open minded...
SOPHIST GROUP

Babylon im 21. Jh.: Profiles – das Ende einer standardisierten UML?

Jürgen Hahn, SOPHIST GROUP

Was war Was wird ?

- UML-Erweiterungsmechanismen
- UML 2.0: Definition und Notation von Profilen
- Zwei Profile im Detail
- Profile: Motivation & Diskussion - Historie & Ausblick
- Erstellen von Profilen
- Profile & Tools

Ernüchterndes

- Es wird nie eine vollständige UML geben.
- Nach der UML 2.0 kommt die UML 2.1.
- Der Markt ist immer schneller als die Standardisierung.
- Individuelle Anpassung war, ist und wird immer nötig sein.
- Die aktuelle UML-Version passt nach Murphy sowieso nicht zu meinem Projekt.

0.8
0.9
1.0
1.1
1.2
1.3
1.4
1.5
2.0
?

Lösung?

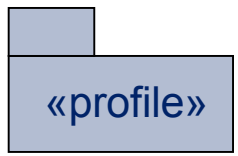
Wir modellieren unsere eigene UML!

UML Erweiterungsmechanismen
2 Möglichkeiten

1. Der mühsame Weg: Beliebige Veränderung des UML-Metamodells

Heavyweight – Extension

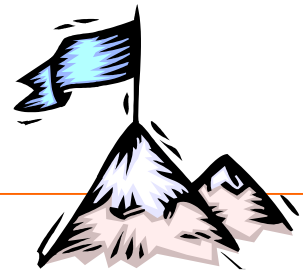
2. Der sanfte Weg: Kontrollierte Ergänzung des UML-Metamodells

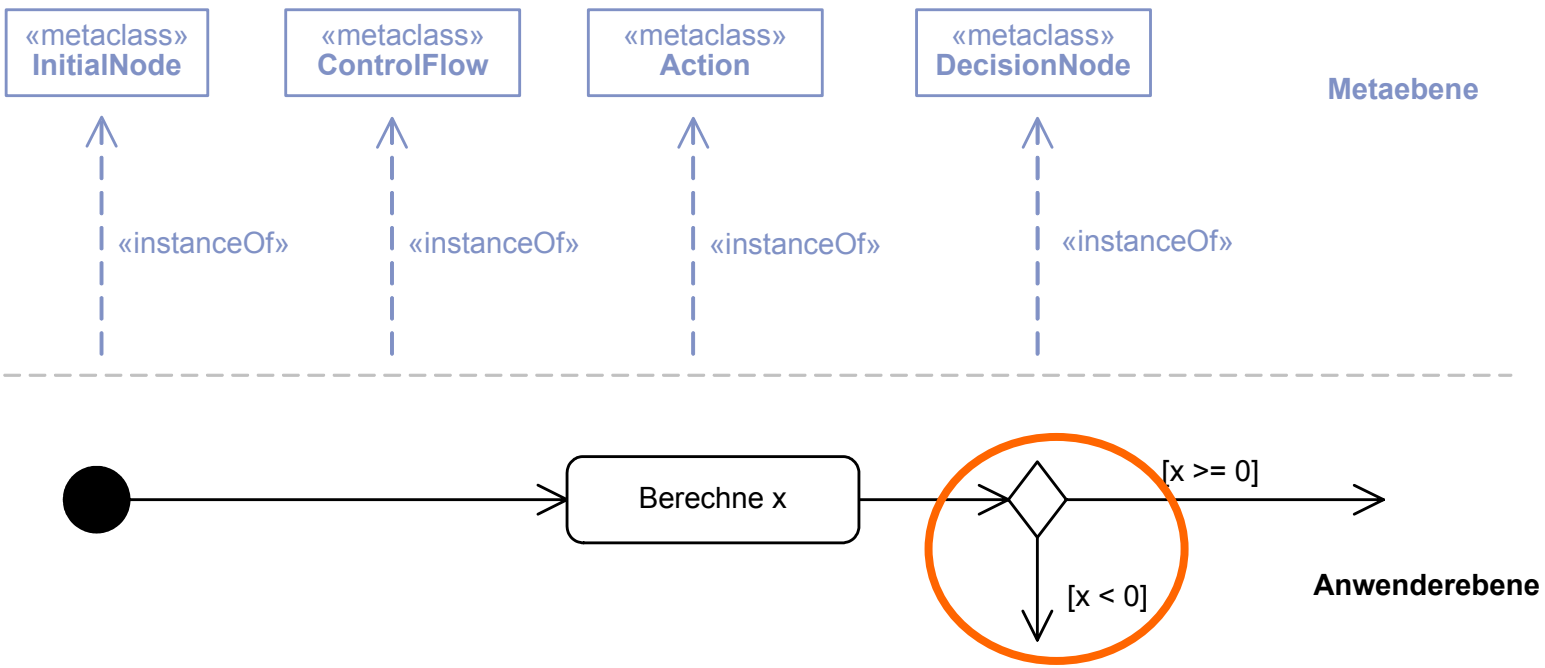


Lightweight – Extension

Heavyweight-Extension

Meta- und Anwenderebene

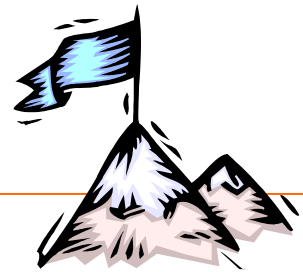


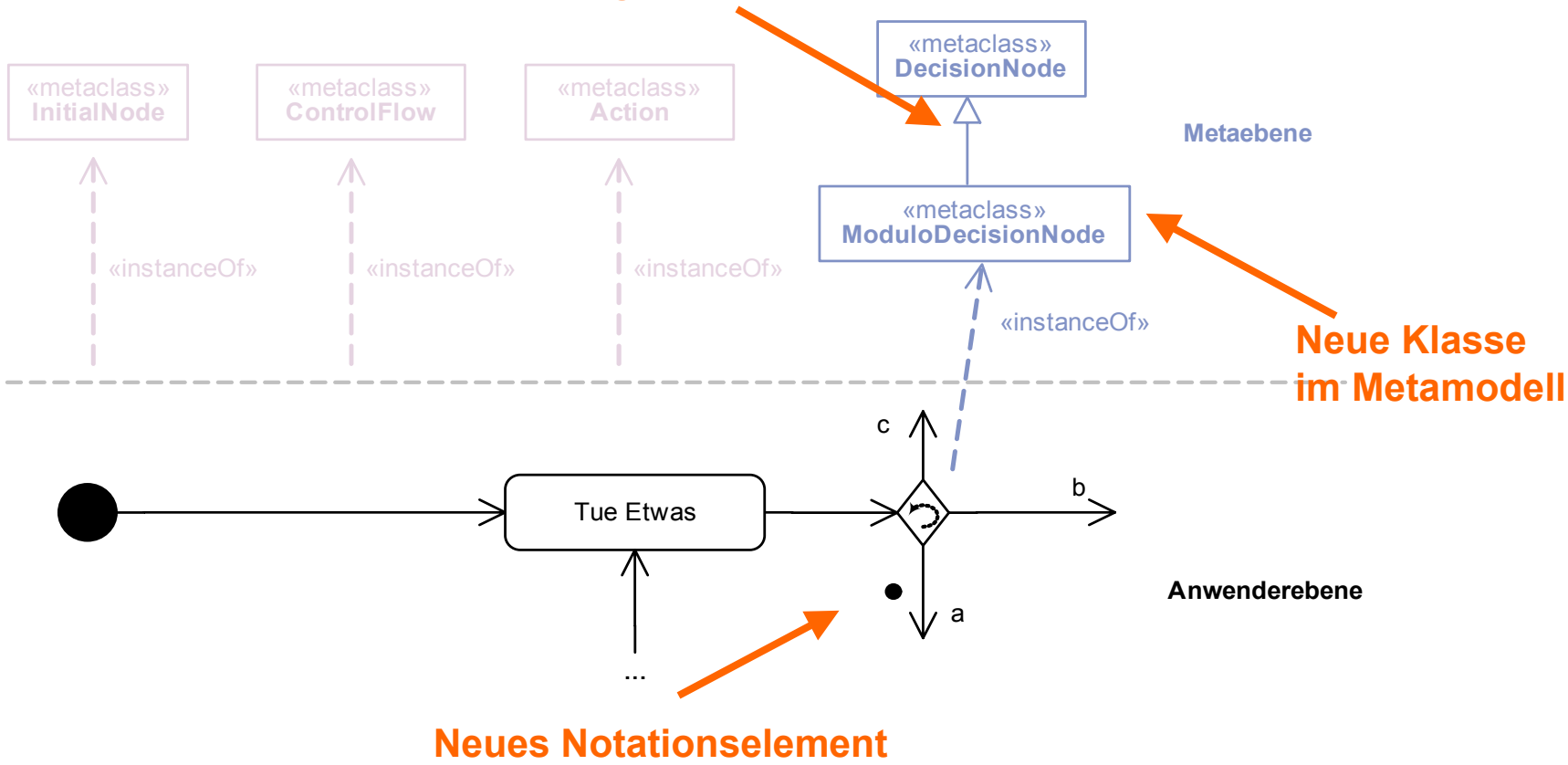


Wunsch nach neuer Semantik, z. B. modulo-Verhalten

Heavyweight-Extension

Änderungen im Metamodell





Heavyweight-Extension

Vor- und Nachteile

Vorteile

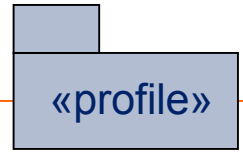
- Hohe Flexibilität (Hinzufügen & Entfernen von Metamodellelementen)
- Sie schaffen IHRE Sprache
- Basis bildet ein stabiler Modellkern

Nachteile

- Sehr gutes Grundverständnis des UML-Metamodells notwendig!
- Kleine Änderung, große Wirkung
- Tool muss Änderung des Metamodells erlauben
- In Praxis meist des Guten zuviel ...
- Die Anwender müssen eine neue Sprache lernen!

Lightweight-Extension

Was ist das?

