Formal models of Fractal Component Based Systems for performance analysis

Nabila Salmi*,** Patrice Moreaux* and Malika Ioualalen**

*LISTIC - Université de Savoie - BP 80449, 74944 Annecy le Vieux Cedex, France
{nabila.salmi,patrice.moreaux}@univ-savoie.fr, http://www.listic.univ-savoie.fr
**LSI - USTHB - BP 32, El-Alia, Bab-Ezzouar, 16111, Alger, Algérie
ioualalen@lsi-usthb.dz, http://www.lsi-usthb.dz

Abstract. Component based system (CBS) development is now a well accepted design approach in software engineering. Although specific tools used for building CBS perform several checks on the built system, few of them provide formal verification of behavioural properties nor performance evaluation. In this context, we have developed a general method associating to a CBS a formal model, based on Stochastic Well formed Nets, a class of high level Petri Nets, allowing qualitative behavioural analysis and performance evaluation of the CBS. The definition of the model heavily depends on the (run time) component model used to describe the CBS. In this paper, we apply our method to Fractal CBS and its reference Java implementation Julia, concentrating on performance evaluation. The main interest of our method is to take advantage of the compositional definition of such systems to carry out an efficient analysis, starting from the Fractal architectural description of a CBS.

1 Introduction

Component based technology (Szyperski et al., 2002) is an attractive paradigm, widely used for the development of software and hardware systems. In this paradigm, components are developed in isolation or reused from previous works, and are then assembled to build a *Component Based System* (CBS). Since the mid'70, a lot of component models have been proposed in the literature, among them EJB, CCM and CORBA, COM+/.NET, Fractal (Sun Microsystems, 2007; Object Management Group, 2000; Microsoft, 2007). CBSs are either directly defined by the code of their components or they are built with the help of sets of tools associated to each component model. These tools allow description of the CBS through Architecture Description Languages (ADL) (Medvidović and Taylor, 2000) and provide the architect with several checking tools mainly based on syntactic analysis of the description and the source code of the elementary units of the component model. Beyond this "static" analysis, the complexity of many CBSs requires verification of behavioural properties such as deadlock-freeness, reachability of some states and so on. This is achieved by defining a formal semantics to the component model and by (model) checking required properties against the semantic model of the CBS. We emphasize that such an analysis should be based on a *runtime*