Report on PhD Thesis:
Variability of Execution Environments for Components-based Systems

Doctoral candidate: Michal Malohlava, Charles University in Prague
Thesis Supervisor: Dr Thomas Bures, Charles University in Prague
Thesis Reviewer: Pr Laurence Duchien, University of Lille, France

Content and contribution of the thesis

The submitted PhD thesis explores the limits of reuse in the context of component-based approaches by an analysis of contemporary component models. Based on this observation, the thesis introduces a meta-component system that defines a set of commonalities and variation points. This meta-component system could be seen as a software product lines or a configuration system that defines a tailored component system based on a set of requirements in accordance with the target application domain. This system is complete, from modeling to exection environment preparation.

The study presented by Mr Michal Malohlava contributes to the definition of a meta-component system from modeling point of vew to the execution environment and their associated tools. The thesis brings the following contributions described in Chap. 3, Chap. 4, Chap. 5, and Chap. 6:
- A specification of the meta-component system (Chap. 4) based on a rigourous analysis of domain (Chap. 3);
- A model-driven approach based on the meta-component system for execution environment construction by configuration and its corresponding deployment process (Chap. 5);
- A study of modelling interoperability through DSLs for the code generation (Chap. 6).

After a state of the art on modern software engineering artifacts such as configurable component systems, reflective midleware, software product lines and component systems and their associated execution environment, the candidate presents in Chapter 3 an analysis of contemporary components models which support a complete application lifecycle (from design till runtime). He identifies four application domains in which component models are widely used for various reasons: enterprise application, user interfaces, configuration frameworks, and embedded systems. The studied approaches are essentially different on three topics: the component model, the support of non-fonctional requirements, and the form of execution environment and their corresponding deployment process. Secondly, the case-studies continue the analysis by demonstrating three different forms of the execution environmement: JpapaBench, RTSJ connector and SOFA 2 runtime extension. Lessons learned conclude each case-study and a summary gives a set of observations for the roadmap of the next chapters.

The Chapter 4 aims at building a meta-component system that will be the material based of a software product line for creating customized component systems in agreement with the target application domain. Associated tools are based on generative software development technics for automating as much as possible the development. The scope covered by the meta-component system is deduced from the Chapter 3 analysis. The commonalities and the variation points provide the identification of features of the component meta-model and the application requirement configuration tool. Secondly, from the problem space determination and the configuration tool, instances of the meta-component system could be

generated with the right core and optional assets. They define the component system instance used then for generated development and design tools, deployment tools and environment execution. The chapter end on a focus on charateritics of deployment and execution enviroments. The ideas developed in this chapter 4 are very interesting and show the interest of a comprehensive approach from its meta-model and associated variations points till the tools for monitoring runtime performance. It is unfortunate that this chapter is not more developed and justified.

The Chapter 5 elaborates the approach with more details. The candidate uses a model-driven approach named μSOFA method for preparing the configurable execution environment. This configurable execution environment should include the execution infrastruture model with functional and extra-functional concepts and an automated transformation from an input component assembly to the infrastructure model and its realization. An example illustrates the approach by following all the steps of the method. The execution environment model uses micro-components for greater flexibility in the expression of functional elements and infrastructure aspects for extra-functional concepts. The candidate concludes the chapter by a discussion on his choices and defenses the models at runtime, the technology independence and unanticipated back-end extensions.

Finally, the Chapter 6 clarifies the code generation in this context and presents a contribution named EcoGen that is a general control element generation approach based on AST transformation strategies and DSL interoperability. The complete method puts together a set of element specifications described in several DSL languages to generate an implementation in a selected language. The structured approach is interesting because it classifies and unified elements issues from different DSLs languages and generates code controls.

The thesis concludes with an evaluation of the μSOFA method on jPapaBench RTSJ connectors and SOFA 2 Runtime extensions case-studies.

Presentation and writing styles

The thesis is well-written en structured. It takes the result of three major publications during the thesis (Chapters 3, 5 and 6). It would be interesting to better unify text so that it is a whole.

Issues and questions for the defense of the thesis

1. Your choice is focused on micro-components to prepare the execution infrastructure rather than other forms more substantial. This may lead to problems of component compositions difficult to control. Can you argue this choice?
2. In Chapter 3 you did an analysis of second generation component models (EJB, Koala, Fractal Gravity, CCM, JavaBeans, Proactive, etc.) and four application domains for identifying variation points in the meta-component system. Can you say more about the definition of these variation points? At what level of semantics did you work ?
3. In Chapter 4, you define the meta-component system as a software product line for building a component system from the model till the execution enviroment in accordance with the domain requirements. Do you think you can evolve dynamically your execution model? What should you add in your scenario on page 65?

4. In Chapter 7, you give a qualitative evaluation of three examples used in Chapter 3 (jPapabench, RTSJ connector and SOFA2). Have you set up benchmarks to evaluate quantitatively your proposal? if not, could you define which elements could potentially evaluated (code, execution time, memory space, etc.) and how to implement this evaluation framework ? Do you think you could integrate itautomatically in your method and process ?
5.

Judgment

Finally, there is no doubt that the candidate knows his research area, he proposed an original and complete contribution on the variability study of execution environments for component-based systems which he knows the benefits and limitations. The main contribution is a meta-component system that could be instanciated in component systems adapted at each requirement. This thesis present a significant and novel contribution in the areea of component-based environments. The PhD's work has been validated and published in several international conferences and journals. For all these reasons, I recommend the thesis for a defense and judge Michal Malohlava worthy the degree of PhD.

Laurence Duchien
Professeur des Universités, Lille 1
Villeneuve d'Ascq, August 16th, 2012