Zwischenbericht zur Diplomarbeit über das Thema:

Modellierung CORBA basierter Anwendungssysteme mit der UML

Andreas Mucha

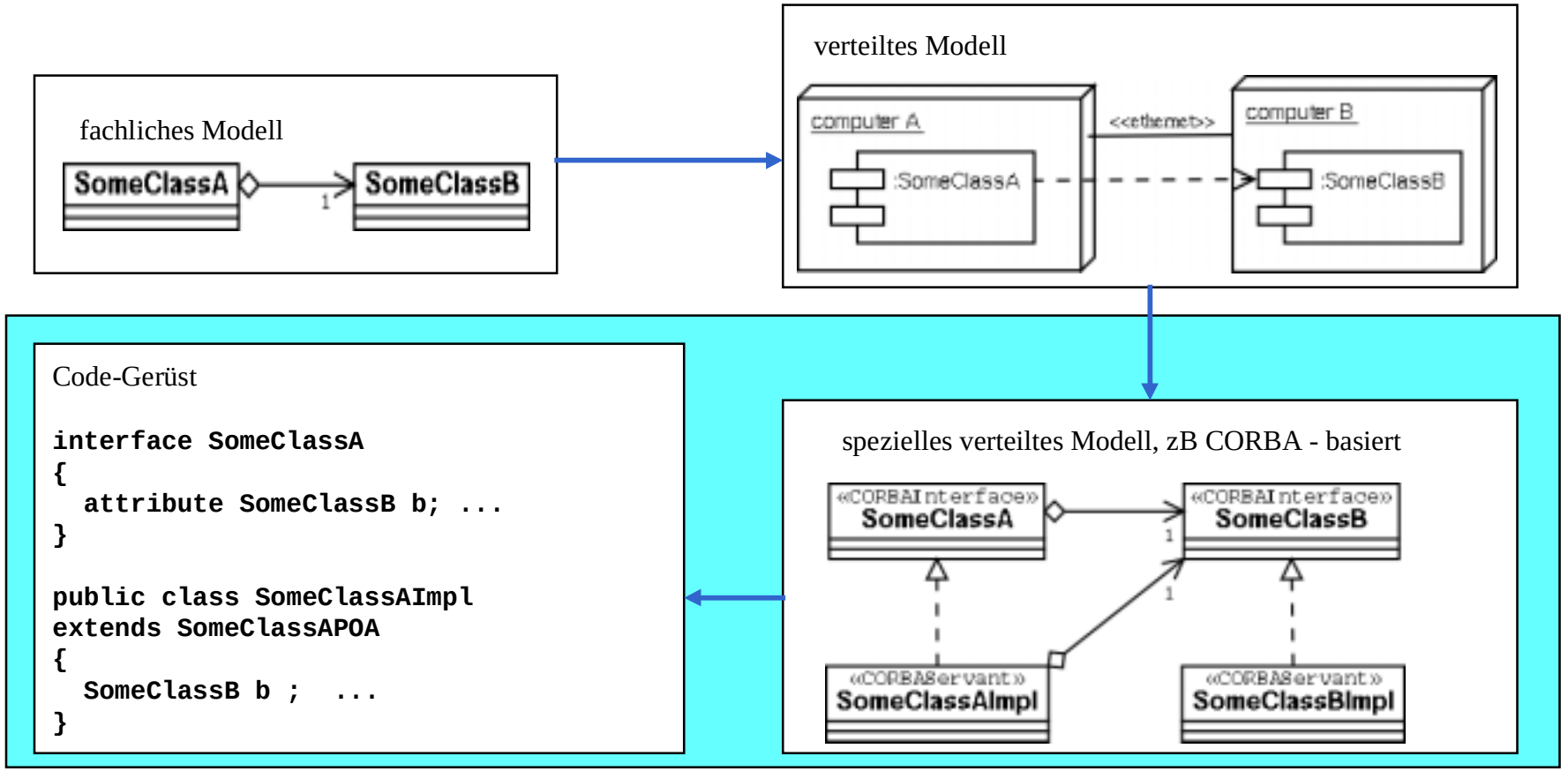
Technische Universität Dresden
Institut für Software- und Multimediatechnik
Lehrstuhl Softwaretechnologie

15. März 2002

Inhalt des Vortrags

* Motivation und Zielstellung
* UML-Erweiterungsmechanismen
* Implementierungsaspekte CORBA-basierter Anwendungen
* Modellierungsansätze CORBA-basierter Anwendungen
* Anforderungen an ein implementierungsbezogenes UML-Profile für CORBA (IUPC)
* Strukturierung des IUPC
* Metaelemente des IUPC
* Beziehungen zwischen IUPC und EDOC
* Was ist noch zu tun ...

