Chapter XI

Specification and Checking of Dependency Relations between UML Models

Claudia Pons
LIFIA, Universidad Nacional de La Plata, Argentina

Roxana Giandini
LIFIA, Universidad Nacional de La Plata, Argentina

Gabriel Baum
LIFIA, Universidad Nacional de La Plata, Argentina

José Luis Garbi
LIFIA, Universidad Nacional de La Plata, Argentina

Paula Mercado
LIFIA, Universidad Nacional de La Plata, Argentina

ABSTRACT

During the object-oriented software development process, a variety of models of the system is built. All of these models are not independent; they are related to each other. Elements in one model have trace dependencies to other models; they are semantically overlapping and together represent the system as a whole. It is necessary to have a precise definition of the syntax and semantics of the different models and their relationships since the lack of accuracy in definition can lead to wrong model interpretations and inconsistency between models. In this chapter, we classify relationships between models along three different dimensions and propose a formal description of them. The goal of the proposed formalization is to provide formal foundations for tools that perform intelligent analysis on models thereby assisting software engineers throughout the development process. In this direction, we discuss...
the construction of a tool based on the formalization and support the verification of traces between requirement and analysis models specified in the Unified Modeling Language.

INTRODUCTION

A software development process, e.g., The Unified Process (Jacobson et al., 1999) is a set of activities needed to transform users’ requirements into a software system. Modern software development processes are iterative and incremental; they repeat over a series of iterations making up the life cycle of a system. Each iteration takes place over time and consists of one pass through the requirements, analysis, design, implementation, and test activities, to build a number of different artifacts (i.e., models). All of these artifacts are not independent; they are related to each other and they are semantically overlapping. Together they represent the system as a whole. Elements in one artifact have trace dependencies to other artifacts. On the other hand, due to the incremental nature of the process, each iteration results in an increment of artifacts built in previous iterations.

Different relationships existing between models can be organized along the following three dimensions:

- Internal dimension (artifact-dimension).
- Vertical dimension (activity-dimension).
- Horizontal dimension (iteration-dimension).

The internal dimension deals with relationships between sub-models that coexist, consistently making up a more complex model. For instance, an analysis model consists of an analysis class diagram, interaction diagrams, and collaboration diagrams. All of the artifacts within a single model are related and must be compatible with each other.

The vertical dimension considers relationships between models belonging to the same iteration in different activities (e.g., a design model realizing an analysis model). Two related models represent the same information, but from different abstraction level. They coexist and should be syntactically and semantically compatible with each other.

The horizontal dimension considers relationships between artifacts belonging to the same activity in different iterations (e.g., a use case is extended by another use case). In this dimension, new models are built or derived from previous models by adding new information that was not previously considered or by modifying previous information.

Figure 1 illustrates the three dimensions described above. It lists the classical activities (requirements, analysis, design, implementation, and test) in the vertical axis and the sequence of iterations in the horizontal axis.

Relationships between models should be formally defined since the lack of accuracy in their definitions can cause problems, for example:

- Wrong model interpretations and discussion regarding the model, i.e., the interpretation done by the user that reads the model may not coincide with the interpretation of the model’s creator.
- Inconsistency among the different models, i.e., if the relationship existing among the different sub-models is not accurately specified, it is not possible to analyze whether its integration is consistent or not.
Towards the Integration of a Formal Object-Oriented Method and Relational Unified Process
Jing Liu, Zhiming Lui, Xiaoshan Li, He Jifend and Yifeng Chen (2005). *Software Evolution with UML and XML* (pp. 101-133).
[www.igi-global.com/chapter/towards-integration-formal-object-oriented/29611?camid=4v1a](www.igi-global.com/chapter/towards-integration-formal-object-oriented/29611?camid=4v1a)