**Torben Weis**

**QCCS**

Quality Controlled Component based Systems

# QCCS

European IST Project

http://www.qccs.org

IRISA

TU-Berlin

SEMA

KD Software

# QoS (Quality of Service)

> Examples for QoS

  > Bandwidth

  > Latency

  > Security

  > Scalability (concurrent invocations)

> QoS is a non-functional property of a system


> Our Goal:
  Modeling of QoS-aware components

# QoS & Components

> Disadvantages of pure objects

   > Connected by type (in the model)

   > Connections at runtime are hardly trackable

   > Therefore, contract management is difficult

> Advantages of components

   > Components reside on a higher level of abstraction

   > Coupled with connectors (in the model)

   > Reflection allows tracking of connections

   > Contract management is easier
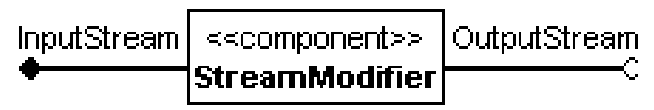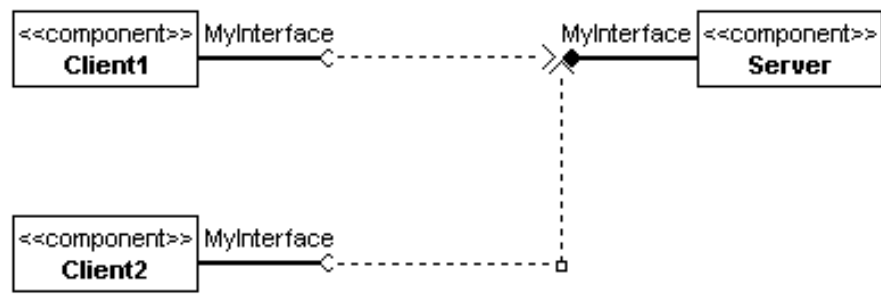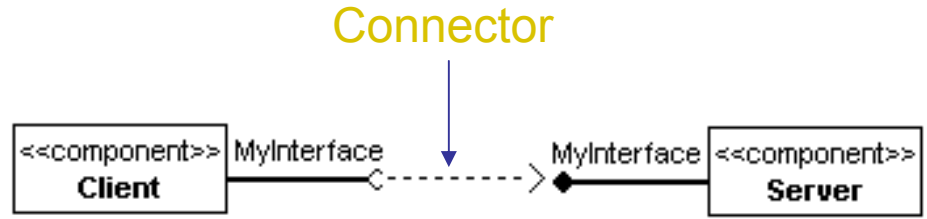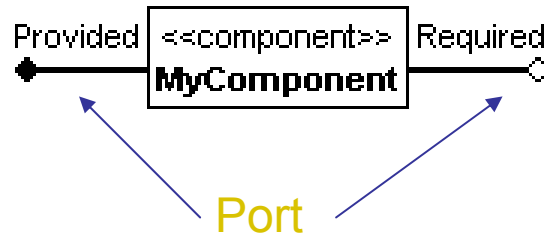
# QoS & Adaption

> Modern applications must *adapt* to a changing environment

>> Bandwidth

>> (Un)secure network connections

>> Device constraints (Screen size, CPU, …)

