Template Based MDE

Matthieu Allon, Gilles Vanwormhoudt, Bernard Carré, Olivier Caron

Template Based MDE

Matthieu Allon[1] , Gilles Vanwormhoudt[1,2] , Bernard Carré[1]
and Olivier Caron[1]

[1] University of Lille 1 - CRIStAL Lab. (UMR CNRS 9189)
[2] Mines-Telecom Institute
France, Villeneuve d'Ascq
{Gilles.Vanwormhoudt, Bernard.Carre, Olivier.Caron}@univ-lille1.fr
{Matthieu.Allon}@etudiant.univ-lille1.fr

Abstract

In MDE, design of systems can be improved and accelerated thanks to reusable models which are made available in model repositories or libraries. This paper focuses on the construction and exploitation of "off-the-shelf" model template bases. Model templates are parameterized models which are adaptable to various application contexts. Due to their parameterization, model templates have their own modeling space. In this paper, we present the main construction and composition operations that underlie this space while presenting its dedicated engineering processes and actors. A software environment is shown to illustrate template based engineering in Eclipse.

Keywords: Parameterized Models, Model Templates, Model Reuse, Model Space.

1 Introduction

In MDE, model reuse is a big challenge that aims to facilitate the capitalization of technology independent design efforts and logics ("off-the-shelf" model component libraries [7]), then to accelerate system design and improve their quality. In existing works on model reuse, main approaches are based either on the use of composable model pieces, either on parameterized models which are adaptable to various application contexts [5, 11, 6].

We ourselves contributed to this research by studying model parameterization techniques such as the one offered by the UML "Template" construct. We have defined an approach that allows the design of systems by assembling templates which model reusable functionalities through parameterization [9, 10]. This approach uses an application operator allowing to add functionalities specified by a template to a model after parameter substitution.

Starting from this work, we focus now on the construction and exploitation of model template bases and the related engineering processes. Our objective is to increase the capacities for creating, composing and reuse templates within such bases. Resulting model spaces must be systematically characterized to master and exploit their structuring properties. After a reminder on model templates, we present our vision of model template spaces and the related operations. Then, we describe a software environment in Eclipse to construct and exploit model template spaces in UML.

2 Reminder on model templates

UML Templates [1] allow to capture modeling constructs which expose some of their elements as parameters. Such constructs can be classes or packages (but not only), so called respectively class templates or package templates. To specify its parameterization, a template

owns a signature, which is a list of formal parameters where each parameter designates an element that is part of the templated model. It is the intent of templates to be instantiated, so reused. For template application, the standard defines a specific "template binding" relationship. This relationship allows to specify how the content of a base model is derived from a template through the substitution of its parameters.

In UML, template parameters are only individual and form an unstructured set of model elements of the template so that the construct is general and permissive enough to represent much of model parameterization needs such as the modeling of generic classes (such as C++ templates) [2], the capture of Design Patterns [12], View [5] or Aspect Oriented Modeling [8]. In [14], we proposed a compatible enhancement of UML templates which consists in enforcing templates to have a full model as parameter. The aim of this enhancement is to improve the consistency of templates, notably for aspectual usages, but also to better specify the model of systems to which the functionalities will apply. Following this enhancement, the binding mechanism has been adapted to enable substitution of the model parameter by a conforming substructure of the base model.

Figure 1 gives an example of such an enhanced template as well as its application for injecting stock management functionalities to a base model. One can observe that the template parameters (see the superimposed dashed box) form a full model and one can verify that its structure is well-preserved by the substituted elements in the binding: "Stock" by "Agency", "Resource" by "Car", "identifier" by "name", "ref" by "number" and "in" by "ac". The expected result of this template application is shown in the lower right.

