Middleware - Quo(S) Vadis?

Kurt Geihs
TU Berlin

Overview

> Motivation and Requirements

> Middleware QoS Approaches

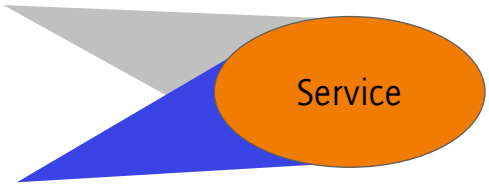
> Software Components with QoS Aspects

> Conclusions

Quality of Service (QoS)

What? *stock ticker, route finder, equation solver, …*

functional properties



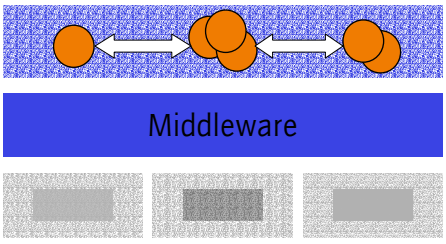
non-functional properties

→ Quality of Service

How? *response time, thruput, precision, availability, security, …*

Middleware

- Platform support for distributed applications
  - interaction, integration, interoperability, coordination
  - distribution transparency
- Examples: CORBA, J2EE/EJB, .NET, …



Overview

- Motivation and Requirements
- Middleware QoS Approaches
- Software Components with QoS Aspects
- Conclusions

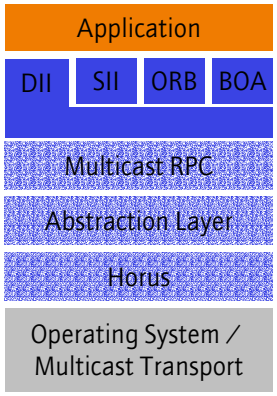
Middleware QoS Approaches

- Specialised (single category) approaches
  - build customised ORB with integrated mechanisms for QoS
  - Examples
    Real-time (TAO / D. Schmidt et. al.)
    Availability (Electra / S. Maffeis)

Example: Electra (S. Maffeis, 1995)

>Availability through replication

- > CORBA extensions
- > service may be an object group
- > atomic multicast (e.g. Horus Toolkit)
- > highly customised design: special ORB objects and interfaces



Middleware QoS Approaches / 2

- > Frameworks to support arbitrary QoS requirements
  - > Provide support for extensible and adaptable middleware
  - > Architectural requirements
    - expandable
    - separable
    - reusable
    - comprehensive
  - > Examples
    - MAQS (Uni Frankfurt / TU Berlin)
    - DotQos (TU Berlin)
    - QuO (BBN), Aspectix (Uni Erlangen), Dynamic TAO (UIUC), OpenORB (Lancaster Univ.), …

Middleware QoS Approaches / 3

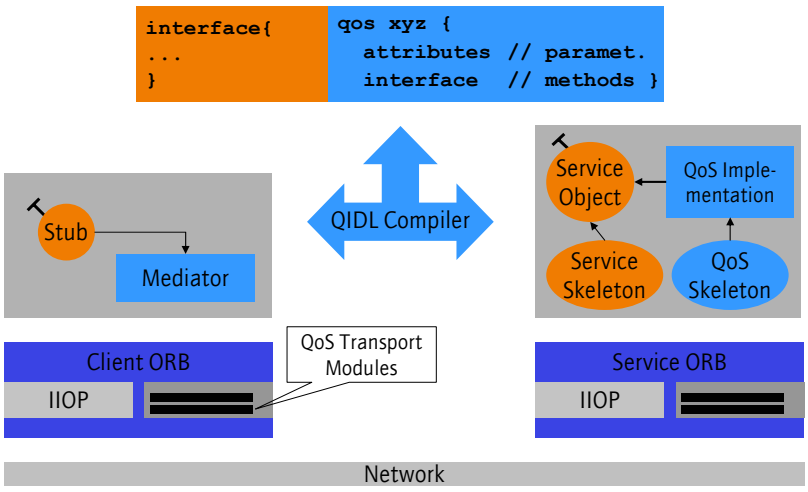
- QoS in OMG
  - QoS Green Paper (1997)
  - real time CORBA (since CORBA 2.4, 2000)
  - miscellaneous QoS provisions in OMG standards, e.g. Notification Service

Example: MAQS (Management Architecture for Quality of Service)

