12.2 A Framework for Change Management of Business Process Models

Christian Gerth (gerth@uni-paderborn.de)
Supervisor: Prof. Dr. Gregor Engels

In model-driven software development approaches, business process models are used at different levels in the development process. For instance, in Service-Oriented Architectures (SOA) high-level business process models become input for the development of IT systems and in running IT systems executable process models describe choreographies of Web Services. A key driver behind this development is the necessity for a closer alignment of business and IT requirements, in order to reduce the reaction times in software development on frequent changes in competitive markets.

Similar to other software artifacts, business process models underlie constant change, i.e., they are created and refined by different business modelers and software architects in distributed environments. This results in different versions reflecting the different views of the involved stakeholders. At some point in time, different versions of a process model have to be compared and merged with each other to obtain an integrated version. This task is called change management of business process models.

Change Management of business process models can be compared to widely used concurrent versioning systems (CVS) for textual documents. However, in contrast to textual documents, that are compared syntactically line by line, the comparison and merging of business process models must consider the graph-like structure of process models and can be improved by considering the semantics of process models.

In my thesis, I develop a framework for change management of business process models that solves the following problems: First, in typical business process modeling scenarios, no change log is available that records changes applied to different model versions. The reasons for this are the use of different modeling tools by different stakeholders and the distributed environment of large software development projects. As a consequence, different versions of a process model need to be compared to identify differences between the versions before they can be merged. For each difference detected, appropriate change operations have to be derived which together can be considered as a reconstructed change log. These change operations must be close to the intended meaning of the change to be understandable by a business user. I approach this problem by decomposing process models into so called single-entry-single-exit (SESE) fragments, which are non empty subgraphs of a process model with a single entry and a single exit edge. These fragments improve the comparison of different process model versions and help to group differences into intuitively understandable compound change operations [1].
Second, to enable a high degree of automation within integration of different process model versions, it is important to understand dependencies and conflicts of changes. Informally, if two changes are dependent, then the second one requires the application of the first one. If two changes are in conflict, then only one of the two can be applied. As a consequence, an approach for computing dependent and conflicting compound changes is required. To address this issue, I capture our set of compound operations in terms of model transformations and apply a critical pair analysis to identify dependent and conflicting transformations [2].

The third problem that I approach in my thesis arises from the fact that change management is a modeling language-dependent problem, i.e., a solution for a particular modeling language cannot be reused easily for another language, due to different syntax and semantics of the languages. To solve this problem, I investigate how a change management solution for a specific business process modeling language can be generalized in terms of a framework for change management of process models [3]. The framework contains an intermediate representation for process models that serves as a common denominator for different process models. Based on the intermediate representation, differences, dependencies, and conflicts are computed. This way, the framework can be instantiated for change management of different process modeling languages.

There are some issues that are currently not covered by my approach to change management of business process models which have to be addressed in future work. For instance, the framework does not support change management across modeling language boundaries, yet. In those scenarios, a (partial) mapping between the meta models of different modeling languages is required.

**Bibliography**

