Modelling Support for Quality of Service

Department for Cooperative and Trusted Systems
Information and Communication Technology,
SINTEF, Forskningsveien 1, N-0314 Oslo, Norway
http://www.sintef.no

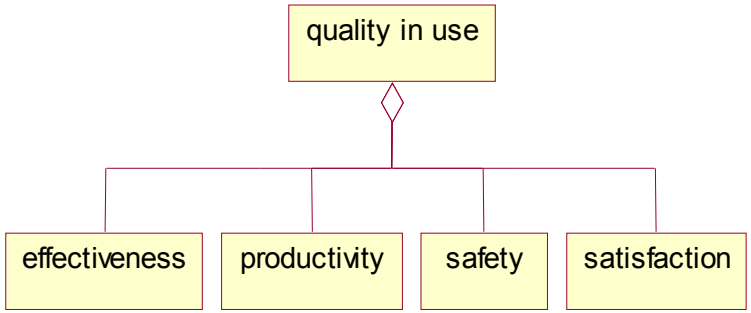
Context of this work

- The present courseware has been elaborated in the context of the MODELWARE European IST FP6 project (http://www.modelware-ist.org/).
- Co-funded by the European Commission, the MODELWARE project involves 19 partners from 8 European countries. MODELWARE aims to improve software productivity by capitalizing on techniques known as Model-Driven Development (MDD).
- To achieve the goal of large-scale adoption of these MDD techniques, MODELWARE promotes the idea of a collaborative development of courseware dedicated to this domain.
- The MDD courseware provided here with the status of open source software is produced under the EPL 1.0 license.

Quality

- Software architecture defines a evolutionary envelope within which acceptable quality is maintained
  - But what kind of qualities are affected?
- Qualities define the "goodness" of the system (or its architecture)
- Heaps of "-ilities"
  - Subjective vs. objective quality
  - Design-time vs. run-time qualities