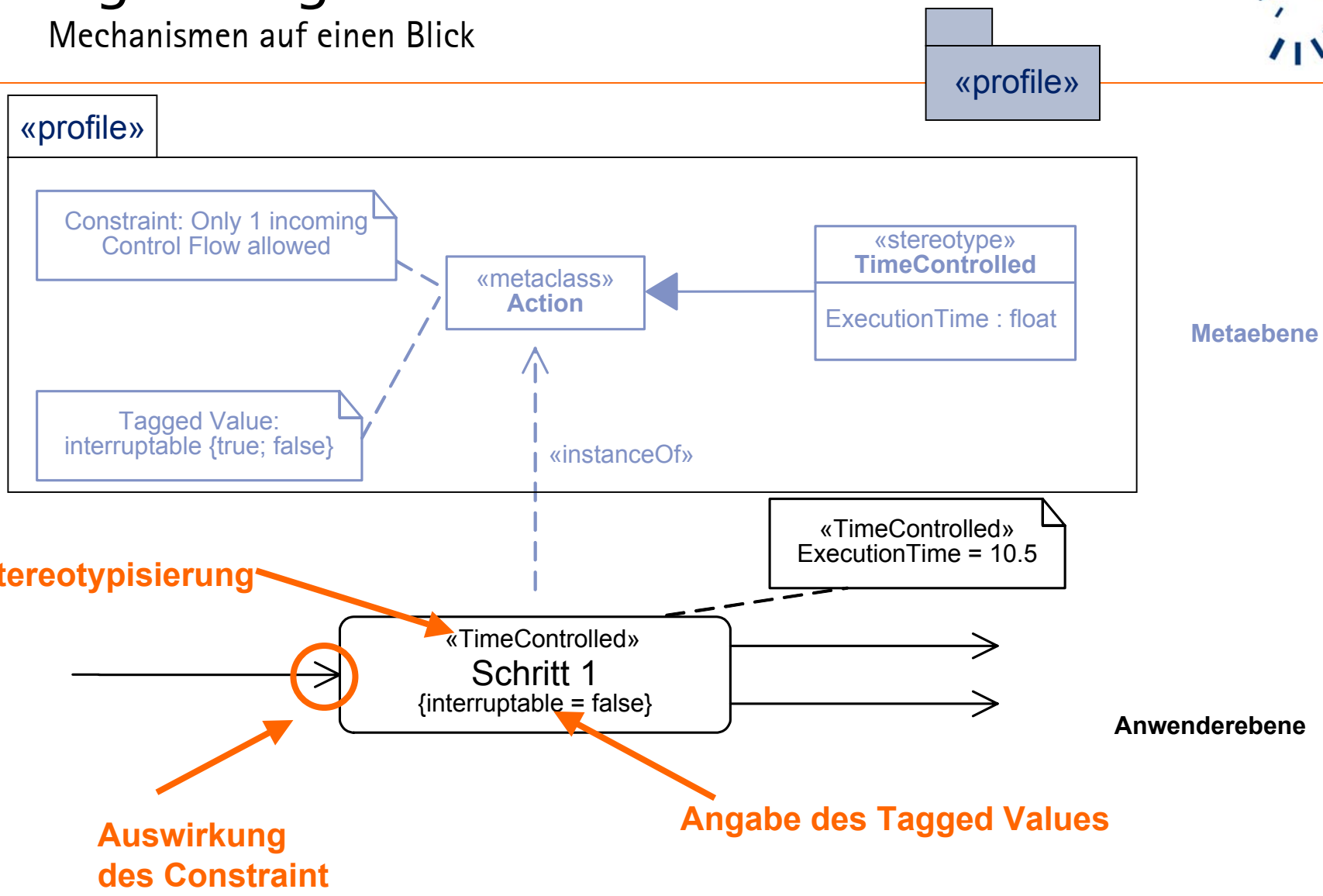


- Mechanismen zur „kontrollierten“ Anpassung des Metamodells
- Häufig als virtuelle Metamodelländerung bezeichnet
- Unterschied zur Heavyweight-Extension schwimmend
- In der UML gibt es 3 Lightweight-Erweiterungsmechanismen:
  - Constraints
  - Tagged Values
  - Stereotypen (inkl. Notationssymbole)

Ein UML Profile ist ein vordefinierter Satz von Constraints, Tagged Values und Stereotypen, die das UML Metamodell für eine spezielle Umgebung, einen Anwendungsbereich oder einen Prozess anpassen.

Lightweight-Extension

Mechanismen auf einen Blick



Was war Was wird ?

- UML-Erweiterungsmechanismen
- UML 2.0: Definition und Notation von Profilen
- Zwei Profile im Detail
- Profile: Motivation & Diskussion - Historie & Ausblick
- Erstellen von Profilen
- Profile & Tools

Constraints (Randbedingungen)

Definition

Ein Constraint präzisiert oder beschränkt die Semantik eines beliebigen UML (Meta-)Modellelements.

- Es muss boole‘sch auswertbar sein.
- Sehr informell (natürliche Sprache) ...
- ... aber auch sehr formal (Programmier- oder Formelsprache, Object Constraint Language (OCL)) definierbar.
- Die Idee steht und fällt mit der Interpretierbarkeit durch Tools.
- Zu viele Constraints reduzieren die Lesbarkeit des Modellbilds.

Constraints (Randbedingungen)

Notation & Anwendung

- Constraints werden häufig in einem Profil als Text formuliert.
- Im Modell werden Sie in geschweifte Klammern gesetzt.

Object Constraint Language

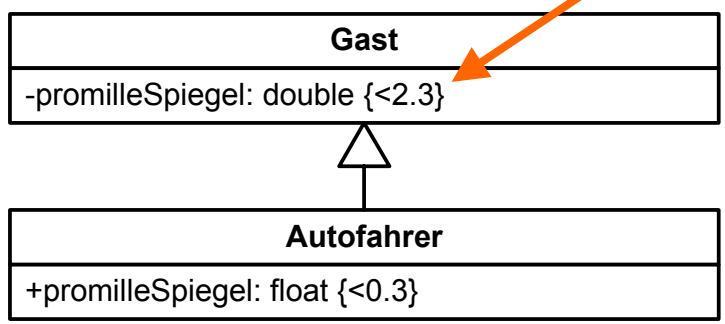
```
slot->forAll(s |
    classifier->exists(c | c.allFeatures()->includes(s.definingFeature)
    )
```

[1] The type of the structural feature of the action must be Timezone.

[2] The type of the result must be Timezone.

Constraint als Textaufzählung

Constraint im (Meta-)Modell



Tagged Values

Definition

Ein Tagged Value repräsentiert eine Eigenschaft (Property) eines UML Elements. Ein Tagged Value besteht aus dem Namen (Tag) und dem Wert (Value) der Eigenschaft.

- Sind vergleichbar mit Attributen.
- Liegen außerhalb des Anwendungsgebiets.
- Liefern Managementinformationen (Aliasnamen, Autor, Version, Datum, ...)
- Steuern Modellierungstool oder Codegenerator (Programmiersprache, Name im Code, Pfad, ...)
- Seit UML 2.0: kein eigenes Metamodellelement

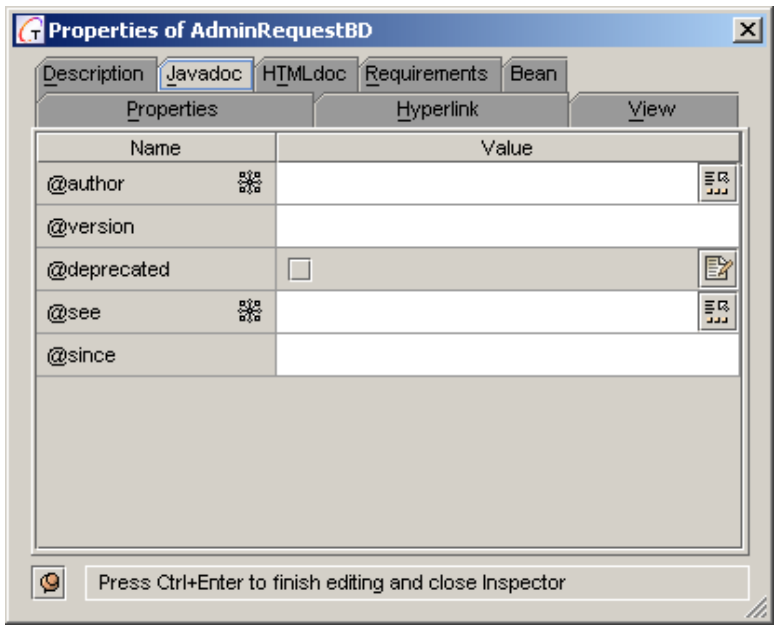
Tagged Values

Notation & Anwendung

- Constraints werden als Text: {Tag = value} notiert
- Bei Boolschen Werten meist nur {value}, z. B. {persistent}
- Sehr viele toolabhängige Notationen.

Tag Value

EineKlasse {author=Jürgen Hahn, company = SOPHIST, date = 04-01-01}
Attribute ...
Operationen ...



Stereotype

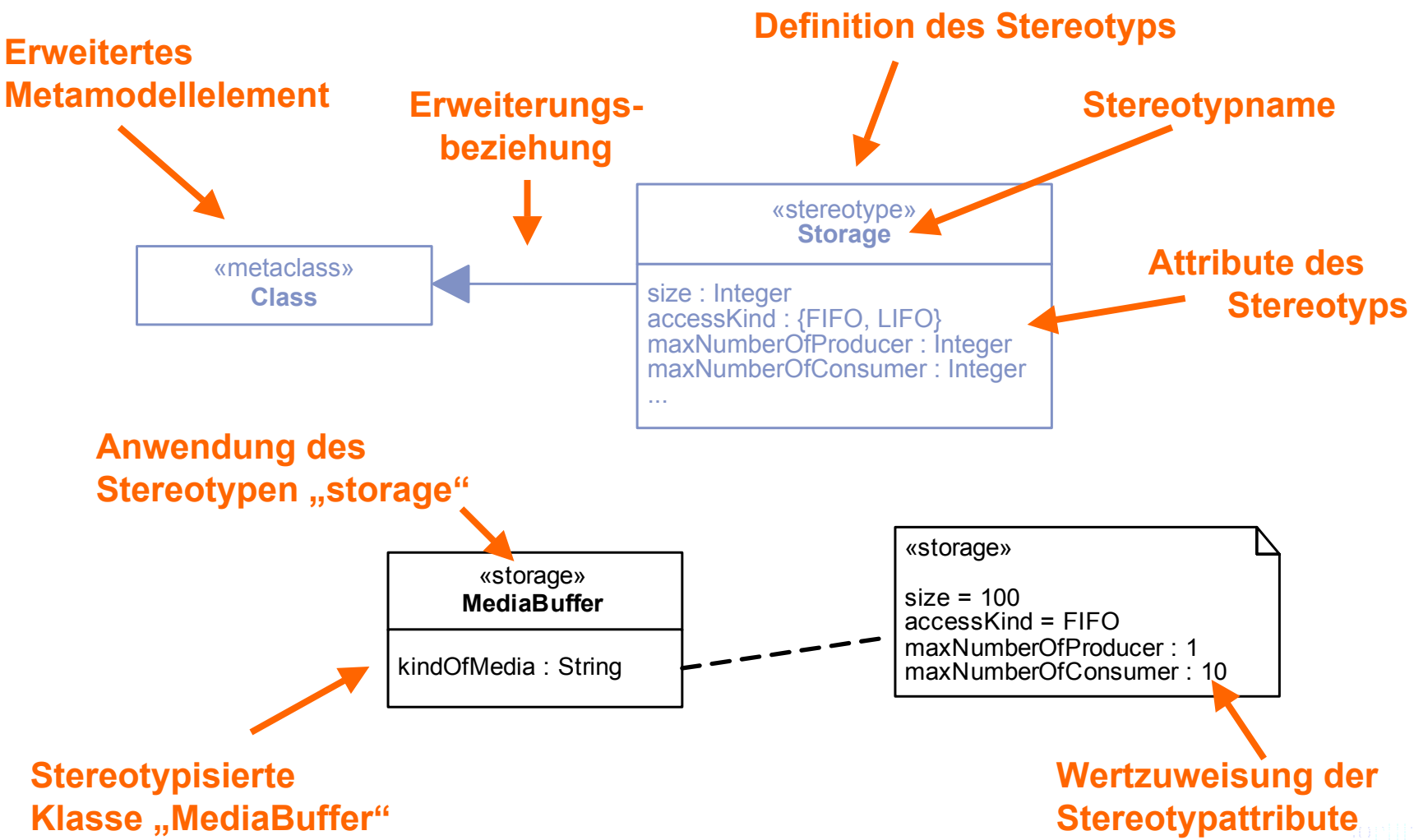
Definition

Ein Stereotyp ist eine Metamodellklasse, die beschreibt wie andere Metamodellklassen erweitert werden dürfen.

- Stereotypen besitzen Eigenschaften „normaler“ Klassen.
- Sie dürfen daher Attribute besitzen und sind spezialisierbar.
- Stereotypen sind (auch) auf andere Stereotypen anwendbar.
- Für Stereotypen dürfen eigene Notationssymbole definiert werden.
- Stereotypen dürfen selber mit Constraints oder TaggedValues versehen werden.
- Stereotypisierung ist der bevorzugte Erweiterungsmechanismus in Profilen.

