Applying Model-Integrated Computing and DRE Middleware to High Performance Embedded Computing Applications

Dr. Douglas C. Schmidt[*] schmidt@uci.edu
Dept. of Electrical and Computer Engineering
University of California
616E Engineering Tower
Irvine, CA 92697

Dr. Aniruddha Gokhale a.gokhale@vanderbilt.edu
Institute for Software Integrated Systems
Vanderbilt University
P O Box 36, Peabody
Nashville, TN 37203

Dr. Christopher D. Gill cdgill@cs.wustl.edu
Dept. of Computer Science
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130

This paper appeared in the proceedings of the 6th Annual Workshop on High-Performance Embedded Computing (HPEC), September 24–26, Boston, MA.

Areas: (1) Automated tools for embedded system development (2) Software architecture, reusability, scalability, and standards (3) Middleware libraries and application programming interfaces.

1 Introduction

Emerging trends. High performance embedded computing (HPEC) systems are widely used for sensor-driven, signal and image processing (SIP) applications with stringent simultaneous quality of service (QoS) requirements, such as predictability, scalability, high performance, and reliability. Applications of SIP include the defense domain (*e.g.*, intelligence, surveillance, and reconnaissance (ISR) and electronic warfare) and medical domain (*e.g.* real-time image processing for magnetic resonance (MR), digital X-ray, and computed tomography (CT)). Sensor types include sonar, radar, visible and infrared images, signal intelligence and navigation assets, and ultra-sound or MRI sensors based on CMOS or CCD technologies.

Due to constraints on weight, power consumption, memory footprint, and performance, development techniques for HPEC application software have lagged those used for mainstream desktop and enterprise software. In particular, HPEC applications have historically been custom-programmed to implement their required QoS properties, making them expensive to build and maintain. Moreover, they are often so specialized that they cannot adapt readily to meet new functional or QoS requirements, hardware/software technology innovations, or market opportunities. To address these problems, there is growing interest in building HPEC systems using commercial off-the-shelf (COTS) hardware (such as COTS DSPs and RISC CPUs) and software (such as real-time operating systems and real-time middleware services).

[*]Currently on leave as a program manager at DARPA's Information Exploitation Office (IXO).

There are, however, a range of COTS choices, including *hardware*, such as CPUs, bus standards, communication standards; *real-time operating systems*, such as VxWorks, MC/OS, and various flavors of real-time Linux; and *component middleware*, such as CORBA CCM, J2EE, and COM+. These myriad options complicate the task of choosing the right mix of COTS hardware and software while balancing costs. Moreover, the high performance demands of SIP applications entails the need for sophisticated parallel processing techniques supported by the underlying hardware, OS, and middleware infrastructure.

Promising solution $\rightarrow$ Model-Integrated Computing. A promising way to address the challenges described above is to apply *Model-Integrated Computing* (MIC) technologies [1] to synthesize, assemble, and deploy QoS-enabled component middleware that is custom-configured automatically for HPEC applications. MIC is a paradigm for expressing application functional requirements and QoS constraints at higher levels of abstraction than is possible with programming languages like C, C++, or Ada. Popular examples of MIC tools are the Generic Modeling Environment (GME) [2] and Ptolemy [3], which are targeted for the real-time and embedded domain.

MIC tools can be applied to analyze different—but interdependent—characteristics of HPEC application behavior, such as performance, predictability, scalability, and safety. Tool-specific model interpreters can then translate the models into the input format expected by analysis tools. Next, these tools can check whether the requested behavior and properties are feasible given the constraints. Finally, MIC tools can synthesize platform-specific code that is optimized for specific platforms comprising standards-based, yet QoS-enabled component middleware, that is custom configured to run on specific real-time operating systems and hardware.

Several emerging technologies are driving R&D efforts to integrate the power of standards-based component middleware and MIC tools to address the needs of HPEC applications:

- **MDA.** The Object Management Group (OMG) has recently adopted the Model Driven Architecture (MDA) [4]. MDA standardizes the integration of the MIC paradigm with a variety of component middleware technologies, including CORBA CCM, J2EE, and COM+.

• **RT-CORBA.** The increasing acceptance of the platform- and vendor-neutral OMG Real-time CORBA (RT-CORBA) [5] model into many distributed, real-time, and embedded (DRE) systems, including avionics mission computing, software defined radios, radar systems, surface mount systems, and hot rolling mills. RT-CORBA provides predictable, efficient, and fine-grained control over system processing, communication, and memory resources.

• **DP-CORBA.** The OMG has also recently adopted the Data Parallel CORBA (DP-CORBA) [6] specification. DP-CORBA enhances the CORBA model by incorporating proven patterns from parallel computing. Highly scalable, parallel computation in DP-CORBA is accomplished via *parallel CORBA objects* that are responsible for segmenting, reorganizing, and disseminating the data (such as SIP data) to individual *parts* that perform the complex computations in parallel.