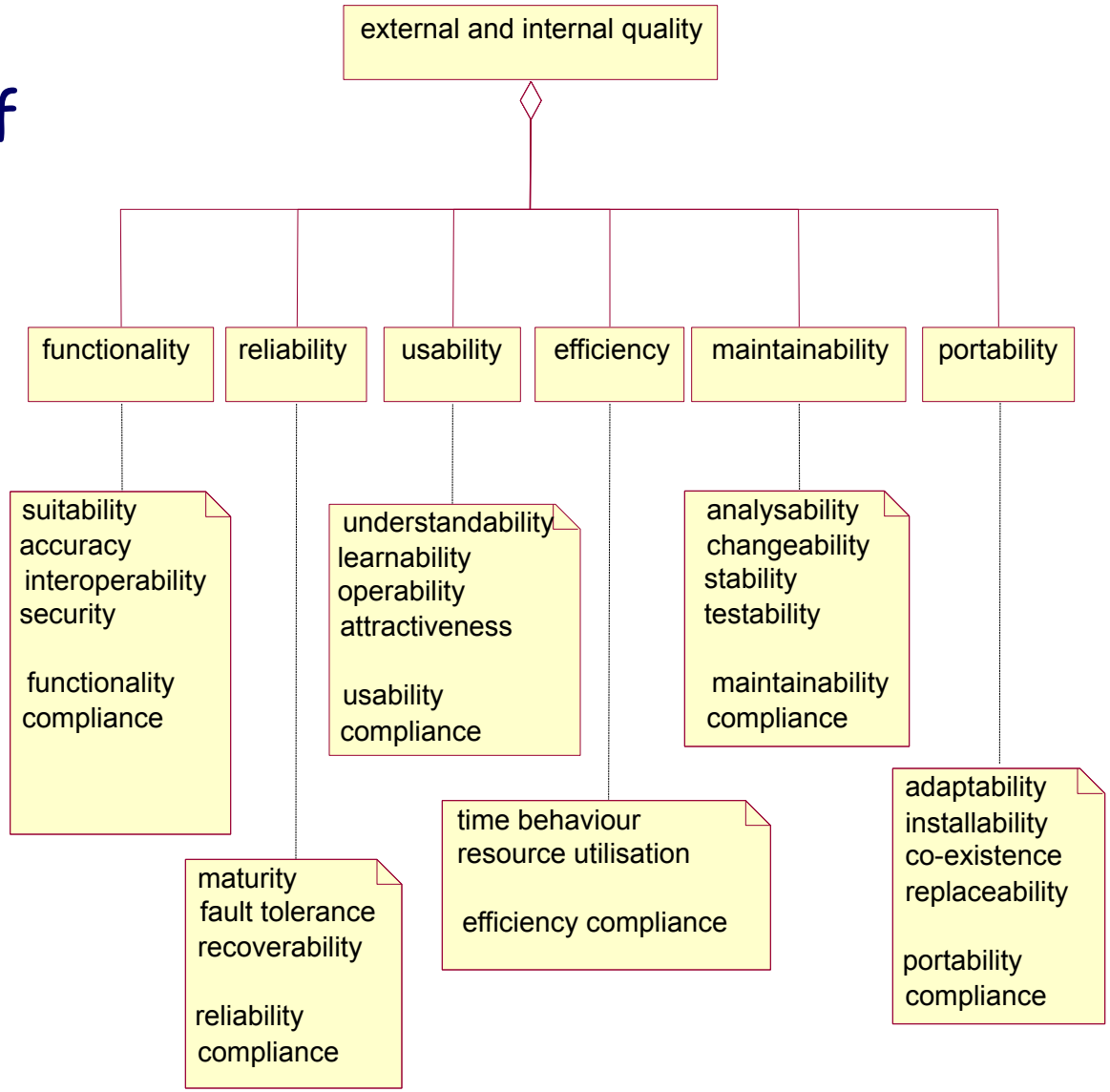
ISO 9126 - Software Product Quality



- Quality in use - related to a specific context of use
  - effectiveness – the capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.
  - productivity – the capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in the specified context of use.
  - safety – the capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use.
  - satisfaction – the capability of the software product to satisfy users in a specified context of use.

ISO 9126 - External and Internal Quality Metrics

- Irrespective of context of use



Functionality

- The capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.
  - Suitability
    - The capability of the software product to provide an appropriate set of functions for specified tasks and user objectives.
  - Accuracy
    - The capability of the software product to provide the right or agreed results or effects with the needed degree of precision.
  - Interoperability
    - The capability of the software product to interact with one or more specified systems.
  - Security
    - The capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them
  - Functionality compliance
    - The capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality.

Reliability

- The capability of the software product to maintain specified level of performance when used under specified conditions.
  - Maturity
    - The capability of the software product to avoid failure as a result of faults in the software.
  - Fault tolerance
    - The capability of the software product to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
  - Recoverability
    - The capability of the software product to re-establish a specified level of performance and recover the data directly affected in the case of a failure.
  - Reliability compliance
    - The capability of the software product to adhere to standards, conventions or regulations relating to reliability.

Usability

- The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.
- Understandability
  - The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.
- Learnability
  - The capability of the software product to enable the user to learn its application.
- Operability
  - The capability of the software product to enable the user to operate and control it.
- Attractiveness
  - The capability of the software product to be attractive to the user.
- Usability compliance
  - The capability of the software product to adhere to standards, conventions, style guides or regulations relating to usability.

Efficiency

- The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions.
- Time behaviour
  - The capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.
- Resource utilisation
  - The capability of the software product to use appropriate amounts and types of resources when the software performs its function under stated conditions.
- Efficiency compliance
  - The capability of the software product to adhere to standards or conventions relating to efficiency.

Maintainability

- The capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.
- Analysability
  - The capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.
- Changeability
  - The capability of the software product to enable a specified modification to be implemented.
- Stability
  - The capability of the software product to avoid unexpected effects from modifications of the software.
- Testability
  - The capability of the software product to enable modified software to be validated.
- Maintainability compliance
  - The capability of the software product to adhere to standards or conventions relating to usability.

Portability

- The capability of the software product to be transferred from one environment to another.
- Adaptability
  - The capability of the software product to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered.
- Installability
  - The capability of the software product to be installed in a specified environment.
- Co-existence
  - The capability of the software product to co-exist with other independent software in a common environment sharing common resources.
- Replaceability
  - The capability of the software product to be used in place of another specified software product for the same purpose in the same environment.
- Portability compliance
  - The capability of the software product to adhere to standards or conventions relating to portability.