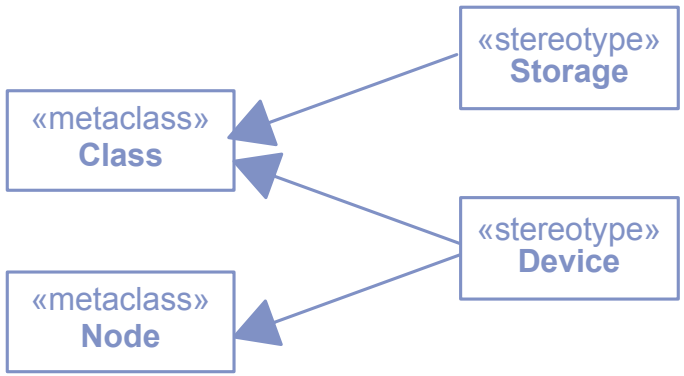
Stereotype

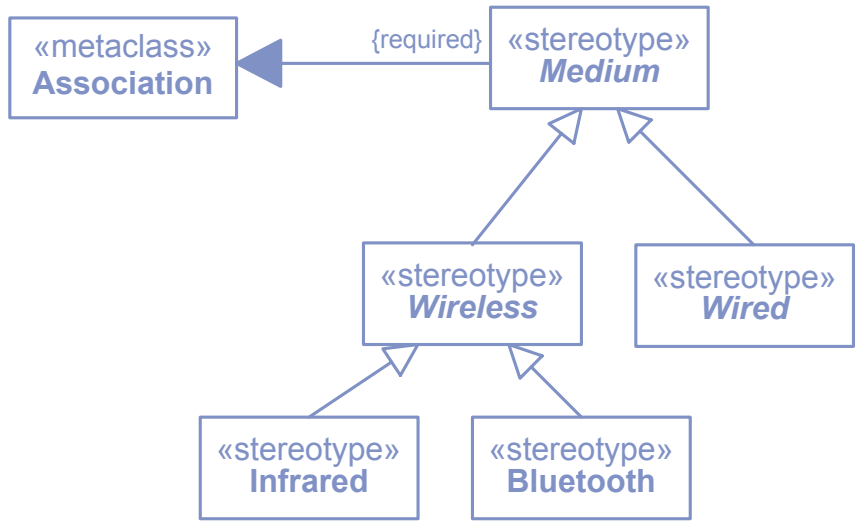
Notation & Anwendung

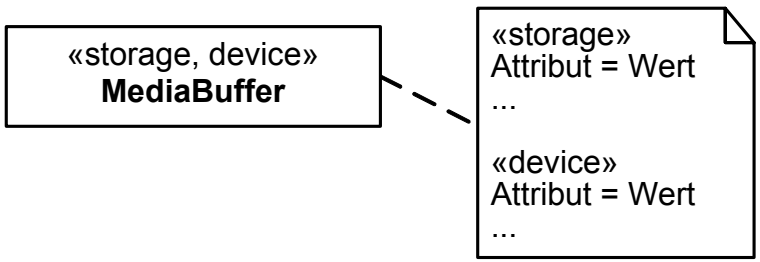


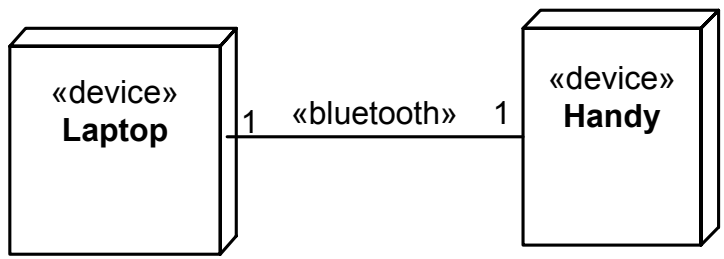
Stereotype

Notation & Anwendung II









Stereotype

Notation & Anwendung III

- In vielen Tools wird eine Tabellenform zur Definition der Stereotypen gewählt.
- Zu jedem Stereotyp ist eine semantische Definition und Erläuterung im Profil zu erstellen.

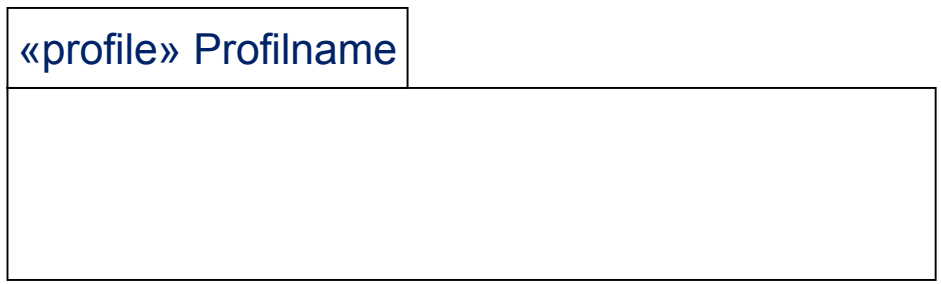
Stereotyp	**Definition**	**Metamodellelement**	**Attribute**	**...**
«Storage»	...	Class	size	...
		Node	accessKind	
		Component	...	

Profile

Definition

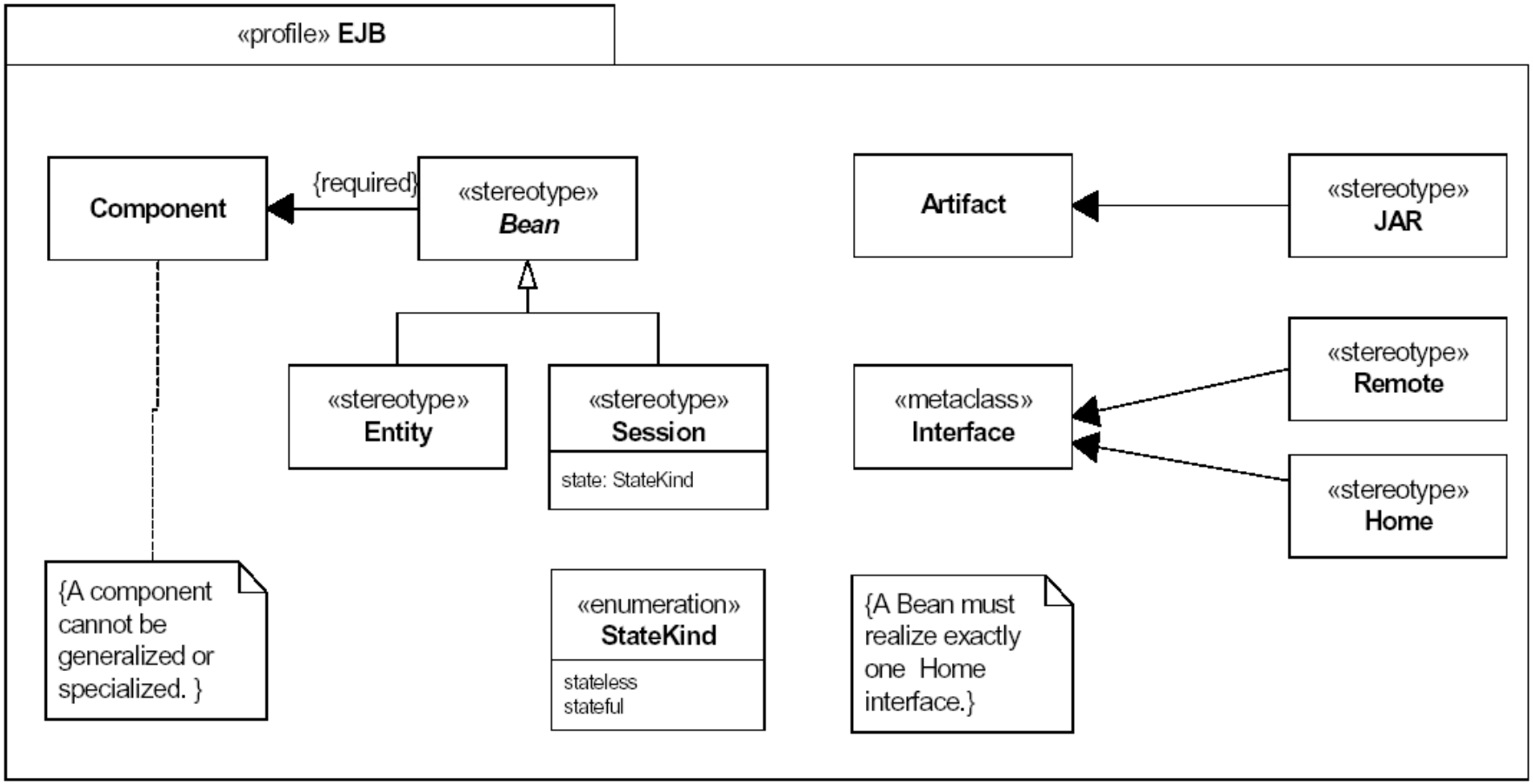
Ein Profile im Sinne der UML 2.0 ist Paket (Package), dass Konstrukte (vornehmlich Stereotypen) zur Erweiterung eines Metamodells enthält.

- Profile sind erstmals im Metamodell der UML 2.0 verankert.
- Ein UML 2.0 Profil übernimmt Notation und Semantik eines Pakets.
- Profile werden durch «profile» vor dem Paketnamen gekennzeichnet:



Profile

Notation & Anwendung I

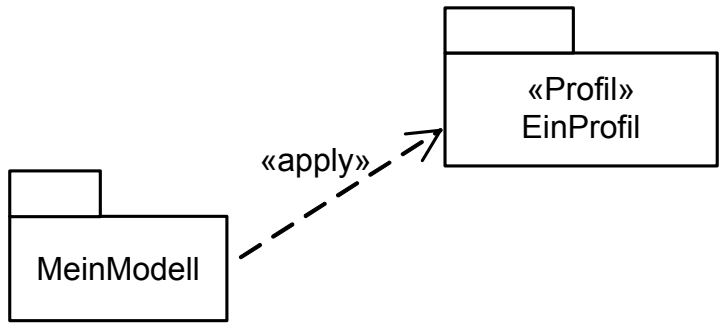


Aus: Unified Modeling Language: Superstructure Version 2.0 (ptc/03-07-06)