The confluence of these technologies is making it possible to model the interfaces and semantics of various HPEC application and system components via standard middleware, rather than language-, OS, or hardware-specific features or proprietary APIs. The remainder of this abstract describes how we are applying MDA/MIC to synthesize, assemble, and deploy HPEC applications based on the RT-CORBA and DP-CORBA component middleware.

2 Integrating Modeling and Middleware for HPEC Systems

Figure 1 illustrates the relationships between the MDA model, model-driven synthesis and assembly of application components and infrastructure elements, and our supporting component middleware infrastructure based on MIC tools and DP-/RT-CORBA we have developed at Vanderbilt University, Washington University, St. Louis, and University of California, Irvine. By virtualizing the relationship between the components and the component middleware infrastructure, the model-driven approach allows seamless composition of HPEC application functionality with the mechanisms to enforce desired application QoS properties.

As noted in Section 1, HPEC systems are strongly affected by stringent constraints of embedded endsystems. In particular, the *granularity* of resource availability on each endsystem requires that resources be allocated in a coordinated manner across endsystems. DP-CORBA offers such coordination to achieve high-performance computation by partitioning data and computing subsets of the data in parallel. This work complements research underway on adaptive small-footprint middleware for the Boeing open experimentation platform (OEP) under several DARPA programs. DP-CORBA will also complement research on a QoS-aware CORBA Component Model for the Boeing OEP under other DARPA programs.

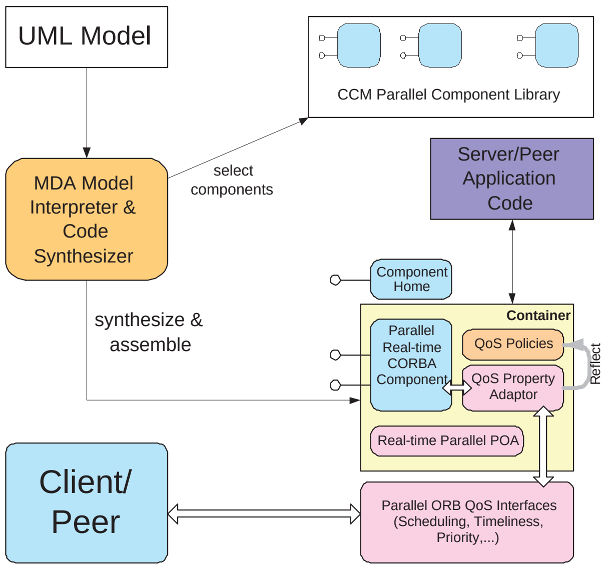
UML Model
CCM Parallel Component Library
select components
MDA Model Interpreter & Code Synthesizer
Server/Peer Application Code
Component Home
synthesize & assemble
Container
Parallel Real-time CORBA Component
QoS Policies
QoS Property Adaptor
Reflect
Real-time Parallel POA
Client/ Peer
Parallel ORB QoS Interfaces (Scheduling, Timeliness, Priority,...)

Figure 1: **Applying MDA/MIC to HPEC Systems**

A key difference between DP-CORBA and other canonical high-performance computing architectures is that DP-CORBA allows distribution for parallel computation on an object-by-object basis *i.e.*, disseminates computation to the part objects. Object-level distribution supports high-performance computing across the finer granularity of resource clustering expected in embedded computing environments. QoS-enabled component middleware, such as RT-CORBA, amplifies the benefits of DP-CORBA by virtualizing resource access and controlling processor, communication, and memory resources end-to-end across endsystems boundaries.

We are currently integrating DP-CORBA with *The ACE ORB* (TAO), which is our widely used open-source implementation of RT-CORBA. Our presentation will discuss our approach and present lessons learned to date on our model-based synthesis and integration of (1) HPEC application functionality, (2) DP-/RT-CORBA component middleware configurations, and (3) data partitioning for parallelization and demonstrate empirically how this architecture offers a powerful and novel approach to HPEC computing environments.

References

[1] Janos Sztipanovits and Gabor Karsai, “Model-Integrated Computing,” *IEEE Computer*, vol. 30, no. 4, pp. 110–112, Apr. 1997.

[2] Akos Ledeczi, Arpad Bakay, Miklos Maroti, Peter Volgysei, Greg Nordstrom, Jonathan Sprinkle, and Gabor Karsai, “Composing Domain-Specific Design Environments,” *IEEE Computer*, pp. 44–51, Nov. 2001.

[3] J. T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, “Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems,” *International Journal of Computer Simulation, Special Issue on Simulation Software Development Component Development Strategies*, vol. 4, Apr. 1994.

[4] Object Management Group, *Model Driven Architecture (MDA)*, OMG Document ormsc/2001-07-01 edition, July 2001.

[5] Douglas C. Schmidt and Fred Kuhns, "An Overview of the Real-time CORBA Specification," *IEEE Computer Magazine, Special Issue on Object-oriented Real-time Computing*, vol. 33, no. 6, June 2000.

[6] Object Management Group, *Data Parallel CORBA Specification*, ptc/2001-11-09 edition, Nov. 2001.