Distributed Object and Component-based Software Systems

Barrett R. Bryant  
The University of Alabama at Birmingham  
bryant@cis.uab.edu

Rajeev R. Raje  
Indiana University Purdue University Indianapolis  
rraje@cs.iupui.edu

This mini-track addresses modeling, interoperability, and middleware design to support distributed object and component-based software systems (DOCS). The mini-track chairs are grateful to the authors and referees for their participation in making this mini-track possible. Each of the 18 submissions was reviewed by 4-5 referees and 9 of these submissions appear here.

In initial construction of DOCS, models and meta-models are created for various domains and components to be implemented for the domains. Current practice is to use UML (Unified Modeling Language). Roussev improves on the lack of seamless integration between different UML models by defining a process to generate formal specifications of the models in OCL (Object Constraint Language). Tan, Zaslavsky, Ewald, and Bond point out that to re-engineer legacy systems in a heterogeneous distributed environment, domain-specific meta-models are required which transcend the heterogeneity among different models. They present such meta-models with associated semantics and translation schemes to support automation of system integration. An additional aspect of distributed systems is Quality of Service (QoS), which includes non-functional issues such as timing and security. Ritter, Born, Unterschüttz, and Weis present an approach to enable QoS modeling and realization for component-based middleware platforms. Their meta-model is integrated with both UML and the CORBA (Common Object Request Broker Architecture) Component Model. Chung and Lee apply modeling processes to web applications. Their approach is UML-based and shows the versatility of modeling across various web application software technologies.

Heterogeneity of DOCS introduces many interoperability issues. Löhr identifies incompatible component interaction styles as a major problem in component interoperability. He proposes a language for describing various component interfaces in order to automatically produce code that mediates among components. Chiang is concerned with component design for reusability across various middleware architectures. He presents an adapter model which allows component interactions to be separated from components themselves and hence facilitates both reusability and component integration. Aleksy, Schader, and Schnell have researched interoperability in messaging between the CORBA Notification and Java Messaging Services. They have defined a bridge to enable message passing between these two services.

Middleware is central to DOCS. Gokhale and Natarajan developed a middleware architecture for the Grid based upon CORBA, improving the software engineering of Grid applications by raising the level of abstraction while providing a high degree of portability and interoperability. Klefstad, Rao, and Schmidt are concerned with memory footprints of middleware implementations, especially problematic for distributed, real-time, and embedded applications. They have designed a dynamically configurable general inter-ORB protocol which minimizes the memory footprint of messaging protocols.