Experience with MOF-Based Meta-modeling of Component-Based Systems

Petr Hnetynka¹ and Frantisek Plasil¹,²

¹ Department of Software Engineering
Faculty of Mathematics and Physics
Charles University in Prague
Malostranske namesti 25, Prague 1,
118 00, Czech Republic
{hnetynka,plasil}@dsrg.mff.cuni.cz
² Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodarenskou vezi 2, Prague 8
182 07, Czech Republic
plasil@cz.cas.cz

Abstract. Component-based development has become a widely used technique for developing not only large enterprise applications, but in fact for any type of applications, including embedded ones. To allow comfortable and easy development, component systems have to provide a rather a big set of development supporting tools including at least a tool for composition and repository for storing and retrieving components. In this paper, we evaluate and present advantages of using MOF and meta-modeling during definition of component system and also during development of the supporting tools. Most of the presented arguments are based on a broad practical experience with designing the component systems SOFA and SOFA 2; the former designed in the classical ad hoc “manual” way, while the latter via meta-modeling.

Keywords: Component-based development, model-driven development, meta-models.

1 Introduction

Component-based development (CBD) has become a well-understood and widely used technique for developing not only large enterprise applications, but in fact for any type of applications, including embedded ones. Using this technique, applications are built by composing already developed components. Every component system (i.e. a system and/or framework allowing to develop and compose components) uses a different view as to what a software component is, but a generally agreed consensus is that “component” means a black-box entity with well-defined interface and behavior. The interface of a component comprises the services provided by it and the services required from other cooperating components and/or an environment (container). To specify its particular view on
components, a component system defines its component model, i.e. a set of abstractions, which together define components, their composition, etc. Thus, the term component has to be always interpreted in the scope of a given component model.

In order to allow really fast and comfortable development and management of component-based applications, component systems should provide rather a big set of development supporting tools and infrastructure. These tools and infrastructure usually comprise of at least a tool for developing and composing components and a repository storing and serving already developed components.

However, creating such an infrastructure is rather tedious and time-and-other-resources-consuming task. This is probably why especially academia-based component systems provided sophisticated component models with plenty of advanced features, but with no or very limited support for real development of components at a large scale. In order to overcome this problem, modern component systems try to heavily employ modeling and meta-modeling approaches that allow automatic generation of many supporting tools.

The component models of classical ("old") component systems were usually defined by an ADL (Architecture Definition Language). Since these ADL languages were proprietary, the development tools were developed completely manually from scratch. Another related problem was that the semantics of a component model had to be typically defined in a natural language. Finally, as the cores of component models had been very similar (in many case in fact the same), a straightforward idea was to allow interoperability between models and use component from one model in another. This issue is not only making components exchangeable between the systems but also it requires interoperable tools to allow developing and managing such heterogeneous applications. But with hand-made tools and infrastructure, the interoperability was quite difficult.

As stated above, the modern component systems usually use meta-modeling approaches to define their component models and, more interestingly, to automatically generate repositories, tools for development, editors for designing and composing components, etc.

Additionally, meta-models provide means for defining semantics in a formal way (at least partially), and also there are approaches supporting easy interoperability and transformations between different models.

All these advantages of the meta-modeling approaches bring faster development and maintenance of the component systems themselves and therefore faster adoption of the systems to the production.

### 1.1 Goal and Structure of the Paper

In this paper, based on our experience with designing and developing component systems (SOFA in particular) and analysis of several existing component systems, such as Fractal and Koala, we present the advantages of meta-modeling approach in component systems. Also we evaluate and compare the meta-modeling approach with the classical one by comparing the SOFA [22] (based on ADL) and SOFA 2 [7] (based on meta-model) component models. To achieve the goal, the