TECHNISCHE UNIVERSITÄT ILMENAU

Fakultät für Informatik und Automatisierung
Fachgebiet Softwaresysteme/Prozessinformatik
Betreuer: Dr.-Ing. Detlef Streitferdt

Komponentenentwurf mit der UML 2.0

Bearbeiter: Andreas Bielert

Gliederung

1. Einleitung
2. Die Modellierungssprache UML
3. komponentenbasierte Softwareentwicklung
4. Komponenten in der UML 2.0
5. Softwareunterstützung
6. Zusammenfassung

1. Einleitung

Softwareentwicklung früher:

- wenige Entwickler, arbeiten alle am selben Ort
- jeder Entwickler kennt das gesamte Programm
- große, monolithische Programmstrukturen
- geringe analytische Vorarbeit

Dadurch:

- Struktur der Programme wird nur noch von wenigen Spezialisten verstanden
- keine Wiederverwendbarkeit
- Pflege und Bug-Fixing sehr schwierig
- trotz geringerer Komplexität erhöhter Programmieraufwand

1. Einleitung

Softwareentwicklung heute:

- viele Entwickler, geographisch oft getrennt
- Entwicklung erfolgt in Klassen und Komponenten
- jeder Entwickler kennt nur den für ihn relevanten Teil des Programmes
- verbesserte analytische Vorarbeit, teils computerunterstützt

Dadurch:

- höhere Wiederverwendbarkeit
- bessere Pflege, leichteres Bug-Fixing
- relativ verständliche Programmstrukturen

Aber:
Modellierung von Komponentenstrukturen bisher nur bedingt möglich.

1. Einleitung

Deshalb:

Entwicklung einer neuen UML Spezifikation
-> UML 2.0

Ziel dieser Arbeit:

Wie werden Komponentenstrukturen in UML 2.0 realisiert?

Wie weit ist die Softwareunterstützung für diese Komponenten bisher realisiert?

2. Die Modellierungssprache UML

Problem:

- Softwareingenieure müssen sich untereinander verständigen
- Softwareingenieure müssen sich mit anderen verständigen

Unified Modelling Language (UML):

- Festlegung auf eine gemeinsame Menge von Sprachkonstrukten und Diagrammstrukturen
- dadurch programmiertechnischer Hintergrund egal
- auch „Nichtprogrammierern" können Programmstrukturen erklärt werden

2. Die Modellierungssprache UML

Entwickelt von den *„drei Amigos"*:

Grady Booch

OOAD - Object-Oriented Analysis and Design
Design und Implementierung

Ivar Jacobson

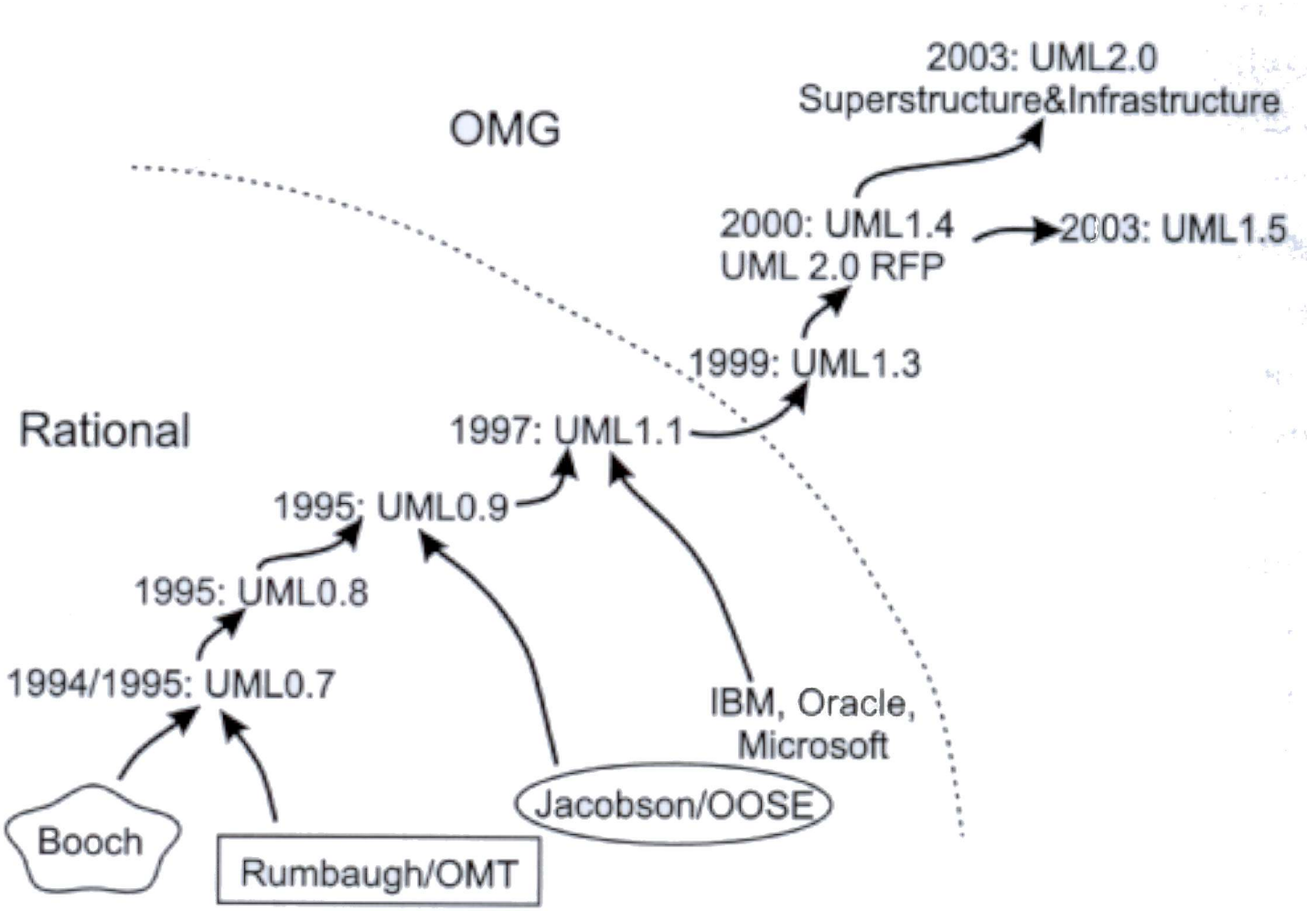
OOSE - Object Oriented Software Engineering
Anwendungsfälle und Klassen

James Rumbaugh

OMT - Object Modelling Technique
dynamisches Verhalten eines Systems

2. Die Modellierungssprache UML

Entstehung der UML:



Bildquelle: „Softwareentwicklung mit UML 2“ Marc Born, Eckhard Holz, Olaf Kath; Addison-Wesley 2004

2. Die Modellierungssprache UML

UML 2.0

- basiert auf Vorschlägen der U2 Partners
  Alcatel, Ericsson, HP, Motorola, Oracle und andere
- immer noch nicht vollständig spezifiziert und abgesegnet

- 13 Diagrammarten:
  - 6 für Modellierung der Systemstruktur
  - 4 Interaktionsdiagramme
  - 3 Verhaltensdiagramme

- viele Veränderungen gegenüber Version 1.4
- Einführung des Petri-Netz basierten Aktivitätsdiagrammes
- Beseitigung von Redundanzen
- Einführung neuer Programmkonzepte wie Komponenten und Ports
- 4 vollkommen neue Diagrammarten

3. komponentenbasierte Softwareentwicklung

Vorteile:

- Aufbau einer Bibliothek von Problemlösungen
- Vermeidung von Doppelt- und Dreifachlösungen
- Einkauf bereits fertiger Komponenten
- einfache Integration externer Komponenten
- Verteilte Programmentwicklung
- Verbesserte Testbarkeit
- Verbessertes Bugfixing
- einfache Erweiterbarkeit (bei Beibehaltung der Schnittstellen)

3. komponentenbasierte Softwareentwicklung

Nachteile:

- hohe Anforderungen an den Programmierer
- gesteigerte Komplexität der Programme
- Vorhandene Komponenten müssen gepflegt, überwacht und katalogisiert werden
- lange Einarbeitungszeit
- gesteigerter Arbeitsaufwand
- „Abhängigkeitsverhältnis“ bei Benutzung von Black-Box Komponenten

4. Komponenten in der UML 2.0

Das Komponentendiagramm

Beantwortet folgende Fragen:
- Wie ist mein System strukturiert?
- Wie werden diese Strukturen erzeugt?
- In welche funktionalen Bestandteile kann ich mein System aufteilen?
- Wie ist die innere Struktur meiner Komponenten? (optional)
- Welche Wechselwirkungen treten zwischen meinen Komponenten auf?