> Adaption becomes an important feature and crosscuts the entire design
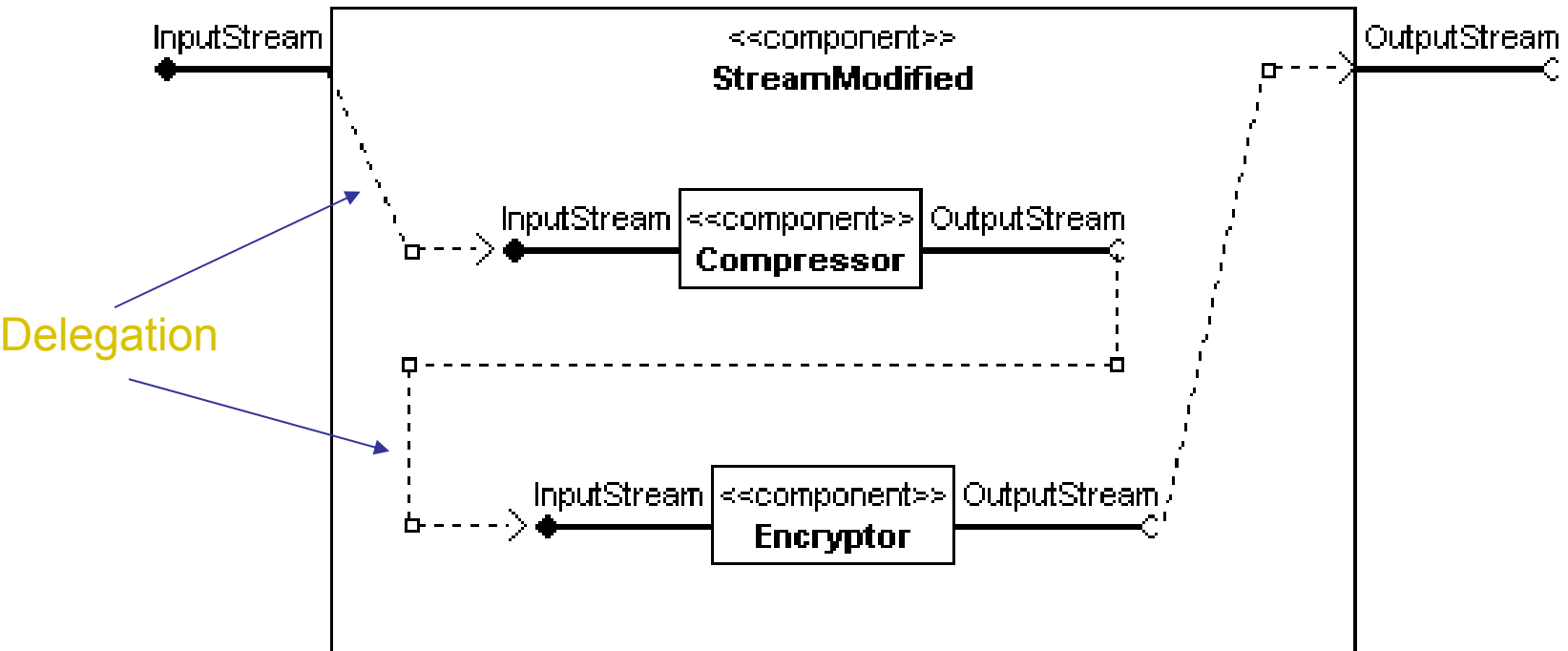
> Adaption should be addressed at the modeling level

# UML2 Components – Black Box View

Connector

Provided <<component>> Required
**MyComponent**

Port

<<component>> MyInterface MyInterface <<component>>