Development-time qualities (from Catalysis)

- Affected by architecture:
  - Modifiability- Can the system be modified efficiently?
    - E.g., low coupling and high cohesion
  - Reusability - Are there units in the system that can are candidates for use elsewhere?
    - E.g., does the system use standards?
  - Portability - The ability to change platform
    - E.g., layering
  - Buildability - Is it easy to implement?
    - E.g., use existing frameworks
  - Testability - Can one easily define test scenarios?
    - E.g., precise requirement specifications

Runtime qualities (from Catalysis)

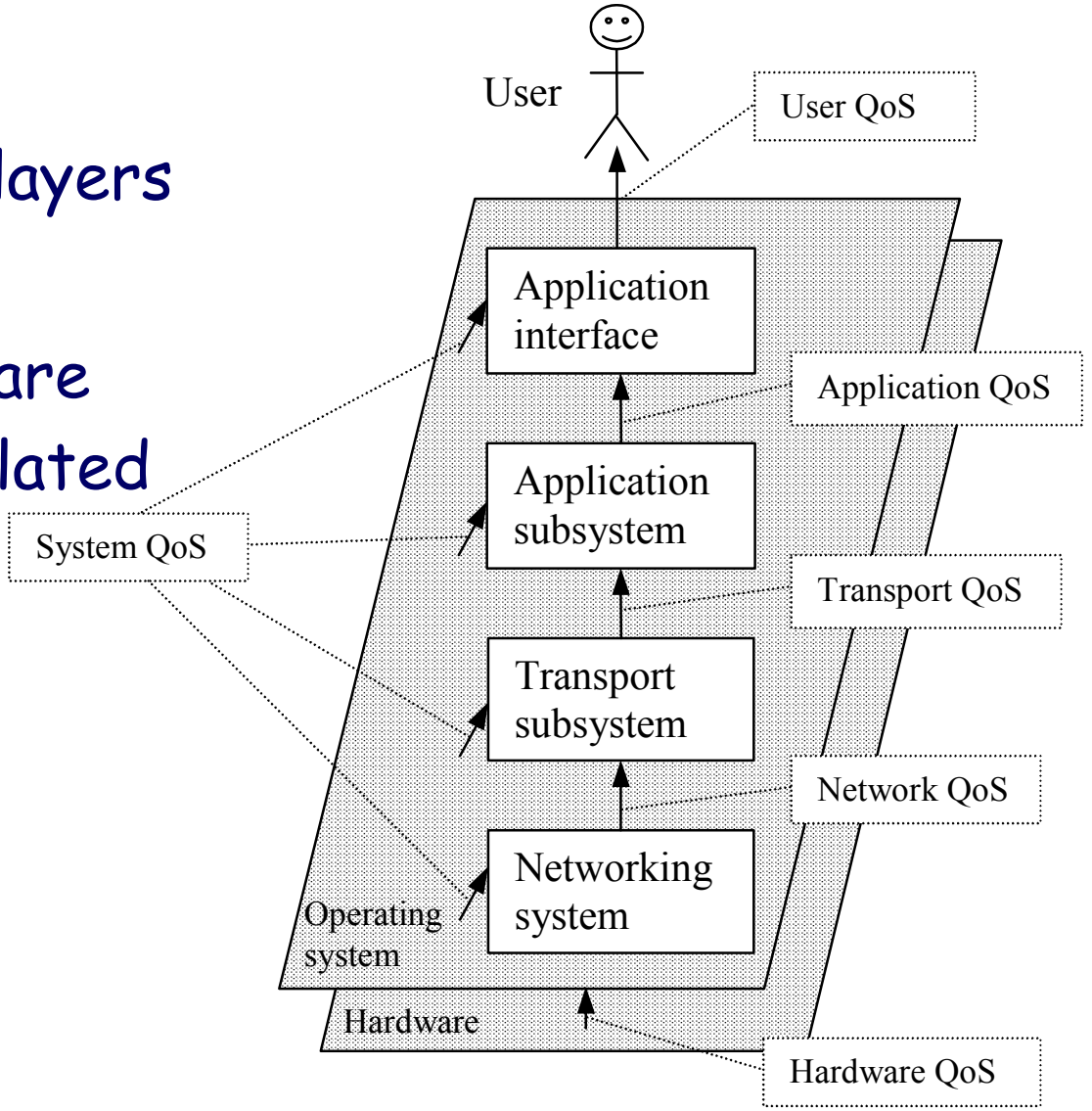
- Affected by architecture:
  - Functionality - does the system assist users in their tasks?
  - Usability - is it intuitive to use for all users?
  - Performance - does it perform adequately when running?
    - E.g., response time, transaction volume, ...
  - Security - does it prevent unauthorised access?
  - Reliability and availability - is it available and correct over time?
  - Scalability - can it cater for increased volume?
  - Upgradability - can it be upgraded at runtime?
- Single key quality: Conceptual integrity of an architecture

Quality of Service (QoS)

- QoS is a general term that covers system performance, rather than system operation (i.e., functionality)
- Extra-functional properties
  - degrees of satisfaction as opposed to satisfied / not satisfied
- Examples:
  - availability, reliability, precision, fault-tolerance, capacity, throughput, delay, …
- Most common for multimedia, command and control, simulations, distributed systems, …
- Less common for information systems
  - there are needs! (transaction-based, security aware, etc.)