Das Komponentendiagramm wurde in der Version 2.0 neu in die UML eingeführt

4. Komponenten in der UML 2.0

Komponenten:

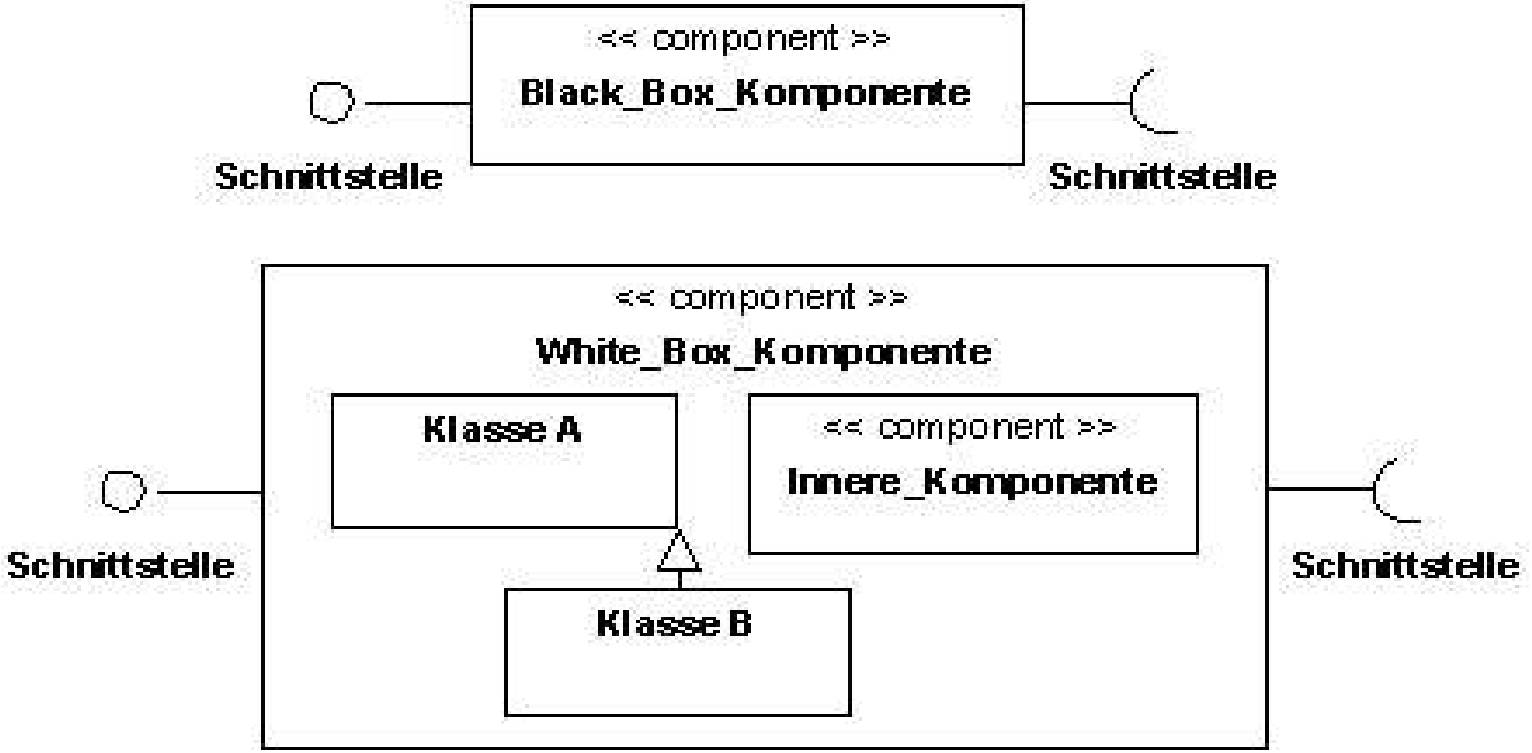
- wichtigstes Modellierungsobjekt im Komponentendiagramm
- von Klassen abgeleitet (*Classifiers*)
- kann Klassen, Objekte, Komponenten oder Subsysteme enthalten
- Darstellung in allgemeiner und in Listendarstellung möglich
- Black-Box und White-Box Komponenten

Black-Box Komponente:
nur Schnittstellen der Komponente sind sichtbar

White-Box Komponente:
das Innere der Komponente ist vollständig modelliert

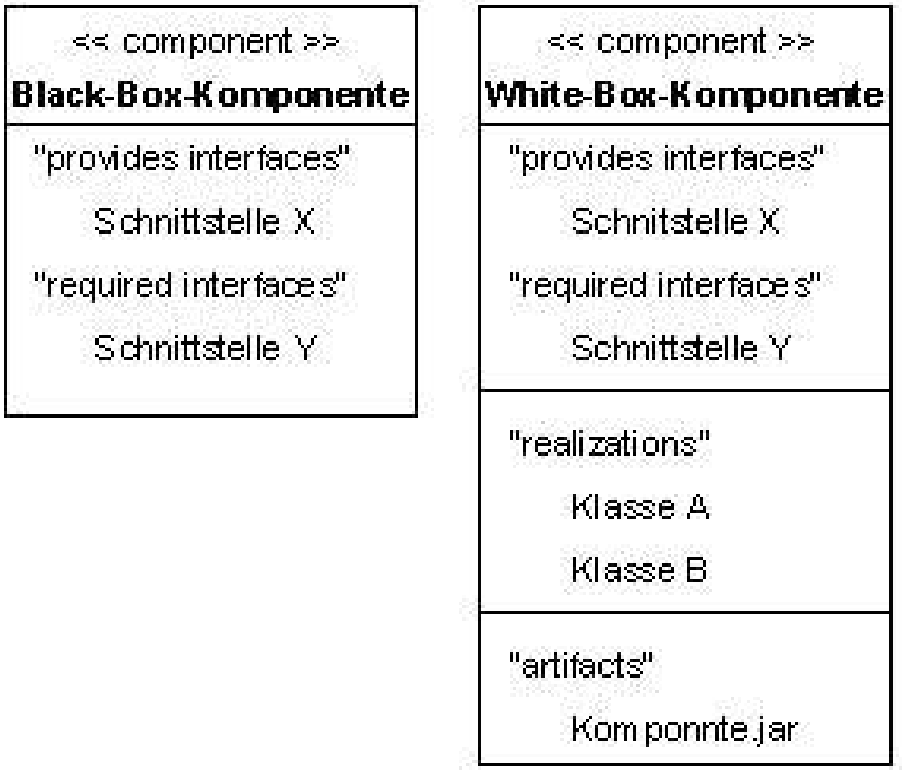
4. Komponenten in der UML 2.0

Komponentendarstellung
– allgemeine Darstellungsform -



4. Komponenten in der UML 2.0

Komponentendarstellung – Listendarstellung -



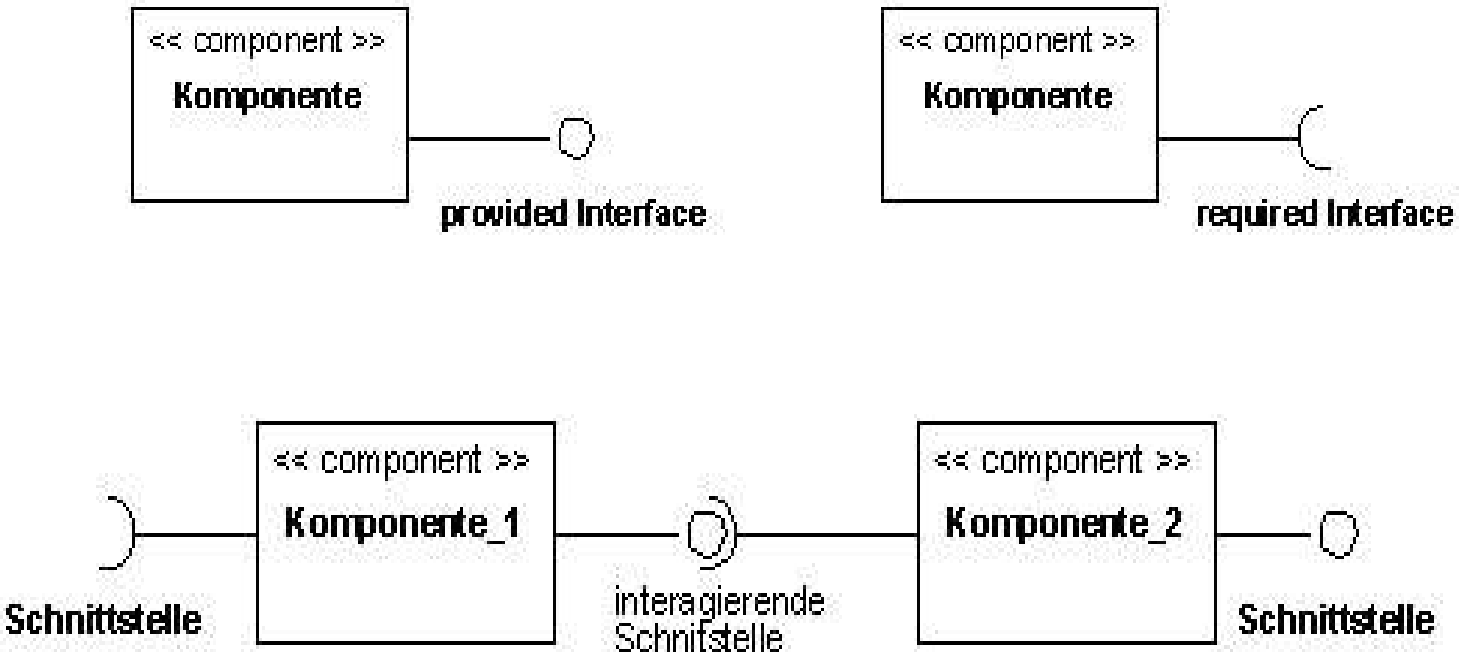
4. Komponenten in der UML 2.0

Schnittstellen:

- bilden die Interaktionspunkte meines Programmes
- können nicht für sich allein existieren,sondern immer nur zusammen mit ihrer Komponentenausprägung
- in UML 2.0 zwei verschiedene Arten von Schnittstellen: bereitgestellte (provided) und angeforderte (required)
- können auch ererbt werden
- jede Schnittstelle steht für sich alleine und ist unidirektional

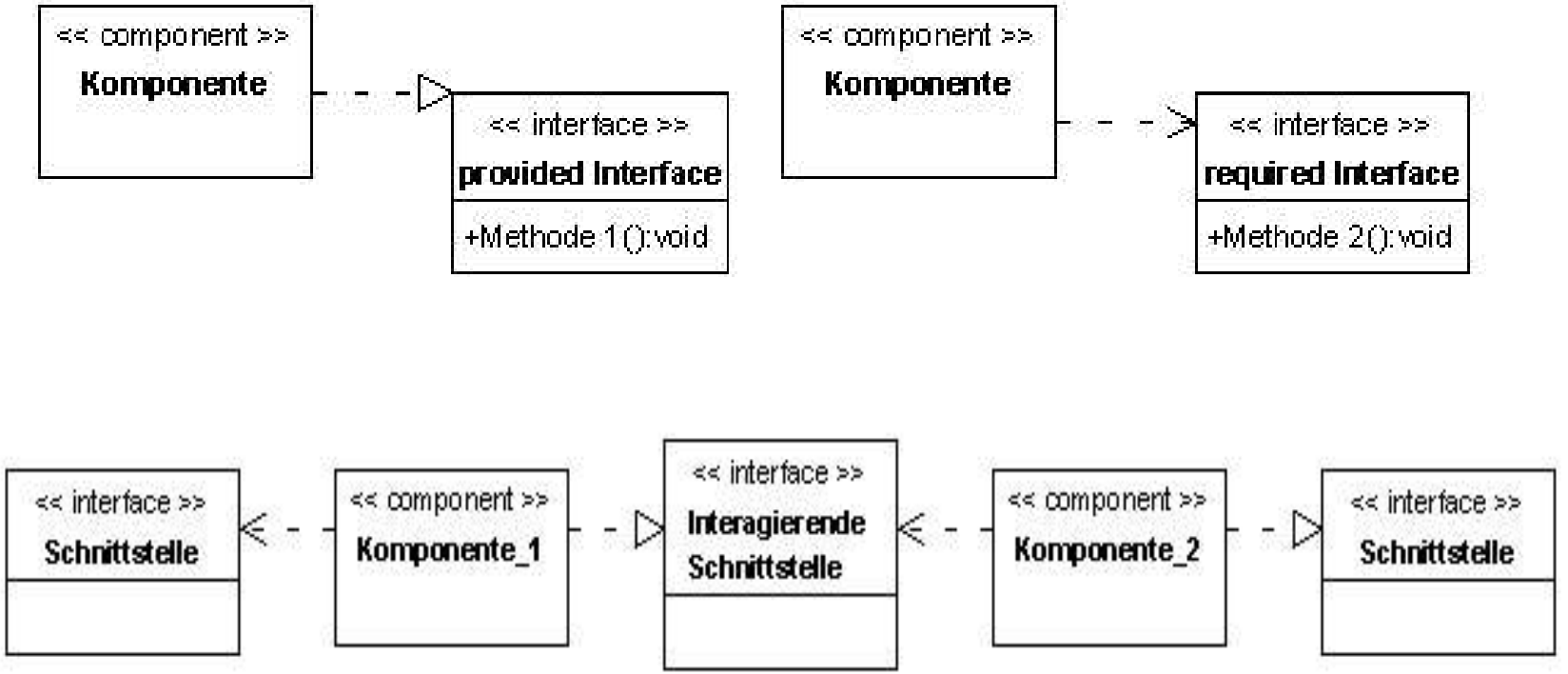
4. Komponenten in der UML 2.0

Schnittstellen
– Ball/Socket Notation -



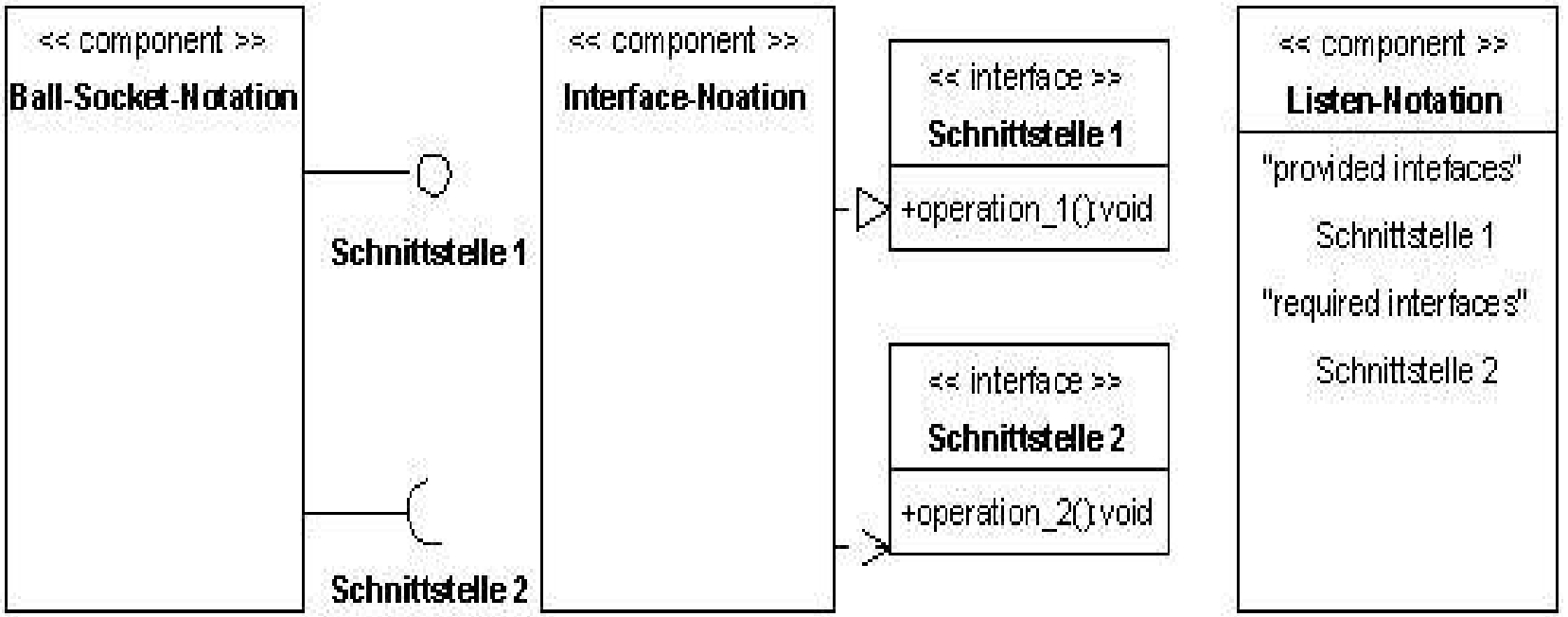
4. Komponenten in der UML 2.0

Schnittstellen – Interfacedarstellung -



4. Komponenten in der UML 2.0

-> 3 verschiedene Darstellungsformen für Schnittstellen



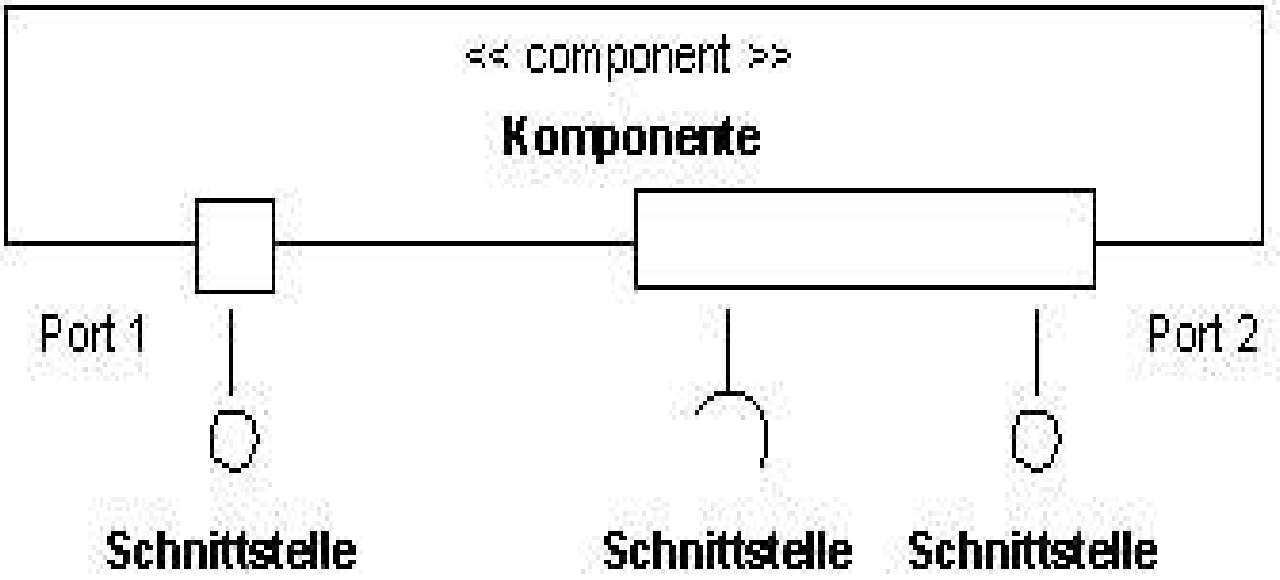
4. Komponenten in der UML 2.0

Ports:

- dienen zum Zusammenfassen von Schnittstellen
- erlauben logische bzw. funktionale Gruppierungen von Schnittstellen
- ein Port kann über mehrere Schnittstellen verfügen -> bidirektional
- werden oft benutzt um die Kommunikation einer größeren Komponente (wie z.B. ein Subsystem) mit der Außenwelt zu modellieren