**Client** **Server**

<<component>> MyInterface MyInterface <<component>>
**Client1** **Server**

<<component>> MyInterface
**Client2**

<<interface>>
**InputStream**
+write( bytes )

<<interface>>
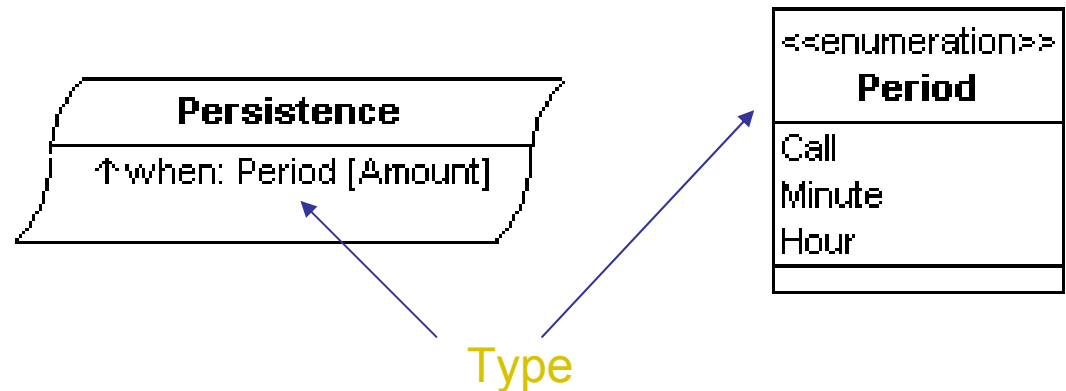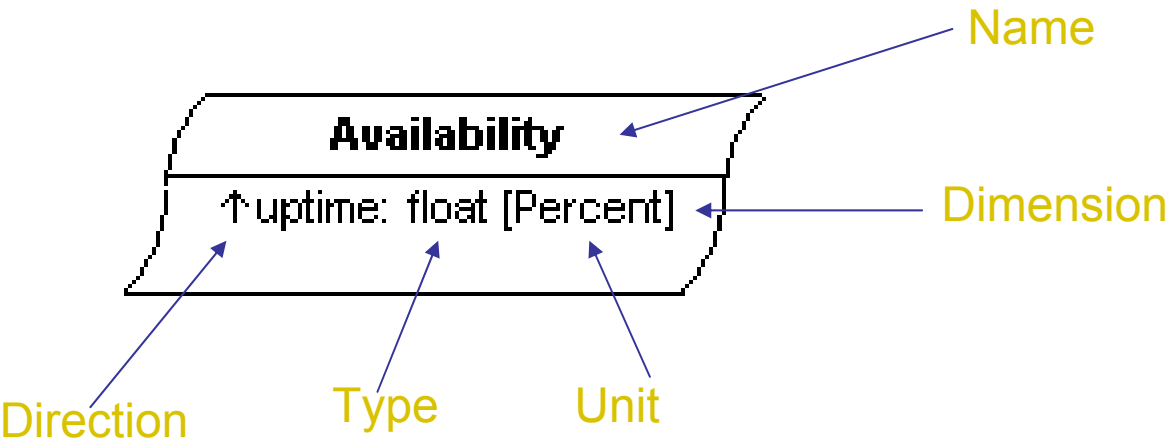**OutputStream**
+read( )

