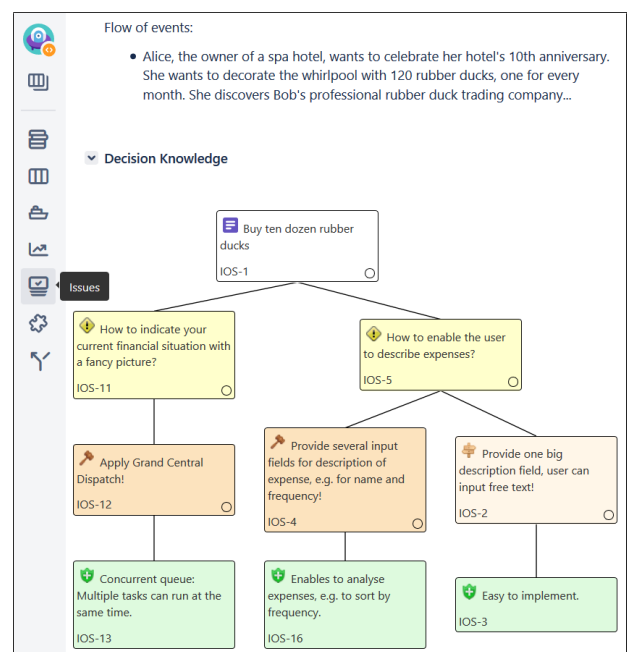

Figure 2: The JIRA issue view including the interactive rationale tree.

and the instant messaging services Slack[5]. Further, the ConDec JIRA plugin[6] needs to be installed in JIRA and be enabled for the specific projects that the student teams work in. Rationale elements that can be explored by the students need to be added to the respective JIRA projects, e.g., the elements shown in Figure 1 and Figure 2. Slack is used as a communication tool between the instructors and students. In particular, polls can be created with the Polly Slack app[7] through which students participate during the lecture.

---

[4]An alternative would be to run the servers locally and to give students access to the intranet (not possible for Slack).

[5]https://slack.com
[6]https://github.com/cures-hub/cures-condec-jira
[7]https://polly.ai

The first part of the lecture covers background information about rationale management, such as its definition, expected benefits, and the rationale elements. We advise students to use certain phrases when talking about and capturing rationale (Table 1). Further, we recommend to phrase issues as questions ending with a question mark and to end alternatives with an exclamation mark.

3.2 Capturing, Visualizing, and Filtering Rationale in JIRA

In the second part of the lecture, the students are introduced to the JIRA ConDec plugin including the JIRA issue types for rationale elements (Figure 3).
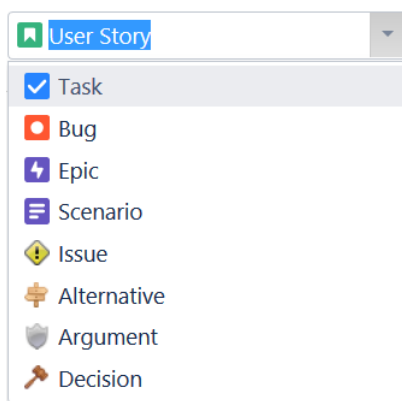

Figure 3: JIRA issue types available in the project.

Two views for rationale management are shown to the students: the decision knowledge view (Figure 1) and the JIRA issue view (Figure 2). The decision knowledge view is a separate view that holds all rationale elements and their links for the given project. In this view, a user can select single rationale elements and visualize the graph of rationale as a tree, called rationale tree. In the JIRA issue view, rationale attached to JIRA issues can be explored. For example, Figure 2 shows the rationale tree for a scenario.

Then, the instructor demonstrates how to create and link rationale elements shown on the right side of Figure 1. Afterwards, the students gather in teams and perform the first exercise:

Exercise 1 *Gather your team, open your JIRA project, and find the scenario already documented in the project. Answer the question: How many issues are linked to the scenario?*

The students can answer the question via a poll. In our example, the correct answer is that two issues are linked to the scenario (Figure 2). The next exercise is to discuss the content of the solution proposals, i. e., the alternatives and decisions.

Exercise 2 *Answer the question: Which of the proposed alternatives and decisions focus on requirements, which of them on implementation?*

In our example, the decision and alternative on the left are more requirements-related, whereas the decision on the right is implementation-specific (Figure 2). With this exercise, the students should learn that rationale can be captured for all steps in the software engineering process, including requirements elicitation, implementation, deriving test cases, and when processing user feedback.