- Classifier kann beliebig viele Ports besitzen

4. Komponenten in der UML 2.0

- Ports -



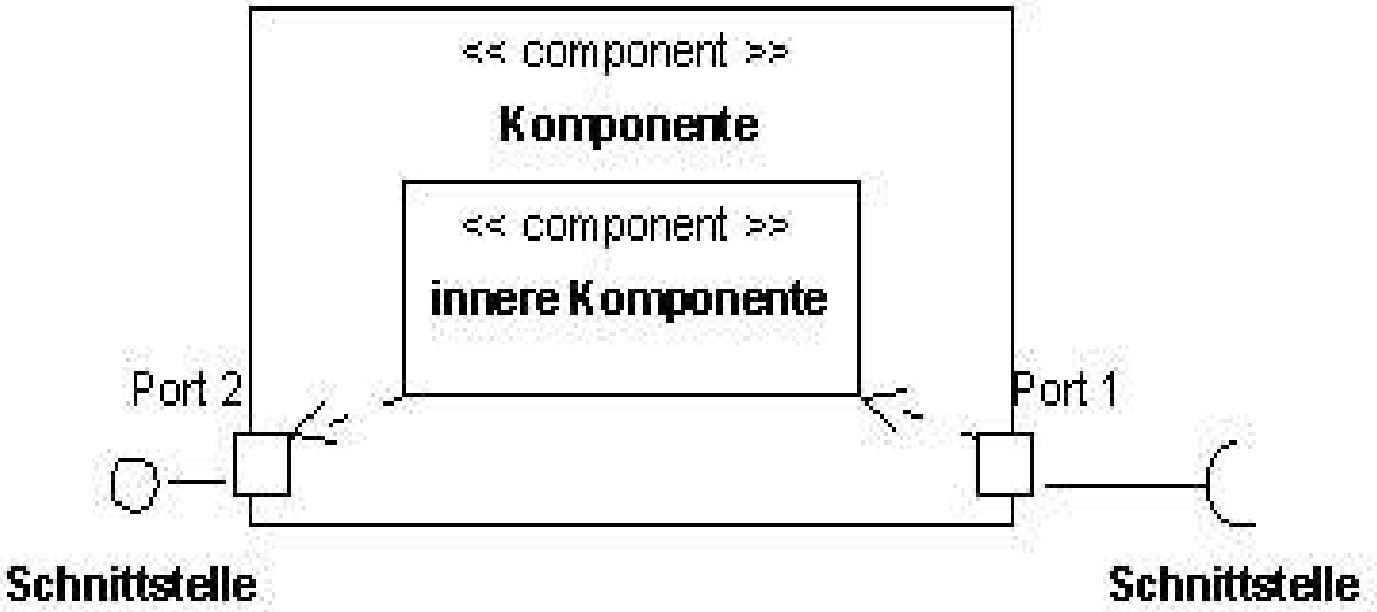
4. Komponenten in der UML 2.0

Delegierende Konnektoren:

- stellen Verbindung zwischen Port / Schnittstelle und dem eigentlichen Element her
- haben nur Informationsweiterleitende Funktion

4. Komponenten in der UML 2.0

- Delegierende Konnektoren -



4. Komponenten in der UML 2.0

- Artefakte -

- Komponente ist nur Abstraktion
- deshalb: Manifestierung in Artefakt

4. Komponenten in der UML 2.0

UML-Profile

- UML von ihrer Idee her eigentlich implementierungs-unabhängige Modellsprache
- wird gelegentlich aber gewünscht die Sprache an die eigenen Bedürfnisse anzupassen

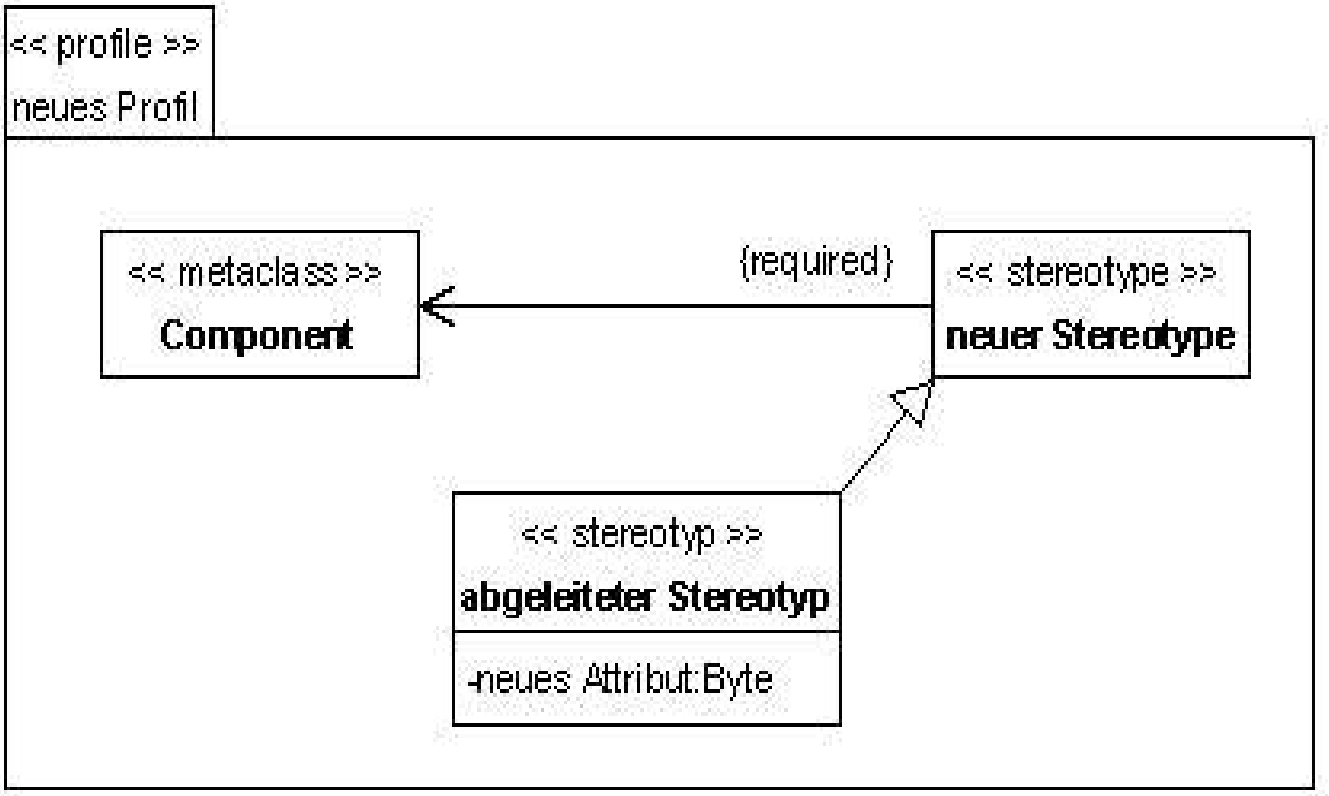
mit Profilen: eigenes Konstrukt als UML Stereotyp definieren

- erlaubt es dem Modellierer sogar bestimmte Konstrukte vollständig durch eigene, erweiterte zu ersetzen
- wird vor allem bei Arbeit mit Frameworks wie .NET oder J2EE benutzt

Aber: keine Änderung am UML 2.0 Meta-Modell möglich!