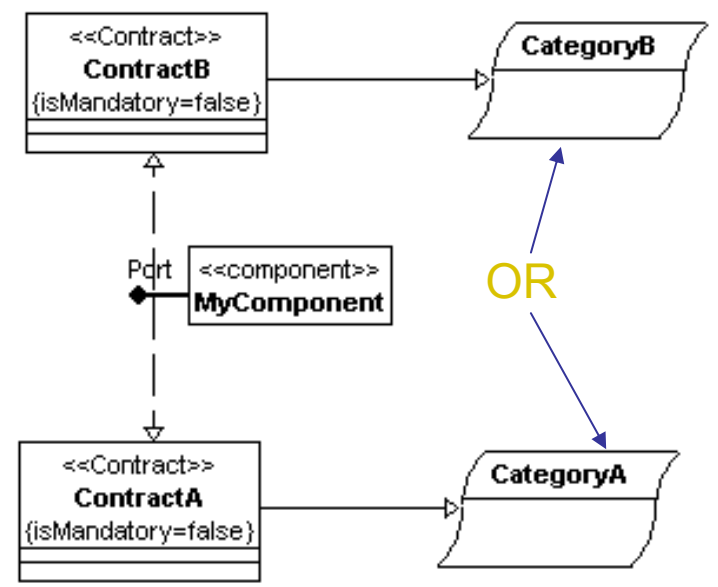InputStream <<component>> OutputStream
**StreamModifier**

# UML 2 Components – White Box View

# Modeling QoS-Categories

Name

**Availability**

↑uptime: float [Percent]

Dimension

Direction    Type    Unit

**Persistence**

↑when: Period [Amount]

<<enumeration>>
**Period**

Call
Minute
Hour

Type

# … putting things together …



QoS-Category

Contract

AND

OR
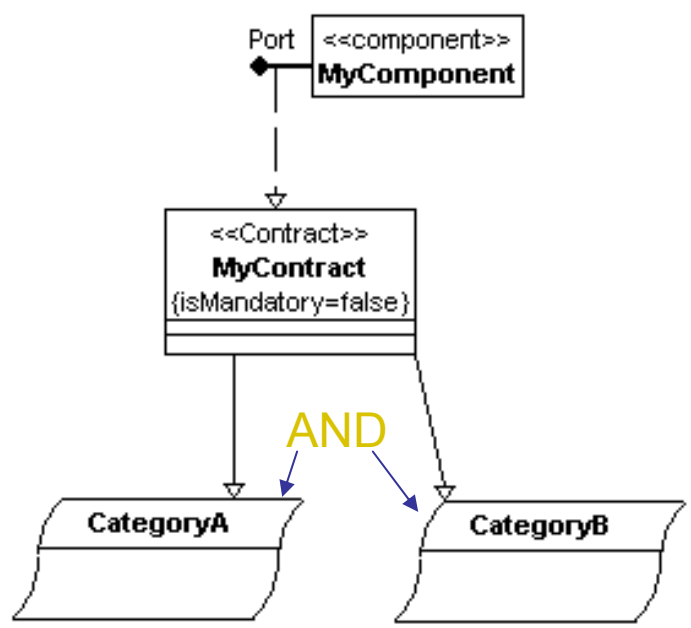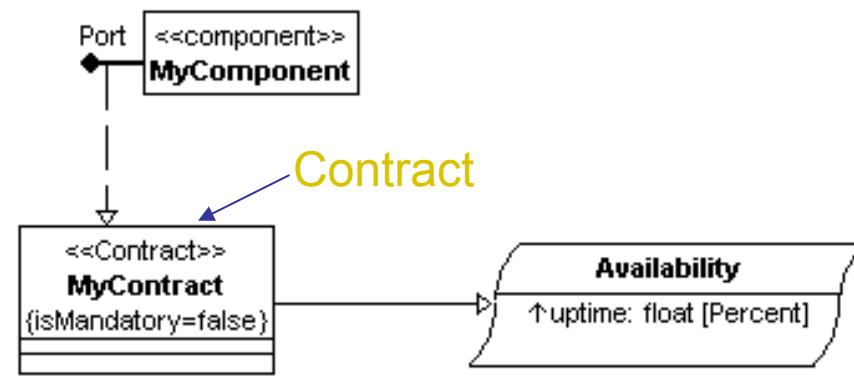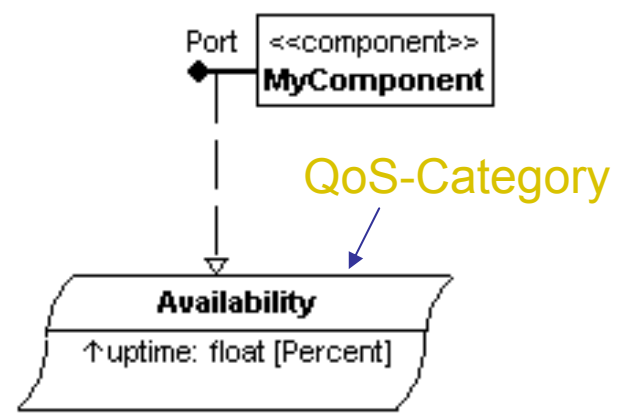
# Modeling Contract Dependencies