Motivation


fachliches Modell
SomeClassA
1
SomeClassB
verteiltes Modell
computer A
:SomeClassA
<<ethernet>>
computer B
:SomeClassB
Code-Gerüst
interface SomeClassA
{
attribute SomeClassB b; ...
}
public class SomeClassAImpl
extends SomeClassAPOA
{
SomeClassB b ; ...
}
spezielles verteiltes Modell, zB CORBA - basiert
«CORBAInterface»
SomeClassA
1
«CORBAInterface»
SomeClassB
1
«CORBAServant»
SomeClassAImpl
«CORBAServant»
SomeClassBImpl

Zielstellung

Konzepte für die Erweiterung der UML 1.4 für:

- Implementierung von CORBA-Objekten
- Integration von CORBA-Diensten in Anwendungssystemen
- Möglichkeiten der Kodegenerierung aus dem UML-Modell

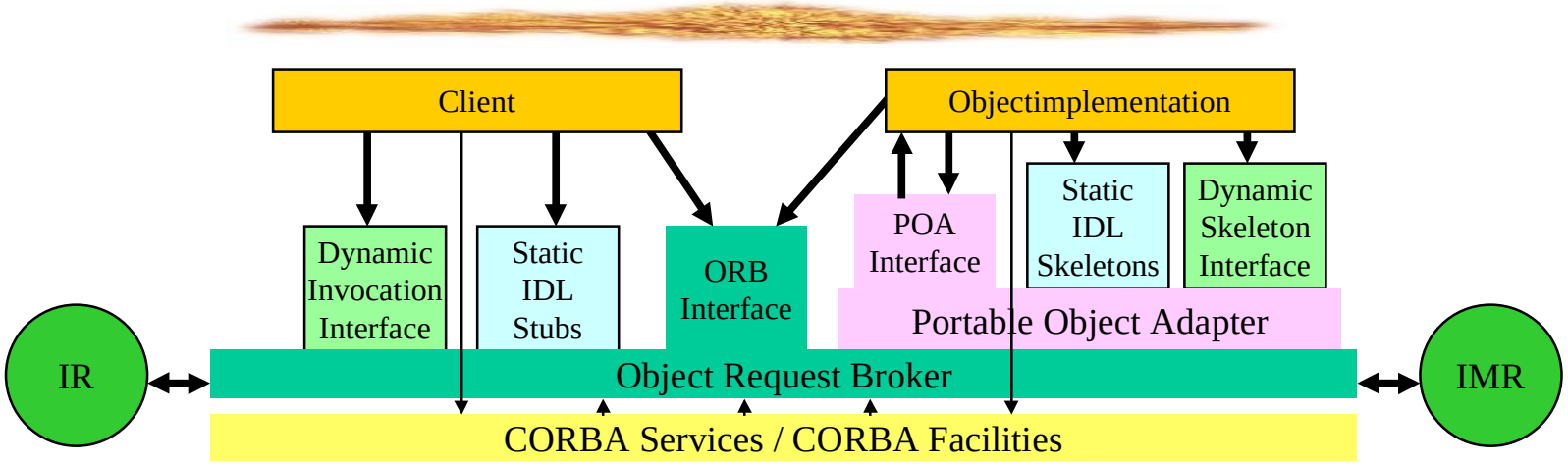
UML-Erweiterungsmechanismen

Leichtgewichtserweiterungen:
Stereotype, Constraints, TagDefinition/TaggedValue

Modelmanagement:
Model, Subsystem, Package,

UML-Profile :
- Package <<profile>>
- Zusammenfassung von UML-Erweiterungen
- Semantik der Metaelemente formal und natürlichsprachlich („well formdness rules“)