Profile

Äbhängigkeiten

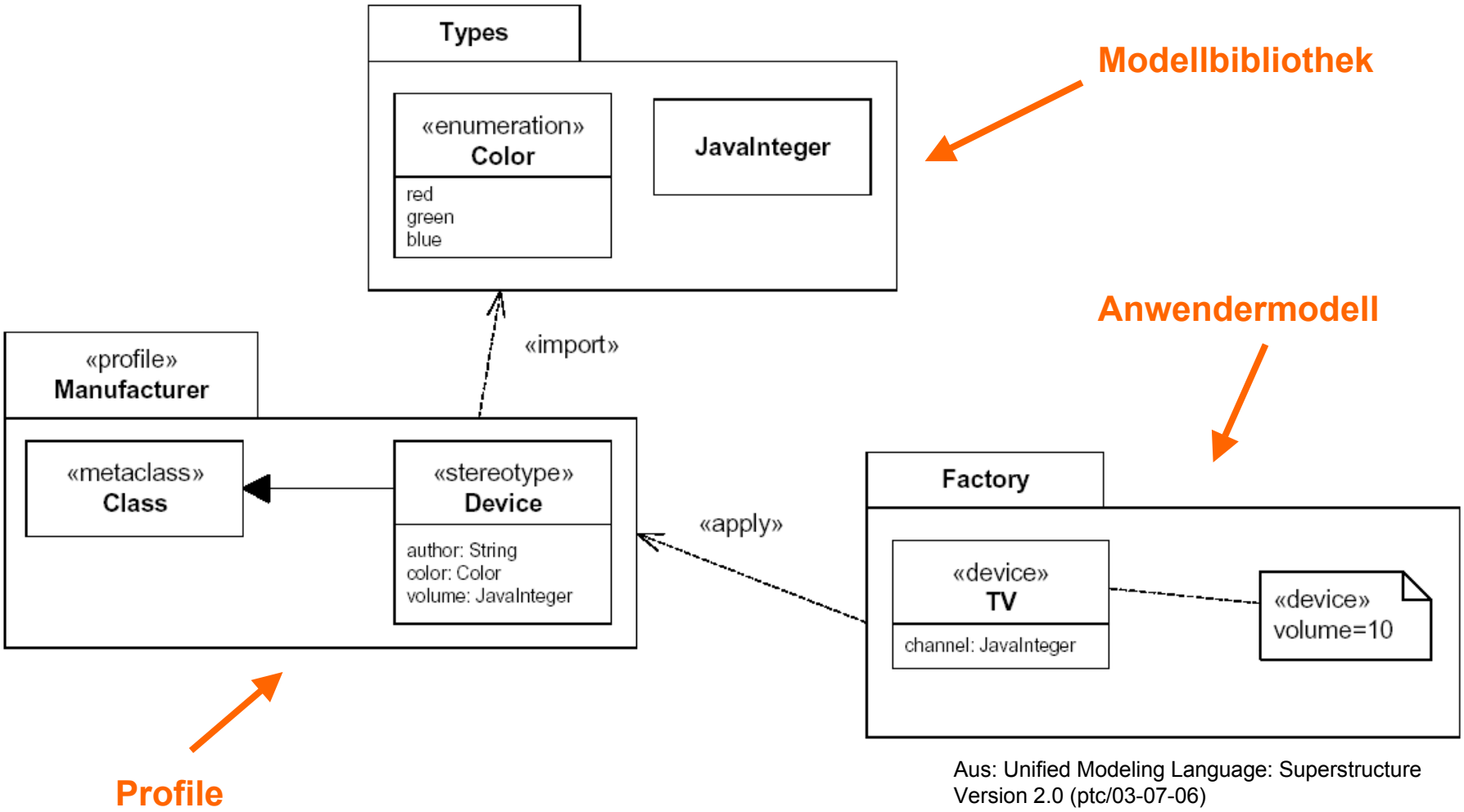
- Um die Elemente (Stereotypen, Constraints, ...) eines Profils zu nutzen, muss das Profil angewendet (applied) werden:



- Ein Profil kann auch auf ein anderes Profil angewendet werden.
- Die Anwendungsbeziehung ist eine spezielle Importbeziehung.
- Auf ein Modell/Profil könne gleichzeitig mehrere Profile angewendet werden.
- Die Anwendung des Profils verändert im Zielpaket nichts.

Profile

Notation & Anwendung II



Aus: Unified Modeling Language: Superstructure Version 2.0 (ptc/03-07-06)

Was war Was wird ?

- UML-Erweiterungsmechanismen
- UML 2.0: Definition und Notation von Profilen
- Zwei Profile im Detail
- Profile: Motivation & Diskussion - Historie & Ausblick
- Erstellen von Profilen
- Profile & Tools

UML Testing Profile

Fakten

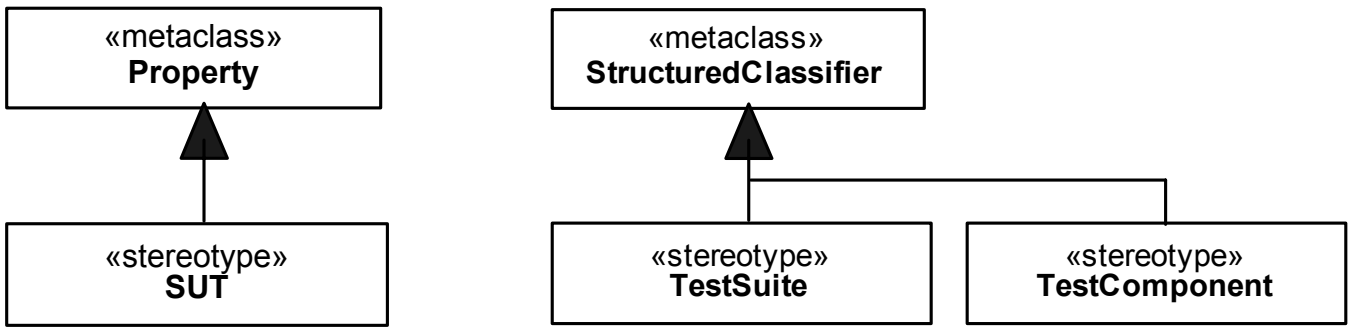
- Derzeit in der Finalisierung (ca. bis 03/05)
- Umfasst momentan 89 Seiten
- Ziel: Entwurf, Visualisierung, Spezifizierung, Analyse Dokumentation der Artefakte eines Testsystems mit der UML
- Basiert auf UML 2.0 Superstructure
- Besteht aus 4 logischen Gruppen:
  - Testarchitektur
  - Testdaten
  - Testverhalten
  - Zeitaspekte beim Test
- Definiert zusätzlich zum UML-Profil ein eigenes Metamodell (heavyweight-Extension)

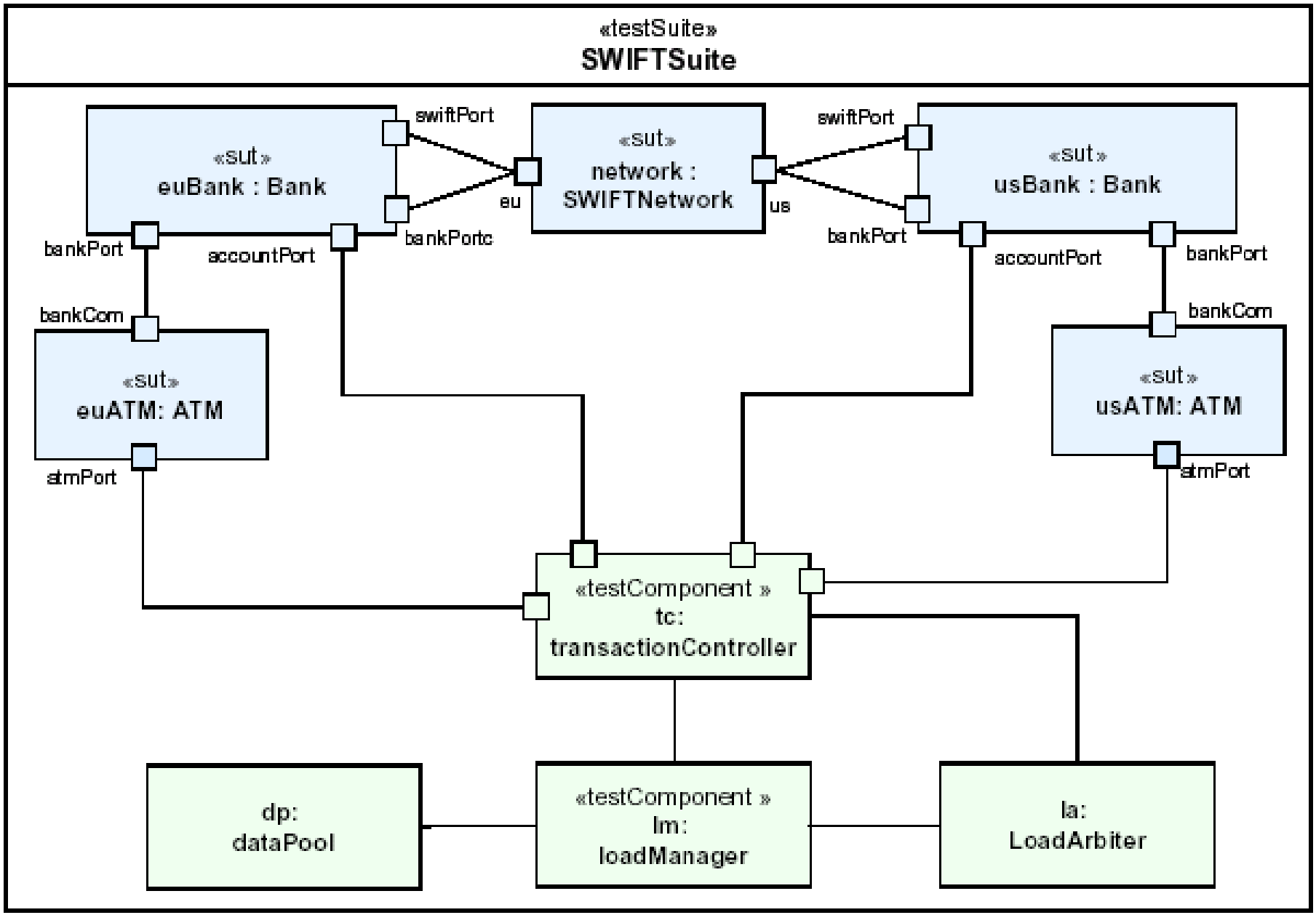
UML Testing Profile

Definierte Stereotypen

- Stereotypen für Testkomponenten, TestSuites, SUT, Testziel, Testfälle, Logaktionen, Zeit, Dauer, Start/Stopptimer, ...

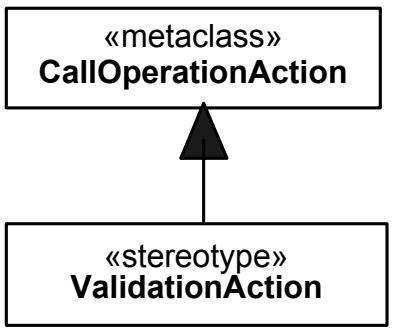
- Beispiele (UML 2):

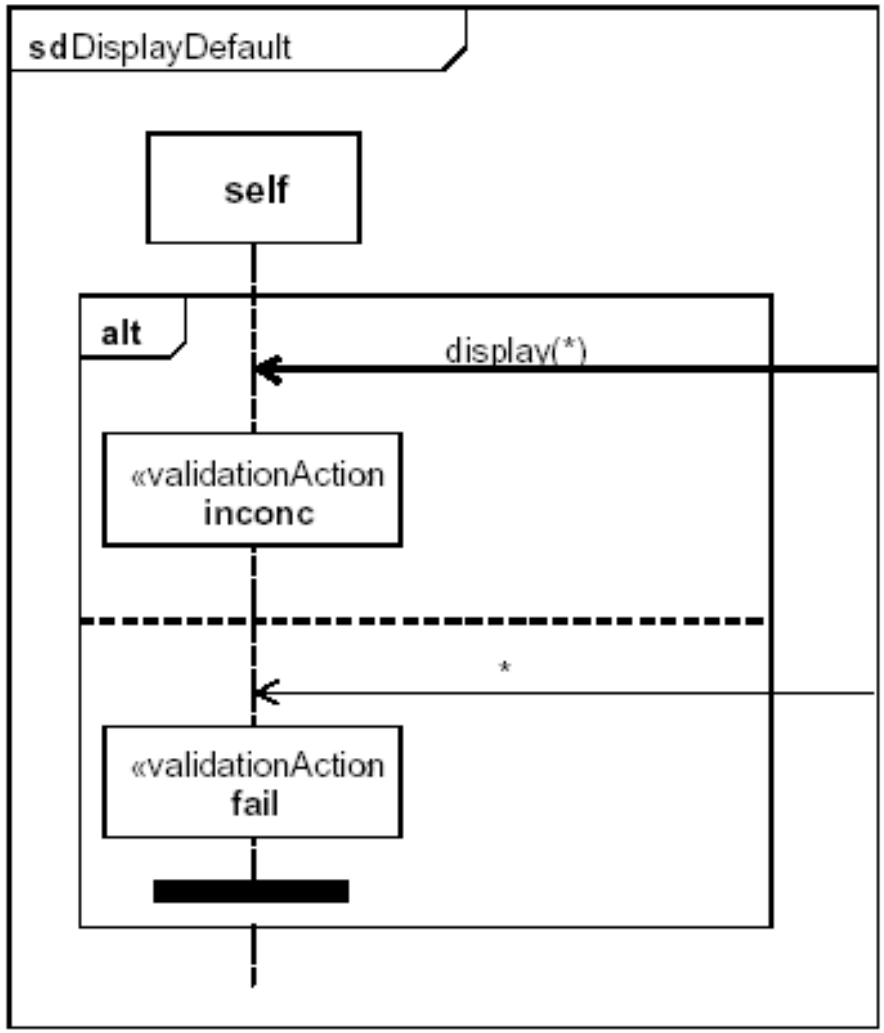



«testSuite»
SWIFTSuite
«sut»
euBank : Bank
swiftPort
bankPort
accountPort
bankPortc
«sut»
network :
SWIFTNetwork
eu
us
swiftPort
bankPort
«sut»
usBank : Bank
accountPort
bankPort
bankCom
«sut»
euATM: ATM
atmPort
bankCom
«sut»
usATM: ATM
atmPort
«testComponent »
tc:
transactionController
dp:
dataPool
«testComponent »
lm:
loadManager
la:
LoadArbiter

