Continuous Rationale Management Using the ConDec Tools

Anja Kleebaum[1] Jan Ole Johanssen[2] Barbara Paech[3] Bernd Bruegge[4]

Abstract: We summarize the paper *Continuous Management of Requirement Decisions Using the ConDec Tools*, which we presented at the 26th *International Working Conference on Requirements Engineering* (REFSQ2020). We summarize ConDec's features and a usage scenario. In addition, the REFSQ paper provides an overview of the views, the architecture, and it sketches evaluation efforts.

Keywords: Rationale Management; Decision Knowledge; Knowledge Graph; Requirements; Code; Design Decisions; Change Impact Analysis; Agile Software Development; ConDec

Summary

While developing a software system, developers – as well as stakeholders of other roles, such as requirements engineers, architects, or testers – continuously make decisions and have to reflect on previous decisions. For example, they decide on how to implement functional requirements, how to fulfill quality requirements, and how to solve trade-offs between them. They build up important *decision knowledge*, which is also referred to as *rationale*. It is important to manage and share decision knowledge in the context of requirements and code for the success of a software development project. However, decision knowledge management is challenging: 1) The documentation of decision knowledge is an additional effort, i.e., it is intrusive in the development process. 2) The documented knowledge can be of low quality in terms of completeness and consistency. 3) It might be distributed across many documentation locations, such as issue comments and commit messages, and thus difficult to access, share, and use. 4) The documented knowledge can become very large with a lot of knowledge elements and links in between them.

The ConDec tools for the *continuous management of decision knowledge* aim to overcome these challenges. ConDec provides features and views for documenting and exploiting decision knowledge in the context of software artifacts, in particular requirements and code. ConDec integrates into the workflows of agile software development. The developers document decision knowledge, e.g., decision problems, solution options, and arguments, within the issue tracking system, commit messages, and code comments. ConDec builds up a knowledge graph of requirements, decision knowledge, code, and other artifacts (Figure 1).

[1] Heidelberg University, Heidelberg, Germany, kleebaum@informatik.uni-heidelberg.de
[2] Technical University of Munich, Munich, Germany, jan.johanssen@tum.de
[3] Heidelberg University, Heidelberg, Germany, paech@informatik.uni-heidelberg.de
[4] Technical University of Munich, Munich, Germany, bruegge@in.tum.de

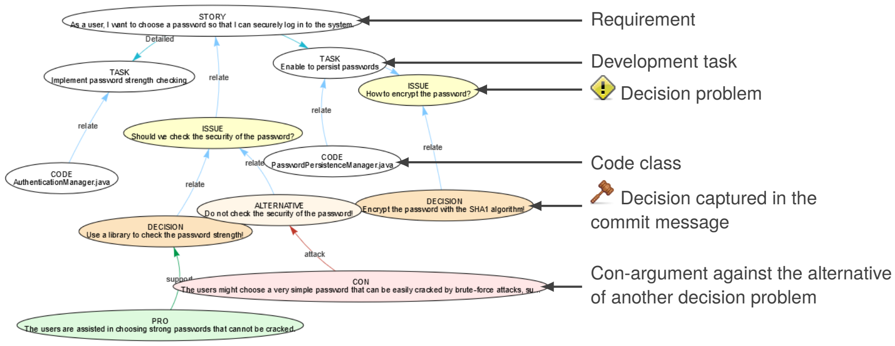
STORY
As a user, I want to choose a password so that I can securely log in to the system.
Detailed
TASK
Implement password strength checking
relate
TASK
Enable to persist passwords
ISSUE
How to encrypt the password?
relate
relate
ISSUE
Should we check the security of the password?
CODE
PasswordPersistenceManager.java
relate
CODE
AuthenticationManager.java
relate
relate
ALTERNATIVE
Do not check the security of the password!
DECISION
Encrypt the password with the SHA1 algorithm!
DECISION
Use a library to check the password strength!
attack
support
CON
The users might choose a very simple password that can be easily cracked by brute-force attacks, su...
PRO
The users are assisted in choosing strong passwords that cannot be cracked.
Requirement
Development task
Decision problem
Code class
Decision captured in the commit message
Con-argument against the alternative of another decision problem

Fig. 1: View on the knowledge subgraph starting from a requirement.

Imagine the following scenario about the development of a user log-in system: An important decision problem is to select an algorithm to encrypt passwords to ensure their confidentiality. Imagine that a developer captures the question ⚠ *How to encrypt the password?* in the comment of a Jira issue. The icon indicates that the developer uses a markup language-like syntax to highlight the question as a decision problem. Subsequently, the developer captures the solution for the decision problem together with a Jira issue identifier (key) in a commit message. Now, ConDec performs three actions automatically: 1) It transcribes the commit message into a new comment of this Jira issue. 2) It highlights the solution decision using a supervised text classifier, which can be trained and evaluated on the underlying software development project. 3) It automatically links the solution decision to the respective decision problem within the knowledge graph. In addition to the automatic linking, developers can manually create and maintain links between knowledge elements in the knowledge graph.

ConDec supports decision knowledge sharing and collaborative decision making using a rationale backlog and in meeting agendas. ConDec introduces a definition of done for high-quality rationale documentation. It supports the fulfillment of the definition of done with the help of dashboards and automated quality checks integrated into the development workflow, e.g., when finishing a requirement. For example, the developer can only finish the log-in requirement if all the decision problems that are documented in a certain link distance in the knowledge graph are solved. The dashboard shows metrics regarding the intra-rationale completeness (e.g., *are there arguments for the decisions?*), the rationale coverage of requirements and code (e.g., *how many decisions are documented for a requirement?*), the rationale-to-artifact traceability (e.g., *is it possible to navigate to the requirements affected by a decision?*), and the feature branch quality regarding the rationale documentation.

References

[Kl20] Kleebaum, A.; Johanssen, J. O.; Paech, B.; Bruegge, B.: Continuous Management of Requirement Decisions Using the ConDec Tools. In: 26th International Conference on Requirements Engineering Workshops, Doctoral Symposium, Live Studies Track, and Poster Track (REFSQ20). CEUR-WS.org, Pisa, Italy, 2020.