

Towards a Real-time CORBA Component Model

Nanbor Wang, Krishnakumar Balasubramanian, Chris Gill
{nanbor,kitty,cdgill}@cs.wustl.edu
Department of Computer Science
Washington University
One Brookings Drive
St. Louis, MO 63130

Commercial of-the-shelf (COTS) middleware is gaining acceptance in the distributed real-time and embedded (DRE) community as (1) the cost and time required to develop and verify DRE systems preclude developers from implementing these systems from scratch and (2) implementations of the OMG's Real-time CORBA (RT-CORBA) specification mature. Although RT-CORBA standardizes the mechanisms to configure and control underlying OS supports for application's real-time requirements, it does not yet provide sufficient abstractions to separate RT quality-of-service (QoS) policy configurations from application functionality. Developers are therefore forced to configure QoS policies in an *ad hoc* way and the code to configure these policies is often scattered throughout parts of a DRE system, which makes it hard for developers to configure, validate, modify, and evolve complex DRE systems consistently.

The tight-coupling of policy aspects with application logic is inherent to the CORBA 2.x object model. This coupling is carried forward into derivative specifications that focus on QoS properties, such as the Real-Time CORBA 1.0 and 2.0 specifications. The OMG has addressed many of the limitations of the CORBA 2.x object model in the CORBA Component Model (CCM). However, QoS properties are not considered in the CCM specification.

In the CCM, application functionality can be defined in *components* installed in generic application servers. Application servers use *containers* to interact and provide QoS support for installed components. Component composition and QoS requirements are specified using metadata stored in XML format that need not be committed by the developers until deployment time. Exposing this information in XML format allows us to extend the standard Document Type Definitions (DTDs) in the CCM to support custom policies unobtrusively using distinct namespaces.

It is our position that the DRE community can benefit from CCM capabilities to provide a more robust DRE development environment. By extending the CCM to support component deployment metadata for QoS policies such as real-time priorities, DRE systems can be composed from existing components while applying various QoS policies. This capability not

only reduces the cost of developing DRE applications, it also makes it possible to analyze the consistency of QoS policies applied throughout a system.

To validate our approach, we are developing an implementation of the CCM called the Component-Integrated ACE ORB (CIAO), which combines mechanisms to configure Real-time CORBA features more flexibly and robustly. Below, we outline how we support deployable QoS policies in CIAO:

- A *component descriptor*, among other information, describes the type of container a component expects. We are extending the CCM component descriptor to specify the RT-CORBA **Priority Model** that a deployed component can accept.
- An *assembly descriptor* specifies how to compose and deploy an application server. We are extending the CCM assembly descriptor to support the following RT-CORBA mechanisms:
 - Overriding the priority model and protocol properties of an installed component, *e.g.*, for client-specified versus server-specified priority
 - Specifying the priority level of an installed component.
 - Defining thread pools that an application server should create (with or without lanes) and associating them with installed components.
 - Specifying the QoS policies associate with component inter-connections. These policies specify whether a connection should be pre-connected using `_validate_connection()`, have priority bands, or avoid demultiplexing the connection by specifying it as private.
- An *assembly* is a collection of component implementations and metadata that defines an application server. We are extending an assembly to include the custom RT-CORBA priority mapping implementation that an application server should use to provide another dimension in adjusting priority levels. Likewise, custom protocol implementations can also be included in an assembly.

This presentation will explain by example how the CIAO capabilities are developed and will present empirical results illustrating their performance and predictability for representative DRE application use cases. This research is sponsored by Boeing under the Bold Stroke project.