Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

Concept and pragmatics of an intuitive visualization-oriented metamodeling tool

Dirk Draheim [b,*], Melanie Himsl [a], Daniel Jabornig [a], Josef Küng [a], Werner Leithner [a], Peter Regner [a], Thomas Wiesinger [a]

[a] FAW-Institute, Johannes Kepler University, Linz, Austria
[b] Software Engineering Group, University of Mannheim, Germany

ARTICLE INFO

Article history:
Received 16 December 2007
Received in revised form
3 March 2010
Accepted 9 March 2010

Keywords:
Visual modeling
Visual modeling languages
Business process modeling
CASE tools
Model-driven architecture

ABSTRACT

In this article we present a metamodeling tool that is strictly oriented towards the needs of the working domain expert. The working domain expert looks for intuitive metamodeling features. In particular, these features include rich capabilities for specifying the visual appearance of models. Our research has identified an important design rationale for metamodeling tools that we call visual reification, which is the notion that metamodels are visualized the same way as their instances. Our tool supports both standard and innovative metamodeling features oriented towards the principle of visual reification. In this paper we present an unbiased discussion of the pragmatics of metamodeling tools against the background of this design rationale.

1. Introduction

In successful projects today, modeling activities are seen in business reengineering, logistics, supply chain management, industrial manufacturing and so on. Models foster communication among stakeholders, because they enforce a certain standardization of the respective domain languages. Therefore, models speed up fulfillment of requirements and then serve as long-term documentation of system analysis efforts. From this perspective, modeling is here to stay. Models add value even when they are not intended as blueprints for software development projects. For example, in major enterprises today huge projects are being directed toward redocumentation of the business process. Research in model-driven engineering is important; however, we have a focus on modeling that is different from model-driven engineering. For example, we focus primarily on the working domain [15] expert. It is often necessary to adapt the modeling method and, in particular, to adapt the modeling language used to the current needs of the domain. It might also be necessary to introduce new modeling elements, to eliminate an existing model element, to add attributes to an existing modeling element, to detail the semantics or to change the appearance of a model element.

2. Motivation and requirements for a visualization-oriented meta- and instance-modeling tool

Unlike most research done in domain-specific modeling, metamodeling and model transformation, our study focuses on the area of business or corporate modeling. In numerous projects, from business process management to enterprise-wide IT architectures, modeling is an essential prerequisite for success. Moreover, it is not possible to

* Corresponding author.
E-mail addresses: draheim@acm.org (D. Draheim), mhimsl@faw.uni-linz.ac.at (M. Himsl), djabornig@faw.uni-linz.ac.at (D. Jabornig), jkueng@faw.uni-linz.ac.at (J. Küng), wleithner@faw.uni-linz.ac.at (W. Leithner), pregner@faw.uni-linz.ac.at (P. Regner), twiesinger@faw.uni-linz.ac.at (T. Wiesinger).

1045-926X/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2010.03.002

achieve sustainable improvements without an appropriate abstraction of the real corporate structures and processes; thus, modeling should be used. On the other hand, modeling must not become an end in itself; rather, it should focus strongly on the elements that need to be analyzed. Otherwise, it will be nearly impossible to maintain the results, considering that corporate structures and processes are frequently subject to change.

As a consequence of organizational changes, the model repository and even modeling methods may have to be adapted to keep them suitable. This adaptation process is more than a tool function; it is an organizational process that has to be implemented. We will discuss this organizational integration [1] in greater depth later in this paper (see also [1]).

An important issue with modeling is that most popular corporate modeling tools do not allow the creation or adaptation of metamodels. This issue is quite interesting if we consider how much effort has been made to "invent" metamodeling methods, and if we look at the list of tools supporting these metamodeling methods. Especially well-established (Meta)-CASE Tools [11,7,5] offer metamodeling features, and the most recent developments, such as Eclipse GMF [25] and Microsoft DSL [26], offer outstanding possibilities for creating domain-specific languages.

To provide metamodeling features for the business domain and directly to end-users, we have determined that the primary principle is intuitivity. This priorization may be the main reason that user-enabled metamodeling is currently not found in business modeling tools. Methods such as OMG's MOF [27] or proprietary methods implemented by, e.g., (Meta)-CASE tools [28], are rarely accepted by users in the business domain. Nevertheless, metamodeling features would add substantial value if applied in a user-friendly, intuitive style.

In this paper, we introduce a metamodeling methodology that is strongly focused on visual representation, supporting what we call the visual reification principle. With visual reification, metamodeling becomes intuitive WYSIWYG modeling rather than an abstract visualization-independent task.

3. Concept of a visualization-oriented meta- and instance-modeling tool

The concept described below is the basis for visual reification. To demonstrate the basics, we introduce modeling elements defined in a generic meta-metamodel. These elements are used to create structurally and visually valid metamodels. If the visual reification principle is applied completely, a metamodel can be seen simultaneously as its minimal valid instance. However, we show that it may not always be adequate to support visual reification to achieve the full power of metamodeling. We present an in-depth discussion about this point in Section 5 below.

Nevertheless, a metamodel is still "only" a modeling-language definition that acts as a schema for instance models. To respect this conceptual separation, we divide the following definition of metamodel into two distinct layers: the meta layer and the instance layer.

3.1. MetaObject

The MetaObject represents an object type on the meta layer. For the definition of attributes, various attribute types are available, such as text, choices, lists, file-system references, URL references and references to other model elements. It is also possible to order the attributes hierarchically. For visualization of the MetaObject, various graphical notations can be created. Attributes can be visualized (e.g., the name should be displayed) relative to the object's bounds or embedded in the object's figure. MetaObjects can also be defined as "abstract." In this case, an instantiation of the abstract MetaObject is not possible. Abstract MetaObjects break the visual reification principle. However, abstract MetaObjects are optional and only available to support the definition of inheritance hierarchies.

Every MetaObject can act as a container for child MetaObjects (see Fig. 1). Several container properties exist to define the container layout, create visual sub-containers and restrict the number of allowed children.

3.2. MetaConnection

The MetaConnection represents a connection type on the meta layer. A MetaConnection is an edge that visually connects exactly two MetaObjects (see Fig. 2). However, under the references copy concept it is possible to state that a connection can connect several different elements. We will discuss reference copies in greater depth later in this paper (see Section 3.4).

Like objects, connections can have different visualizations and attributes. Additionally, it is possible to decide whether the connection is directed and cyclic, as well as to specify its source and target multiplicities. In summary, a MetaConnection can be defined as follows:

- A connection visualizes a relation between model objects.
- A connection is represented as an edge with identifier and direction.

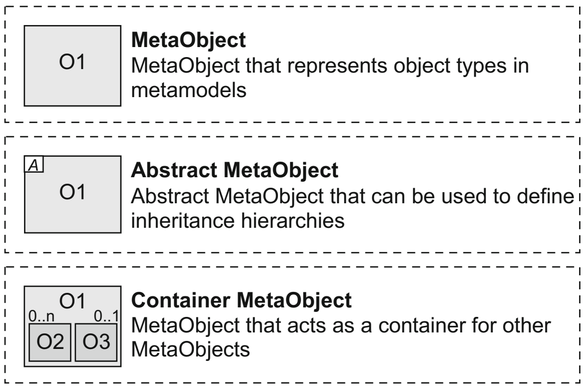
O1
MetaObject
MetaObject that represents object types in metamodels
A
O1
Abstract MetaObject
Abstract MetaObject that can be used to define inheritance hierarchies
O1
0..n
0..1
O2
O3
Container MetaObject
MetaObject that acts as a container for other MetaObjects

Fig. 1. Three types of MetaObjects.