Now, the students will make their first experience in capturing rationale:

Exercise 3 *Link the existing issue "How to save transactions?" to the scenario.*

This issue is already part of the JIRA project (Figure 1) but not linked to the scenario yet. The students can link the issue via a context menu on the scenario node (root node in Figure 2).

The students realize that graphs of rationale can become large and complex. Therefore, filtering is important. The next exercise addresses filtering:

Exercise 4 *Filter the element types so that only the scenario and decisions are shown in the rationale tree.*
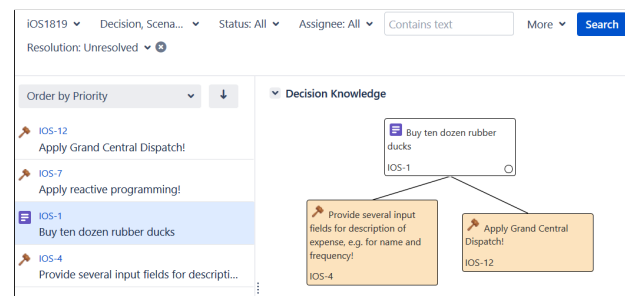

Figure 4: Filtered rationale tree.

Figure 4 shows the solution of this exercise. Now, students discuss rationale in their team:

Exercise 5 *Create a new issue "How to persist data?". Add the following alternatives: "JSON!", "SQLite!". Discuss and capture pro and cons of each persistence alternative in your team. Add more alternatives and make a decision.*

To solve this exercise, the students can add rationale elements collaboratively from different devices. This exercise takes about 10 to 15 minutes.

Rationale can be captured in many places. JIRA issues are just one possible documentation location to store rationale. In order to demonstrate that there are other possible documentation locations, the students are presented with capturing rationale in JIRA issue comments (Figure 6). In this lecture, this is for information only, but, as part of our future work, we integrate various documentation locations typical for continuous software engineering and support the identification of rationale elements with a supervised text classifier.
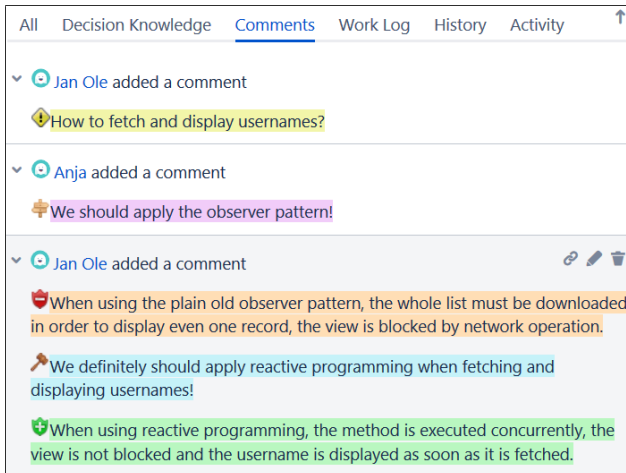
Figure 6: Explicit rationale in the comments of the scenario.

3.3 Rationale-based Meeting Management

Capturing rationale pays off during sprint meetings. The meeting agenda has an information sharing section that lists issues discussed and decisions made during the last sprint. If meeting agendas are managed in Confluence, the rationale elements captured in JIRA can be easily imported.

Exercise 6 *Gather your team, open your Confluence space and create a new page called <Rationale Lecture> (one per team). Create a sub-page called <Your Name> (every team member). Use the JIRA Issue/Filter macro to display decisions from your JIRA project. Answer the question: How many decisions do you see?*

The JIRA Issue/Filter macro is used in the exercise and the search string is expressed using the JIRA query language (JQL). The JQL string is:

```
project = <Project Key> AND
issuetype = Decision AND
created > -7d.
```
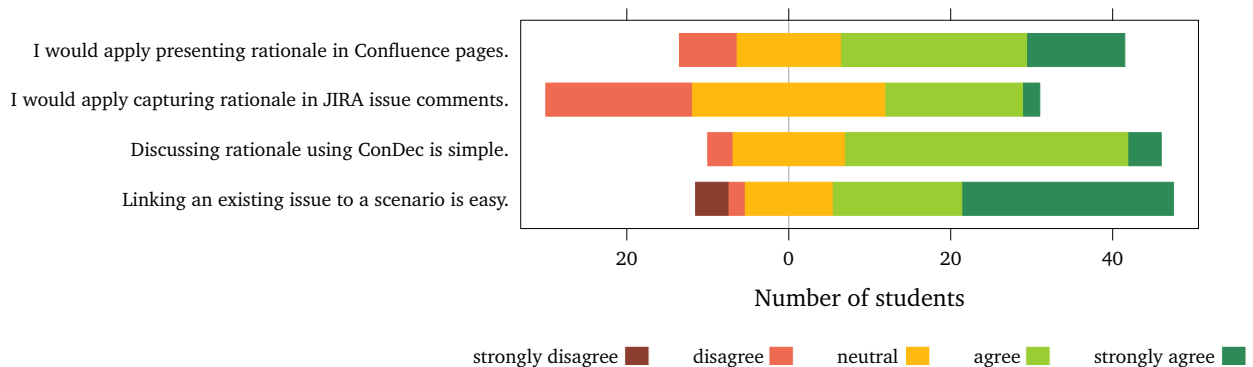
In our case, two decisions are shown.