UML Testing Profile

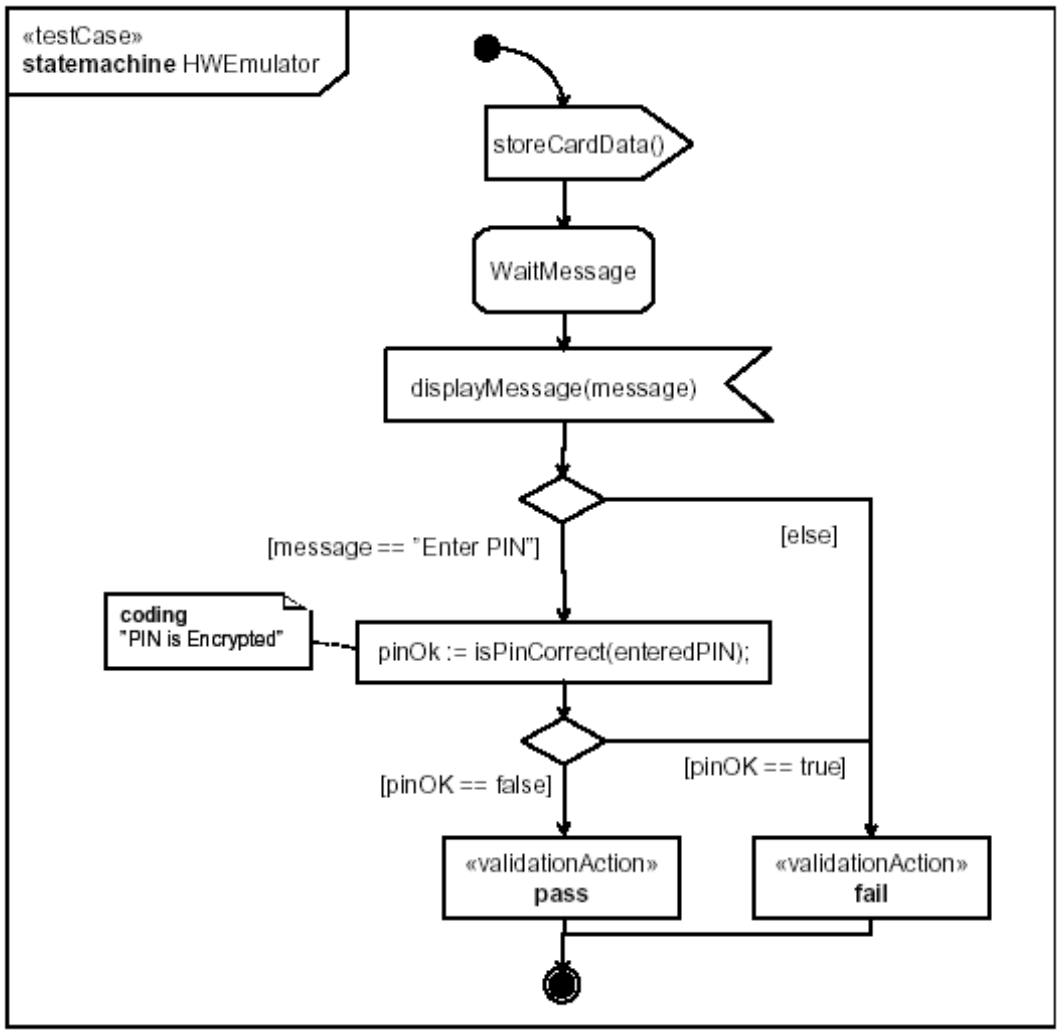
Beispiel Validierungsaktion I

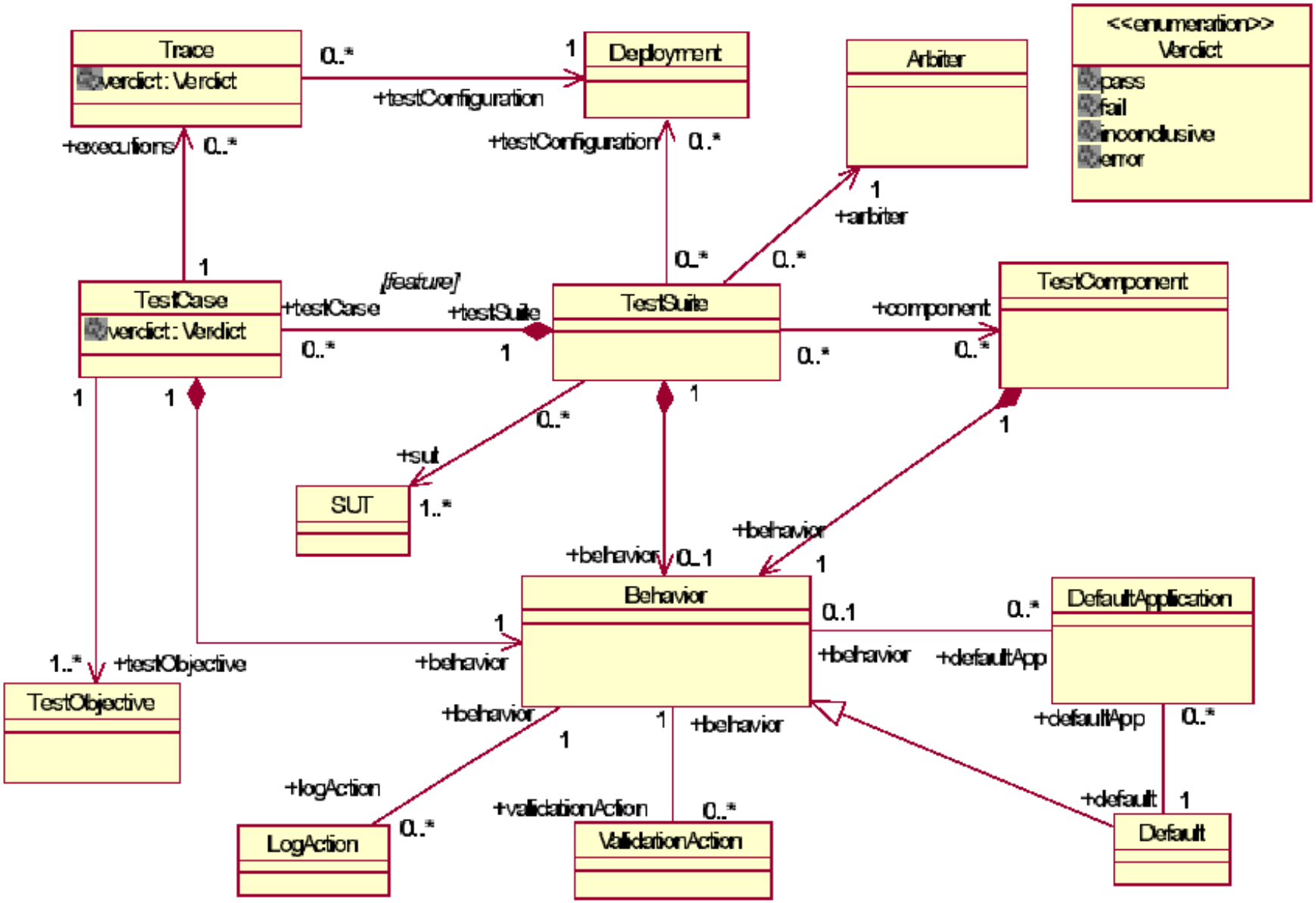




UML Testing Profile

Beispiel Validierungsaktion I



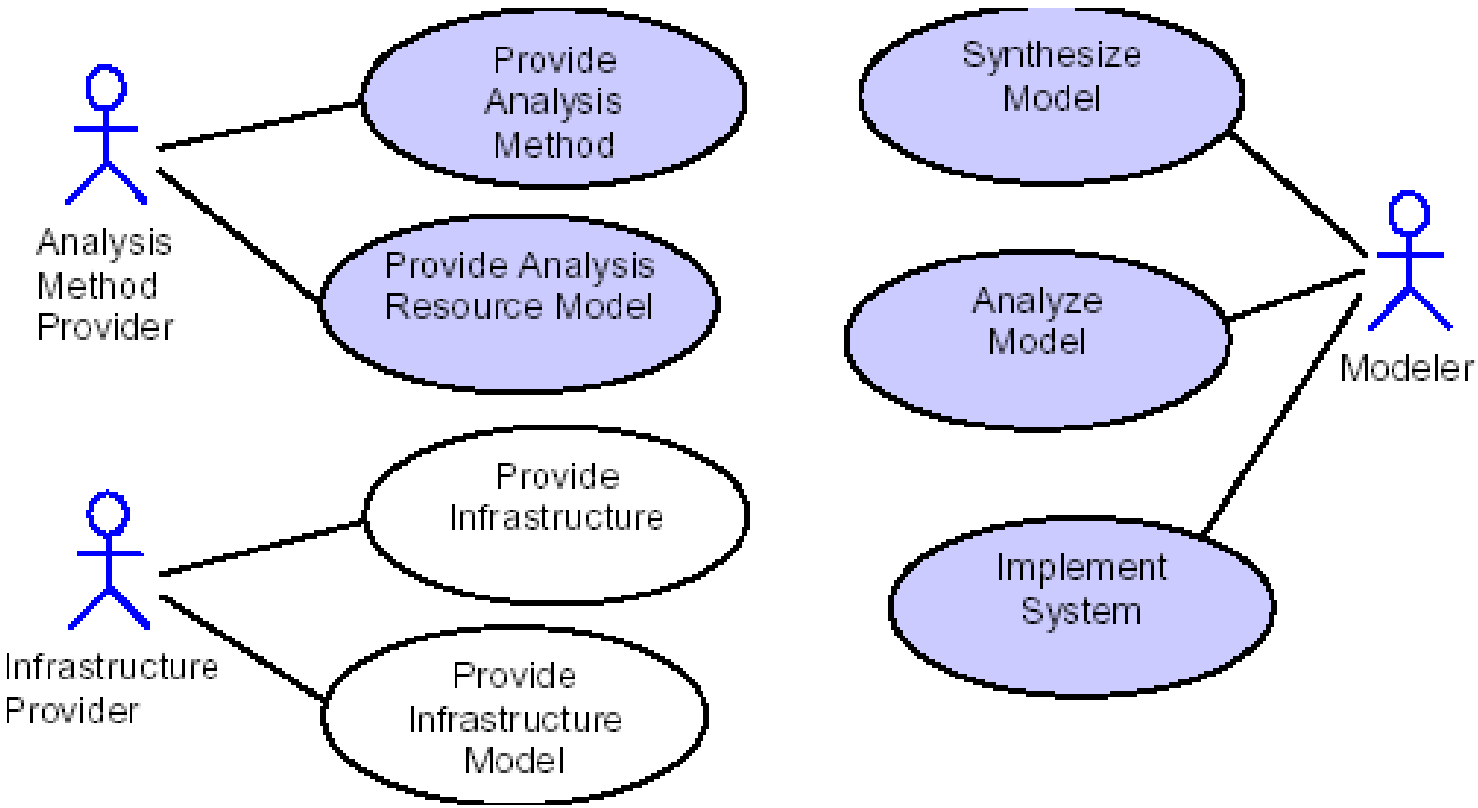


Unified Modeling Language: Testing Profile
Version 2.0 (ptc/03-08-03)