Implementierungsaspekte CORBA-basierter Anwendungssysteme



„Statische Anwendungen“
> aus IDL Kode erzeugen
> Objektimplementierung realisiert IDL-Schnittstelle durch
> Ableitung von speziellen Klassen oder
> Implementierung best. Interfaces
> aktivieren,registrieren,nutzen

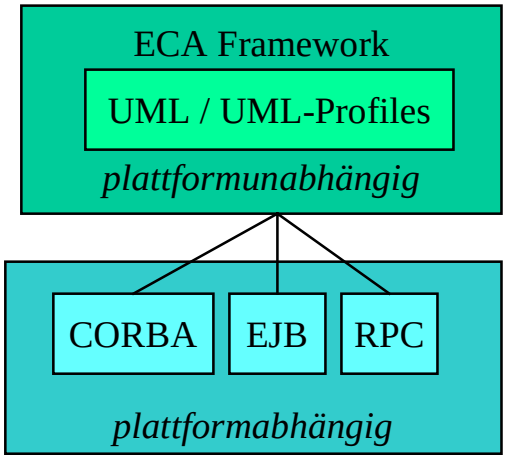
„Dynamische Anwendungen“
> IDL-Beschreibung nicht notwendig
> Objektimplementierung realisiert definierte Schnittstelle
> Ableitung von DynamicImplementation
> Implementierung „invoke“
> aktivieren,registrieren,nutzen

Verschiedene POA (Policies) ,
Nutzung von IR und IMR

Modellierung CORBA-basierter Anwendungssysteme

UML-Profile für CORBA : IDL->UML
`<<CORBAInterface>>,<<CORBAStruct>>,<<CORBAArray>>,...`

[1] Stereotype: `<<CORBAInterface>>,<<CORBAServantRealization>>,<<CORBAServant>>`
Verknüpfungsmuster: *Generalization,Delegation,Usage*
[2] Erweiterung der Verknüpfungsmuster + *Realization*
typ. Rollen von CORBA-Dienst-Objekten



EDOC - Profile (Enterprise Distributed Object Computing)
- Enterprise Collaboration Architecture (ECA)
  - Component Collaboration Architecture
  - Business Processes Profile
  - Events Profile
  - Entities Profile
  - Relationship Profile
- Standard Pattern Profile
- Technologiespezifische Modelle und deren Abbildung auf technologieunabhängige Modelle

[1] Fischer Mike. *Abstract modeling of CORBA based applications with UML*. OMG workshop: UML in the .com Enterprise, Paper, Dresden 2000
[2] Mucha Andreas. *Muster bei der Kombination von CORBA-Objektdiensten mit Anwendungssystemen*. Großer Beleg, TU-Dresden 2001

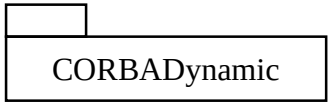
Anforderungen an ein **i**mplementierungsbezogenes **U**ML-**P**rofile für **C**ORBA (IUPC)

- > Standardkonformität
  CORBA,IDL,...
- > Vollständigkeit
  Modellierung DSI,SSI, POA, IMR, ...
- > Grad der Detailiertheit
  „grobes“ Modell -> „feines“ Modell
- > Strukturiertheit
  Zusammenfassung funktional abhängiger Komponenten
  überschaubar , verständlich
- > Implementierungsunabhängigkeit
  ein Modell -> verschiedene ORBs
- > Quellkodegenerierung
  ein Modell -> verschiedene ORBs , sprachunabhängig