# Invocation-Protocols / 1



ResponseTime
↓time [Sec]

<<ContractType>>
**MyContract**
+lines
**Interactions**
Protocol

IGraphics

<<component>>
**MyComponent**

<<interface>>
**IGraphics**
+begin( )
+drawLine( )
+end( )

Protocol
Parameter

Protocol            Interface

# Invocation-Protocols / 2



Interface

Protocol
Parameter

Client

:IGraphics

begin()

loop [lines]

drawLine()

end()
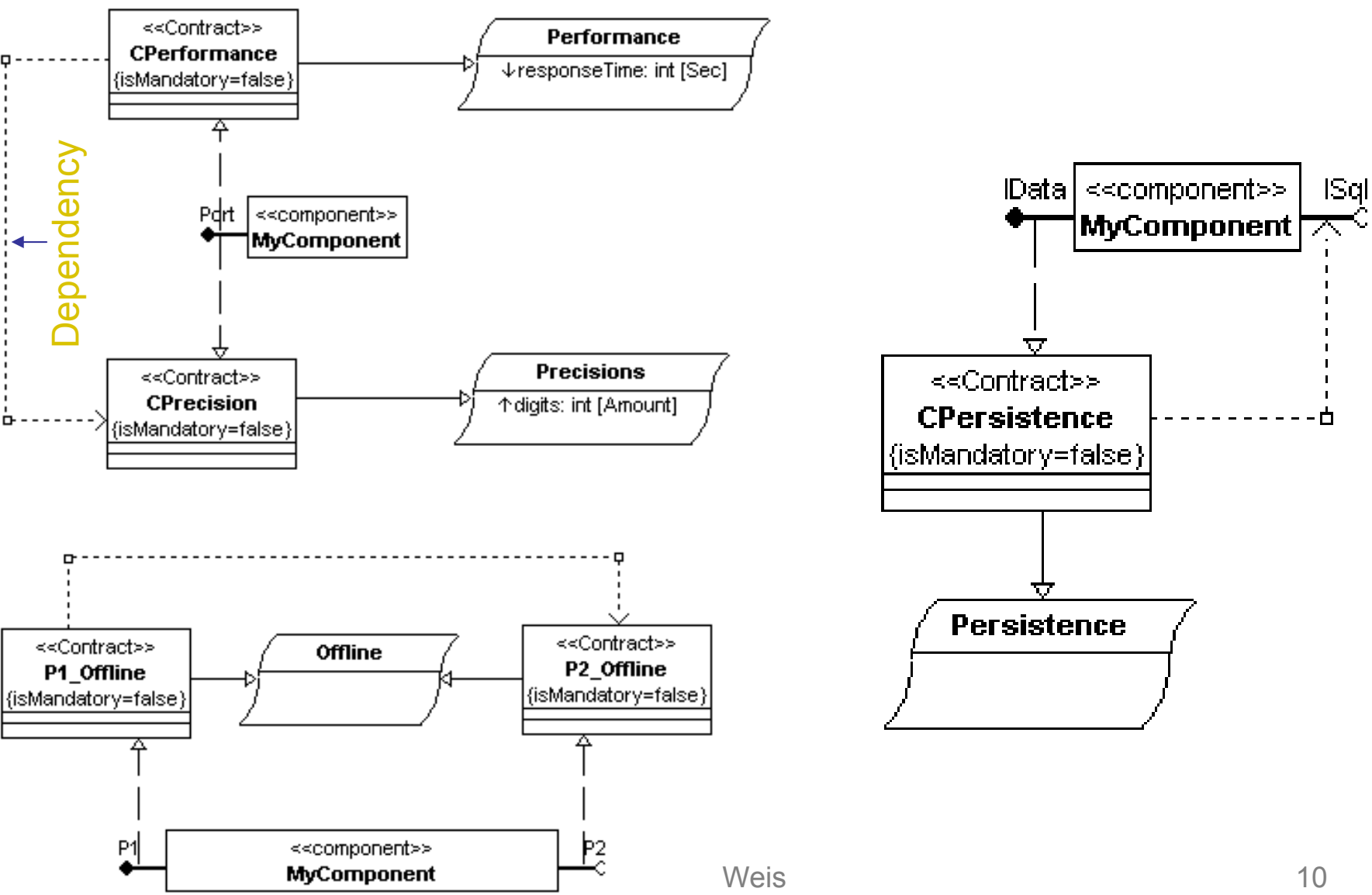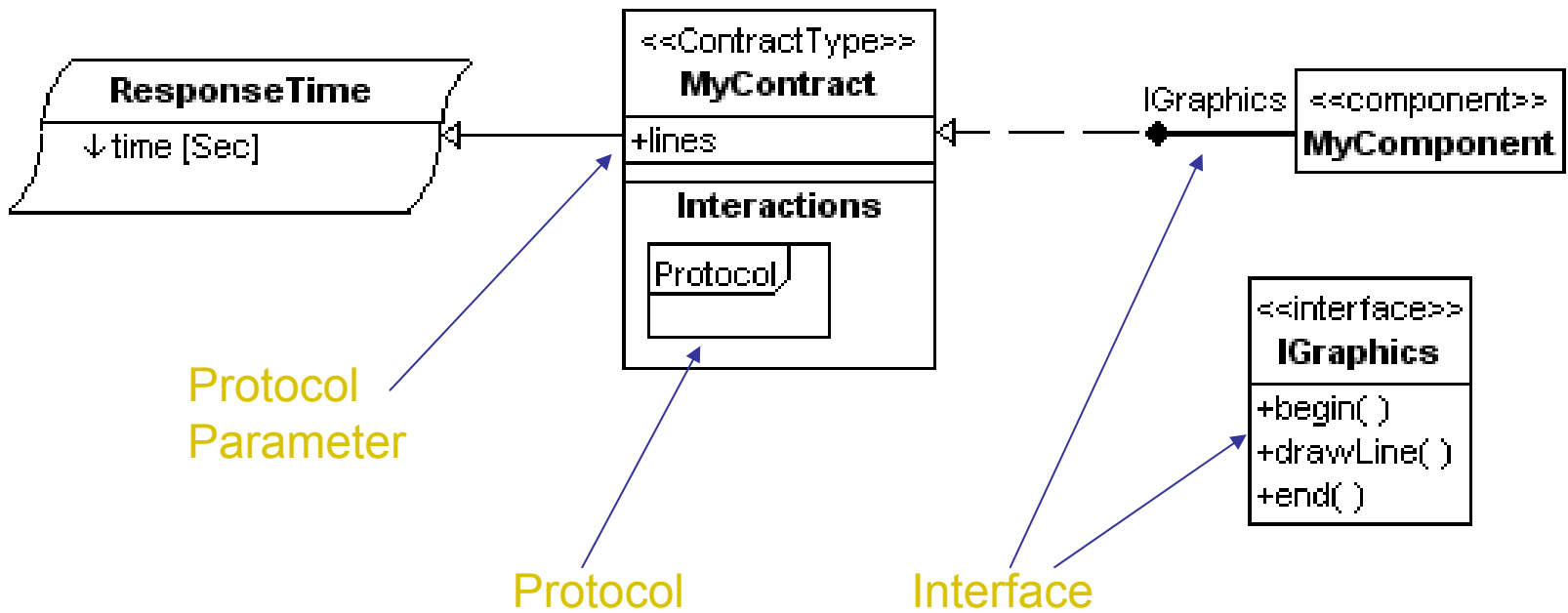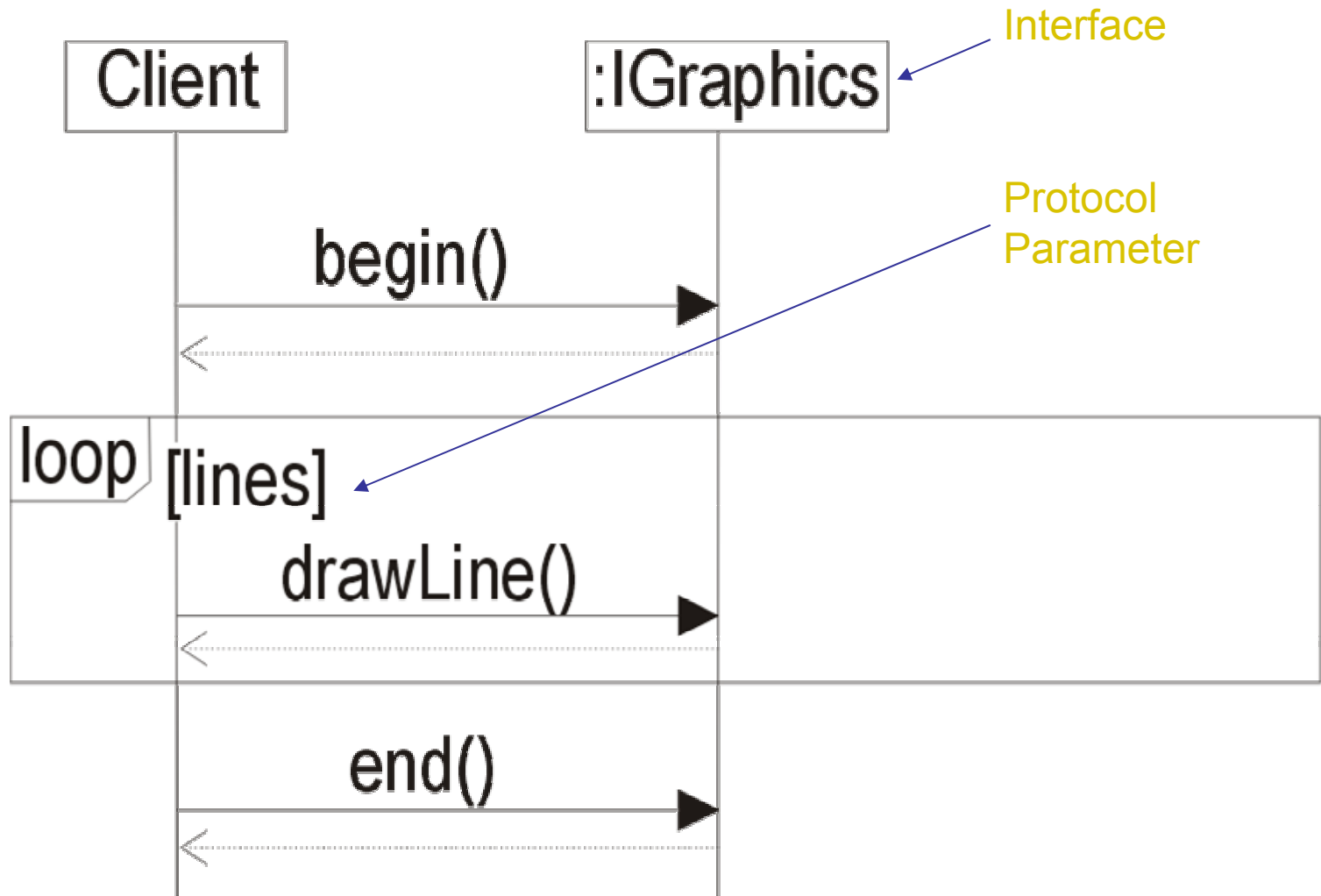
# Entity Components