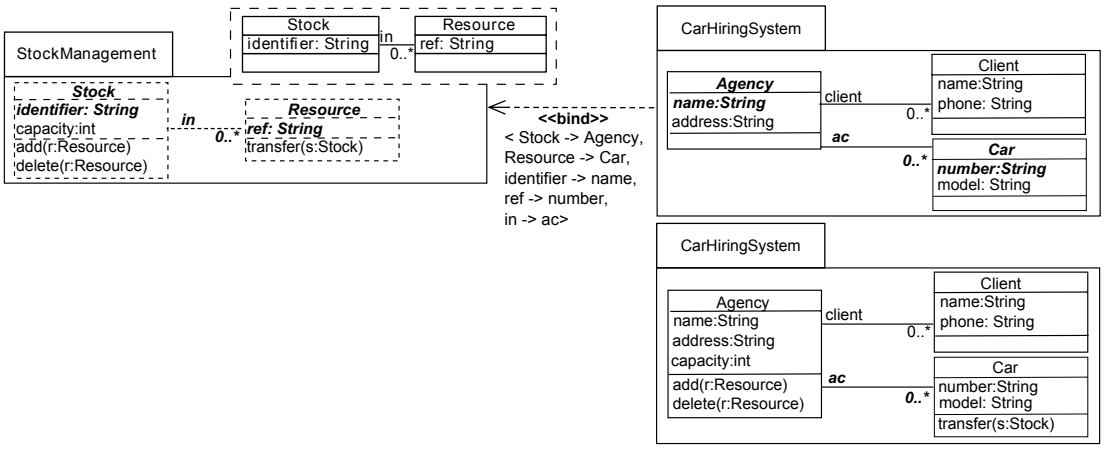

Figure 1: Template application

The previous template construct and application mechanism constitute a useful building block in the context of model reuse. When composed in the large and hierarchically, it allows to design complex systems from assemblies of templates but also to obtain richer templates from existing ones. A formalization in OCL of this semantic variation of the standard has been proposed in [14]. This formalization aims at capturing the common grounding of model templates with the "parameter as model" requirement, so that they can be exploited in any approach which use UML templates [8, 4, 11].

3 Template Based Model Engineering

On the basis of the previous model template technique, specific modeling spaces with their dedicated engineering practices and automatic processes emerge. Fig.2 shows an illustration of such a modeling space with typical involved actors and the respective activities around a model repository containing templates and models that they share.