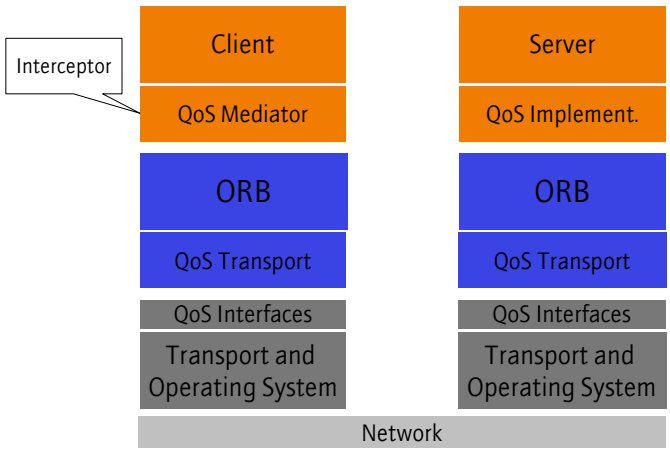
MAQS Objectives

- Framework for QoS Management in CORBA
  - generic support
- Separation of
  - application → QoS mechanisms
  - QoS specification → QoS implementation
- View QoS as an aspect → Aspect Oriented Programming (AOP)

MAQS System Architecture



Integration of QoS Mechanisms



Example: DotQos – QoS for .Net

www.dotqos.org

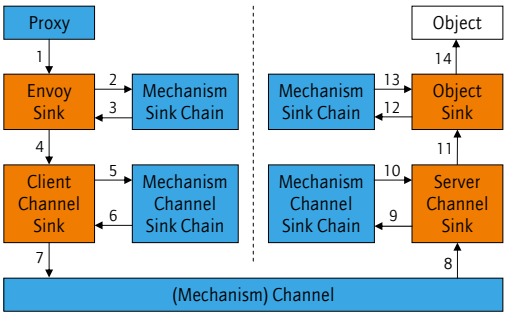
- Built on top of .NET Remoting
  - Flexible, reflective middleware
  - Supports costum meta data, message formats, and transport protocols
- Designed with component-based applications in mind
- Provides extensions to enable
  - QoS Specification
  - QoS Negotiation
  - QoS Provision
  - Working on resource management and dynamic reconfiguration
- Aims at multi-category QoS

Configuring the Middleware

- Need to insert QoS mechanisms into the framework
- Mechanisms invoked by sinks (= interceptors)
- Multi-category QoS by combining sinks in a chain
  - Architectural reflection is required (see G.S. Blair, CACM Vol. 45 No. 6)
  - Use .NET custom metadata
- Reconfiguration of sink chains at runtime
- Resource management as component decomposition
  - Mechanisms and resources are components themselves
  - Resource allocation can be seen as agreement on the QoS a component delivers (→ QCCS)

QoS Mechanisms in .NET

- Standard .NET sink chain is static
  - Added dynamic sink chain for QoS mechanisms
- Support QoS mechanism at
  - Request-level
  - Message-level
  - Transport-level
- Generic QoS sinks
  - QoS negotiation
  - Mechanism invocation



DotQoS Conclusions

- Reflection is important
  - Required for dynamic reconfiguration
- Middleware should be customizable
  - .NET Remoting far more flexible than CORBA
  - Eases integration of mechanisms
  - No need to mess with the middleware itself
- Custom Meta-data
  - No need to add new languages (i.e. QIDL)
  - Can be used for advanced reflection
- QoS fits nicely into the concepts of components
  - See QCCS …

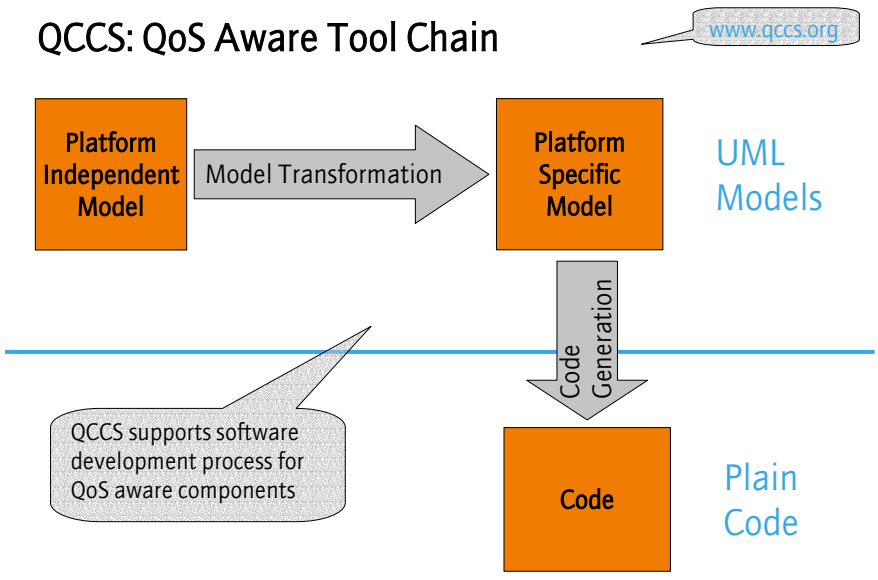
Overview

- Motivation and Requirements
- Middleware QoS Approaches
- Software Components with QoS Aspects
- Conclusions