# Doing the MDA thing …



PIM

PSM

QoS Mechanisms

Mechanism Interface

Contract Interface

Business Logic
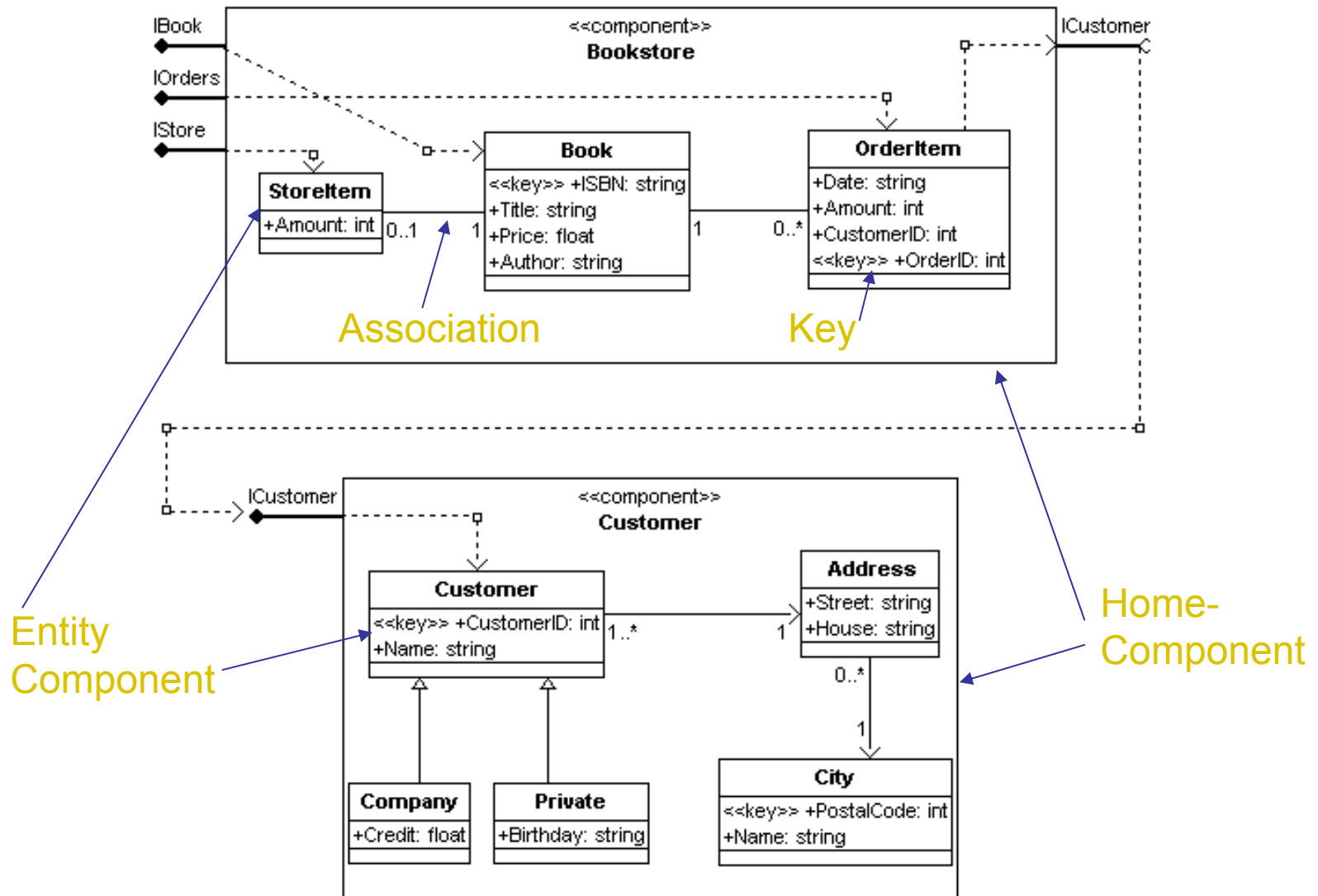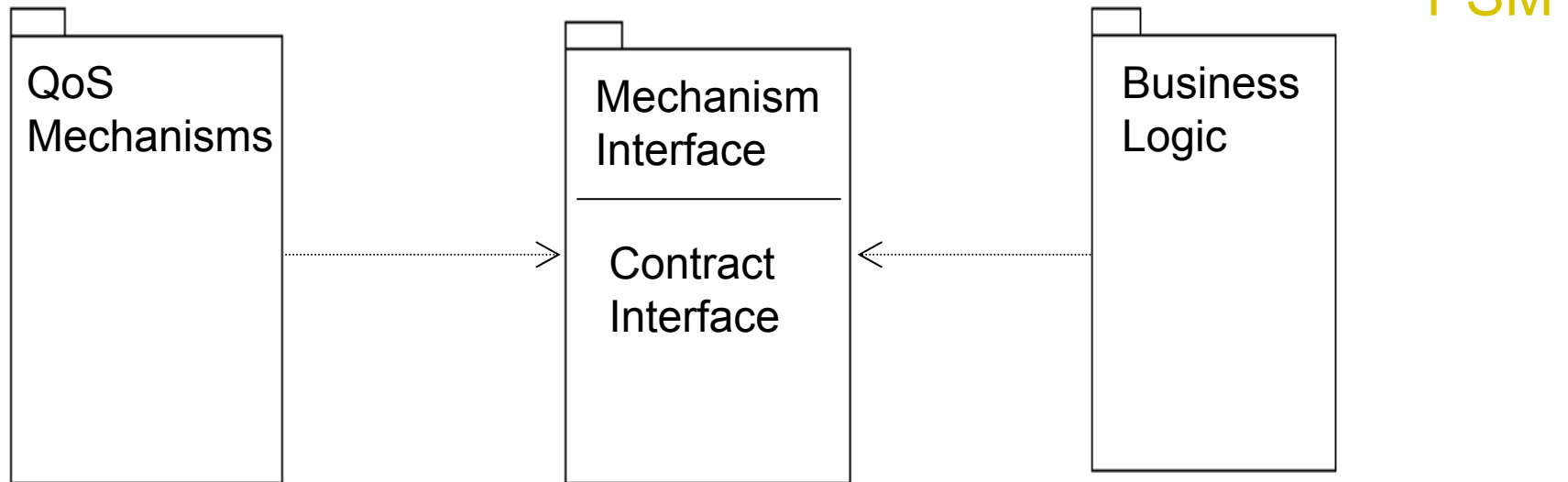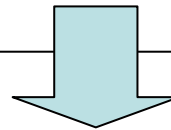
# Aspects in UML / 1