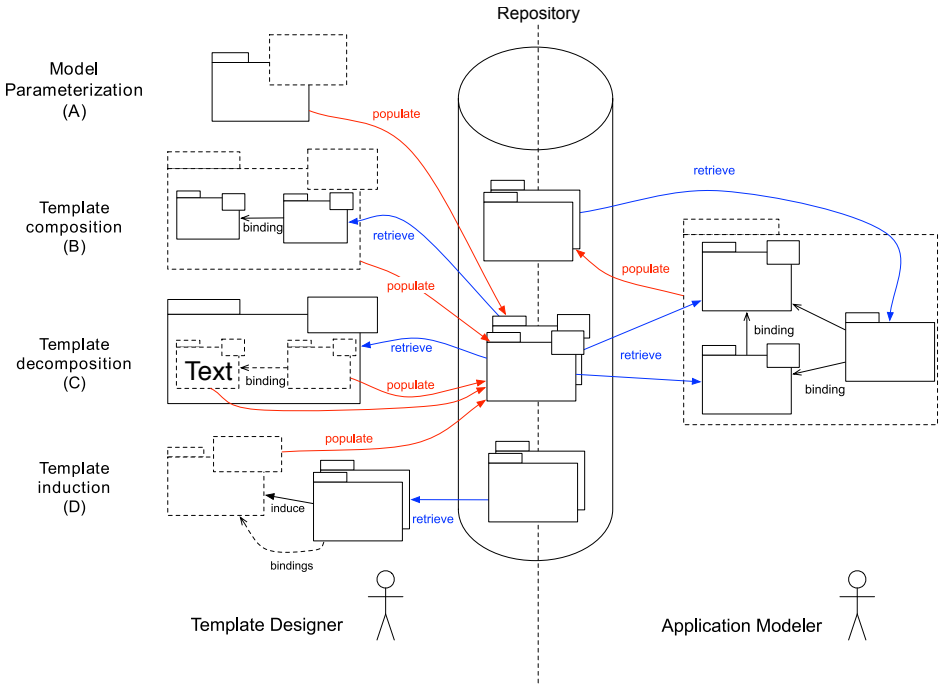

Figure 2: Template Based Model Engineering

In this modeling space, designers of model templates are mainly concerned with "design for reuse" and the constitution of libraries ("off-the-shelf models"). For this purpose, they have to identify candidate model of functionnalities and represent them as model templates for enabling their reuse. Several activities (see Fig.2) are of interest for this engineering task :

- Parameterization of models from the selection of elements corresponding to the model of systems to which the specific functionalities may apply (activity (A)).
- Composition of existing templates in order to build richer ones (activity (B)), notably through partial template binding as this mechanism produces new templates.
- Decomposition of a previously identified complex template (activity (C)) which leads to identify finer ones.
- Induction of new templates from previously designed models of systems which share common functionalities (activity (D)).

Application modelers are much concerned with "design by reuse" methodology (right of Fig.2). From the efforts of template designers they get the possibility to exploit model templates for their application needs in a safe manner through "template binding". Binding activity

is iterative and compositional, allowing the construction of complex systems by successive application of model templates following rich model assemblies.

All these engineering practices must be controlled in an homogeneous and consistent manner. This requires a precise formalization and characterization of model templates as well as their relationships and dedicated operators. More generally, this issue is concerned with the question of model inclusion and model typing within the same meta-modeling space [3]. We are currently investigating this question in the specific context of model templates.

4 EMF technology

We are designing reusable technology and a software environment dedicated to template based model engineering in Eclipse. This environment is composed of plugins which are based on the official EMF (Eclipse Modeling Framework), UML and OCL plugins. These plugins offer core functionalities to specify and verify templates well-formedness and their binding in a compliant way with the UML plugin thanks to a specific profile. In addition, the plugins provide general and original facilities to support other modeling tasks targeting templates or user assistance. For instance, there are facilities to determine the missing parameters in template signatures and bindings. Some facilities also exist for automatic completion of signatures and bindings as well as for inference of parameter substitutions. All the plugins functionalities are reusable for modeling tools that handle model templates[1]. In our case, they were used to build an interactive tool (see Fig. 3).

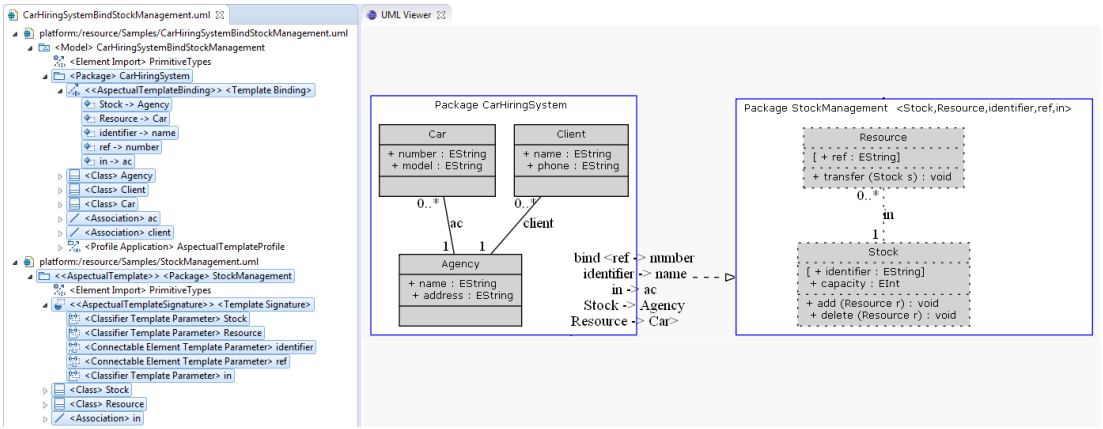

Figure 3: Tool snapshot

Following main features are currently under study in order to support plain template based engineering:

- An engine for determining inclusion and typing relationships between templates and their constituents. Such relationships should permit hierarchical structuring of templates that can be beneficial to many template-based tasks ranging from application to search. For the development of this feature, we will exploit our submodel engine [3].
- A richer set of template operators. Currently, only parameterization and binding with their checking, completion and inference facilities are available. Other operators such as template decomposition and template induction are under study.