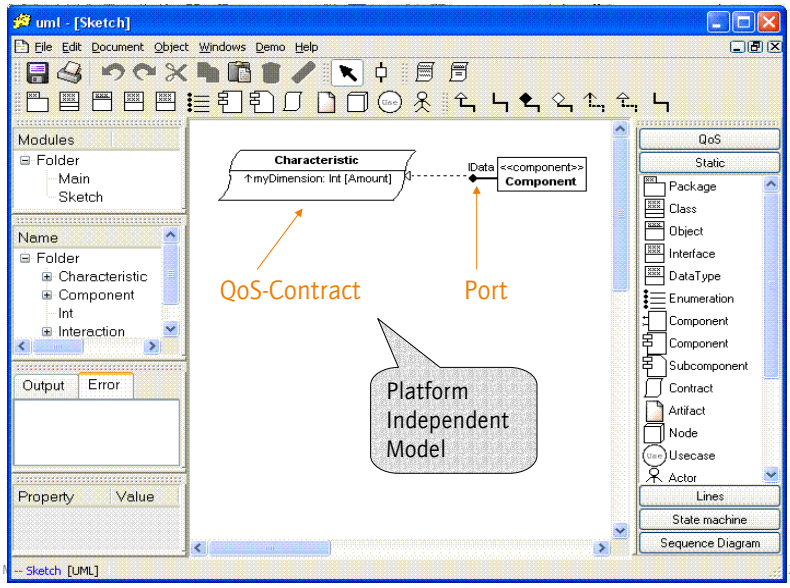
QCCS: From Model to Implementation

- QoS should be captured early in the design phase
  - The entire tool chain needs to be QoS aware !
- Modeling languages need to become QoS aware
  - The UML is a reasonable candidate to start with
- MDA (Model Driven Architecture)
  - Capture design ideas and QoS-requirements in a platform independent way
  - Don't get muddled with implementation issues too early
  - Transform these models (semi)-automatically into platform dependent models
- QoS-contracts can be attached to components
  - Components are often used for PIM models
  - QoS-contracts specify non-functional properties of components

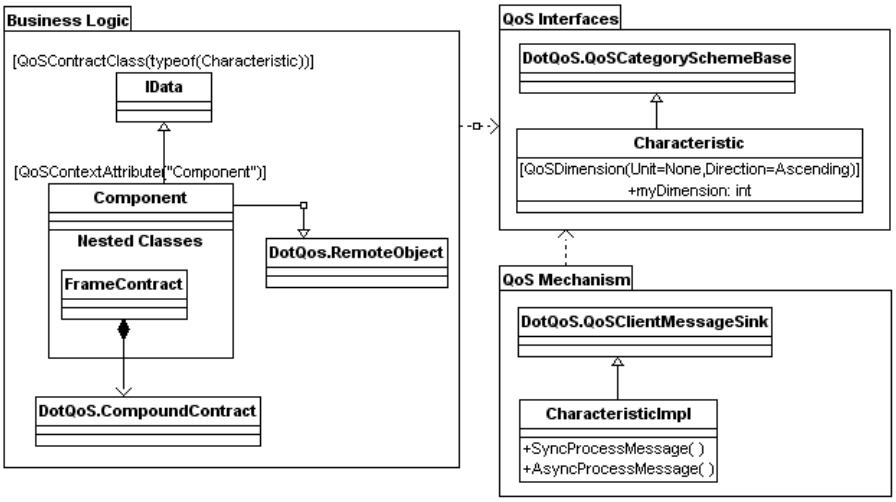
QCCS: QoS Aware Tool Chain



KASE: UML Tool for QoS Modeling and Code Generation



Platform Specific Model



Overview

- Motivation and Requirements
- Middleware QoS Approaches
- Software Components with QoS Aspects
- Conclusions

Conclusions

- Best effort is not enough, users want quality of service guarantees.
- Middleware needs to handle QoS aspects.
- Whole software development process needs to consider QoS aspects → QoS aware tool chain !
- CORBA is a suitable architecture for middleware QoS; .NET Remoting even more so.
- Applications need to adapt, if possible.

Acknowledgements

Christian Becker

Andreas Ulbrich

Torben Weis

Thank you.

Questions and comments are welcome!

Prof. Dr. Kurt Geihs
geihs@ivs.tu-berlin.de
Intelligent Networks and Distributed Systems Management
TU Berlin
www.ivs.tu-berlin.de

Institut für Telekommunikation
Fakultät IV – Elektrotechnik und Informatik
TU Berlin

T: +49 30 314-79333
F: +49 30 314-24573
E: office@ivs.tu-berlin.de

Sekretariat EN 6
Einsteinufer 17
D-10587 Berlin / Germany