Chapter 5
Expressing and Validating OCL Constraints using Graphs

Najet Zoubeir
Institut Supérieur d'Informatique, Tunisia

Adel Khalfallah
Institut Supérieur d'Informatique, Tunisia

Samir Benahmed
Faculté des Sciences de Tunis, Tunisia

ABSTRACT

The definition of the semantics of visual languages, in particular Unified Modeling Language (UML) diagrams, using graph formalism has known a wide success, since graphs fit the multi-dimensional nature of this kind of language. However, constraints written in Object Constraint Language (OCL) and defined on these models are still not well integrated within this graph-based semantics. In this chapter, the authors propose an integrated semantics of OCL constraints within class diagrams, using graph transformation systems. Their contribution is divided into two parts. In the first part, they introduce graph constraint patterns, as the translation into graphs of a subset of OCL expressions. These patterns are validated with experimental examples using the GROOVE toolset. In the second part, the authors define the relation between OCL and UML models within their graph transformation system.

1. INTRODUCTION

The formal definition of the semantics of visual languages has been the focus of many works, in order to extend the scope of such languages to more critical domains. For example, UML diagrams have been formalized using different formalisms, such as formal specification languages (PVS [Aredo, 1999; Ledang & Souquires, 2001], CSP [Ng & Butler, 2003], Z [Dupuy, 2000; France & Bruel, 2001; France, Bruel, Larrondo-Petrie, & Grant, 1997],...). Among these formalisms, graph transformation systems have known a fair success representing the visual languages seman-

DOI: 10.4018/978-1-4666-3922-5.ch005

tics, since this formalism is formal, universal and easily understood. Many graph-based semantics were proposed for UML diagrams, such as the Dynamic Meta-Modeling approach introduced by Hausmann, which formalizes many of the UML diagrams, such as the statechart diagrams (Engels, Hausmann, Heckel, & Sauer, 2000; Hausmann, 2001), the sequence diagrams (Hausmann, Heckel, & Sauer, 2002; Hausmann, Heckel, & Sauer, 2004) and the activity diagrams (Hausmann, 2005). Other works try rather to define integrated semantics for a number of UML diagrams. For example, (Kuske, Gogolla, Kollmann, & Kreowski, 2002), and (Gogolla, Ziemann, & Skuske, 2003) propose a graph-based integrated semantics for UML class, object and statechart diagrams, and (Holscher, Ziemann, & Gogolla, 2006) works on a larger subset of UML diagrams, including further the use cases and interaction diagrams.

In this context, expressing constraints on UML diagrams, written in the Object Constraint Language (OCL) is studied in many works (Bauer, 2008; Rutle, Rossini, Lamo, & Wolter, 2012; Dang & Gogolla, 2009; Rensink & Kleppe, 2008; Bottoni, Koch, Parisi-Presicci, & Taentzer, 2002). In general, the purpose of these works is to provide a semantics of OCL constraints using graphs. The work in (Bauer, 2008) defines the notion of conditions on attributes in DMM graphs, without considering the OCL syntax, since he does not directly manipulate UML diagrams. In fact, the author considers conditions on graph nodes as an additional refinement of the matching in graph transformation rules. He uses the GROOVE toolset (Groove, 2012) for the representation and manipulation of graphs and constraints. Rutle et al. in (Rutle, Rossini, Lama, & Wolter, 2012) propose constraint aware model transformations based on the diagram predicate framework (DPF), which is a generic graph-based specification framework. The authors propose to define constraints in the transformation rules specified by a joined modeling language, used to join the source modeling language to the target modeling language. The constraints are written in First-Order Logic (FOL), and represented in diagrams by diagrammatic signatures. Although this approach is based on the formal meta-modeling framework DPF, we consider that the proposed constraint semantics (1) is not well integrated with models, since it uses a different notation (FOL), and (2) is restricted to the meta-level, because constraints are used only to control which structure to create in the target model, and which constraints to add to the created structure.

The work of Dang et al. in (Dang & Gogolla, 2009) propose to integrate OCL constraints with Triple Graph Grammars (TGG), and realize this approach by a tool as an extension of USE (UML based Specification Environment). OCL conditions are used in meta-models in order to precisely represent well-formed models, and in models as their properties. However, OCL constraints are represented in the proposed approach in their textual form. So the integration of OCL with TGG is carried out by the tool, and not in a visual form. The work presented in (Resink & Kleppe, 2008) formally extends type graphs to be more compliant with UML. This extension includes a set of object oriented notions such as inheritance and multiplicity, and defines the notion of graph constraints. The work organizes these constraints in categories, such as bidirectionality and multiplicities on associations, and acyclicity of containments, and proposes formal definition for each category. However, the authors do not specify the details of OCL constraints since they do not aim to define a semantics for them, and just classify them as a general category. Bottoni et al. in (Bottoni, Koch, Parisi-Presicci, & Taentzer, 2002) propose an OCL constraints visualization based on collaboration diagrams, and an operational semantics based on graph transformation units. They propose to express constraints by graph transformation rules, such as a constraint satisfaction is represented by the matching between the rule and the instance graph. However, the OCL constraints are manipulated as expressions, without defining how