CoSMIC: Addressing Crosscutting Deployment and Configuration Concerns of Distributed Real-time and Embedded Systems

Aniruddha Gokhale
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN, USA
gokhale@dre.vanderbilt.edu

Krishnakumar Balasubramanian
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN, USA
kitty@dre.vanderbilt.edu

Tao Lu
Institute for Software Integrated Systems
Vanderbilt University
Nashville, TN, USA
lu@dre.vanderbilt.edu

ABSTRACT

This paper describe a model-driven development (MDD) toolsuite called *Component Synthesis using Model-Integrated Computing* (CoSMIC), which configures and deploys distributed real-time and embedded (DRE) systems using quality of service (QoS)-enabled component middleware. We show how CoSMIC addresses crosscutting configuration and deployment concerns at multiple layers of middleware and applications in component-based DRE systems. We also discuss how CoSMIC leverages model checking and analysis tools to validate key properties for configured DRE systems.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling Methodologies*; D.2.4 [**Software Engineering**]: Software Program Verification—*model checking*

General Terms

Design, Experimentation, Standardization

Keywords

Crosscutting Concerns, Deployment & Configuration, Model-Driven Development

1. INTRODUCTION

Despite advances in quality of service (QoS)-enabled component middleware, such as the CORBA Component Model (CCM) [3] and Real-time CORBA [4], key challenges must be addressed before this middleware can be used to build large-scale distributed real-time and embedded (DRE) systems. For example, QoS provisioning of DRE applications requires appropriate configuration and deployment of system resources, such as (pre)allocating CPUs, reserving network bandwidth/connections, and monitoring/enforcing the proper use of system resources at runtime to meet or exceed application and system QoS requirements. These QoS provisioning activities typically crosscut multiple layers of middleware, OS, networks and hardware.

Conventional middleware lack standard mechanisms, policies, and tools to address these crosscutting concerns. It is therefore unduly hard to configure, deploy, and validate large-scale DRE systems. In particular, prior *ad hoc* approaches to deploying middleware used proprietary mechanisms (such as Perl scripts to deploy applications), lacked standard management interfaces (which greatly reduces flexibility at deployment time), and forced tight coupling between the system being configured and the configurator. With the advent of component middleware, there are now standard configuration and deployment mechanisms, such as the recently adopted OMG Deployment and Configuration (D&C) Specification [5] for component-based DRE systems.

The following challenges remain, however, before component middleware can be used effectively for DRE systems:

1. Lack of tools for effectively composing DRE systems from components. QoS-enabled component middleware enables application developers to develop individual components that can be composed together into *assemblies* that form complete DRE systems. Although this approach supports the use of "plug and play" components in DRE systems, system integrators then face the daunting task of composing the right set of compatible components that will deliver the desired semantics and QoS to applications that execute in large-scale systems.

2. Lack of tools for configuring component middleware. In QoS-enabled component middleware frameworks, application components and the underlying component middleware services can have a large number of attributes and parameters that can be configured at various stages of the development lifecycle, such as (1) *during component development*, where default values for these attributes could be specified, (2) *during application integration*, where component defaults could be overridden with domain specific defaults, and (3) *during application deployment*, where domain specific defaults are overridden based on the actual capabilities of the target system. It is hard, however, to manually ensure that all these parameters are semantically consistent throughout a large-scale DRE system.

3. Lack of tools for automated deployment of DRE systems on heterogeneous target platforms. The component assemblies described above must be deployed in the distributed target environment before applications can start to run. DRE system integrators must therefore perform the complex task of mapping the individual components/assemblies onto specific nodes of the target environment. This mapping process involves ensuring semantic compatibility between the requirements of the individual components, and the capabilities of the nodes of the target environment.

The OMG D&C specification itself is quite complex, with seven

OOPSLA'04, Oct. 24-28, 2004, Vancouver, British Columbia, Canada.
ACM 1-58113-833-4/04/0010.

types of XML configuration files that are tedious and error-prone to write manually. For example, information pertaining to a single component can be spread across four different types of descriptors. As the number of components in a large-scale system increases, these XML files are complex to comprehend without tool support. This complexity is exacerbated by the fact that the information present in these files is inter-related, and hence an error in one descriptor often results in errors in multiple crosscutting descriptors.

The remainder of this paper describes how the CoSMIC [1] toolsuite resolves the deployment and configuration challenges of DRE systems described above, focusing on the generative programming and analysis capabilites of CoSMIC.