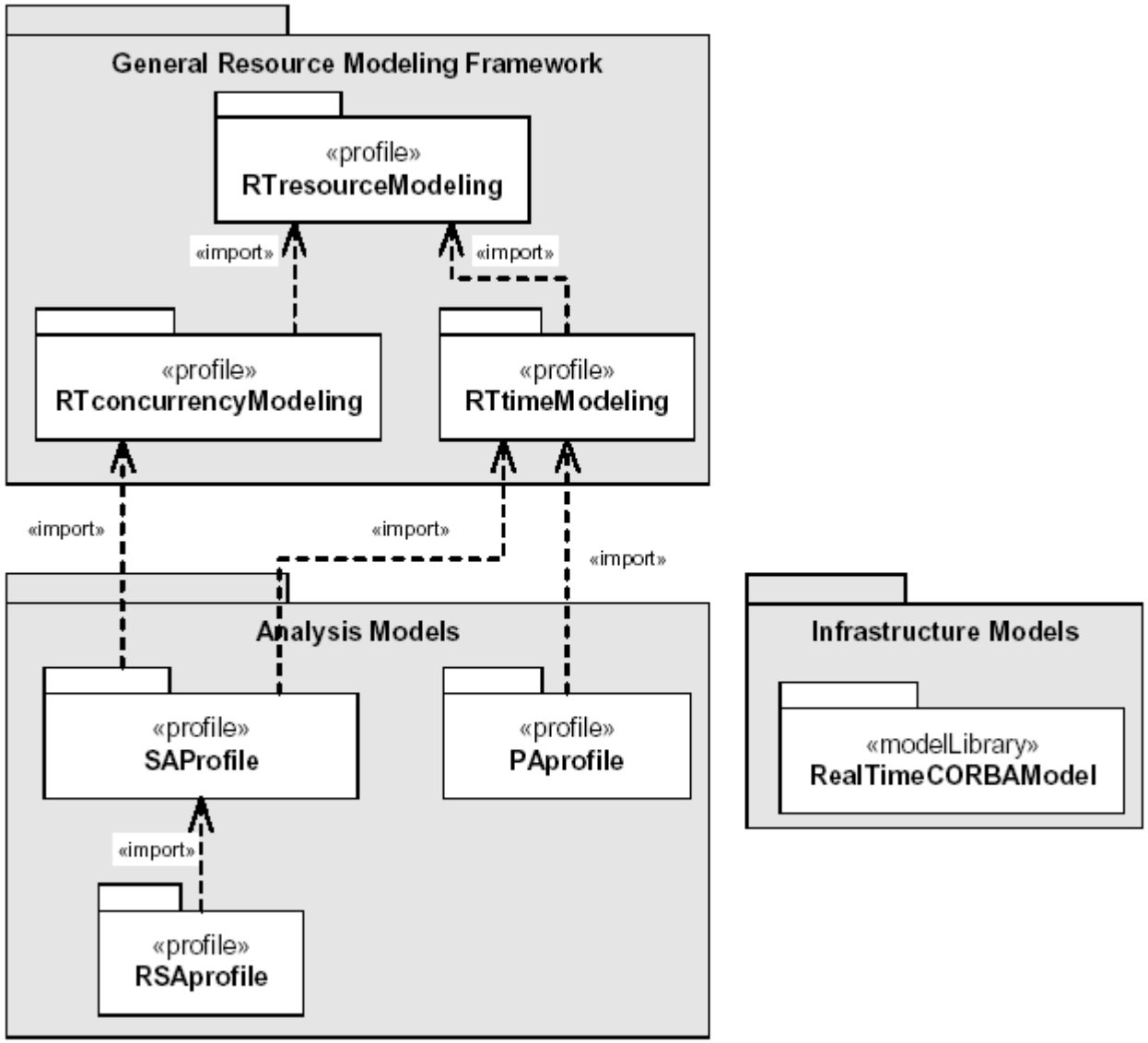
UML Profile for Schedulability, Performance and Time Fakten

- Derzeitige gültige Version: 1.0 (von 2001) (ca. 232 Seiten)
- Überarbeitung im Hinblick auf UML 2.0 Metamodell nötig
- Ziel: Leichtere Modellierung von Real-Time Anwendung mit der UML (ohne Festlegung auf das Vorgehen, Methoden oder das Design) durch standardisierte Elemente und toolneutral
- Untergliedert sich in logische Gruppen:
  - Resource Modeling
  - Time Modeling
  - Concurrency Modeling
  - Scheduling Modeling
  - Performance Modeling

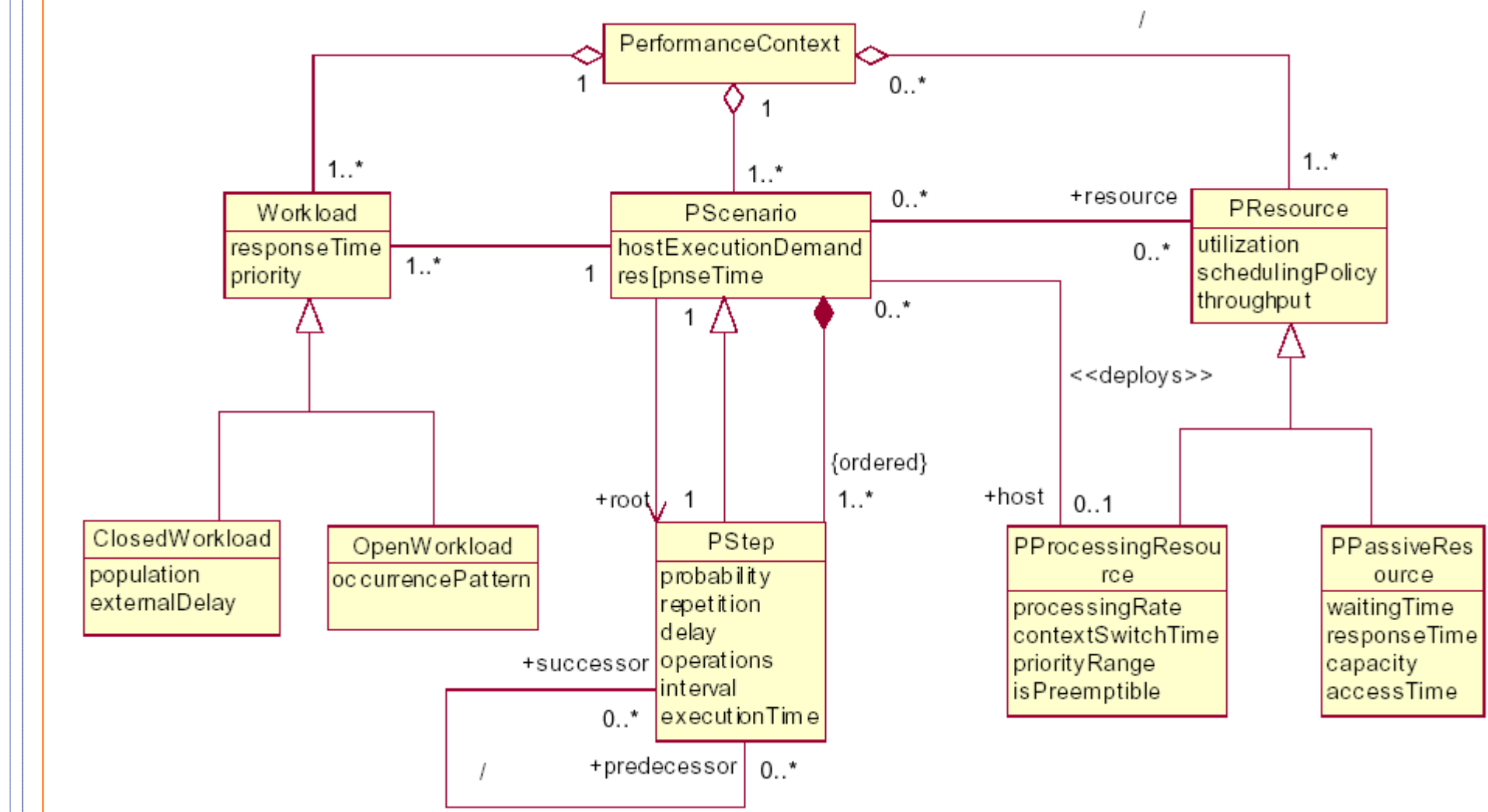
UML Profile for Schedulability, Performance and Time UseCases



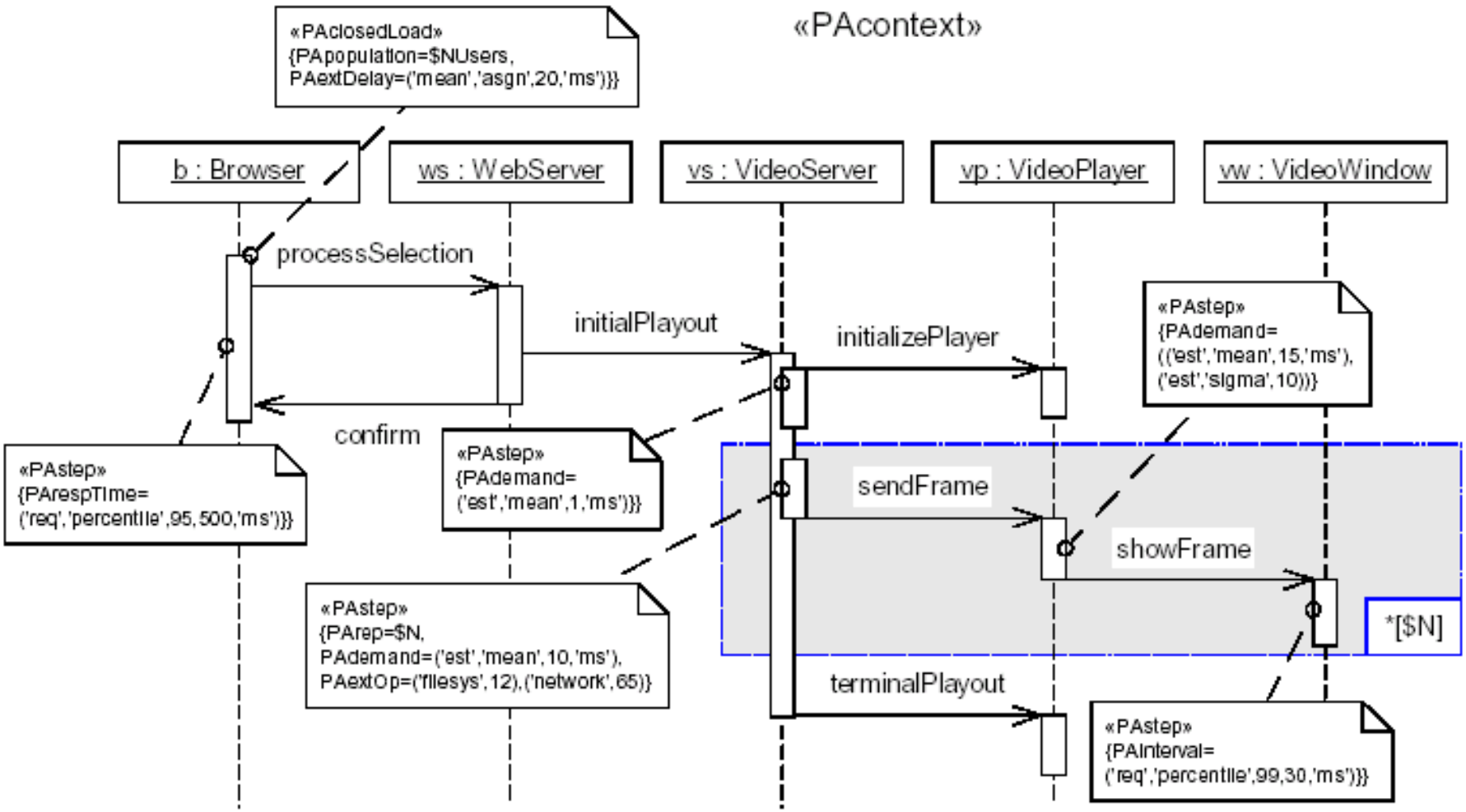
UML Profile for Schedulability, Performance and Time Profilstruktur



