

QuA: a QoS-Aware Component Architecture

Richard Staehli¹ Frank Eliassen¹ Gordon Blair²
Jan Øyvind Aagedal³

¹*Simula Research Laboratory*, ²*University of Tromsø*, ³*SINTEF*

{richard, frank}@simula.no, gordon@comp.lancs.ac.uk, jan.aagedal@sintef.no

QoS-Safe Deployment. A component architecture guarantees that components will function correctly when deployed on any sufficiently provisioned architecture-compliant platform. We refer to this as the *safe deployment property*. Unfortunately, current architectures have well known limitations: closed platform services are inefficient for applications like streaming continuous-media, real-time constraints are not communicated, and tolerance for imprecision or other data loss cannot be traded for better real-time performance.

We refer to applications that tolerate bounded error or imprecision as *QoS-sensitive applications*. Such applications cannot be safely deployed on current component architectures without careful engineering of deployment resources and communication protocols. How can we extend component architectures so that the safe deployment property is preserved for QoS-sensitive applications?

The crux of the problem is that *application QoS* is a property not of a component, but of a deployed composition of components. In current architectures, application QoS is a function of both services provided by the middleware platform and of deployment descriptors provided by application developers. Neither the platform nor the application developer have enough information to guarantee safe deployment.

Platform-Managed QoS. The QuA project is investigating the idea that an open platform can effectively assume all responsibility for QoS management.

In the QuA architecture, applications declare instantiation and binding requirements in service specifications. A service specification identifies both new and existing objects that participate in a service, and declares their initialization, composition with other objects, and exported interfaces. New

objects in a specification are identified only by type, leaving the platform free to select an appropriate implementation. A type may be implemented by a primitive software component, or by another service specification, allowing arbitrarily complex compositions.

The *QuA Platform* handles service creation requests as a meta operation. A service request includes a *service specification*, *quality specification*, and *input guarantee* [1]. The platform has the responsibility for identifying implementation alternatives that satisfy the required service types and for planning a configuration that best satisfies the quality specification.

Open Service Planner Architecture. The QuA architecture builds on the experience of the OpenORB project [2] by adopting an open-reflective model to enable runtime platform configuration. The key innovation of the QuA component architecture is the pluggable service planner. The QuA Platform delegates the task of service creation to the service planner associated with the requestor's service context. We believe QuA will support non-QoS-sensitive applications efficiently with a simple generic planner while allowing system administrators to configure a QoS-sensitive application with specialized planners that encode as much knowledge as needed to optimally satisfy service specifications.

References

- [1] Richard Staehli, Frank Eliassen. QuA: A QoS-Aware Component Architecture. Technical Report Simula 2002-13, Simula Research Laboratory, 2002.
- [2] G. Coulson, G. S. Blair, M. Clarke, N. Parlavantzas. The design of a configurable and reconfigurable middleware platform. *Distributed Computing*, 15:109–126, 2001.