3.4 Results

Two authors of this paper gave the lecture on November 8, 2018. In this section, we present results of this first instantiation of the lecture.

The first poll was to answer the question what team a student belongs to. With this poll, we wanted the students to get warmed up in voting and to get the number of participating students. In total, 88 students answered the poll, i.e., about 88 students participated in the lecture. This number serves as a reference point for the following, quantitative evaluation results.

Exercise 1 was answered by 64 students, i.e., 73 % of the participating students, of whom 63 gave the correct answer.

Regarding the content of the solution proposals, i.e., the alternatives and decisions, we initially asked a different question than given in Exercise 2. We asked the students to describe the difference between the alternatives and decisions. Since this question seemed to be hard to answer, we restated the question as given in Exercise 2. This exercise was verbally performed and we did not collect any results for it.

After performing Exercise 3, we asked students to assess how easy it was to link an existing issue to a scenario via a poll. They were asked to rate the statement *Linking an existing issue to a scenario is easy* with one answer from a five point Likert scale. 59 students participated in this poll, of whom 6 disagreed, 11 were neutral, and 32 agreed with the statement (Figure 5). After the lecture, we checked whether the teams correctly performed Exercise 3. In every team project, the issue was correctly linked to the scenario.

We did not collect any results for or attitudes towards Exercise 4.

For Exercise 5, we created a poll in which the students could rate the statement *Discussing rationale using ConDec is simple*. 56 students participated in this poll, of whom 7 disagreed, 13 were neutral, and 35 agreed with the statement (Figure 5).

Figure 5: Students' attitude towards the presented methods and tools for rationale management.

We asked the students to provide written feedback on discussing rationale (Table 2) and we analyzed the rationale graphs. A total of 126 rationale elements were documented, i. e., a mean value of 12.6 rationale elements per team with a standard deviation of 9.4 elements. That means that each student contributed a mean value of 1.4 rationale elements. A mean value of 3.3 alternatives were documented per team (standard deviation 1.2). 61 % of the alternatives correctly ended with an exclamation mark. Seven of the ten issues correctly ended with a question mark. Four teams documented the decision. The types of the elements were correctly chosen, e. g., arguments were not accidentally classified as alternatives.

Student Feedback	Our Rationale
Deletion of elements is not possible.	Permission scheme in the JIRA project forbids deletion for non-admin users.
If there are many alternatives and arguments, it is hard to get an overview. The user needs to scroll very far to the right to see all the content.	A better graph visualization and better, easy-to-apply filter possibilities are needed.
Rationale elements should not always be a single ticket. This seems to end up in a ticket overflow. Pro- and con-arguments should be comments.	Capturing rationale elements in separate JIRA issues has the advantage that they can easily be imported into Confluence. We also develop the ConDec Confluence plugin to import rationale from comments.

Table 2: Summarized feedback provided by students.

After demonstrating that rationale could also be captured in the comments of the scenario, we asked the students to rate the statement *I would apply capturing rationale in JIRA issue comments*. 61 students participated in this poll, of whom 18 disagreed, 24 were neutral, and 19 agreed with the statement (Figure 5).

After the students performed Exercise 6, we asked them to rate the statement *I would apply presenting rationale in Confluence pages*. 55 students participated in this poll, of whom 7 disagreed, 13 were neutral, and 35 agreed with the statement (Figure 5).

A mean value of 58 students participated in the polls analyzed in Figure 5, which represents 66 % of all students.

3.5 Discussion and Lessons Learned

Clearly, the results that we collected during the lecture only represent the students' first impression about the presented methods for rationale management and the ConDec JIRA plugin. As described in the next section, we will apply a more thorough evaluation process during the semester.

