Applying Meta-Programming Techniques to Dynamically Order Equivalence Classes in Open Distributed Real-Time Systems

Angelo Corsaro, Ron Cytron, and Chris Gill
{corsaro, cytron, cdgill}@cs.wustl.edu
Department of Computer Science
Washington University
St. Louis, MO 63130, USA

The maturation of the CORBA specification and standards-based CORBA implementations has simplified the development of open distributed systems with complex functional requirements. The recent emergence of the Real-time CORBA (RT-CORBA) specification and standards-based RT-CORBA implementations should likewise simplify the development of open distributed systems with complex quality of service (QoS) requirements. For example, future combat systems will involve heterogeneous collections of mobile autonomous robots that must collaborate to perform coordinated maneuvers in support of time-critical missions, such as reconnaissance, perimeter defense, and suppression of enemy air defense. Other applications that will benefit from real-time middleware are distributed multimedia applications, distributed network games, distributed real-time databases etc.

A key challenge that arises in these types of distributed real-time systems involves communicating and reasoning about the relative importance of various entities that compete for system resources at a given point in time. Resolving this challenge is essential to build distributed and real-time open systems (i.e., systems where components can connect and interoperate in a flexible manner without having to be preconfigured statically) that can also preserve key end-to-end QoS properties, such as timeliness and resource constraints. For instance, mobile autonomous robots should be able to collaborate in a predictable and efficient manner, despite the fact that their scheduling policies and implementations may not be homogeneous.

The forthcoming Dynamic Scheduling Real-Time CORBA Joint Revised Submission (DSRT-CORBA JRS) addresses some aspects of the challenge outlined above. For example, the DSRT-CORBA JRS defines a distributable thread mechanism that enables the thread of control—i.e., an entity—to transition from ORB to ORB while serving an operation request. A distributable thread competes with other entities for the use of different resources (such as CPU time, memory, or network bandwidth) in the various ORBs it traverses in a dynamic call graph. Each distributable thread contains certain scheduling information that are embedded with other fields in a GIOP request. This information can be used by the ORBs visited by the thread to ensure the thread is processed with the appropriate priority end-to-end.

However, the DSRT-CORBA JRS does not fully address the interoperability aspect of the challenge outlined above. In particular, it consciously under-specifies how to map scheduling parameter elements when distributable threads pass through ORBs that are configured with (1) heterogeneous scheduling policies or (2) different scheduling parameters for the same scheduling policy. As yet there is no official standard proposal concerning how to provide interoperability between heterogeneous (but still composable) schedulers, which potentially limits the openness of distributed real-time systems using DSRT-CORBA JRS middleware.

We believe that meta-programming techniques can resolve the problems outlined above by enabling dynamic ordering of priority equivalence classes. The equivalence classes consist of entities that have the same relative importance at any given point in time. Each entity will reflect properties by means of a meta-layer that can enable DSRT-CORBA JRS implementations to enforce request orderings, even in the face of heterogeneous scheduling policies and implementations. Although we focus on DSRT-CORBA JRS, our approach applies to any situation that require a partial ordering of entities where entities belong to different domains or their properties are expressed using different descriptive mechanisms, such as the scheduling properties that describe different algorithms, or scheduling properties that describe the same policies but using different attributes.

The key components of our solution are comparators and adapters. Comparators encapsulate the logic needed to compare the criticalness of different entities. Different comparators can compare combination of entities described using different scheduling parameters. Adapters convert scheduling parameters used by a given scheduling policy into those used by another. The use of a meta-layer that describes the scheduling properties makes the conversion easier and more flexible. Our solution approach enables both the installation of custom comparators/adapters, while simultaneously enabling ORB developers to specify how to map universal scheduling properties like deadlines, periods and so on to the scheduling parameters used by the scheduling policies available in their ORB.