4. Komponenten in der UML 2.0

- Profile -



4. Komponenten in der UML 2.0

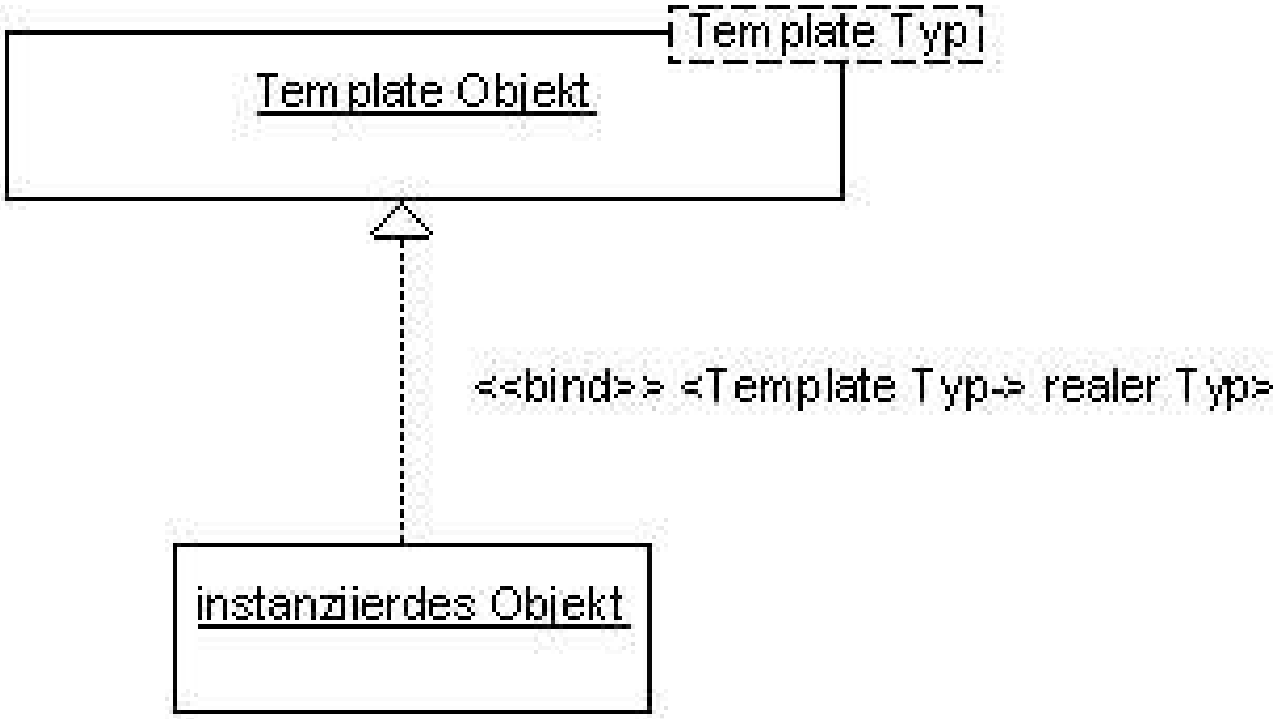
Templates:

- bestimmte Funktionsabläufe wiederholen sich ständig
- aber manchmal Datentypen auf denen gearbeitet nicht klar definiert oder ändert sich während der Abarbeitung
- oder Modell soll möglichst offen gehalten werden

- bilden Spezifikationslücke im Modell
- Spezifikation wird erst zum Zeitpunkt des Bindungsprozesses ausgefüllt
- Template Wert wird konkreter Typ übergeben um korrekte Abarbeitung zu gewährleisten
- durch Parameter konkrete Ausgestaltung eines Classifiers beeinflussen

4. Komponenten in der UML 2.0

- Templates -



5. Softwareunterstützung

- UML Spezifikation immer noch sehr neu
- immer noch nicht vollständig Spezifiziert
- Verbreitung UML 1.4 sehr groß

deshalb:

- Anzahl der Programme mit 2.0 Unterstützung immer noch recht überschaubar
- viele Programme realisieren 2.0 Spezifikation nur zum Teil

->Überprüfung der Programme durch eigene Komponente

5. Softwareunterstützung

Spezifikation:

- Komponente „Student“
- 5 Schnittstellen, davon eine Dienst anbietend
- Schnittstellen sind auf 2 Ports aufgeteilt
- im Inneren eine weitere Komponente „Gesundheit“ und eine Klasse „Wissen“
- „Gesundheit“ und „Wissen“ agieren über Schnittstelle miteinander
- Ports sind über Konnektoren mit den Classifiern verbunden
- „Gesundheit“ besitzt verschieden Methoden und ein Attribut Namens „Wissensstand“
- „Student“ wird über Artefakt „Student X“ manifestiert
- Template Klasse „Getränk“
- interagiert mit „Student“ über Schnittstelle

5. Softwareunterstützung

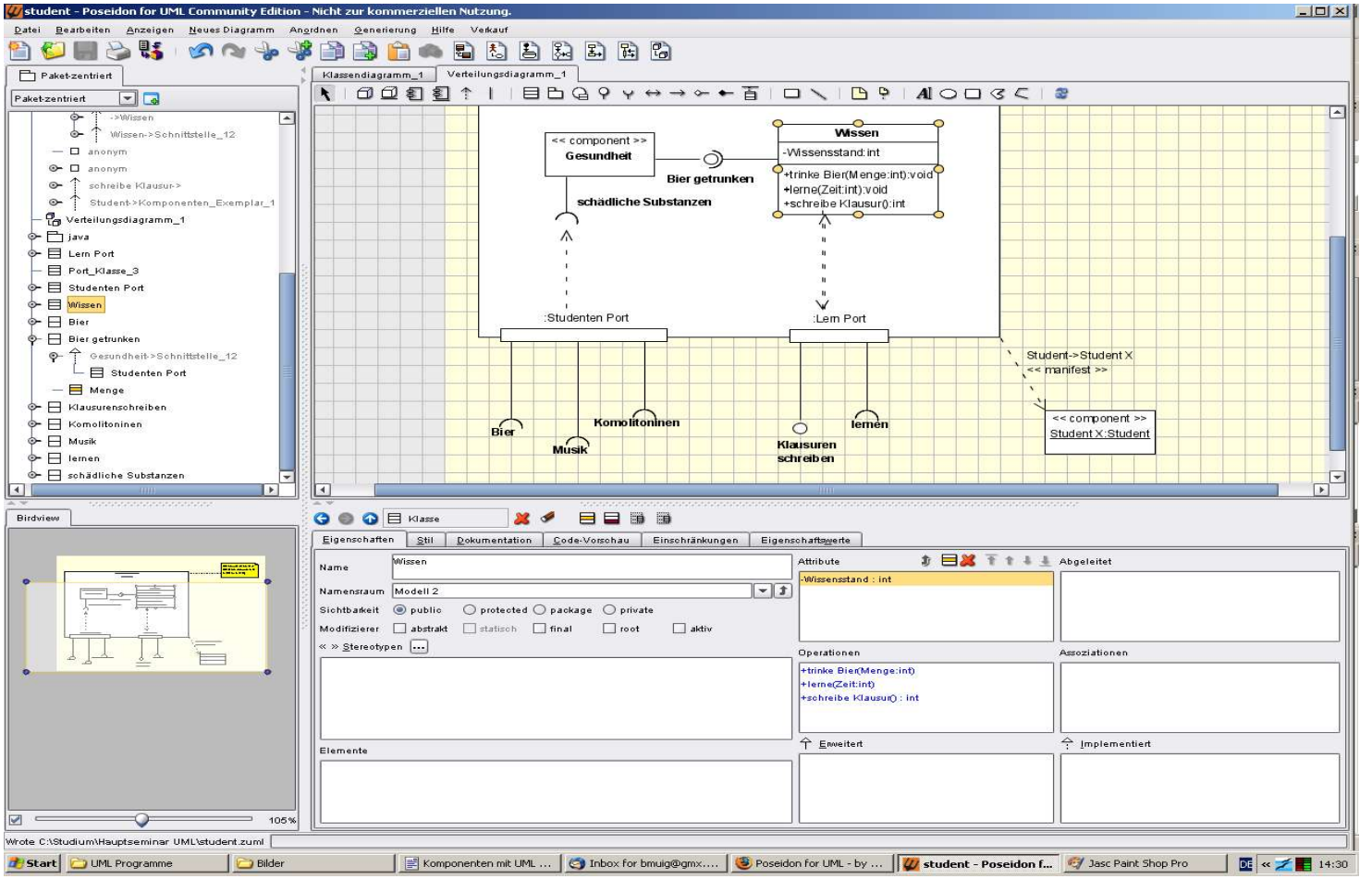
Poseidon for UML 3.1

- entwickelt von Gentleware
- Windows, Linux, Mac OS
- „Community Edition“ beliebig lange kostenlos
- aber diverse Einschränkungen
- „Professional Edition“ für 875$

