Chapter IV
Graphical Notations for Rule Modeling

Sergey Lukichev
Brandenburg University of Technology at Cottbus, Germany

Mustafa Jarrar
University of Cyprus, Cyprus

ABSTRACT

This chapter describes various graphical notations for rule modeling. Rule modeling methodologies, empowered with graphical notations, play an important role in helping business experts and rule engineers to represent business rules formally for further deployment into a rule execution system. Rules, represented graphically, can be easier understood by business people and by technicians without intensive technical learning. In this chapter we mainly focus on three graphical notations for rules: UML/OCL, URML and ORM. UML/OCL is a mainstream modeling technology in software development, which is also accommodated by some business experts when modeling a system at the semi-formal, platform independent level. URML extends UML with additional graphical symbols and the concept of a rule, which allows visualization of different rule types on top of UML class diagrams. ORM is an alternative methodology with a rich graphical notation for modeling a domain at the conceptual level. The methodological power, graphical expressivity, and verbalization capabilities of ORM have made it the most popular language within the business rules community. This chapter introduces each of these graphical notations, explain how it can be used, and compare them against each other.

MOTIVATION

In order to deploy rules, initially communicated by business experts in natural language, into a software system, these rules have to be formalized and translated into a rule engine's language. The process of rule translation from informal language into a technical language is usually performed by IT professionals, who might not be familiar enough with the original business

domain. Therefore, the resulting rules may contain mistakes or unintended conceptions. To bridge this gap between IT and business experts, several rule modeling methodologies (empowered with graphical notations) have been proposed. The goal of such methodologies is that rules, represented graphically, can be easier understood by business people and by technicians without intensive technical learning. Through their set of well-defined graphical notations, these methodologies guide rule modelers not only on what to model, but also on validating what have been intended is exactly there.

Furthermore, while doing high-level, conceptual rule modeling and when technical details of a particular rule platform are not yet important, it is easier to work with rules, expressed graphically.

BACKGROUND

The idea of the graphical representation of knowledge is known for a long time and there are several methodologies for rule modeling. The mainstream technologies are the Unified Modeling Language (UML) (OMG, 1999) and the Object Role Modeling (ORM). Speaking about graphical representation of rules by means of UML, it is usually considered in conjunction with the Object Constraints Language (OCL) (OMG, 2003). There is a new upcoming methodology for rule modeling called UML-based Rule Modeling Language (URML) (URML, 2008), which is based on UML and can be used by rule modelers and business experts for graphical modeling of rules.

Before introducing these graphical notations in details, we classify rule types and then give an overview about the major rule modeling approaches, namely UML-based methodologies and the ORM methodology.

Rule Classification

The main types of rules, which we shall consider in this chapter and which are the most used in practice, are integrity rules, derivation rules, production rules, and reaction rules (see Wagner et al., 2004). Rules are considered at three different modeling levels of the Model Driven Architecture (MDA) (MDA, 2008): the Computational Independent Model (CIM) level, which contains semi-formal models, the Platform Independent Model (PIM) level, which contains platform-independent models and Platform Specific Model (PSM) level, which contains implementation-specific aspects of a system. According to (Wagner et al., 2004), derivation rules, integrity constraints, and reaction rules are "*meaningful both as (computation-independent) business rule categories and as (platform-independent) computational rule categories, whereas the concepts of production rules and transformation rules appear only to be meaningful as computational rule categories*". In this chapter we focus on derivation rules, integrity constraints, reaction rules, and production rules, however, transformation rules are not included, since they are a particular case of "technical" rules and from the perspective of a business expert such rules are less important.

Integrity Constraints (Integrity Rules)

An integrity rule, also known as (integrity) constraint, consists of a constraint assertion, which is a sentence in a logical language such as first-order predicate logic or OCL. An example of an integrity rule is "*If rental is a one way rental then its pickup branch is different from its return branch".* The well-known graphical notations for representation of such rules are UML/OCL and ORM. Representing such rules graphically is sometimes problematic for some languages because the general logical formula, which represents a constraint, may have a complex structure, which is not easy to visualize.