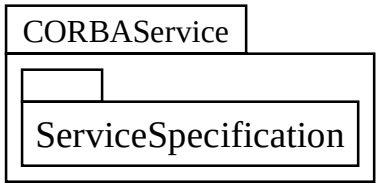
Strukturierung des IUPC

: Abbildung der CORBA Spec. / IDL in UML

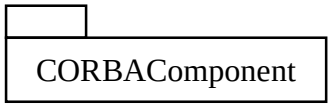
: Standardanwendung / SII,SSI



: Anwendung mit DII,DSI,versch. POA , IR, IMR

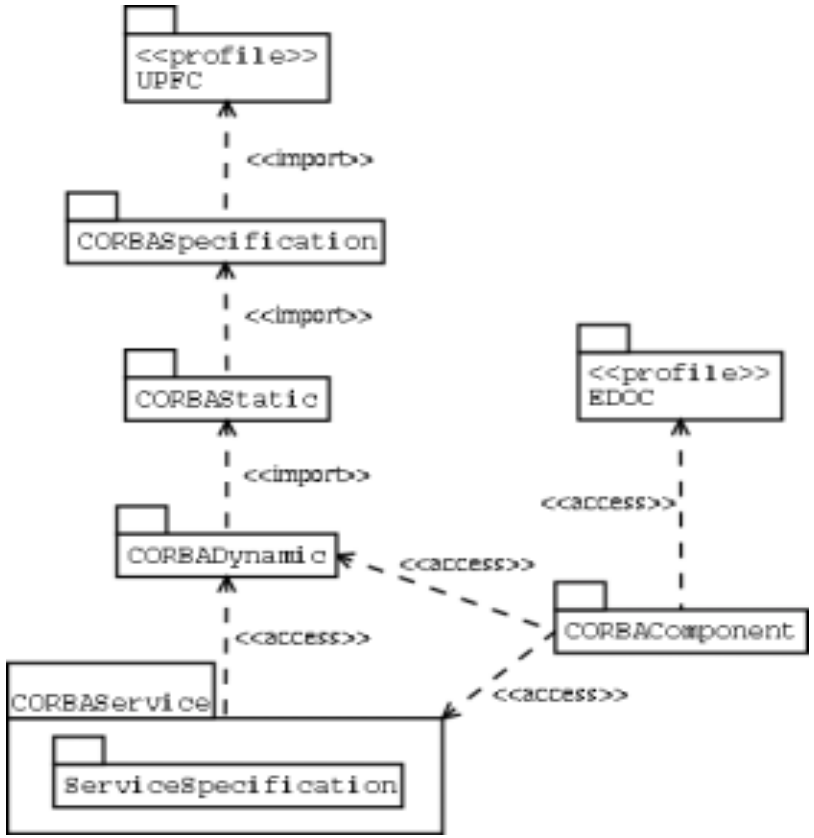


: Metaelemente für Arbeit mit CORBA-Diensten
z.B. Verknüpfungsmuster,
alle IDL-Beschreibungen der Dienste in UML



: Speziell Abbildung EDOC / allgemeine Verteiltheit

Strukturierung des IUPC(2)



Metaelemente des IUPC

`<<CORBAActivator>>` : Class (Operation) -> Aktivieren einer Objektimplementierung
`<<CORBARegistrator>>`: Class (Operation) -> Referenz verfügbar
`<<CORBAObjectImpl>>` : Zusammenfassung von <<CORBAInterface>>,
<<CORBAServantRealization>> und <<CORBAServant>>
! Unterscheidung der Beziehungen <<servant>> / <<interface>>
`<<CORBAClient>>` : Class (Operation) -> Referenz auf Server -> Methodenaufrufe

Statische Anwendungen (SSI,SII)

Dynamische Anwendungen (DII,DSI,POA ...)

Dienstintegration:

NamingService
für `<<CORBARegistrator>>`
Tag registration = string | **naming** | trading
->Anwendung Verknüpfungsmuster
für NamingService

EventService
stereotype:<<EventSupplier>>,<<EventConsumer>>
->Anwendung Verknüpfungsmuster
für EventService

Das IUPC und EDOC

CCA:

<<ProcessComponent>> → <<CORBANode>>
<<Interface>> mit <<FlowPort>>, <<OperationPort>> → <<CORBAInterface>>
<<Protocoll>> mit <<ProtocollPort>> → <<CORBAInterface>>

Entity Profile:

<<Entity>> → <<CORBAUserDefinedType>> z.B. <<CORBASequence>>
<<Key>> → <<CORBAPrimitive>>,<<CORBAConstant>> <<CORBAConstructedType>>

Events Profile:

<<Publisher>> → <<EventSupplier>>
<<Subsciber>> → <<EventConsumer>>
<<PubSubNotice>> → EventChannel
<<Subscriber>> → Event Filter (NotificationService)

Was ist noch zu tun...

- Verfeinerung der Metaelemente (IUPC,EDOC-Mapping,Dienst-Integrierung)
- Aspekte der Kodegenerierung
- Beispiel-Profile ?