http://www.gentleware.com

5. Softwareunterstützung

Poseidon for UML 3.1

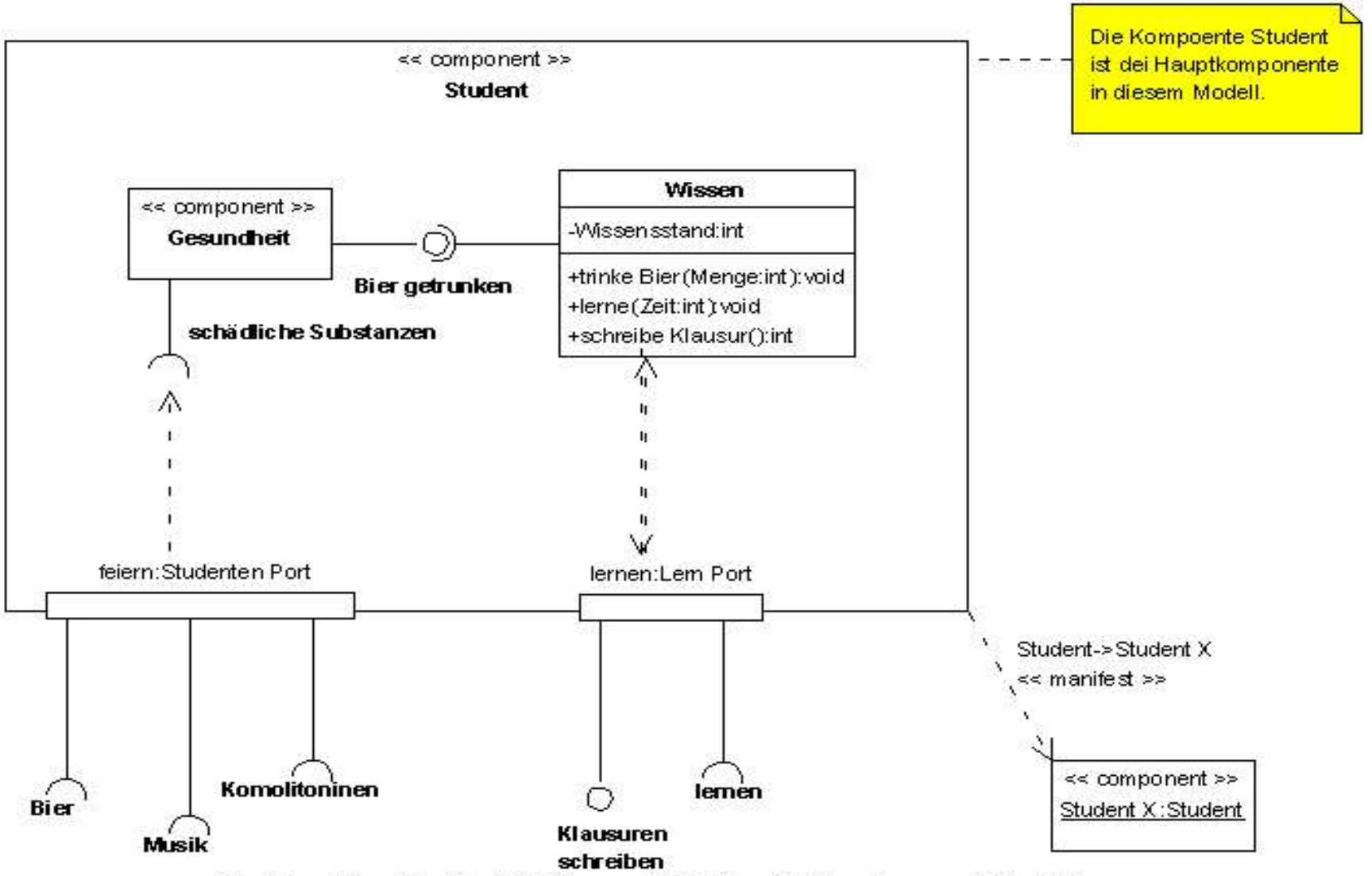


5. Softwareunterstützung

Poseidon for UML 3.1

- einfachen Rechteckdarstellung
- keine Listendarstellung
- Schnittstellen und Ports
- Ball/Socket und allgemeine Interfacenotation
- komfortables Eigenschaftenmenü
- keine Verzicht auf Typen möglich
- Artefakte werden als Komponenten bezeichnet
- teilweise eigene Stereotypen
- Profile nur als Modellierungselement
- keine Templates
- Codegenerierung nur in Java möglich

5. Softwareunterstützung



5. Softwareunterstützung

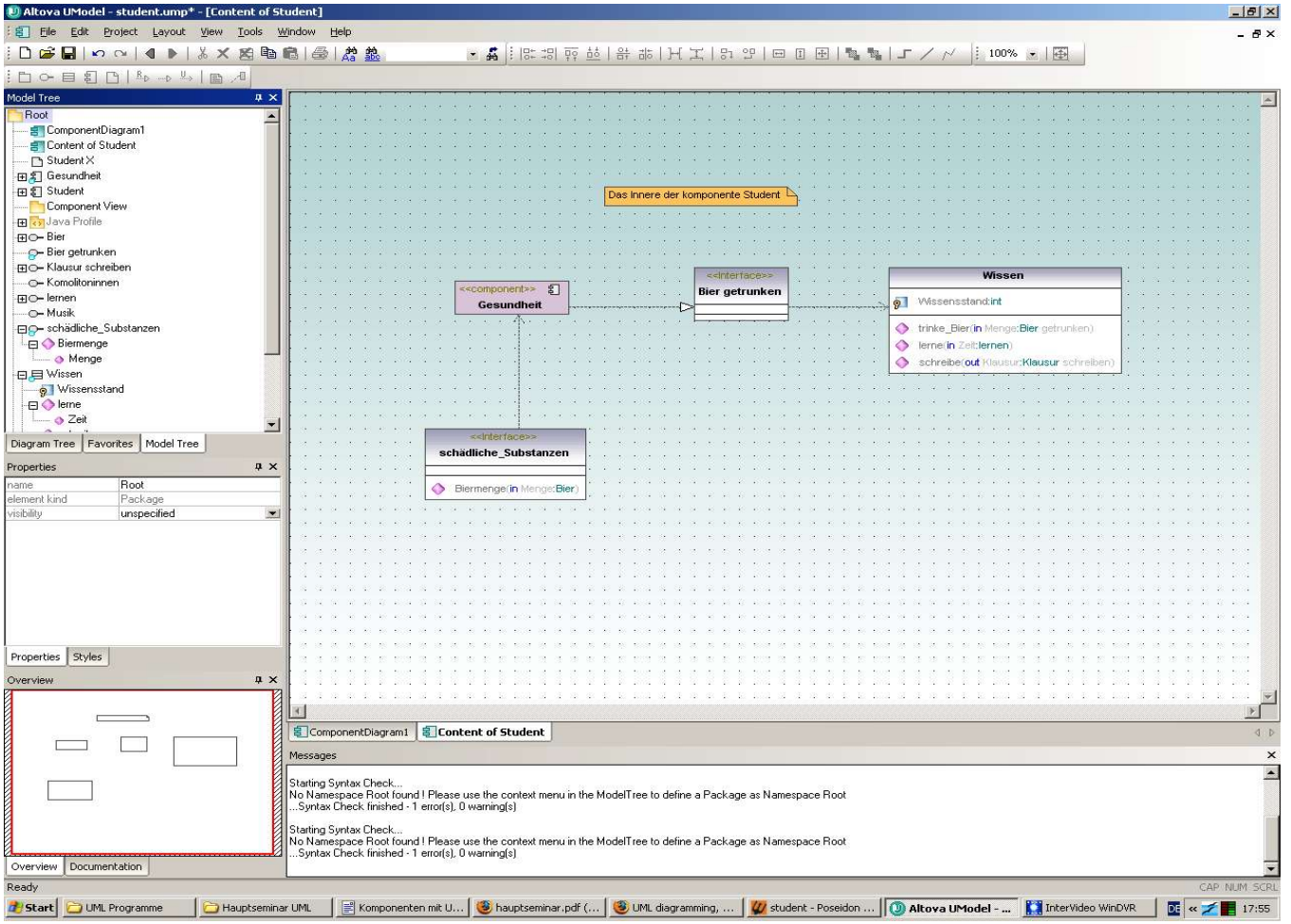
Umodel 2005

- von Firma Altova
- nur für Windows
- 30 Tage Evaluierungsversion
- dafür aber Registrierung nötig
- Evaluierungsversion ohne größere Einschränkungen
- Vollversion (Single License) 123,75€

