Production of Domain Oriented Graphic Modeling Environments

Bassem KOSAYBA, Raphael MARVIE, Philippe MERLE, Jean-Marc GEIB

Laboratoire d'Informatique Fondamentale de Lille
UPRESA CNRS 8022
59655 Villeneuve d'Ascq
{kosayba,marvie,merle,geib}@lifl.fr

16 May 2003

Abstract

Modeling environments dedicated to specific domains clearly increase the productivity of their users compared to general modeling environments. (especially in the domains where the life of the products is very short). This paper present a meta-tool for generating a graphic modeling tool from a domain's specification.

1 INTRODUCTION

Most current graphic environments which help designers to express their models are rather general and they are aimed at a large number of domains. Thus, they are never really adapted to a specific domain. These environments do not take the specifities of the domains into account and thus must be adapted. Without adaptation, these environments have an impact on the design of the solution suggested by the user. This latter must express the concepts of his/her particular domain using the concepts proposed by the general modeling tool. It becomes significant to specialize the modeling environments for the user's needs. The concepts provided by these environments must be watching those of the user's domain. In other words, the environment must be adapted to the user's domain instead of the user adapts himself to the environment. Domain oriented environments become availables, but they are often produced manually. Their development is complex and expensive. This approach has two limitations:

- The development process is repeated for each domain (a new tool has to be developed).

- The modeling tool manually produced often map its models to a single technology of implementation. When this technology evolves, a new tool thus should be produced because the part that map model to this technology is hardly coded in the tool's program body.

The speed of the development and the quality of the software are very important. Therefore, our goal is to provide a meta-tool to produce automatically graphic modeling tools in order to assist thedesign of applications in a specific domain. This meta-tool needs a specification of the target domain for which one requires to provide a graphic modeling environment. This specification is the domain's concepts, and relations between these concepts. These concepts and their relations vary from a domain to another and can be expressed using meta-modeling.
Moreover, we want the generated graphic tool to be able to produce abstract models independently of technologies of implementation and then to map them to technologic models (for existing technologies or to come). This makes it possible to capitalize models and to follow evolutions of technologies.

In this paper, we present a proposition in order to systematically generate a graphic tool adapted to the user's needs, starting from a meta-model expressing the concepts of a specific domain, and independent of technologies. Section 2 shows the means used to realize our goal. Section 3 presents our motivations and our work. Finally, section 4 concludes this paper.

2 from META-MODELs to SPECIFIC DESIGN LANGUGES

2.1 Our choice : Meta Modeling

A meta-model is used to express the concepts of a specific domain and their interactions. It can be seen as a language making it possible to describe a particular domain. The domain oriented environments use only the domain concepts to define models. So, these environments can follow the rules of the meta-model in order to define models according to the domain's specification.

2.2 The MOF(Meta Object Facility)

MOF is the OMG standard for meta-modeling. The MOF is defined as a four levels architecture. Each level contains information describing the lower level. The application which is being executed, is a group of elements which interact together (M0). These elements and their relations are described at the level of the model (M1). For example, the source of a program is a model which defines the interactions between its components. For describing the model of an application based on components, concepts are required as "component", "port", "connection", etc. These notions are defined at the meta-model level (M2). This

latter is based on the concepts of the MOF ("class", "association", etc.) that are used to describe the concepts necessary to describe models.
Moreover, the MOF provides rules to produce repositories from meta-models. This approach offers means to represent and to store models in the form of an instanciation of the classes of the meta-model. Each repository has an API defined (Mapping) that can be produced automatically.

2.3 The ModFact project

ModFact follow the MOF rules in order to produce a model repository from a meta-mode. Therefore, this repository can be part of a specific modeling environment but there is not a graphic interface to assist this repository (to handle the model stored in this repository or to build it in).

3 PROPOSITION

3.1 Motivation

The models built using directly visual forms which are representations of the domain concepts are easier to be understood and expressed by the people who work in this domain. The repository generated by ModFact is a modeling environment that follows the domain concepts defined in the meta-model and it offers only these concepts to users. So, we work to do a meta-tool which allows, starting from a MOF meta-model to obtain a graphic interface which assists the design in a particular domain and interact with the mdel repository from ModFact.

Also, we search to pass from the model to the code of several technologies of implementation. So, we think that the models produced by these environments will be abstracts, in a way that allow us to map them towards several technologies of implementation (Java, CCM, ..etc.).