**Component realization** →
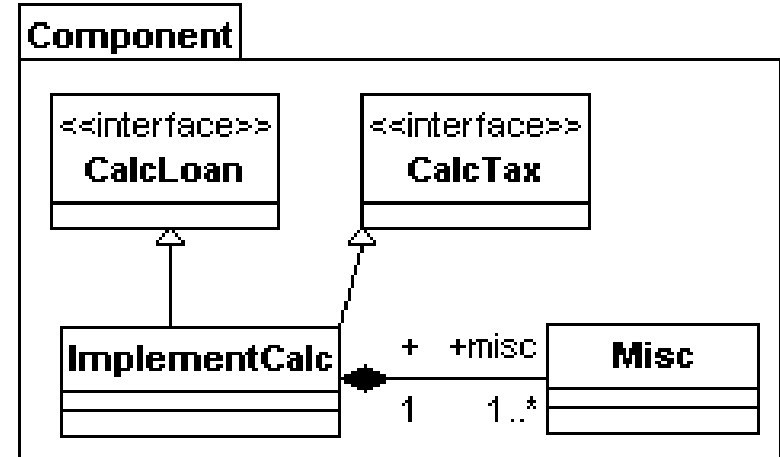
**Aspect invocation**

**Contract realization** →
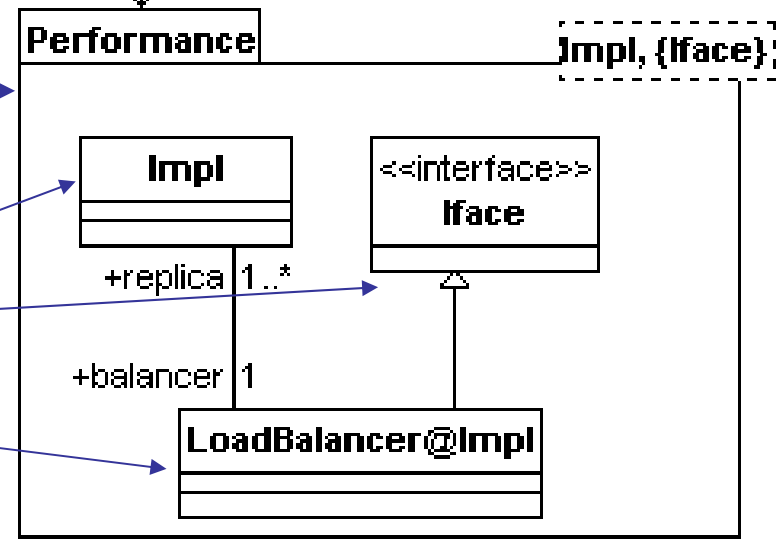
**Parameters**

**New class**

# Aspects in UML / 2

**Woven component**

**New association**

**New generalizations**

Component

Misc

+misc 1..*
+ 1

ImplementCalc

<<interface>>
CalcTax

+replica 1..*

+balancer 1

LoadBalancer

<<interface>>
CalcLoan
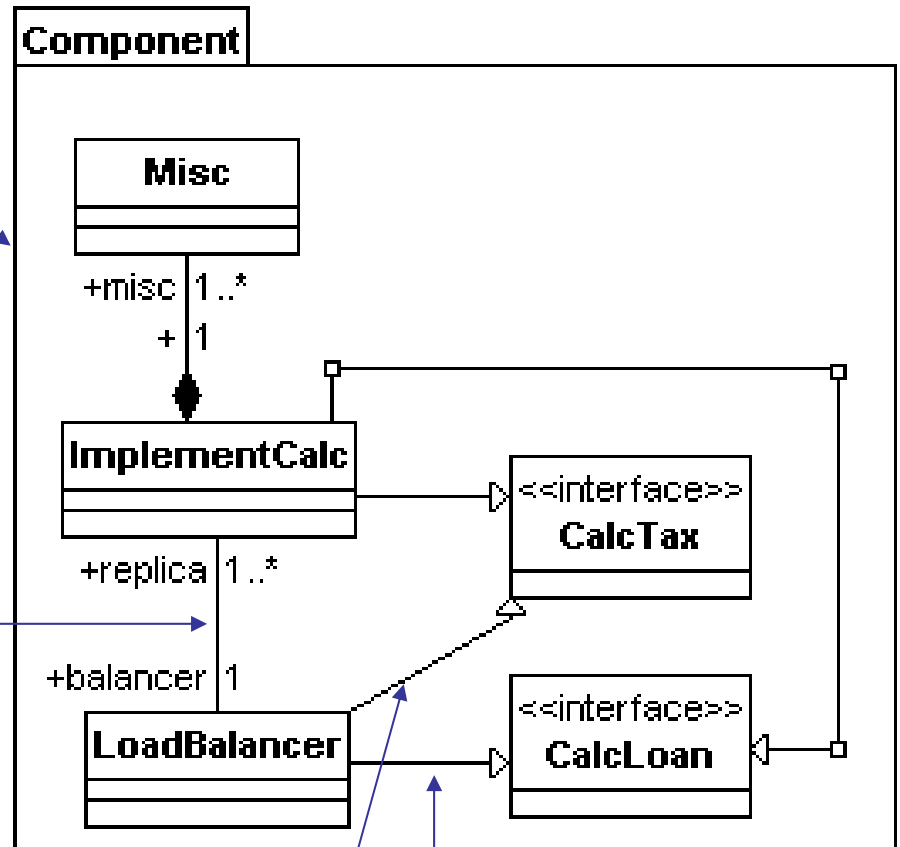
# Thank you.

**Questions?**

Torben Weis
weis@ivs.tu-berlin.de
FG Intelligente Netze und Management verteilter
Systeme
TU Berlin
www.ivs.tu-berlin.de