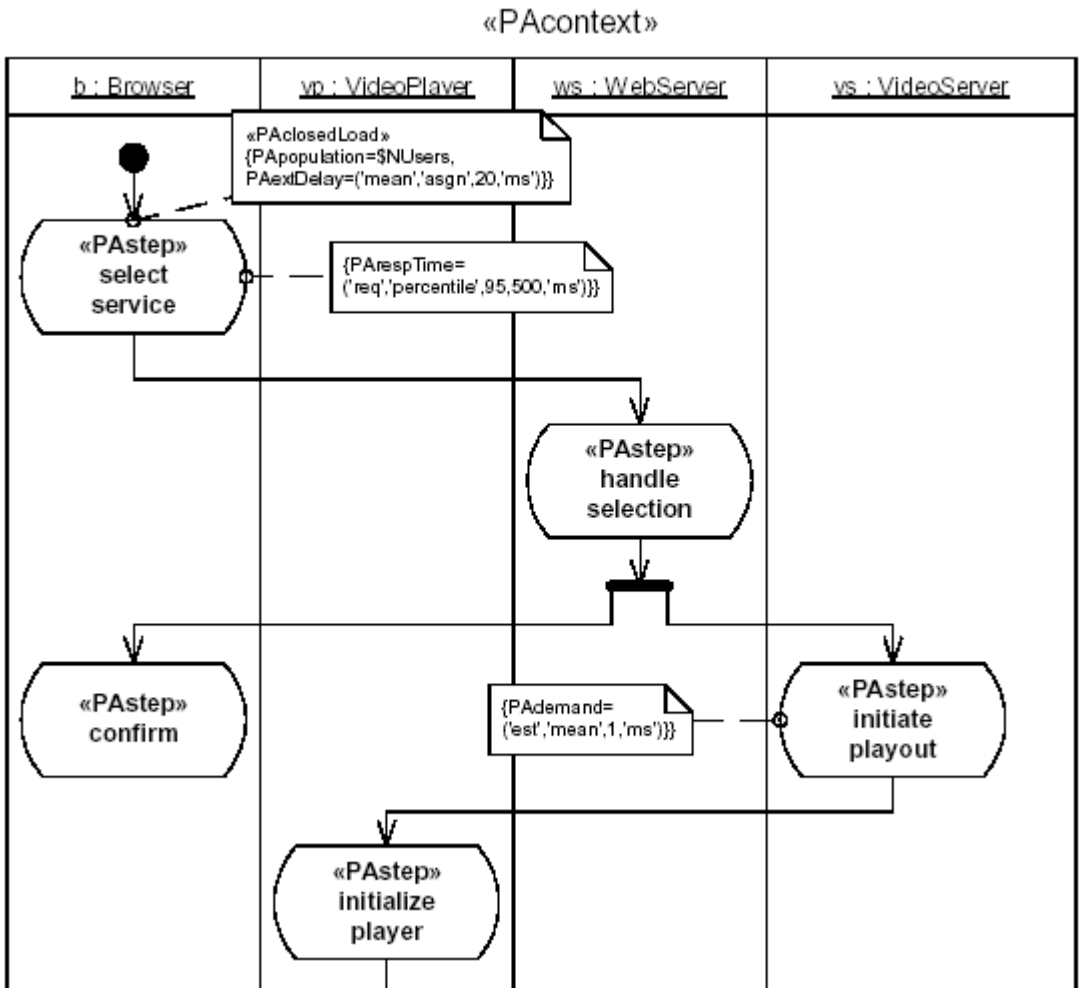
UML Profile for Schedulability, Performance and Time Fachmodell



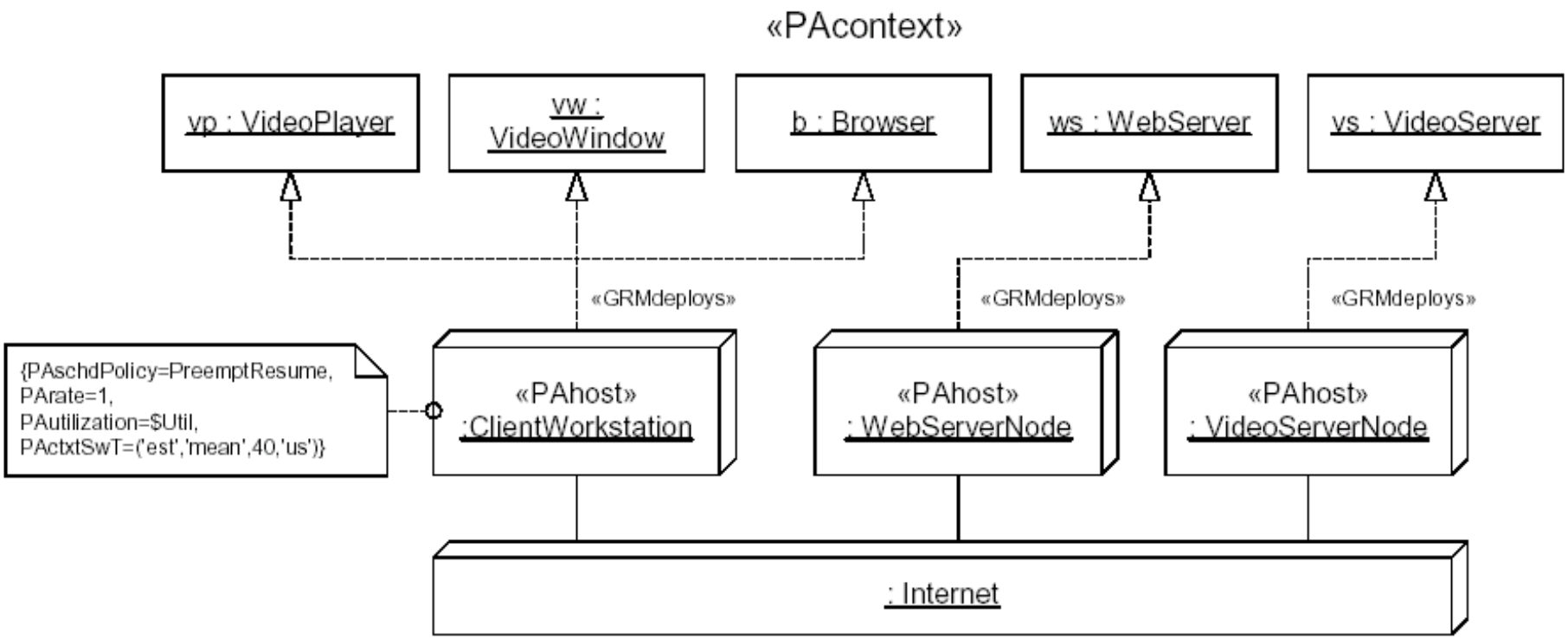
UML Profile for Schedulability, Performance and Time Beispiel Annotation I



UML Profile for Schedulability, Performance and Time Beispiel Annotation II



UML Profile for Schedulability, Performance and Time Beispiel Annotation IV

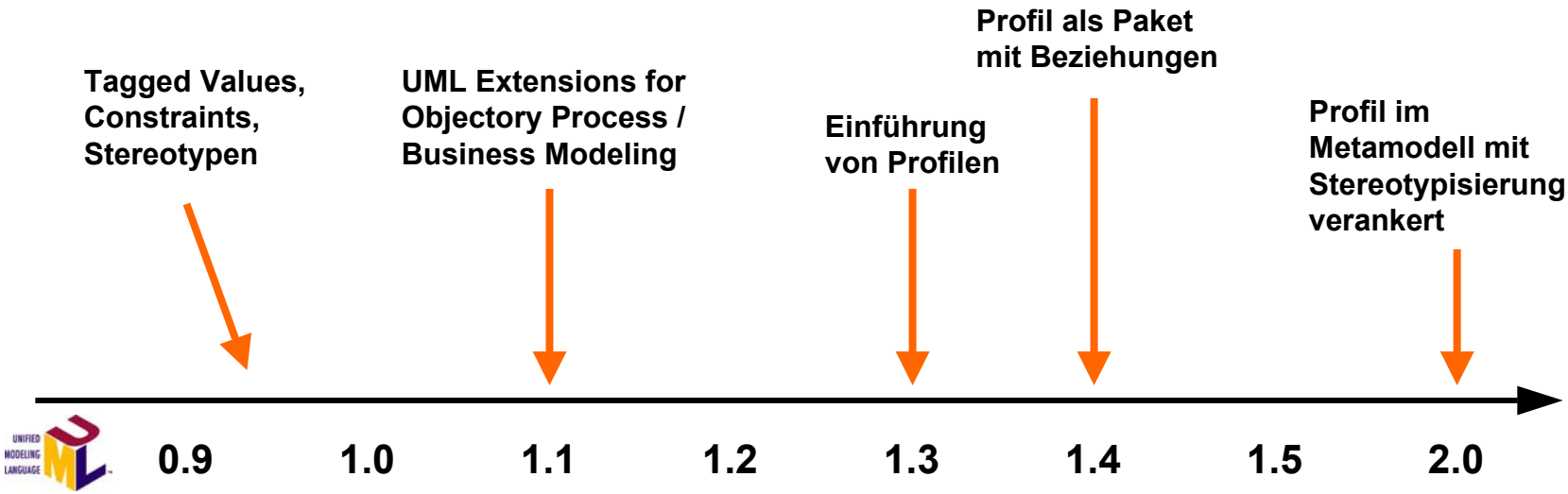


Was war Was wird ?