The results of voting on the statements in Figure 5 lead us to conclude that the majority of the students liked applying rationale management. More important than the votes, the written feedback summarized in Table 2 encourages us to improve the visualization and filtering components of the ConDec JIRA plugin. In general, we had the impression that the students liked voting on the polls and performing the exercises, since this made the lecture more interactive [13].

Studying the rationale trees that the students created in Exercise 5, we noticed that we did not ask the students to make a decision. Thus, only four decisions were documented. As a result, we will restate the exercise for future use. The students seem to understand the elements of the rationale model (Table 1), since they correctly classified the element types in their trees of rationale.

4 Evaluation During Semester

Two teams continue to apply rationale management during the agile project course. Thus, we will evaluate the explained methods for rationale management, especially the ConDec JIRA plugin.

We introduced the role of the rationale manager. The rationale manager is responsible for checking and improving the rationale quality, i. e., they make sure that important elements are documented and that they are consistent. Further, the rationale manager imports issues and decisions important for the last sprint into the meeting agenda in Confluence. They update and add rationale elements after the meeting in JIRA. The role of the rationale manager is taken by one student per team. The role is passed on after a week to a different student, i. e., it is an interchanging role. After the students have completed the role of the rationale manager, we ask them to give us feedback by filling in a questionnaire. We derive the questions in this questionnaire by considering the variables of the technology acceptance model [16]: We consider the perceived usefulness, the perceived ease of use, and the intention to use. Next to rating statements as shown in Figure 5, it is important that the students provide detailed feedback on the features for rationale management they applied. For example, students should provide details on what they think is useful or useless, easy or difficult, and why they think so.

5 Related Work

To the best of our knowledge there is no work that reports on teaching rationale management. Thus, in this section, we present related work on applying rationale management in student projects.

While tools for rationale management have been evaluated in student projects before, e. g., in [8] and [2], the following authors especially focus on the positive effects of rationale management for the success of the student course.

Dutoit *et al.* discuss experiences with an integrated, rationale-based modeling environment in a variety of software engineering courses [6]. By applying rationale management, they aim to enhance the communication between instructor and students, to support students in reflecting their own work, and to enable the instructor to better monitor the students' progress. These are valid benefits that we also expect when students apply rationale management during the semester as part of the agile project course. Similar to their modeling environment, ConDec integrates the rationale elements with the system elements, such as requirements. In addition, we focus on linking rationale with development tasks since they represent the place where developers need to solve issues related to design and implementation.

Malloy and Burge developed the software engineering using rationale tool SEURAT_Edu as a web-based system that replaces the former SEURAT Eclipse[8] extension [15]. Like our tool support, SEURAT_Edu aims to support students in making the best decision for an issue under consideration by explicitly reasoning about design alternatives. During their evaluation, Malloy and Burge found that the students using their tool considered more alternatives and put more thought into decision-making. Similar to their tool, the ConDec JIRA plugin is web-based, which allows students to easily collaborate. Their tool integrates with learning management systems such as Moodle[9], which has the advantage that the learning management system takes care of authentication and assignment creation. In our case, this is handled by JIRA. Similar as we do in the lecture on rationale management, SEURAT_Edu enables the teacher to supply students with incomplete rationale that they are asked to complete. In addition, SEURAT_Edu enables the teacher to create a set of "solution" rationale. It displays the status to which students reached a solution, in order to encourage them during their tasks. SEURAT_Edu performs automatic error checks, e. g., whether there are issues not solved by a decision. The ConDec JIRA plugin provides a report page that lists quality metrics. We plan to integrate support to ensure the rationale quality in the development process.

6 Conclusion

We presented a lecture on rationale management that teaches students to apply a rationale model as well as methods and tool support for rationale management. The students' feedback and the exercise results let us conclude that the students comprehend the usage of the rationale model and that they are motivated to apply rationale management. To validate this first impression, a more detailed evaluation will be part of the remaining duration of the agile project course.

---

[8] https://eclipse.org
[9] https://moodle.de

References

[1] Zoya Alexeeva, Diego Perez-Palacin, and Raffaela Mirandola. Design decision documentation: A literature overview. In *Software Architecture*, volume 5292 of *Lecture Notes in Computer Science*, pages 84–101. Springer, Berlin, Heidelberg, 2016.

[2] Rana Alkadhi, Jan Ole Johanssen, Emitza Guzman, and Bernd Bruegge. REACT: An approach for capturing rationale in chat messages. In *11th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'17)*, Toronto, Canada, 2017. IEEE.

