UNICASE Trace Client: (Semi-) Automatic Tracing of Requirements and Code During Development for Small and Medium Enterprises

Alexander Delater, Barbara Paech
Institute of Computer Science, University of Heidelberg, Im Neuenheimer Feld 326, 69120 Heidelberg, Germany
{delater, paech}@informatik.uni-heidelberg.de

1 Introduction

Corporations are constantly making progress in their efforts towards traceability in software development. One challenge faced by small and medium enterprises (SMEs) is to create traceability links between requirements and code [1], e.g. to use them to prove to their customers that they implemented all requirements. However, these links are often created after development [2], which can, for example, result in increased costs and development time. In [3], we presented an approach to (semi-) automatically create links between requirements and code during development using artifacts from project management. Based on this approach, we developed the lightweight tool UNICASE Trace Client (UTC) [4]. It is an extension to the model-based CASE tool UNICASE [5], which is an Eclipse plug-in developed in an open-source project. As SMEs prefer tools supporting multiple aspects of software development, e.g. requirements, project management and code, UTC is particularly suited for them. In this paper, we introduce UTC and describe why it is valuable for SMEs.

2 Background

We defined a Traceability Information Model (TIM) [3] (see Figure 1) consisting of artifacts from requirements engineering (system model), project management (project model) and code (code model). An extended UML notation was used to represent the three models with their artifacts.

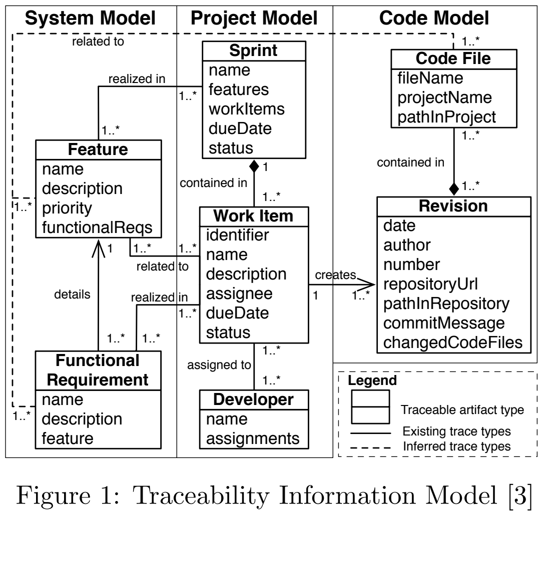

Figure 1: Traceability Information Model [3]

A feature is realized in a sprint and is detailed in one or more functional requirements. Functional requirements are realized in work items. One work item must have one or more linked functional requirements. Work items are contained in sprints and are assigned to developers. A work item can create one or more revisions, each containing one or more changed code files. All these artifacts can be found in common software development projects performed by SMEs [1].

We presume the following situation in a development project. First, a list of features and functional requirements exists. Second, a project manager has planned the implementation of the features in sprints and s/he has broken down the implementation schedule of the functional requirements into work items for the developers. Third, all work items are already assigned to developers. As SMEs often use agile software development techniques and issue management [1], this situation is common to them. Below we use the term *requirement* to refer commonly to features and functional requirements to avoid misunderstandings.

3 UNICASE Trace Client

UTC implements the TIM and integrates itself seamlessly in Eclipse and its supporting plug-ins, e.g. Subversion. It supports various features for the (semi-) automatic creation and usage of links between requirements and code. The features are divided into *basic features* for link creation and *advanced features* for link usage. The basic features for link creation are:

Capturing Traceability Links: First, the developer selects a work item from his/her list of assigned work items. While working on the work item, all requirements the developer looks at during implementation are automatically captured by UTC. After the developer finishes coding, UTC asks him/her to validate all captured requirements. The validated requirements are linked to the work item and the created revision is linked to the work item. This results in traceability between requirements, work items and code.

Inferring Traceability Links: The created links are used to infer direct links between requirements and code. They are represented as dashed lines in Figure 1. In [3], we have presented an algorithm for inferring links between requirements and code. The inference algorithm is executed when the status of a work item is changed by the developer from *assigned* to *done*. The algorithm connects in a brute force manner all linked requirements of a work item with all the code files in the linked revisions of a work item.

