CoSMIC: An OMG MDA Toolsuite for Distributed Real-time and Embedded Applications*

Aniruddha Gokhale Balachandran Natarajan Jeff Parsons
Krishnakumar B. Tao Lu Boris Kolpakov

ISIS
Vanderbilt University
Box 1829, Station B
Nashville, TN 37235

Contact Author: a.gokhale@vanderbilt.edu

Abstract

The Object Management Group (OMG) has adopted the Model Driven Architecture (MDA) to standardize the integration of the modeling and simulation paradigm with middleware technology platforms. The MDA defines platform-independent models (PIMs) and platform-specific models (PSMs) that streamline platform integration issues and protect investments against the uncertainty of changing platform technology. This technology has been most successful to date notably for enterprise and business applications, where modeling techniques using the Unified Modeling Language (UML) have been integrated with component middleware technologies, such as Enterprise Java Beans (EJB), Microsoft's .NET, and the CORBA Component Model (CCM).

The MDA technology is yet to make its impact in the domain of distributed real-time and embedded applications (DRE) in areas such as avionics, telecommunications, industrial process control and defense. Recent efforts, notably within OMG and some DARPA DoD programs, have started addressing these issues.

CoSMIC is an OMG MDA tool suite tailored to the requirements of DRE applications. CoSMIC addresses the following unresolved challenges in using COTS middleware to build mission-critical DRE systems with time and space constraints:

*Work supported by DARPA PCES grant number F33615-00-C-1695

1. **Lack of middleware composability to support multidimensional QoS:** An increasing number of DRE applications, such as controllers for surface-mount component pick-and-place machines, have stringent QoS requirements that must be satisfied simultaneously in real-time. QoS provisioning in large-scale DRE systems crosscuts multiple system layers and requires end-to-end enforcement. This entails the need for separate, modularized, fine-grained, small footprint DRE middleware addressing individual QoS properties that can then be combined and assembled to provision QoS. However, existing middleware lack the tools to compose middleware components tailored to the QoS needs of DRE applications.
2. **Accidental complexities in integrating software systems:** Ad hoc techniques in determining, composing, assembling, and deploying the right mix of semantically compatible, QoS-enabled COTS middleware components do not scale well as the DRE application size and requirements increase. Moreover, ad hoc techniques, such as manually selecting the components, are often tedious, error-prone, and lack a solid foundation to support verification and validation.
3. **Accidental complexities in configuring middleware:** Existing DRE COTS middleware tend to provide a large number of configuration parameters (or knobs) required to customize and fine-tune the mid-

dleware behavior and performance. However, determining the right combination of semantically compatible knobs from among a large set of available options is fraught with accidental complexities that are a deterrent to the use of COTS middleware in DRE systems. A tools-based approach is needed that allows specification and analysis of DRE application requirements, resulting in automated selection of the right set of parameters that must be used to configure the DRE middleware.

4. **Lack of principled methodologies to support reflective middleware capabilities:** DRE applications most often operate in dynamically changing environments. Therefore, it is necessary for the underlying DRE middleware to be adaptive in order to maintain end-to-end QoS. Adaptation requires instrumenting the middleware to reflect upon the runtime middleware resource usage data and adapting the behavior based on the collected data. Existing practice in configuring these QoS meta data collection points i.e., reflective hooks, within DRE middleware are based on ad hoc manual programming techniques that are error prone and hard to verify. Moreover, these hooks tend to crosscut multiple layers of DRE middleware.

To address the aforementioned challenges, the CoSMIC R&D activities involve the following tasks:

1. Development of aspect-oriented modeling tools for high-level specification of DRE application QoS requirements. The interaction of these requirements serve as the basis for the phases involving analysis and aspect-oriented code generation.
2. Development of analysis tools that analyze the end-to-end QoS and time/space constraints of DRE applications. This goal also requires developing analysis, verification and validation tools for the composite models so that the synthesized middleware is already partially validated, thereby reducing testing costs and project schedules.
3. Mapping the end-to-end QoS constraints specified in the models to the target-specific middleware configuration parameters
4. Development of generators that compose optimized and fine-tuned middleware components to meet the end-to-end constraints. The goal is to generate the right set of customized middleware composed from fine-grained QoS-enabled middleware components.
5. Developing aspect weavers and program transformation tools that can weave in application QoS requirements to compose customized middleware from fine-grained QoS enabled components. The goal is to ensure that changes to the application's QoS properties can be validated and woven transparently without requiring expensive, time-consuming and error-prone recoding and experimentation.
6. Development of additional aspect weavers that instrument synthesized code to collect, organize and present QoS meta-information to the QoS adaptation layer that controls the middleware infrastructure. The model-driven synthesis approach helps the user to specify these properties at the modeling level thereby eliminating the accidental complexities involved in existing ad hoc techniques to accomplish this.

If these capabilities are not realized, complex DRE applications will continue to be built using existing non-validated, non-verifiable ad hoc techniques that are tedious and error prone, and almost always result in unwieldy total ownership costs. Moreover, these non-scalable techniques will continue to be used to provision end-to-end QoS for such systems. This will continue to haunt DRE application developers since build schedules will continue to slip, allocated budgets will continue to be insufficient, and total ownership costs will continue to be insurmountable. This will be all the more prevalent with COTS hardware and software refresh.