Different Abstractions

- Peer-to-peer vs. layers
- QoS concepts on different layers are different, but related
  - QoS mapping



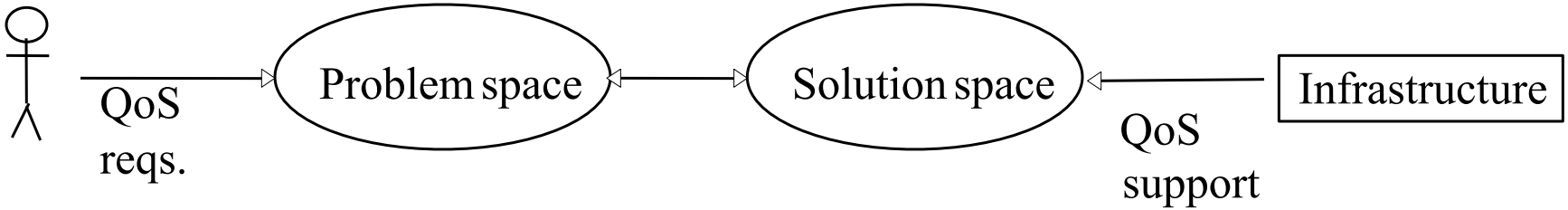
Quality of Service Support in Infrastructures

- Some is already available
  - Networks and protocols (IPv6, RSVP, ATM, …)
  - Real time operating systems (Chorus, VxWorks, …)
  - Middleware (CORBA implementations such as COOL, ACE, …)
- Unified and comprehensive QoS architecture is still a hot research topic
- Several stakeholders
  - End users, system operators, network providers, resource owners, …
- From a software engineering perspective, a unified computational model is needed
  - As target for model transformations / code generation

QoS Management

- The activities needed to support monitoring, control and maintenance of QoS
- Spectrum from static to dynamic management
  - Static
    - Designed and configured into the system
    - QoS are not monitored or controlled
  - Dynamic
    - Adapts to changes
    - Monitoring, resource allocation, adaptation of applications (e.g., caching), re-routing, ...
- Best-effort QoS
  - Threshold or compulsory
- Guaranteed QoS
  - Statistical variations

Quality of Service Specifications to Support QoS management

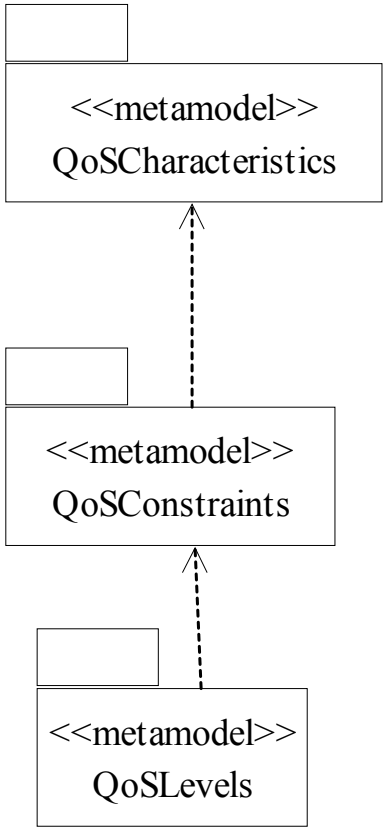


- QoS support in applications requires methodological support during development
  - How to match requirements and solutions
- QoS is a concern of many stakeholders
  - Need to include QoS specification into architectural descriptions
- The Big Question: How do changes in environment affect the agreed upon QoS?
  - I.e., is adaptation needed?

