Towards Automated Verification of Grid Component Model

Alessandro Basso[⋆]
[⋆]University of Westminster
Harrow School of Computer Science
University of Westminster
Watford Road
Harrow HA1 3TP
bassoa@wmin.ac.uk

Alexander Bolotov[†]
[†]University of Westminster
Harrow School of Computer Science
University of Westminster
Watford Road
Harrow HA1 3TP
bolotoa@wmin.ac.uk

Abstract

We continue investigation of formal specification of Grid Component systems by branching-time logics and subsequent application of temporal resolution as a verification technique. We aim at the analysis of average case component system, and work on the specification algorithms as well as on the implementation of the verification tool.

1 Introduction

Component models allow for applications to be designed using modules in order to create software easy to be reused and combined, ensuring greater reliability. This is important in distributed systems where asynchronous components must be taken into consideration. The Grid Component Model (GCM) based on Fractal is the one chosen for our research. In these models, components interact together by being bound through interfaces. GCM ensures basic requirements to be developed within specific parameters. However, there is a need for a method which ensures correct composition and behaviour of components. To express this behaviour, different methods might be used: we consider model checking and deductive reasoning. Both methods require a solid logic structure to define components and interfaces, and the first has been already tested in various circumstances, one of which is (Barros et al., 2005). It has been shown that it is possible to build a specification of the behaviour by starting from each specific component to the system of components and their communication. Then the properties of these components are verified by a model checker. This method however, does not consider the environment in which a component system has been developed; in building a large scale distributed system, we cannot afford anymore not to take into consideration the entire infostructure. It is exactly at this point where model checking hits a complexity wall. To specify a component system can be a very large task on its own, and when the infostructure is also be taken into consideration, the issue escalates dramatically. On the other hand, deductive reasoning is not expensive during the specification process, with the complexity raised at the subsequent verification level. We believe that using branching-time specification we should be able to handle also the specification of the entire infostructure of the component system under consideration.

2 Architecture

There are three parts in the architecture, namely the formal specification of the primitive components, the architecture description language and the infostructure specification (see Figure 1). The first two define the behaviour of a components system. It is partially given as an input by the user, and partially is automatically extrapolated using metadata (this section will ultimately be implemented in the GridComp project IDE). The latter would define the environment behaviour and be defined according to users need. Using these three parts, we are able to create a formal specification in SNF_{CTL} as described in (Basso et al., 2006), which is a suitable input format for our deductive reasoning tool.

3 System Behaviour

To extrapolate the system behaviour, we need to gain two pieces of infomation: the architecture and hierarchical components structure, and the functional behaviour of the primitive components. The first one is gained by extrapolating the information from the XML based ADL file. From this file, we need to gather the possible actions of each component's interface. The user might need to refine this process, for example to keep significant parameters only. The second part is to be assured as a joint effort by the user and by the metadata extrapolator. Since one of the GridComp functionality will include a metadata generator when building component models, we can reuse some of the data it provides to blueprint the behaviour of the component in question, leaving the task to the user to fill in the gaps.

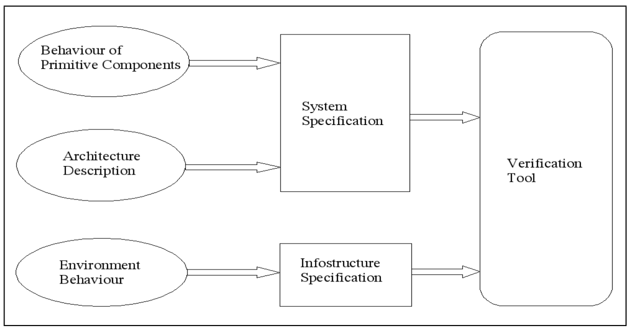

Figure 1: Architecture

4 Infostructure

The infostructure can represent a general purpose environment based on some common grounds, or a specific one, defined by users. Note that in the former case, infostructure must of course leave room for further expansion and adaptation depending on the user's need.

5 Case study

For our average case, we are taking into consideration the same case study analysed in (T. Barros and Madelaine, 2006). We aim at recreating a similar situation as a starting point for our complexity analysis. This case consists of a subsection of a Wifi internet access system developed by the Charles University in Prague. It consists of a client and a wireless network which provides the internet access after the user is authenticated. This subsystem consists of 10 primitive components (basic components).

References

T. Barros, L. Henrio, and E. Madelaine. Verification of distributed hierarchical components. In *International Workshop on Formal Aspects of Component Software (FACS'05)*, pages 51–66, Macao, Oct 2005. Electronic Notes in Theoretical Computer Science (ENTCS).

Alessandro Basso, Alexander Bolotov, Artie Basukoski, Vladimir Getov, Ludovic Henrio, and Mariusz Urbanski. Specification and verification of reconfiguration protocols in grid component systems. In *Proceedings of the 3rd IEEE Conference On Intelligent Systems IS-2006*, 2006. Extended version is available as a CoreGrid technical report 0042, http://www.coregrid.net/mambo/images/stories/TechnicalReports/tr-0042.pdf.

A. Cansado T. Barros and E. Madelaine. Model-checking distributed components: The vercors platform. In *International Workshop on Formal Aspects of Component Software (FACS'06)*, pages 101–114, Prague, Sept 2006. Electronic Notes in Theoretical Computer Science (ENTCS).