Chapter IV
Meta–Modelling and Graph Transformation for the Definition of Multi–View Visual Languages

Esther Guerra
Universidad Carlos III, Spain
Juan de Lara
Universidad Autónoma de Madrid, Spain

ABSTRACT

In this chapter, we present our approach for the definition of multi-view visual languages (MVVLs). These are languages made of a set of different diagram types, which are used to specify the different aspects of a system. A prominent example of this kind of languages is UML, which defines a set of diagrams for the description of the static and dynamic elements of software systems. In the multi-view approach, consistency checking is essential to verify that the combination of the various system views yields a consistent description of the system. We use two techniques to define environments for MVVLs: meta-modelling and graph transformation. The former is used to describe the syntax of the whole language. In addition, we define a meta-model for each diagram type of the language (that we call viewpoint) as a restriction of the complete MVVL meta-model. From this high-level description, we can generate a customized environment supporting the definition of multiple system views. Consistency between views is ensured by translating each one of them into a unique repository model, which is conformant to the meta-model of the whole language. The translation is performed by automatically generated graph transformation rules. Whenever a change is performed in a view, some rules are triggered to update the repository. These updates may trigger other rules to propagate the changes from the repository to the rest of the views. In our approach, graph transformation techniques are also used for other purposes such as model simulation, optimisation, and transformation into other formalisms. In this chapter, we also discuss the integration of these concepts in the AToM$^3$ tool, and show some illustrative examples by generating an environment for a small subset of UML.
INTRODUCTION

Visual languages (VLs) play a central role in many computer science activities. For example, in software engineering, diagrams are widely used in most of the phases of software construction. They provide intuitive and powerful domain-specific constructs and allow the abstraction from low-level, accidental details. The term domain specific visual language (DSVL) (Pohjonen & Tolvanen, 2002) was coined to refer to languages that are especially oriented to a certain domain, limited but extremely efficient for the task to be performed.

Meta-modelling (Atkinson & Kühne, 2002) is a common approach to describe and generate environments for DSVLs. This technique consists on building a model (the meta-model), which describes the set of all valid models of the language. The meta-model is usually built using visual notations such as class or entity relationship diagrams, plus additional restrictions expressed in textual constraint languages such as OCL (Warmer & Kleppe, 2003).

The increasing complexity of software systems makes a common practice its specification by means of a set of smaller diagrams (that we call system views) instead of including this information in a single, monolithic one. Each one of these smaller models is more comprehensible and cohesive, describing some feature of the system from a specific viewpoint. Thus, ever more frequently, visual notations offer different diagram types in order to describe the various aspects of the system. These notations are known as multi-view visual languages (MVVLs) (Guerra, Díaz, & de Lara, 2005). For example, the UML notation (UML, 2006) proposes a set of diagram types to describe the static and dynamic aspects of the application. Note how MVVLs are not described in a separate way, but the language is described through a common meta-model, which includes the meta-models of the different diagram types and their relation. The meta-models of the different diagram types can overlap, and these common elements indicate how the different diagrams are related to each other. For example, in the UML, classes are defined in class diagrams, but can be referenced afterwards in object and sequence diagrams.

An important aspect in the definition of environments for MVVLs is the support for consistency between their different system views. In this way, when the user modifies some system view, these changes may affect other views, which may have to be changed in their turn. For example, if the same class is present in several views of a UML environment, then changing the type of a class attribute in one view has to be reflected in the other views. Additional syntactic checkings (sometimes known as static semantics) can be defined by means of OCL (Warmer et al., 2003) or graph constrains. Besides syntactic consistency, the environment should also support semantic consistency. This can be achieved in several ways. For example, one may define an operational semantics for the different diagram types, which allow testing the behaviour of the system. Another possibility is to translate the diagrams into a semantic domain for analysis. We called the result of this translation semantic view. For example, in the case of UML, it is possible to define a translation from the different Statecharts into Petri nets for analysis or simulation.

As models and meta-models can be described as attributed typed graphs, they can be visually manipulated by means of graph transformation techniques (Ehrig, Engels, Kreowski, & Rozenberg, 1997; Ehrig, Ehrig, Prange, & Taentzer, 2006). This is a declarative, visual, and formal approach to manipulate graphs. Its formal basis, developed in the last 30 years, makes it possible to demonstrate properties of the transformations. A graph grammar is made of a set of rules and a starting graph. Graph grammar rules are made of a left and a right hand side (LHS and RHS), each one having graphs. When applying a rule to a graph (called host graph), an occurrence of the
Related Content

The Effect of Using Class PC and Interactive Boards on Students’ Academic Achievement and Creative Thinking: Applied Study on 2nd Grade Science Students at Private Schools
[www.igi-global.com/article/the-effect-of-using-class-pc-and-interactive-boards-on-students-academic-achievement-and-creative-thinking/206567?camid=4v1a](www.igi-global.com/article/the-effect-of-using-class-pc-and-interactive-boards-on-students-academic-achievement-and-creative-thinking/206567?camid=4v1a)

Seriously Social: Young Adults, Social Media and News
[www.igi-global.com/article/seriously-social/134408?camid=4v1a](www.igi-global.com/article/seriously-social/134408?camid=4v1a)

Design of Believable Intelligent Virtual Agents
Pilar Herrero and Ricardo Imbert (2005). *Developing Future Interactive Systems* (pp. 177-211).
[www.igi-global.com/chapter/design-believable-intelligent-virtual-agents/8264?camid=4v1a](www.igi-global.com/chapter/design-believable-intelligent-virtual-agents/8264?camid=4v1a)

Social Media Usage, Social Relations, and a Sense of Community in Indonesia