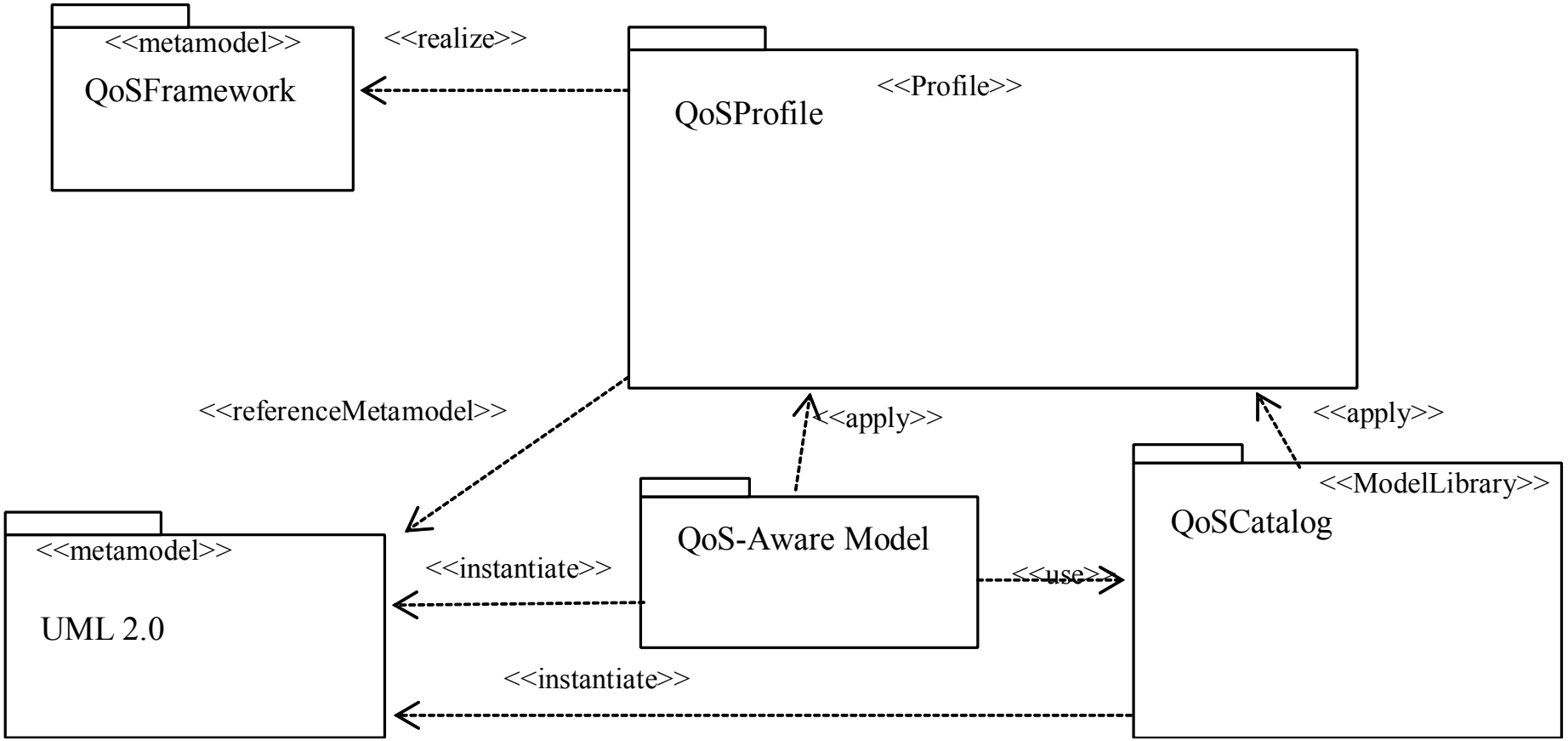
Quantification

- How to measure and quantify quality?
  - Commonly agreed reference units (objective) versus ordinal rankings (subjective)
- Main problem: From subjective to objective
- Vagueness in classification of observations
  - Vagueness is not ambiguity or generality
  - Sorites paradox (The "slippery-slope" argument)
  - Vagueness-as-ignorance or gray areas (fuzzy logic)?
- Commonly chosen approach: Stability threshold