http://www.altova.com

5. Softwareunterstützung

Umodel 2005

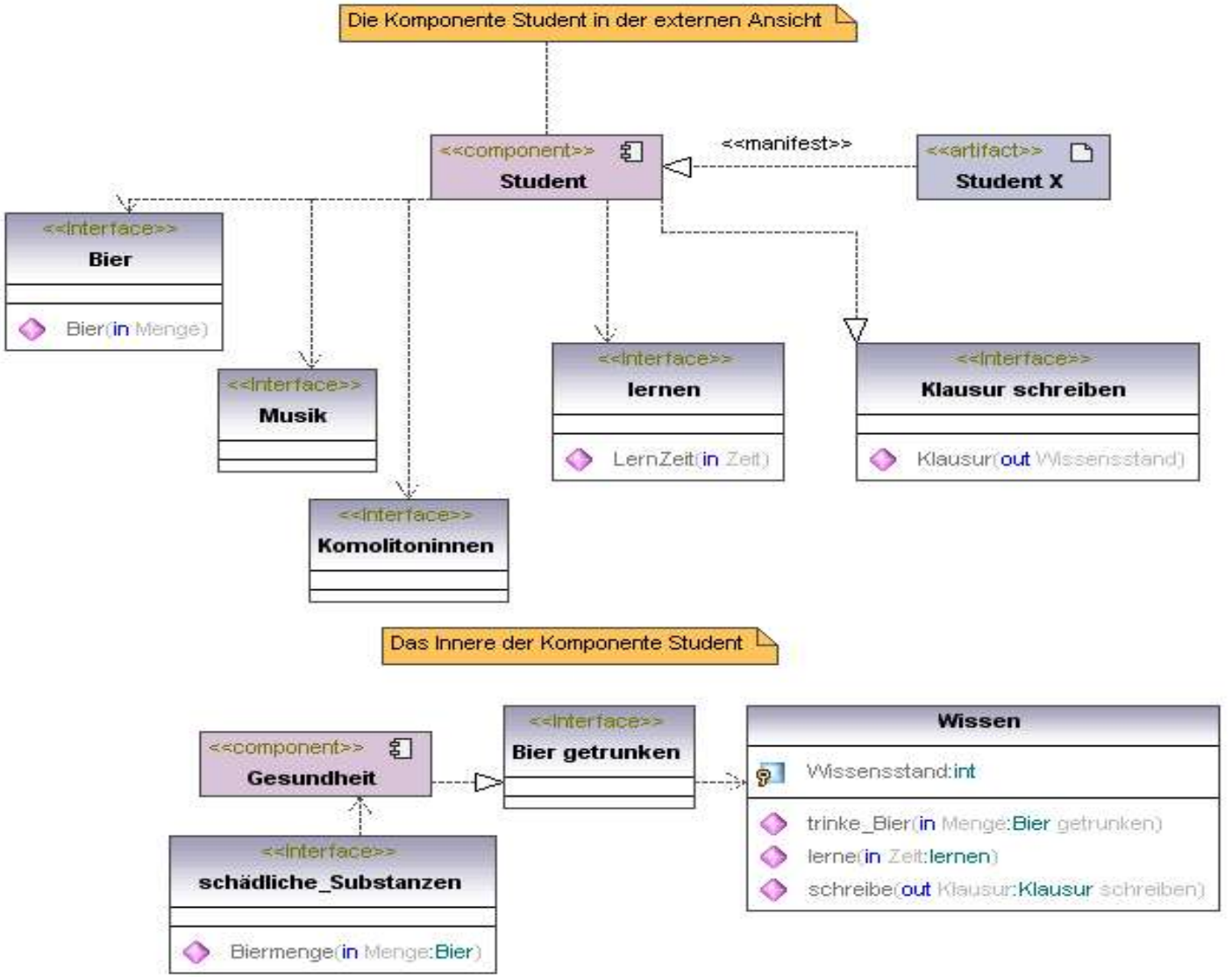


5. Softwareunterstützung

Umodel 2005

- komplizierte Handhabung
- unübersichtliche Darstellung
- Komponenten im eigenen Fenster
- dadurch keine White-Box Darstellung
- Darstellung mit graphischen Stereotypen
- keine Attribute und Funktionen an Komponenten
- Schnittstellen nur in Interfacenotation
- keine Ports
- kein Ball/Socket
- keine delegierenden Konnektoren
- eigene Typendefinitionen
- keine Templatedarstellung
- Profile nur als Darstellungsselemente

5. Softwareunterstützung



5. Softwareunterstützung

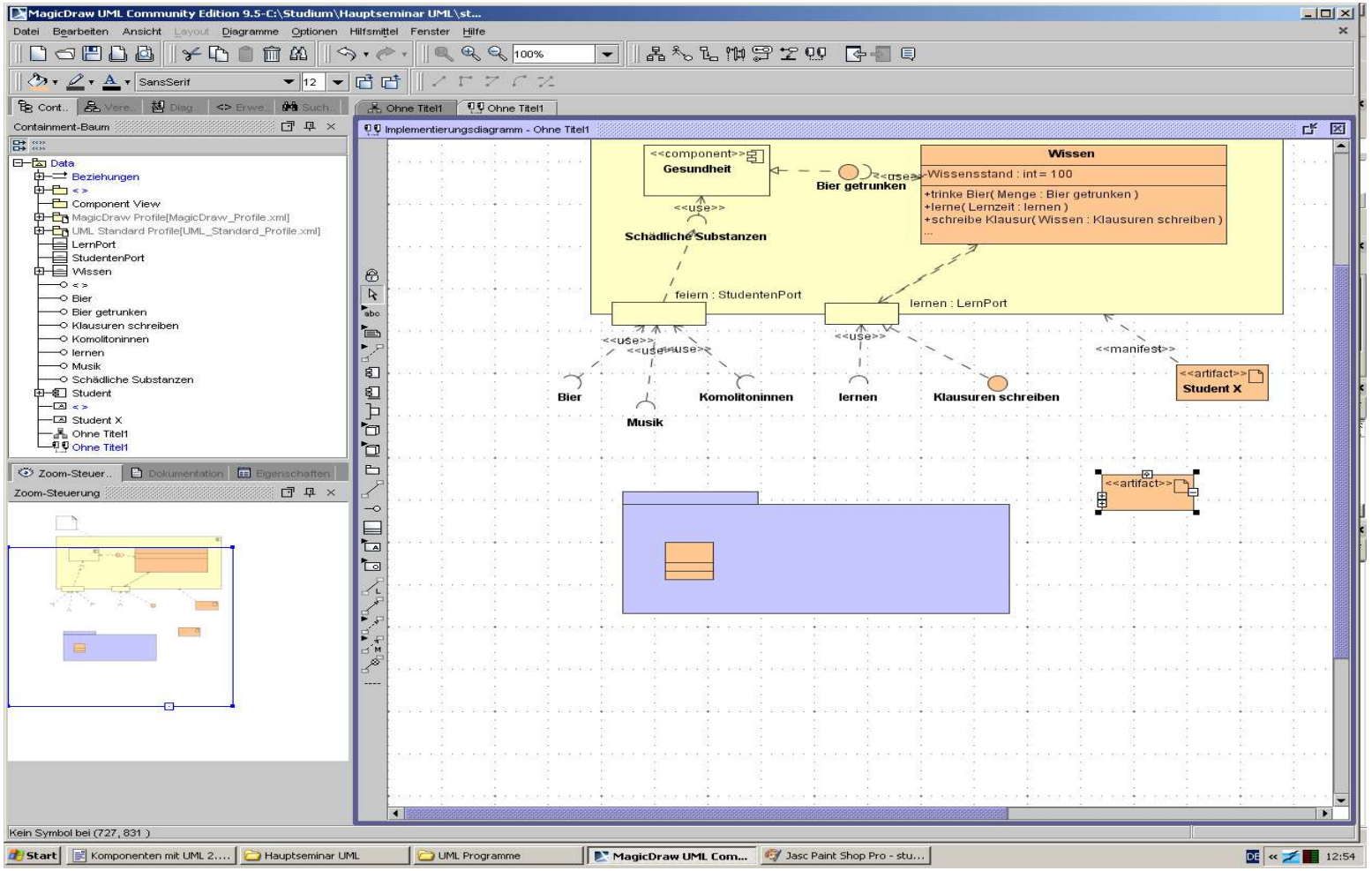
MagicDraw UML 9.5

- von Firma MagicDraw
- Versionen für Windows, Linux/Unix und Mac OS
- kostenlose Community Edition
- unbegrenzt Lauffähig
- aber Registrierung auf Firmenwebseite
- Registrierung des heruntergeladenen Programms
- Professional Edition 899$

