Chapter 2
Change Impact Analysis for UML Model Maintenance

Anne Keller
University of Antwerp, Belgium

Serge Demeyer
University of Antwerp, Belgium

ABSTRACT

Software maintenance is generally considered to be the most costly phase in the software life-cycle. The software system to be maintained consists of numerous inter-dependent artifacts that inevitably undergo various changes during maintenance. What makes planning and executing these changes difficult is that each change may have severe "ripple effects" to other points of the system that are difficult to assess due to the inter-dependent nature of the artifacts. The goal of this chapter is to introduce a lightweight and accurate change impact analysis technique for UML models. Impact analysis rules are created that trace different relationships between UML model elements depending on the type of change applied. We will show that we can achieve good accuracy. To validate the technique, a change scenario that consists of changes that occur during the resolution of inconsistencies between different UML models (corrective maintenance) was chosen. The validation is performed on two case studies, which together contain approximately 5686 UML model element instances on which 3287 inconsistencies are resolved. The validation of the two case studies returns a mean precision and recall of (0.77, 0.95) and (0.97, 0.93).

INTRODUCTION

Software maintenance is generally considered to be the most costly phase in the software life-cycle (Morrissey, 1979). The software system to be maintained consists of numerous inter-dependent artifacts that, inevitably, undergo various changes during maintenance. In the model-driven engineering context, the artifacts are inter-dependent models and their model elements that describe the system at different levels of abstraction, show

DOI: 10.4018/978-1-61350-438-3.ch002

different views and can even use different formalisms (Kleppe, 2003).

What makes planning and executing these changes difficult, is that each change may have severe "ripple effects" to other points of the system that are difficult to assess due to the inter-dependent nature of the artifacts (Yau, 1978; Yau, 1979). Change impact analysis identifies "the potential consequences of a change" and estimates "what needs to be modified to accomplish a change" (Arnold, 1996). This information can be used to estimate the cost of the changes in terms of time, labor and money, reduce potential errors due to unexpected side effects, and finally, efficiently implement the change.

What change impact analysis techniques have in common is that they calculate a so-called *impact set* from a given *change set*. The change set contains the artifacts that are directly changed; the impact set contains the artifacts that require a subsequent change. In order to identify the impact set, the relationships of the artifacts of the change set to the rest of the system are analyzed. This analysis is done recursively starting from the original change set depending on the desired depth of analysis. The most basic change impact analysis techniques traverse all relationships, however, more elaborate change impact analysis techniques exploit knowledge about the planned changes and the actual relationships. For example, a change impact analysis technique might start from fine-grained, atomic changes (such as create, delete, modify) and chain those together in sequences of changes that have to be applied together. Other techniques require manual intervention of a software engineer to assess whether a change will propagate or not.

Since change impact analysis is an approximate technique, a validation of its accuracy is crucial (Hattori, 2008). Indeed, the danger of each change impact analysis technique is that it produces false-positives, that is, artifacts are identified, as belonging to the impact set, yet will not be affected once the planned change is applied. False-negatives are likely as well, that is, artifacts are affected by a change; yet do not appear in the impact set. Therefore, validation of a change impact analysis is done by counting false-negatives and false-positives, or in information retrieval terms by computing the precision and recall. For this, an experiment must compare the impact set with the elements that are actually affected when applying the changes in the change set.

This chapter presents a lightweight change impact analysis technique for UML models, which makes use of a small, generic set of *impact analysis rules.*

The presented change impact analysis technique traces relationships between UML model elements (OMG, 2004). The kind of change (e.g., add, remove, modify) is used to reduce the amount of relationships to trace. For instance, to determine the impact for the creation of a UML model element, not all relationships but only the containment relationships to this UML model element are traced.

Through the exploitation of the UML meta-model and the type of change we can present a small, generic set of impact analysis rules.

Our change impact analysis technique does not require user interaction to apply the impact analysis rules, yet is reasonably accurate, as we will show in the validation section. Accuracy is crucial to judge the quality of the technique and a prerequisite for its acceptance in praxis. Another contribution of the small, generic set of impact analysis rules is that they can be adapted easily. This holds both for the case where the meta-model evolves as well as for the case where the impact analysis rules themselves need to be adjusted or extended. And finally, the small set of impact analysis rules makes it easy for tool builders interested in supporting change impact analysis for UML models to implement the rules.

This impact analysis technique was developed as part of a framework to support inconsistency resolution for UML models. Therefore, the change scenario chosen to validate the technique