Modeling and Meta-modeling of Software Components

František Plášil¹,² and Petr Hnětynka¹

¹Department of Software Engineering
Faculty of Mathematics and Physics, Charles University
Malostranské náměstí 25, Prague 1, 11800, Czech Republic
{hnetytnka, plasil}@nany.ms.mff.cuni.cz

²Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží, Prague 8, 18000, Czech Republic
plasil@cs.cas.cz

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, the applications are built by composing already developed software 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 a component is 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 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. But with hand-made tools and infrastructure, such interoperability was quite difficult.

As stated above, the modern component systems 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.

In this contribution, based on our experience with designing and developing component systems (SOFA in particular) and analysis of the 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 (based on ADL) and SOFA 2 (based on meta-model) component models.

Frantisek Plasil is a professor and the head in the Department of Software Engineering of Charles University Prague. After receiving his Ph.D. degree from the Czech University of Technology in Prague in 1978, he was with that university until 1994 when he joined Charles University. In the meantime, he held several visiting professor positions including 2 years with the University of Denver, 1 year with Wayne State University Detroit, and 1 year in the University of New Hampshire. Being focused on component based programming comprising middleware technologies, his research activities have been mainly carried out in framework of a number of projects, e.g. SOFA and SOFA 2, ITEA/EUREKA projects PEPiTA, OSMOSE, and OSIRIS, the list of industrial partners of which includes IONA Technologies, France Telecom, Bull Grenoble, Alcatel, and MLC Systeme Ratingen.

Petr Hnetynka is a researcher at the Department of Software Engineering of Charles University Prague. He received his Ph.D. degree in 2005 also from Charles University. In the meantime, he held 1 year postdoctoral researcher position at University College Dublin. He specializes in component based development, especially a design of component models and model driven development. He actively takes part in the projects SOFA and SOFA 2 and in the ITEA/EUREKA projects OSMOSE, and OSIRIS.