http://www.magicdraw.com

5. Softwareunterstützung

MagicDraw UML 9.5

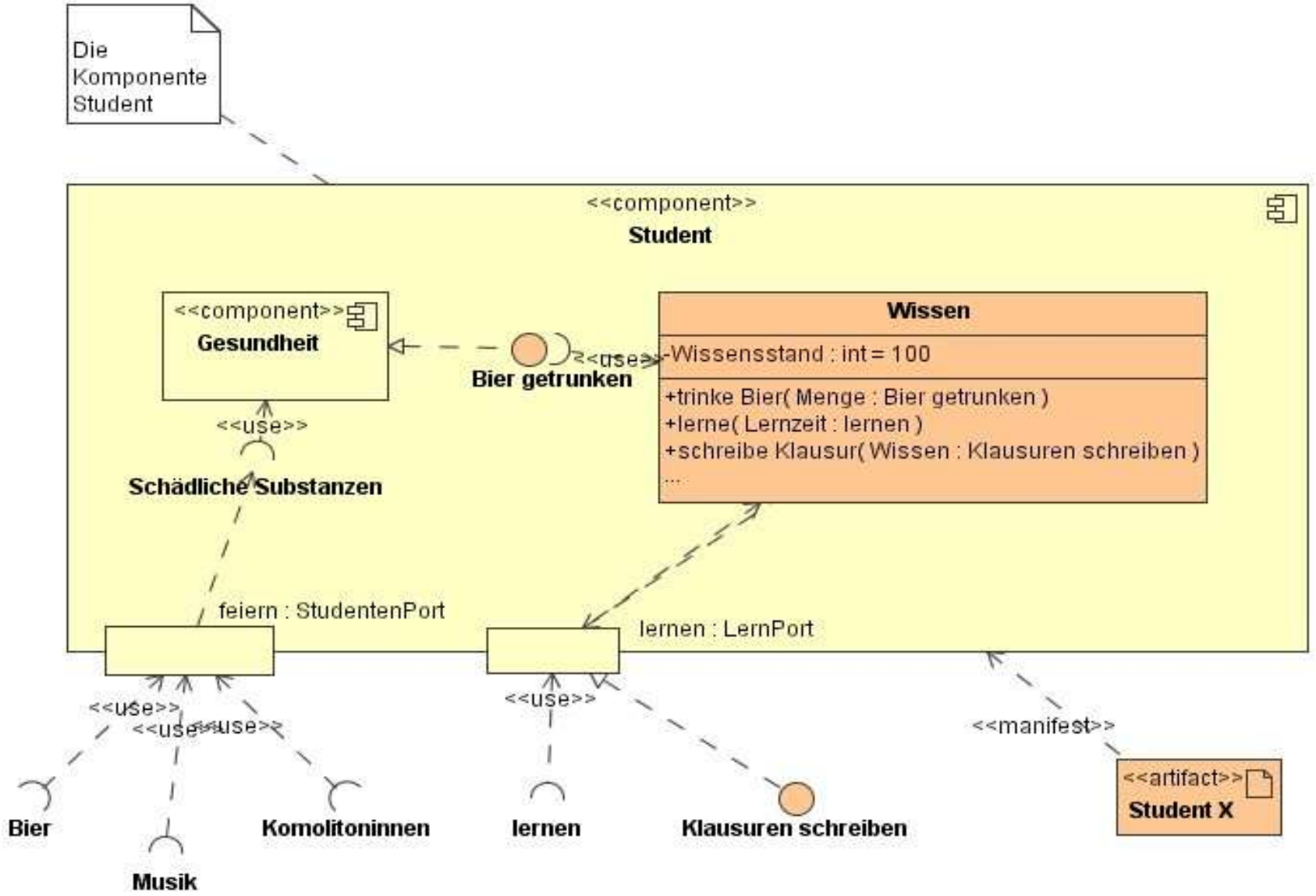


5. Softwareunterstützung

MagicDraw UML 9.5

- auf das Malen von Komponenten ausgelegt
- keine Codegenerierung
- Auswahl zwischen UML 1.4 und 2.0 Notation
- Listendarstellung
- eigene Parametertypen
- Funktionseinstellungen in Pop-Up Menüs
- ungewöhnliche Interfacedarstellung
- Ball/Socket und Interface
- kein „return“ Parameter
- keine Templates
- Unterstützung von Profilen
- Speicherung in XML-Code

5. Softwareunterstützung



5. Softwareunterstützung

Fazit:

- unterschiedliche Modellansätze -> unterschiedliche Modelle
- UML 2.0 nur teilweise
- Poseidon und MagicDraw gute Handhabung und Funktionalität
- Modelle bei Poseidon übersichtlicher
- MagicDraw näher an der UML Spezifikation
- Umodel abgeschlagen
- kein Programm kennt Templates
- Profile nur in MagicDraw unterstützt

5. Softwareunterstützung

Auswertung:

	Poseidon for UML 3.1	***Umodel 2005***	***MagicDraw 9.5***
Modellierung von Black-Box Komponenten	Ja	Ja	Ja
Modellierung von White-Box Komponenten	Ja	*nur in neuem Diagramm*	Ja
Listendarstellung von Komponenten	Nein	Nein	Ja
Attribute/Funktionen auf Komponenten	Ja	Nein	Nein
Interfaces in Ball/Socket Notation	Ja	Nein	Ja
Interfaces in allgemeiner Darstellung	Ja	Ja	Nein
Ports	Ja	Nein	Ja
Erstellung eigener Stereotypen	Ja	Ja	Ja
Erstellung eigener Typen	*Teilweise*	Ja	Ja
Verzicht auf Typenangaben möglich	Nein	Ja	Ja
Unterstützung von Profilen	*Teilweise*	*Teilweise*	Ja
Codegenerierung	Ja	Ja	Nein
Templates	Nein	Nein	*nur über Hilfskonstruktion*

Zusammenfassung

Zusammenfassung:

- UML 2.0 Schritt in die richtige Richtung
- Verbreitung immer noch gering
- aber auch immer noch in Abnahme
- bestimmt Konstrukte noch nicht vollständig spezifiziert
- allgemeine Verwendung erst nach Absegnung

- Programme decken nur Teile ab
- Modellierer muß flexibel sein
- großes Verbesserungspotential

4. Komponenten in der UML 2.0

Basic Components und Packaging Components:

- Spezifikation auf 2 Pakete aufgeteilt:

    1. Basic Componets:
        - Komponenten wie bisher betrachtet
        - abgeschlossenes, austauschbares Stück Software
        - durch Artefakt manifestiert
        - *geleistete Schreibarbeit des Programmierers*

4. Komponenten in der UML 2.0

Basic Components und Packaging Components:

2. Packaging Components
   - von Basic Component abgeleitet
   - muß nicht direkt manifestierbar bzw. ausführbar sein
   - kann durch Instantiierung der sie realisierenden Classifier erstellt werden
   - besteht nicht ausschließlich aus instantiierbaren Objekten
   - darf alle Arten von Modellelementen enthalten
   - erfasst logische Zusammenhänge
   - z.B. Übergabe von Design-Pattern
   - *nicht nur der reine, ausführbar Code, sondern auch die kreative Leistung des Modellierers*

3. komponentenbasierte Softwareentwicklung

Definition Komponente:

„Eine Komponente ist eine fachliche Einheit, mit einer intern aus Klassen oder wiederum Komponenten bestehenden Struktur. Nach außen stellt eine Komponente Funktionalität über Schnittstellen bereit oder fordert Funktionalität über Schnittstellen an. Ein wesentliches Merkmal für Komponenten ist ihre prinzipielle Austauschbarkeit zur Laufzeit."

„Analyse und Design mit UML 2" Bernd Oestreich
Oldenbourg Verlag München 2005