- UML-Erweiterungsmechanismen
- UML 2.0: Definition und Notation von Profilen
- Zwei Profile im Detail
- Profile: Motivation & Diskussion - Historie & Ausblick
- Erstellen von Profilen
- Profile & Tools

Profile

Historische Entwicklung



Begriff des Profils in UML Erweiterungen für

-Enterprise Distributed Object Computing (EDOC) und

-Real-Time-Spezialisierungen eingeführt

Whitepapers, Anforderungen an Profile

Erste OMG Profile entstehen

Mittlerweile mehr als 50 offizielle und „freie“ Profile verfügbar; Toolunterstützung

Profile

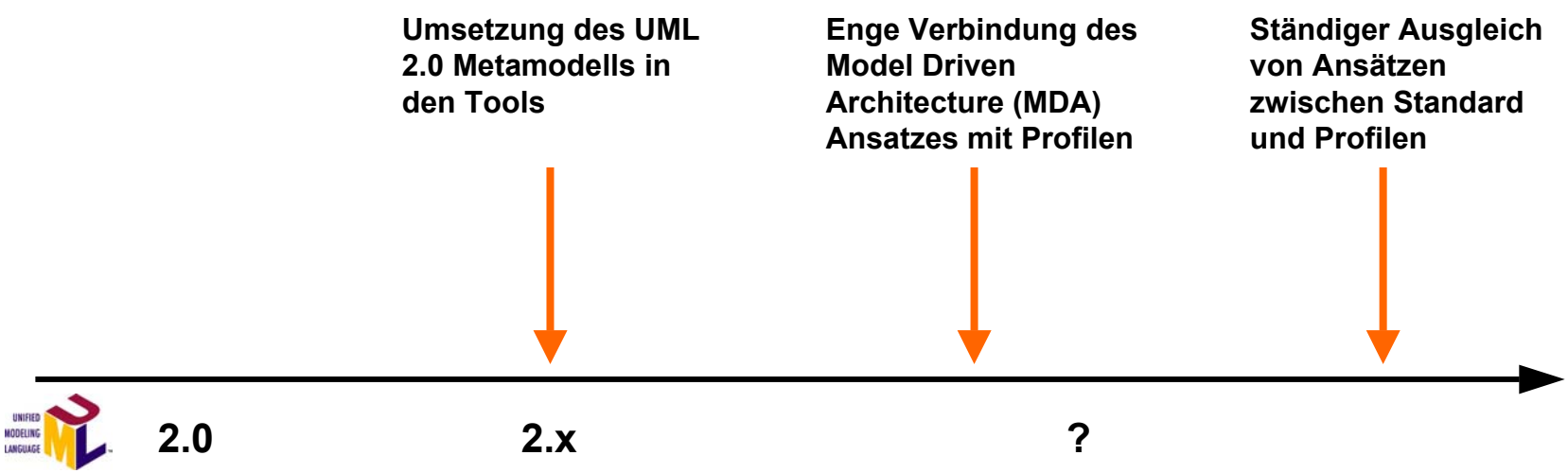
status quo am Markt

- UML Profile for Agent-Oriented Modeling
- UML Profile for Aspect-Oriented Modeling
- UML Profile for Real-time System Formal Design and Validation
- UML Profile for External Agent-Oriented Modeling
- UML Profile for Modeling Workflow and Business Processes
- UML Profile for QoS Management Information Specification
- UML Profile for Real-Time Constraints
- UML Profile for Reverse Engine
- UML Profile to Support Re with KAOS
- UML Profile to Suppo of Software
- UML Profile for the Web Language
- UML Profile for MultiTEL
- UML Profile for Framework Architectures
- UML Profile for Security Assessment
- UML Profile for Interaction Design
- UML Profile for Software Architecture Descriptions
- UML 2.0 Testing Profile
- UML Profile for Business Modeling
- UML Profile for Data Modeling
- UML Pro Enterprise Application Integration
- UM Real-Time Modeling
- stems Engineering
- Modeling
- CORBA Components
- ile for DCL
- ML Profile for Enterprise Distributed Object Computing
- UML Profile for Schedulability, Performance and Time
- UML Profile for Software Development based on the oose.de Method
- UML Profile for Enterprise Java Beans
- UML Profile for Knowledge Representation
- UML Profile for Visualizing Design Patterns
- UML Profile for Data Modeling
- UML Profile for Meta Object Facility
- UML Profile for Communication Systems
- ...

Und noch ein paar mehr ☺

Profile

Wo kann die Entwicklung hingehen?



Fertigstellung der in der Standardisierung (derzeit ca. 10) befindlichen Profile

Profilzahl wird mit der Toolunterstützung rapide wachsen ...

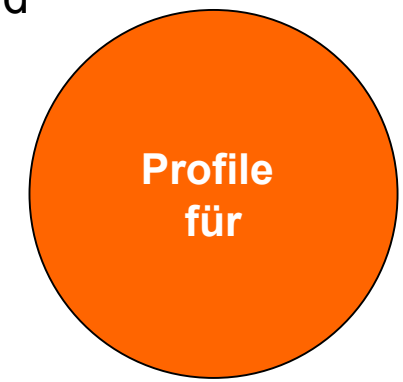
Reduktion der Profilzahl wegen Konsistenz und Kompatibilität nötig

Mit jeder größeren UML Metamodelländerung müssen bestehende Profile angepasst werden

Profile

Denkbare Ausprägungen

- Für spezielle Technologien und Implementierungssprachen
- Für spezielle Methoden und Vorgehensmodelle



- Für spezielle Systemtypen
- Für spezielle Anwendungsbereiche

Profile

Ausprägung: Technologien und Implementierungssprachen

- UML Profile für CORBA, EJB, .NET, ...
- UML Profile für C++, C, Java, ...
- UML Profile für Datenbanken (relational, OO, ...)
- UML Profile für propietäre Lösungen (Oracle, Microsoft, SAP, ...)
- ...

- Beispiele:
  - C++: Umgang mit friend, pure virtual methods, ...
  - Java: statischer Konstruktor, nur Konstanten in Interfaces, keine Mehrfachvererbung, ...
  - Datenbanken: Kennzeichnung von Primärschlüsseln, „Nicht-Null-Werten“, stored procedures

Profile

Ausprägung: Systemtypen

- UML Profile für Real-Time-Systeme
- UML Profile für n-tier Systeme
- UML Profile für verteilte Systeme
- UML Profile für Embedded-Systeme
- UML Profile für Workflow-Systeme
- UML Profile für Hardware-Systeme
- UML Profile für Software-Systeme
- UML Profile für „Real-World“-Systeme
- ...

Profile

Ausprägung: Anwendungsbereiche

- UML Profile für das Bank- und Versicherungswesen
- UML Profile für den Telko-Bereich
- UML Profile für den Automobilbereich (Steuergeräte, Navisysteme, ...)
- UML Profile für den Home-Entertainment-Bereich
- UML Profile für die Raum- und Luftfahrtanwendungen
- UML Profile für das prozessverarbeitende Gewerbe
- ...

Profile

Ausprägung: Spezielle Methoden und Vorgehensmodelle

- UML Profile für die Modelle in verschiedenen Phasen eines SW-Entwicklungsprozesses (Analyse, Design)
- UML Profile für die Unterstützung der Model Driven Architecture
- UML Profile für die Codegenerierung
- UML Profile zur Abbildung von Vorgehensmodellen (RUP, V-Modell, ...)
- ...

Profile

Vorteile

- Modelle und Profile können per XMI in Standard UML Tools ausgetauscht werden
- Tools müssen keine Funktionalität zur Veränderung des Metamodells anbieten
- Basis-UML-Notation kann genutzt werden!
- Durch Beschränkung der Metamodellanpassung auf die Profile ist die Wiederverwendung höher, da das Metamodell unangetastet bleibt!
- Profil-Anwendung ist einfacher als ein Modell-Merge!
- Viele UML Basiskonzepte bleiben unangetastet und müssen nicht neu erlernt werden!

Profile

Anwendung

- Erst prüfen, ob Basis-UML-Mittel ausreichen (Toolunabhängigkeit!)
- Profile sind kein Ersatz für Bibliotheken oder Frameworks
- Achten Sie auf Kompatibilität zwischen den Profilen
- Nutzen Sie Profilkombinationen (z. B.: Real-Time, C++, verteilt)
- Verlagern Sie keine fachlichen Informationen der Anwenderebene in Profile!

Lesbarkeit und Verständlichkeit Ihrer Modelle sind indirekt proportional zur Anzahl der angewendeten Profile!

Was war Was wird ?

- UML-Erweiterungsmechanismen
- UML 2.0: Definition und Notation von Profilen
- Zwei Profile im Detail
- Profile: Motivation & Diskussion - Historie & Ausblick
- Erstellen von Profilen
- Profile & Tools

Erstellen von Profilen

Regeln

- Ein UML Profil darf den Regeln und Bedingungen des UML Metamodells nicht widersprechen.
- Ein UML Profil darf nur die Standarderweiterungsmechanismen der UML (Stereotypen, Tagged Values und Constraints) nutzen.
- Ein UML Profil muss ein austauschbares UML-Modell sein:
  - Organisation der Elemente in Pakete
  - Vollständige Definition von Profilabhängigkeiten mit Namensräume und Abhängigkeitsbeziehungen
  - Weiteres definiert der UML-XMI Standard
- Ein UML Profil muss den verwendeten Subset des UML Metamodells definieren.

Erstellen von Profilen

Regeln II

- Ein UML Profil soll bestehende, relevante Elemente auflisten (wie z. B. Bibliotheken, Typdefinitionen, bestehende Modelle, ...).
- Ein UML Profil soll spezielle Operationen unterstützen:
  - Profil-Spezialisierung: Verfeinerung eines Profiles in ein oder mehrere Subprofile (Konsistenzsicherung!)
  - Profil-Vereinigung: Verknüpfung von kompatiblen Profilen zu einem neuen Profil
- Tagged Values sollten formal definiert werden (Typ, Wert, Beschreibung, evtl. mathematische Operationen, ...).
- Ein UML Profil muss mit einer Bibliothek zu einer logischen Einheit zusammenfassbar sein.

Was war Was wird ?

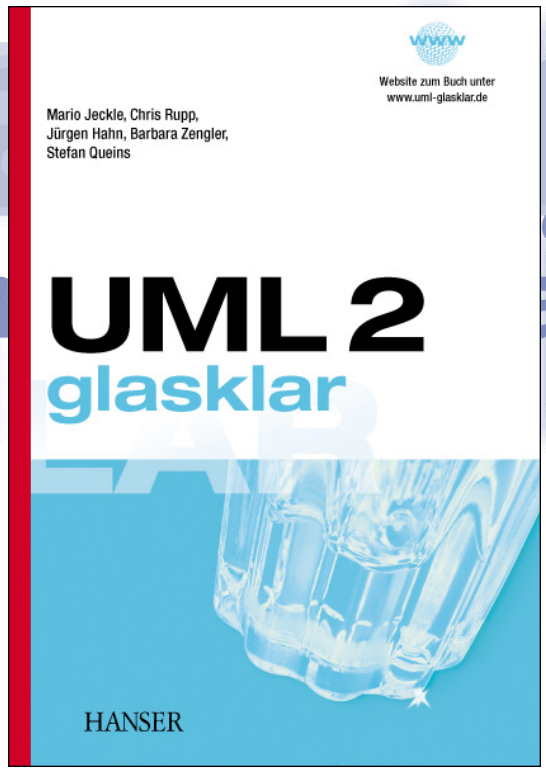
- UML-Erweiterungsmechanismen
- UML 2.0: Definition und Notation von Profilen
- Zwei Profile im Detail
- Profile: Motivation & Diskussion - Historie & Ausblick
- Erstellen von Profilen
- Profile & Tools

Profile & Tools

Anforderungen

- Profildefinition (Definition der Erweiterungsmechanismen, des Profils und seiner Abhängigkeiten)
- Profilimport und Profilexport mittels XMI
- Profilanwendung (Die Anwendung muss nachvollziehbar sein!)
- Profilentfernung (Ohne Verlust des Modells!)
- Filterfunktion (Ausblenden von Nicht-Profilelementen)
- Testfunktion (Prüfen, ob ein Modellelement den Regeln eines Profils folgt)
- Modellintegration (Verbinden von Modellen, die zu unterschiedlichen Profilen gehören)
- Profilintegration (Verbinden unterschiedlicher Profile)

Damit Sie klar sehen!



Diesen Vortrag, weitere Informationen zu Profilen und zur UML 2.0 unter

www.uml-glasklar.de
www.sophist.de