[1] `http://www.cristal.univ-lille.fr/caramel/aspectualtemplates`

- Templates searching capacities in model repositories. Using the relationships mentioned previously, a searching system similar to the one described in [13] but specific to templates can be developed. From a selected template, this system will enable to find similar templates with regard to its constituents into repositories and hierarchically present then.

5 Conclusion

In this paper, starting from our previous work, we sketch the model template modeling space and its associated engineering. We presented an environment that illustrates how these two dimensions can be supported practically. As indicated, fundamental issues need to be investigated to the help of full template-based modeling and engineering. This work will contribute to better understanding and generalization of templates for the quest of model reuse and model space structuring.

References

[1] UML 2.4.1 template chapter. `http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF`. [Accessed 19-March-2015].

[2] J. Bigot C. Pérez. Increasing Reuse in Component Models through Genericity. *Formal Foundations of Reuse and Domain Engineering Lecture Notes in Computer Science*, 5791:21–30, 2009.

[3] B. Carré, G. Vanwormhoudt, and O. Caron. From subsets of model elements to submodels: A characterization of submodels and their properties. *Software & Systems Modeling*, 14(2):861–887, May 2015.

[4] S. Clarke and R.J. Walker. Generic aspect-oriented design with theme/UML. *Aspect-oriented software development*, pages 425–458, 2005.

[5] D. DSouza and A.C. Wills. *Catalysis: Objects, Components, and Frameworks with UML*. Object Technology Series. Addison-Wesley, 1998.

[6] D. Del Fabro and J. Bézivin. Generic model management: from theory to practice. In *First International Workshop on Towers of Models - TOWERS 2007 (co-located with TOOLS EUROPE 2007)*, pages 1–9, 2007.

[7] M. Herrmannsdorfer and B. Hummel. Library concepts for model reuse. *Electronic Notes in Theoretical Computer Science*, pages 121–134, 2010.

[8] J. Kienzle, W. Al Abed, F. Fleurey, J.M. Jézéquel, and J. Klein. Aspect-oriented design with reusable aspect models. In *Transactions on Aspect-Oriented Software Development*, volume VII, pages 272–320. Springer, 2010.

[9] A. Muller. *Construction de systèmes par application de modèles paramétrés*. PhD thesis, University of Lille 1, 2006.

[10] A. Muller, O. Caron, B. Carré, and G. Vanwormhoudt. On some properties of parameterized model application. In *Model Driven Architectureâ"Foundations and Applications*, pages 130–144. Springer, 2005.

[11] Y.R. Reddy, S. Ghosh, R.B. France, G. Straw, J. M. Bieman, N. McEachen, E. Song, and G. Georg. Directives for composing aspect-oriented design class models. In *Transactions on Aspect-Oriented Software Development (I)*, volume I, pages 75–105. Springer, 2006.

[12] G. Sunyé, A. Le Guennec, and J-M. Jézéquel. Design Patterns Application in UML. In *Proceedings of 14th European Conference on Object-Oriented Programming (ECOOP'2001)*, pages 44–62. Springer, 2000.

[13] G. Vanwormhoudt, B. Carré, O. Caron, and C. Tombelle. Recherche de sous-modèles. In *CIEL 2014, Troisième Conférence en IngénieriE du Logiciel*, pages 126–130. HAL, 2014.

[14] G. Vanwormhoudt, O.Caron, and B. Carré. Aspectual templates in UML. *Software & Systems Modeling*, dx.doi.org/10.1007/s10270-015-0463-3:p. 29, 2015.