Summing up, the only manual work in our approach is to establish initial links between work items and requirements (which is typical for issue management) and to validate the captured links (which should be easy as the links refer to the work just finished). Besides this, no other additional work is required by SMEs to achieve traceability between requirements and code. UTC provides the following advanced features for link usage:

Traceability between Requirements & Code: Using inferred links between requirements and code, one can analyze which code contributes to the realization of which requirement. This helps SMEs proving to their customers that they have implemented all requirements.

Requirements Context: During implementation, a developer can look at the requirements context that shows all requirements linked to the currently open code file. Due to agile software development techniques, development teams can change quickly. For example, this feature can support SMEs during development, when new developers are joining and trying to understand the purpose of the implemented code.

Progress of Implementation: Work items have a completion status and are linked to requirements. Thus, work items enable to identify not implemented requirements as well as the progress of their implementation. This helps SMEs to see how far they have already implemented all requirements, as well as identifying not implemented requirements requiring increased attention.

Requirements Impact Analysis: If a SME needs to change a requirement to reflect changed customer demands, all related artifacts potentially affected by this change can be identified. Affected requirements and work items can be identified, e.g. if a change in a requirement is comprehensive, related requirements and their planning of realization described in work items needs to be adapted. An initial set of code files can be identified, which can be a starting point for detailed impact analysis. The changes in the code files can result in additional changes in other code files.

4 Related Work

There exist various commercial tools used by SMEs supporting traceability between requirements and code. We analyzed 12 tools by comparing various criteria, e.g. support for managing requirements, project management and code as well as their ability to (semi-) automatically create links and let developers use these links, especially for impact analysis. In this paper, we only present UTC compared to the best three other tools. Table 1 shows the tools UTC, IBM Rational Team Concert (RTC), Polarion Requirements (POL) and Atlassian JIRA (JIRA). During comparison, an example project with all necessary artifacts was used.

UTC and RTC are based on Eclipse, whereas POL and JIRA are web-based. SMEs prefer integrated tools, e.g. in an development environment (IDE) like Eclipse, which ensure a seamless traceability between all artifacts of the development process and reduced time for switching between different tools. All four tools support requirements specification and project management. However, only UTC supports code files and revisions, whereas all other tools only support code files. Revisions help to identify who changed what in the code files. Furthermore, only UTC supports the manual and (semi-) automatic creation of links, whereas in all other tools links have to be created manually. This means that in all other tools, people from the SME need to link all artifacts by hand, which can, for example, result in increased development time or incorrect traceability links. RTC is the only other tool besides UTC supporting impact analysis, but only for code files. Compared to RTC, UTC supports combined impact analysis between requirements and code. This enables SMEs to estimate the potential impact of a changing requirement to the code, and vice verca.

Table 1: Tools for Tracing Requirements and Code

Criterion	**UTC**	**RTC**	**POL**	**JIRA**
Platform	Eclipse	Eclipse	Web	Web
IDE integration	Yes	Yes	No	No
Requirements	Yes	Yes	Yes	Yes
Project Mgmt.	Yes	Yes	Yes	Yes
Code/Revisions	Yes/Yes	Yes/No	Yes/No	Yes/No
Link Creation	Auto.&Man.	Man.	Man.	Man.
Impact Analysis	Yes	(Yes)	No	No

5 Conclusion

We developed UTC which supports the (semi-) automatic creation of links between requirements and code using project management artifacts. Currently we are in the midst of evaluating our approach in an extensive case study to get a better understanding of the creation and usage of links between requirements and code in practice. We want to compare the effort and quality of our created traceability links to the results of other conducted exploratory case studies.

References

[1] Aranda, J., Easterbrook, S., Wilson, G.: Requirements in the wild: How small companies do it, In RE 07: 15th International Requirements Engineering Conference, pp. 39-48 (2007)

[2] Cleland-Huang, J., Heimdahl, M., Huffman-Hayes, J., Lutz, R., Maeder, P.: Trace queries for safety requirements in high assurance systems, In REFSQ 12: 18th International Conference on Requirements Engineering: Foundation for Software Quality, pp. 179-193 (2012)

[3] Delater, A., Narayan, N., Paech, B.: Tracing Requirements and Source Code During Software Development, In ICSEA 12: 7th International Conference of Software Engineering Advances, pp. 274-282 (2012)

[4] UNICASE Trace Client at Google Code, http://code.google.com/p/unicase/wiki/TraceClient

[5] UNICASE, http://www.unicase.org/