[3] Manoj Bhat, Klym Shumaiev, Andreas Biesdorf, Uwe Hohenstein, and Florian Matthes. Automatic extraction of design decisions from issue management systems: A machine learning based approach. In *11th European Conference on Software Architecture (ECSA'17)*, pages 138–154, Cham, Switzerland, 2017. Springer. ISBN 978-3-319-65830-8.

[4] Bernd Bruegge, Stephan Krusche, and Lukas Alperowitz. Software engineering project courses with industrial clients. *ACM Transactions on Computing Education*, 15(4):17:1–17:31, 2015.

[5] Michael Doyle and David Straus. *How to Make Meetings Work: The New Interaction Method*. Berkley, 1993. ISBN 978-0425138700.

[6] Allen H. Dutoit, Timo Wolf, Barbara Paech, Lars Borner, and Jürgen Rückert. Using rationale for software engineering education. In *18th Conference on Software Engineering Education and Training (CSEE&T)*, pages 129–136, Ottawa, Canada, 2005. IEEE.

[7] Allen H. Dutoit, Raymond McCall, Ivan Mistrík, and Barbara Paech. *Rationale Management in*

---

[10] http://p.yusukekamiyamane.com

Software Engineering: Concepts and Techniques. Springer, 2006. ISBN 978-3-540-30998-7.

[8] Davide Falessi, Giovanni Cantone, and Martin Becker. Documenting design decision rationale to improve individual and team design decision making. In *ACM/IEEE International Symposium on Empirical Software Engineering (ISESE)*, pages 134 – 143, Rio de Janeiro, Brazil, 2006. ACM.

[9] Tom-Michael Hesse and Barbara Paech. Supporting the collaborative development of requirements and architecture documentation. In *3rd International Workshop on the Twin Peaks of Requirements and Architecture*, pages 22–26, 2013.

[10] Tom-Michael Hesse, Veronika Lerche, Marcus Seiler, Konstantin Knoess, and Barbara Paech. Documented decision-making strategies and decision knowledge in open source projects: An empirical study on firefox issue reports. *Information and Software Technology*, 79:36–51, 2016.

[11] Anja Kleebaum, Jan Ole Johanssen, Barbara Paech, Rana Alkadhi, and Bernd Bruegge. Decision knowledge triggers in continuous software engineering. In *4th International Workshop on Rapid Continuous Software Engineering (RCoSE)*, pages 23–26, Gotheburg, Sweden, 2018. ACM.

[12] Anja Kleebaum, Jan Ole Johanssen, Barbara Paech, and Bernd Bruegge. Tool support for decision and usage knowledge in continuous software engineering. In *3rd Workshop on Continuous Software Engineering*, pages 74–77, 2018.

[13] Stephan Krusche, Nadine von Frankenberg, and Sami Afifi. Experiences of a software engineering course based on interactive learning. In *15. Workshop Software Engineering im Unterricht der Hochschulen (SEUH)*, pages 32–40, Hanover, Germany, 2017.

[14] Claudia López, Víctor Codocedo, Hernán Astudillo, and Luiz Marcio Cysneiros. Bridging the gap between software architecture rationale formalisms and actual architecture documents: An ontology-driven approach. *Science of Computer Programming*, 77(1):66–80, 2012.

[15] John Malloy and Janet Burge. SEURAT_Edu: A tool to assist and assess student decision-making in design. In *47th Technical Symposium on Computing Science Education (SIGCSE)*, pages 669–674, Memphis, Tennessee, USA, 2016. ACM.

[16] Nikola Marangunić and Andrina Granić. Technology acceptance model: a literature review from 1986 to 2013. *Universal Access in the Information Society*, 14(1):81–95, 2015.

[17] Thorsten Merten, Bastian Mager, Paul Hübner, Thomas Quirchmayr, Simone Bürsner, and Barbara Paech. Requirements communication in issue tracking systems in four open-source projects. In *6th International Workshop on Requirements Prioritization and Communication (RePriCo)*, pages 114–125, Essen, Germany, 2015.

[18] Benjamin Rogers, Yechen Qiao, James Gung, Tanmay Mathur, and Janet E. Burge. Using text mining techniques to extract rationale from existing documentation. In *6th International Conference on Design Computing and Cognition*, pages 457–474. Springer, 2014.

[19] Carlos Solis and Nour Ali. Distributed requirements elicitation using a spatial hypertext wiki. In *5th IEEE International Conference on Global Software Engineering*, pages 237–246, Princeton, NJ, USA, 2010. IEEE.