Traceability between System Model, Project Model and Source Code

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

Supported by UNICASE
http://www.unicase.org

Abstract

Traceability from source code to system model elements like requirements has been extensively researched. Even though these approaches use different heuristics and methods to compute traceability links between requirements and source code, they do not return very satisfying and dependable results.

We not only consider the system model, but also the project model, which is used for planning and organization in software development projects.

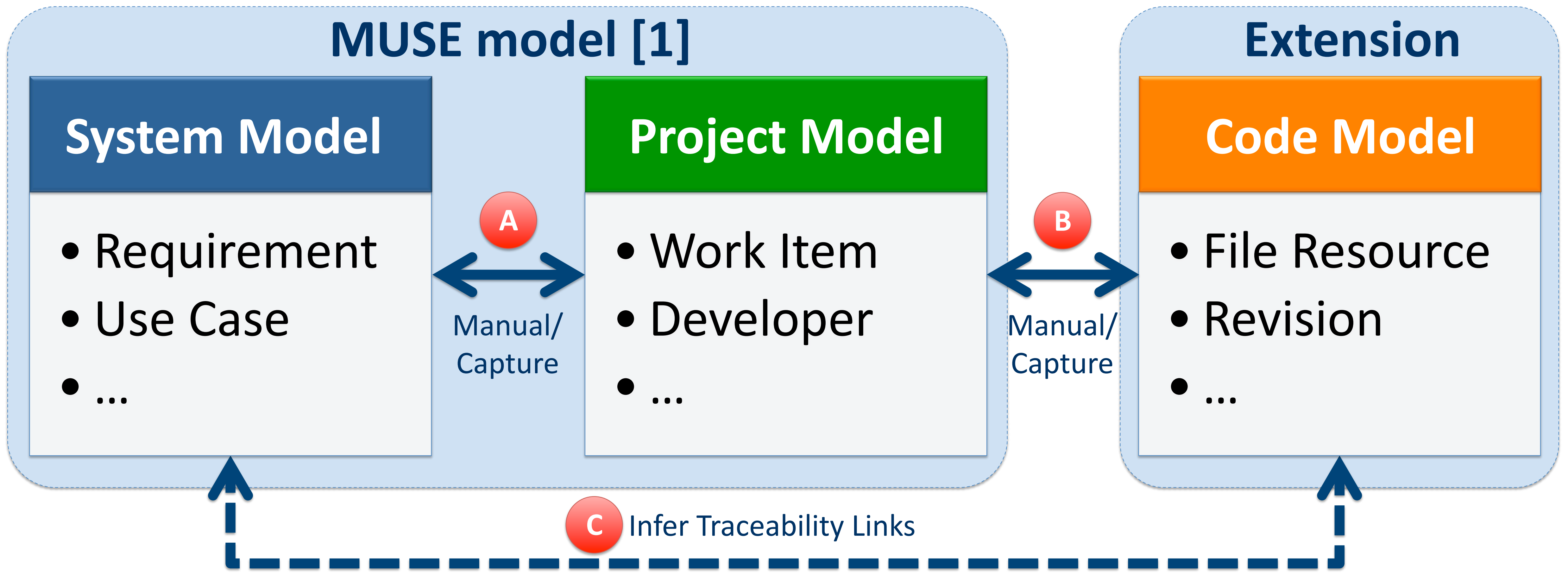
The MUSE model (Management-based Unified Software Engineering) [1] integrates the system model and project model. The MUSE model is implemented in the model-based CASE tool UNICASE [2].

In this thesis, we plan to extend the MUSE model by a code model to support traceability to the source code. We want to create and utilize links between elements from system model, project model and code model.