2. RESOLVING DRE SYSTEMS DEPLOYMENT AND CONFIGURATION CHALLENGES USING COSMIC

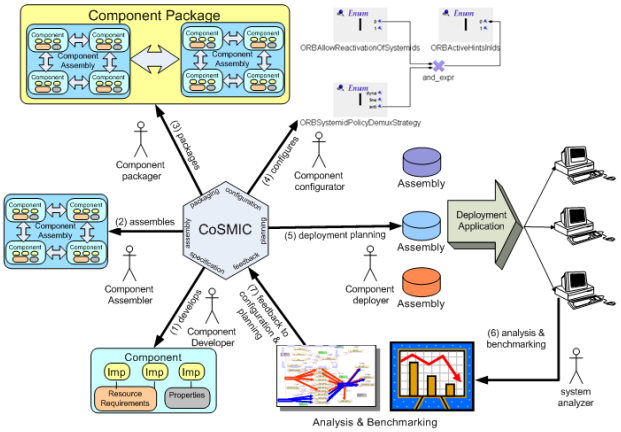

Figure 1: The CoSMIC Toolsuite

Figure 1 illustrates the CoSMIC toolsuite and the remainder of this section describes key tools and capabilities it provides.

Model-driven generative development. At the heart of CoSMIC is the *Platform-Independent Component Modeling Language* (PICML), which is a domain-specific modeling language (DSML) that captures the recurring elements of DRE system design, configuration and deployment. PICML provides generative tools to synthesize the systemic QoS information associated with deployment and configuration as a set of XML descriptors, while enforcing the principle of "correct-by-construction," thereby relieving DRE system developers from design- and run-time mistakes arising due to misconfigurations from manually transforming design artifacts to code artifacts. PICML addresses several concerns including component assembly and packaging, middleware configuration, configuring publisher/subscriber services and component interface generation.

Our MDD-based approach in CoSMIC makes the following contributions to the challenges of configuring and deploying DRE systems using QoS-enabled component middleware:

- It defines and implements a platform-independent modeling language that provides developers with multiple views (*e.g.*, conceptual and logical) of component-based DRE systems. PICML can be targeted to generate systemic QoS data, corresponding to multiple underlying middleware platforms, such as CCM or J2EE.
- It defines and implements a generic data-binding approach that shields DRE middleware developers from having to know multiple XML-based APIs, such as SAX and DOM, and instead allows them to concentrate on other key issues associated with developing their systems, such as using the systemic data to configuring their systems or to annotate these data with QoS properties and pushing them along the pipeline of systemic data processing.
- It implements a standardized mechanism for configuring and deploying component-based applications, with extensions to take advantage of features available in QoS-enabled CCM implementations that provide features needed for DRE systems.

Validating DRE configurations and deployment. The DRE system configurations and deployments synthesized by CoSMIC are validated by integrating CoSMIC with external model checking and analysis tools, such as Cadena [2]. Our tool integration approach is based on the *Web-based Open Tool Integration Framework* (WOTIF) developed as part of the MoBIES program (`www.isis.vanderbilt.edu/projects.asp`) at DARPA. OTIF's tool integration repository stores data in a semantic format understood by one of the communicating tools. Custom semantic translators and tool adaptors can be plugged into the OTIF backplane and used to (1) automatically convert data in a format understood by one tool into of data for another tool and (2) communicate between the tools.

3. CONCLUDING REMARKS

To date we have demonstrated the viability of CoSMIC and the range of DRE systems that can be modeled using this toolsuite by working with industrial partners to develop MDD systems in a number of DRE domains, including avionics mission computing, command and control systems, total ship computing environments, and intelligent warehouses. The source code for CoSMIC including the PICML modeling paradigms and code generators are available at `www.dre.vanderbilt.edu/cosmic`.

4. ADDITIONAL AUTHORS

Additional authors: Jaiganesh Balasubramanian Arvind Krishna Gan Deng Emre Turkay Jeffery Parsons Gabriele Trombetti Balachandran Nataraqjan and Douglas C. Schmidt (All at Vanderbilt University)

5. REFERENCES

[1] A. Gokhale, K. Balasubramanian, J. Balasubramanian, A. Krishna, G. T. Edwards, G. Deng, E. Turkay, J. Parsons, and D. C. Schmidt. Model Driven Middleware: A New Paradigm for Deploying and Provisioning Distributed Real-time and Embedded Applications. *The Journal of Science of Computer Programming: Special Issue on Model Driven Architecture*, 2004.

[2] J. Hatcliff, W. Deng, M. Dwyer, G. Jung, and V. Prasad. Cadena: An Integrated Development, Analysis, and Verification Environment for Component-based Systems. In *Proceedings of the 25th International Conference on Software Engineering*, Portland, OR, May 2003.

[3] Object Management Group. *CORBA Components*, OMG Document formal/2002-06-65 edition, June 2002.

[4] Object Management Group. *Real-time CORBA Specification*, OMG Document formal/02-08-02 edition, Aug. 2002.

[5] Object Management Group. *Deployment and Configuration Adopted Submission*, OMG Document ptc/03-07-08 edition, July 2003.