

# Quality of Service Engineering with UML, .NET, and CORBA

Torben Weis, Andreas Ulbrich, Kurt Geihs  
TU Berlin, iVS, Einsteinufer 17, 10587 Berlin, Germany  
{weis,ulbi,geihs}@ivs.tu-berlin.de

## Abstract

*The concern for non-functional properties of software components and distributed applications has increased significantly in recent years. Non-functional properties are often subsumed under the term Quality of Service (QoS). It refers to quality aspects of a software component or service such as real-time response guarantees, availability and fault-tolerance, the degree of data consistency, the precision of some computation, or the level of security. Consequently, the specification and implementation of QoS mechanisms has become an important concern in the engineering of distributed applications. In this tutorial the attendees will learn how non-functional requirements can be engineered in a systematic way into applications on top of distribution platforms such as CORBA and .NET. The tutorial focuses on two major subjects areas: (1) Specification of QoS properties and (2) implementation of QoS mechanisms in middleware. We present a comprehensive, model-driven approach. It starts with a platform-independent model (PIM) in UML that captures the application QoS requirements. This model is mapped by a tool to a platform-specific model (PSM) tailored for a specific middleware, which is extended with the corresponding QoS mechanisms. Finally, the PSM is translated to code. Participants in this tutorial will get a thorough understanding of general QoS requirements, QoS modeling alternatives and QoS mechanism integration in respect to popular distributed object middleware. Furthermore, we will discuss the pros and cons of CORBA and .NET for QoS engineering. A tool will be demonstrated that eases substantially the modeling stages and the code generation.*

## 1. Motivation

The concern for non-functional properties of software components and distributed applications has increased significantly in recent years. Users and managers demand guaranteed quality properties for software products, and software developers face non-functional requirements that

they need to pay attention to in the design and implementation of distributed software systems. The objective of this half-day tutorial is to teach the attendees guidelines on how to handle non-functional properties (QoS) in distributed systems throughout the entire software development life-cycle.

QoS-enabled middleware platforms support the development of QoS-aware applications. However, middleware can not completely shield applications from the implications of QoS. Therefore, QoS provision affects the application design, too. Hence, the tutorial consists of two sessions. The first session focuses on the design of QoS-aware applications while the second session concentrates on middleware QoS issues. Thus, we aim at the conceptual foundations as well as the implementation implications. A new UML modeling tool [3] and our .NET QoS extension [2] are used during the workshop to illustrate the material. The tools will be available for download after the tutorial.

## 2. Part I: Modeling QoS with UML

There is a large gap in complexity between QoS specification and QoS realization. Furthermore, different QoS-enabled middleware platforms demand special considerations by the designer. Hence, a model-driven approach [1] can simplify the design and development. Application designers can start with a platform independent model (PIM) that captures QoS properties independently of a concrete realization [4]. A model transformation automatically maps the platform independent model to a platform (middleware and implementation language) specific model (PSM).

The advantage of this approach is two fold. First, the gap in complexity can be bridged (at least partially) by a tool that simplifies the development and captures expert knowledge in the model transformation process. Second, the platform independent model can be much clearer since it is not affected by realization details.

The platform independent model presented in the tutorial builds on the proposed UML2 component model since the UML 1.4 component model has several weaknesses. Most notably, the concepts of ports and connectors are missing.

The presented approach provides means to specify QoS properties for tasks. This concept is much more powerful than attaching QoS properties to single methods only. Especially in large distributed enterprise applications the performance or availability of a single method invocation is not that interesting. The important question is whether a certain task can be finished or how long the processing of the task will take. Our approach uses the improved hierarchical message sequence charts (HMSC) to model such tasks and their QoS properties.

We have developed a new CASE-tool that supports these new modeling techniques. Furthermore, the tool supports model transformation. Such transformations are simple scripts that can be adapted by users to customize the transformation process.

### 3. Part II: QoS in Object Middleware

QoS provisioning happens at application and at middleware level. Current standard middleware platforms such as CORBA or .NET do not provide generic QoS support. Hence, they have to be extended.

Some CORBA ORBs have built-in extensions for special QoS categories such as real-time or fault tolerance. In contrast, .NET Remoting is a new middleware platform that is part of Microsoft's .NET SDK. .NET does not feature built-in QoS support and its default use of SOAP and HTTP for message transmission has been considered to be a major drawback for performance oriented applications. However, due to the inherent modularity and component-orientation of .NET, its transport mechanisms can be simply exchanged by more efficient ones. Together with the advanced features of the .NET runtime it is a viable candidate for QoS support.

Our research on the CORBA compliant MICO ORB has shown that QoS-enabling a CORBA ORB usually requires changes to the ORB's implementation. Such ORBs usually introduce an extended IDL or some aspect language to specify QoS properties. New languages or language extensions increase the learning curve.

That is not the case for .NET. DotQoS [2] - our QoS extension for .NET - is built on top of a standard .NET SDK and does not require changes to the implementation of .NET. Furthermore, DotQoS does not require any IDL or aspect language for specifying QoS properties. This is possible due to .NET's support of meta-data, reflection and communication channels. Meta-data can be used to enrich the source code with information about required QoS-properties. Using reflection DotQoS can determine at runtime which QoS mechanisms have to be used for a method invocation. Finally, .NET Remoting allows controlling almost every detail of a remote method invocation by inserting new or replacing existing message handlers in the communication channel.

## 4. Outline

### 1. Introduction

The tutorial starts with an introduction to the QoS terminology and highlights the difficulties in building QoS-aware applications. Then we motivate the usage of a model-driven approach in the realm of QoS-aware applications.

### 2. QoS Specification

An UML extension is presented that supports QoS specification in a platform independent application design. The usage of this extension is illustrated with examples. Finally, a brief tool demonstration shows how such QoS-aware models can be created conveniently and how to transform them to standard UML models.

### 3. QoS Implementation

QoS must be supported by the middleware platform, too. This section discusses how .NET and CORBA can be enhanced with QoS support. The architecture and extensibility of .NET Remoting and CORBA are analyzed and differences discussed. We illustrate how to utilize the advanced features of .NET for QoS provisioning, negotiation and monitoring by means of our DotQoS middleware.

### 4. Conclusions and Final Discussion

We summarize the tutorial material and comment briefly about future directions of the field. An open discussion concludes the tutorial.

## 5. Who Should Attend

The tutorial aims at software engineers, architects and developers who are concerned with innovative software projects and explicit quality requirements. Participants will learn how to approach the engineering of non-functional properties into software, how to model and refine such properties, and what it takes to realize such properties systematically with popular middleware architectures.

## References

- [1] OMG. Model-Driven Architecture, <http://www.omg.org/mda>, 2003.
- [2] A. Ulbrich and T. Weis. [www.dotqos.org](http://www.dotqos.org), 2003.
- [3] T. Weis. [www.kase-tool.org](http://www.kase-tool.org), 2003.
- [4] T. Weis, C. Becker, K. Geihs, and N. Plouzeau. A UML Meta-model for Contract Aware Components. *Springer LNCS*, 2185, 2001.