Problems

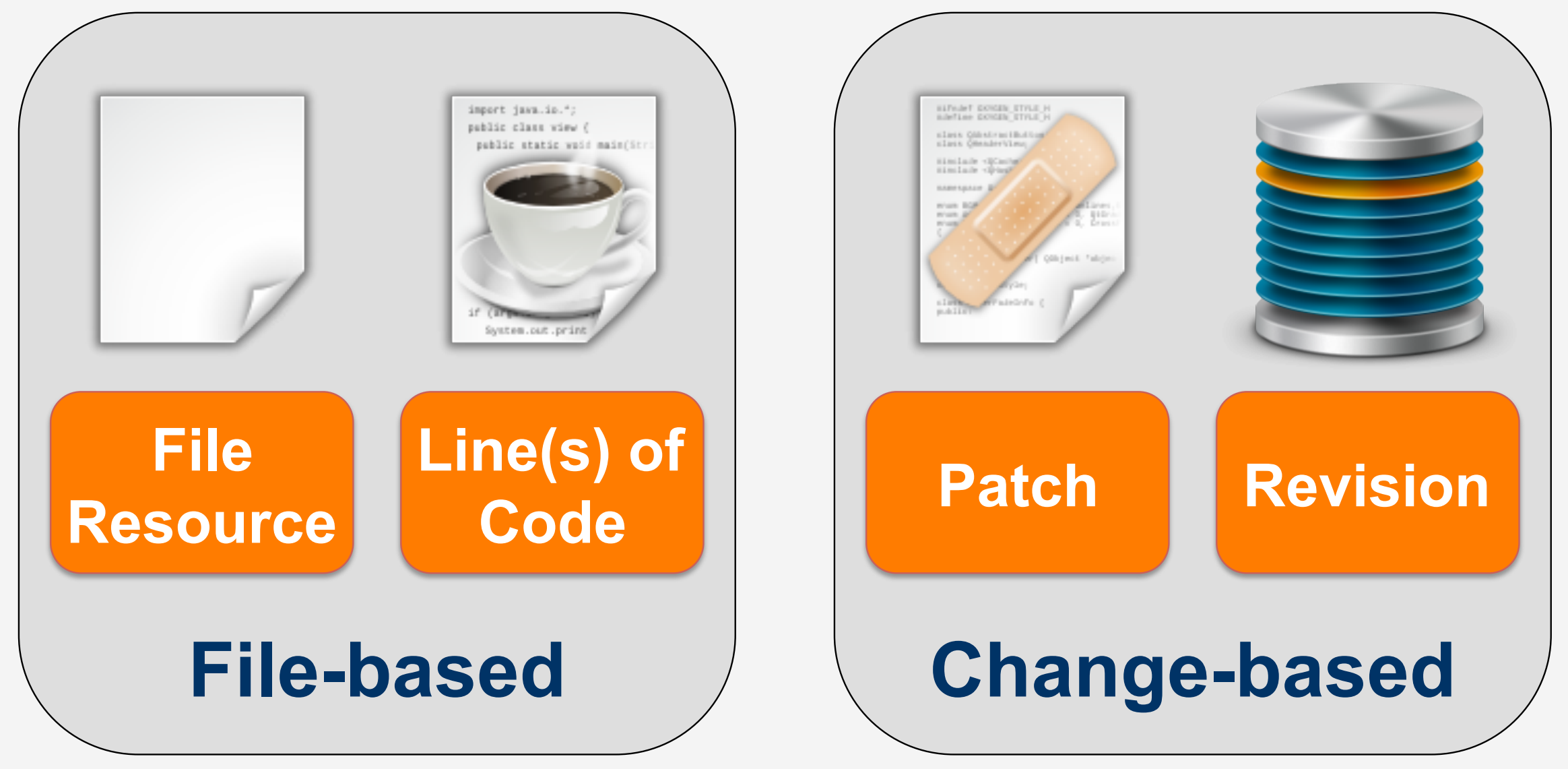
1. Define representations of source code that represent elements of code model.
2. Propose (semi-) automatic approach for creating traceability links between requirements and source code utilizing project model elements.
3. Identify relevant traceability links.
4. Improve impact analysis by using captured and inferred traceability links.

Core Idea



Proposed Solutions

1 Representations of Source Code



2 Capturing & Inferring Traceability Links

A developer selects a work item and starts implementation. While working on the work item, all system elements (e.g. requirements, design documents) the developer looks at during implementation are captured A. After finishing the implementation, all changes in the source code are linked to the work item B. We have to study whether the set of links of one work item supports efficient navigation between all linked elements C.

3 Identifying Relevant Traceability Links

The approach for capturing and inferring traceability links might create a lot of links. Support for the derivation of the most relevant links is necessary. We plan to implement an algorithm that provides a relevance ranking for each link based on the change history of the elements connected by the link. The impact analysis can focus on the most relevant traceability links.

4 Supporting Impact Analysis

We plan to implement an algorithm for impact analysis using the most relevant captured and inferred traceability links. This algorithm bridges the gap between requirements and source code to answer questions as: "What parts of the source code need to be changed based on a change in a requirement?". This algorithm should provide more detailed results during change management than existing algorithms.

Progress

In 2011, we provided (semi-) automatic support for capturing traceability links between project model and code model.

We used patches and revisions in a version control system as two possible types of representation of source code.

Changes to the source code are tracked and when the developer commits some code changes, links between the code changes and the work item are captured.

In 2012, we want to study more options for the representation of source code.

Next, we plan to provide (semi-) automatic support for capturing traceability links between system model and project model in UNICASE. We will implement an algorithm providing a relevance ranking for each link.

We will implement an algorithm for impact analysis using the most relevant captured and inferred traceability links. We plan to evaluate the algorithm using data from the open source project UNICASE.

We expect to finish this thesis by mid 2013.

References

[1] Helming, J., Koegel, M., Naughton, H. Towards traceability from project management to system models. In TEFSE '09: Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, pp. 11-15. IEEE Computer Society (2009)

[2] UNICASE Project, http://www.unicase.org