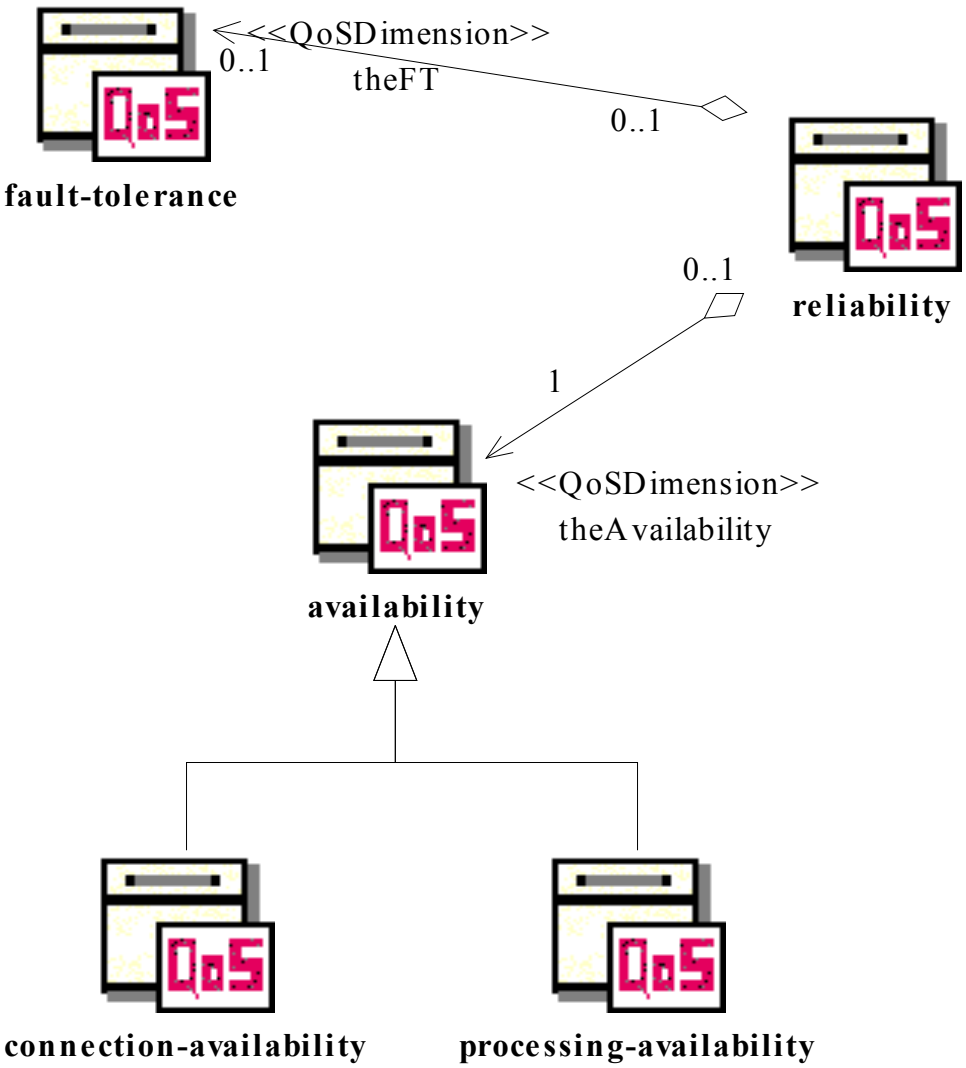
UML Profile for QoS - overview



UML QoS profile



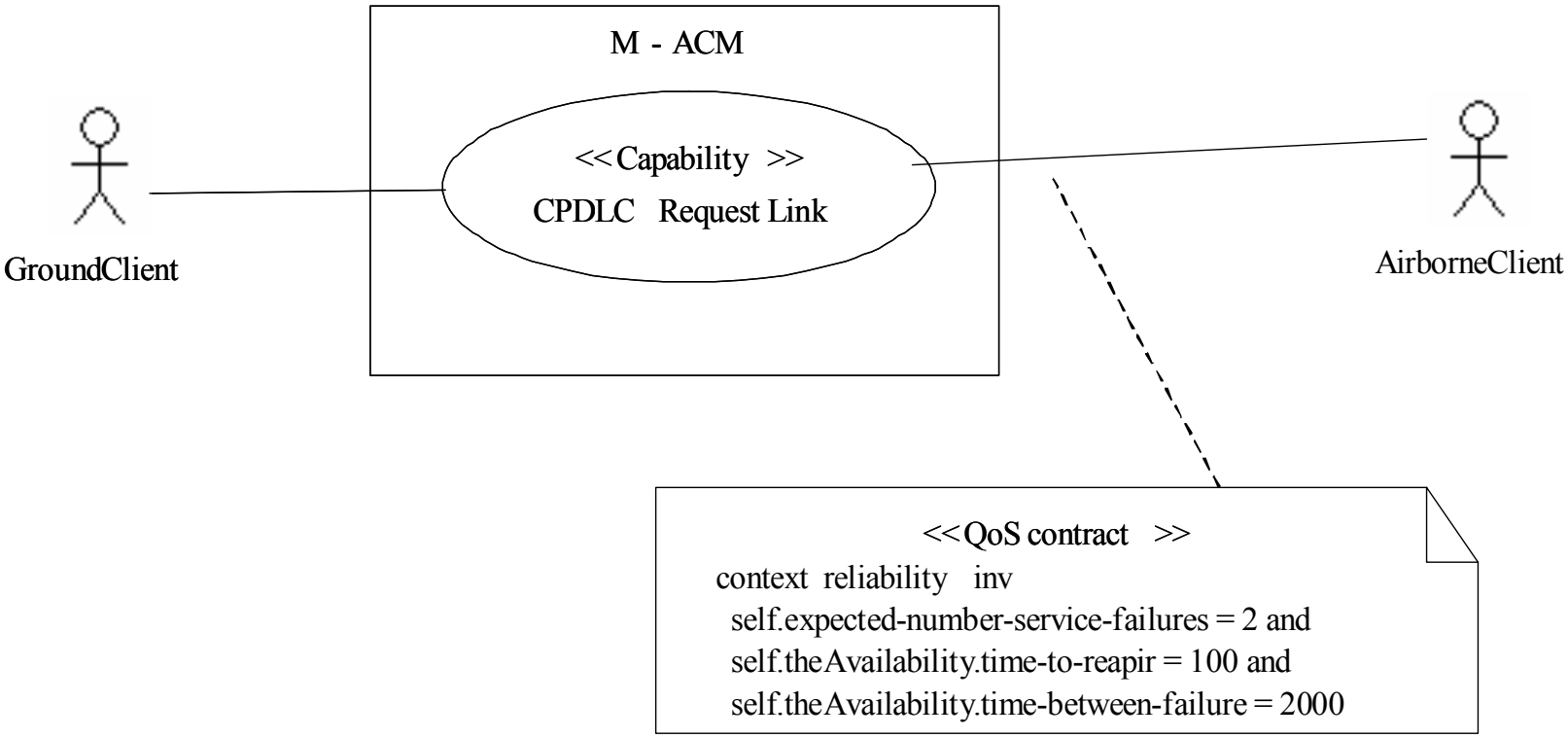
QoS characteristics



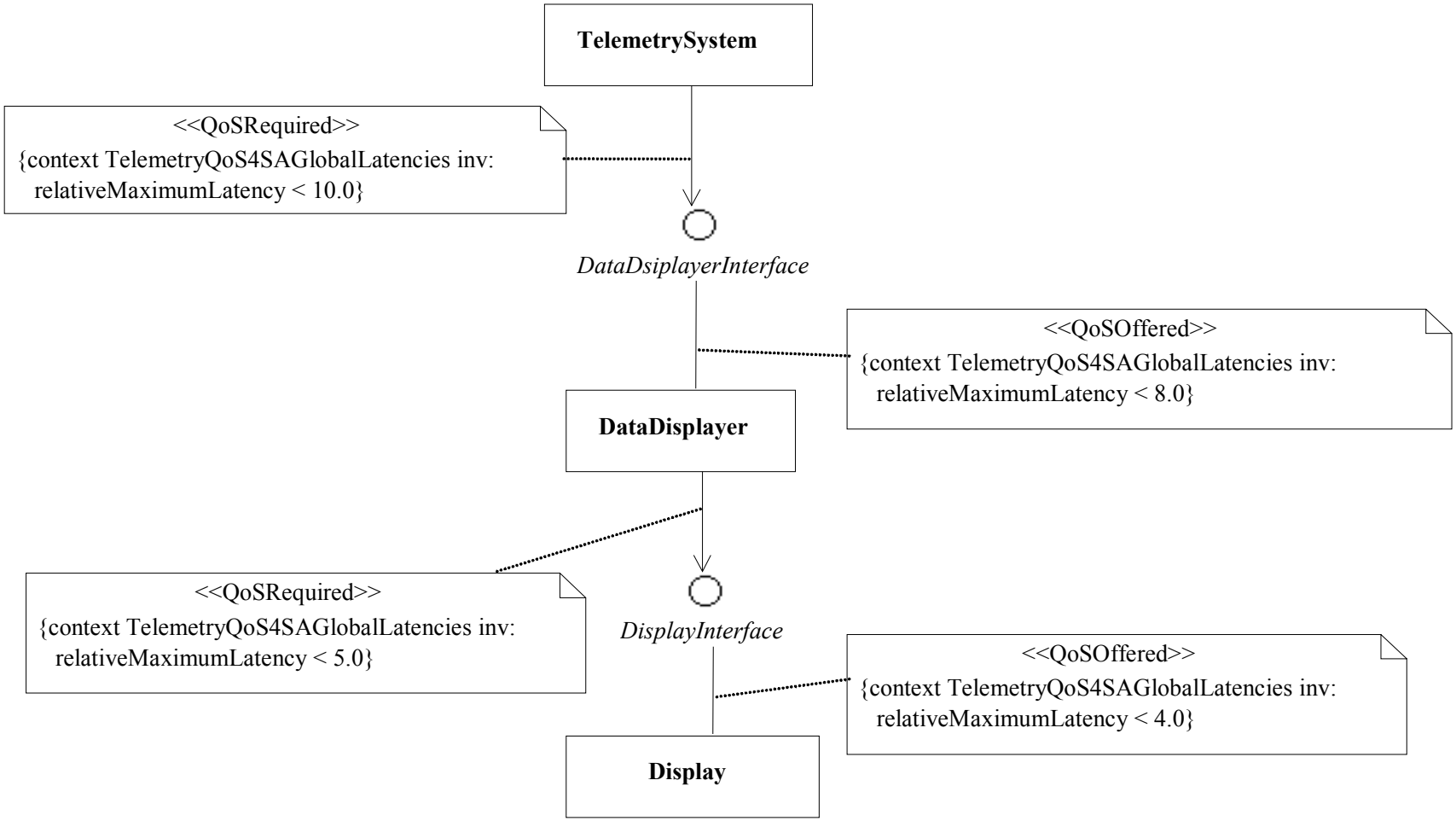
QoS value