3.2 Our Meta-Tool

In the oriented domain graphic modeling environments, the graphic interface must provide a visual form for each concept of the specific domain, and operations to handle these forms. So, we implement a meta-tool that generates a graphic interface starting from a meta-model. In the context of this meta-tool, we propose that the representation of a meta-model concept is fixed according to the MOF concept used to define this meta-model concept. For example, each meta-model's concept defined using the concept "class" in MOF, has a graphic representation as a rectangle containing the name of this meta-model concept. Also, our tool is able to answer all MOF concepts and their interlacing like inheritance between classes, classes composed in others and it present operations to assign the values to the attributes and operations to check if the model is well formed or not

according to the meta-model, for example the generated graphic interface checks if the model entities respect the multiplicity relations "1..*" "m..n" between them.

We have associated one or several graphic components to each MOF concept. The nomber and the nature of those graphic components depend on the nature of the MOF concept ("class", "association", "class composed in another" etc.). And, we organized those graphic components in templates. Our meta-tool detect the MOF concepts existing in a meta-model and generate the graphic components according to each MOF concept. The generation of those graphic components is done by using the according templates using the properties (name, attributes, etc.) of those MOF concepts.

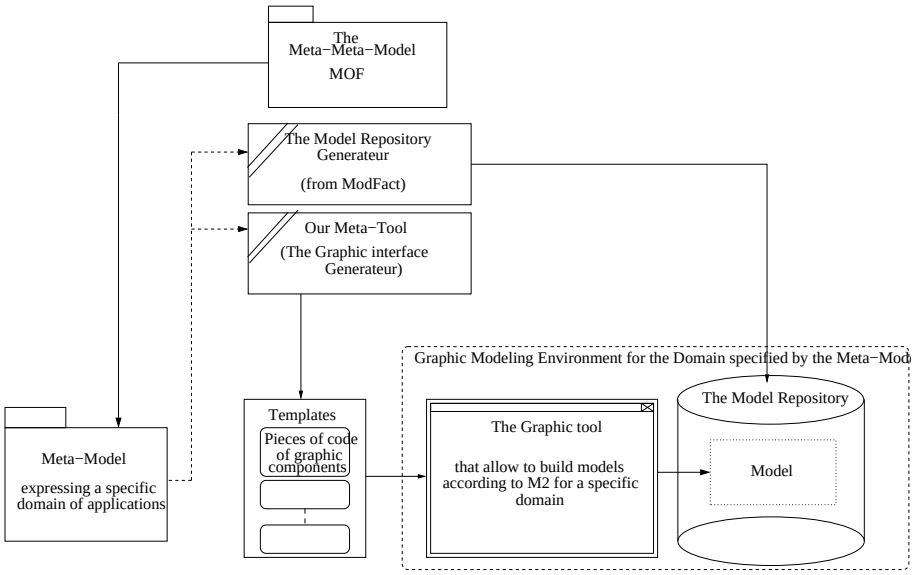

Figure 1: General View

The next step is to bind our generated graphic interface to the repository generated by ModFact. The goal is to generate a graphic modeling tool that is able to store the model built graphically in the form of objects. Such, we can easily (graphically) build the models and map them to the technologies of implementation (mapping from abstract model in the form of objects to pieces of code for technologies).

4 CONCLUSION

This paper discusses the limitations of the current proposals, that it is in the form of general modeling environments or in the form of graphic modeling tools oriented domains but manually developed. Therefore, we try to produce graphic modeling tools oriented domains. This paper discusses a meta-tool to generate graphic interfaces to assist the model repositories generated by ModFact project in order to create domain oriented graphic

environments. We are planing also to specialize the environment to the user (the user can choose his/her own graphic representations for the concepts of his/her specific domain).

References

[1] Mikael Peliter *Transformation entre un profil UML et un meta-modele MOF.*

[2] R. Marvie *Separation of Concerns in Modeling Distributed Component-based Architectures*, In Proceedings of the 6th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2002), EPFL, Lausanne, Suisse, September 2002.

[3] OMG *Meta Object Facilty (MOF) Specification v1.3*, Object Management Group, mars 2000.

[4] X. Blanc *The Specifications Exchange Service of an RM-ODP Framework*, In Proceedings of the 4th International Enterprise Distributing Object Computing Conference (EDOC'00), IEEE Press, Germany, September 2000.

[5] Carine Courbis, Pascal Degenne, Alexandre Fau, Didier Parigot *L'apport des technologies XML et Objets pour un gnrateur d'environnements: SmartTools*,