<<QoSValue>> Availability_of_remote_server:processing-availability
time-between-failures=10 time-to-repair=1

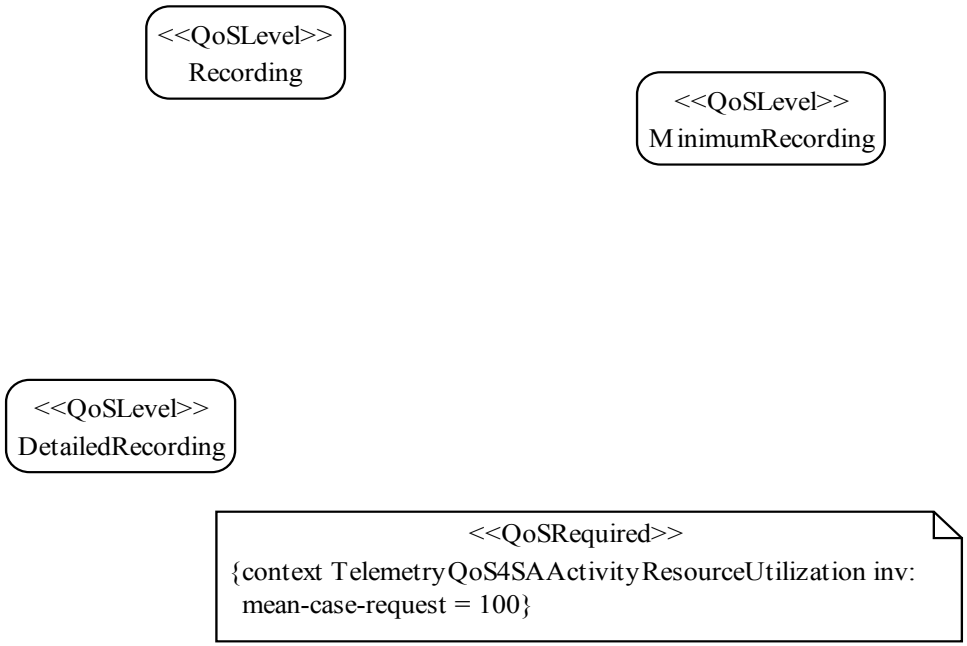
QoS contract



Another contract



QoS behaviour (adaptation)



Conclusions

- QoS should be considered during software design
- QoS should be specified to support QoS management
- QoS specifications should be orthogonal to software architecture
  - but may influence it