Ein Überblick über UML

1 Abriss der Geschichte von UML

1994 Booch und Rumbaugh beschließen, ihre Ansätze (OOAD und OMT) zu vereinigen mit dem Ziel, einen Industriestandard zu schaffen.

1995 Unified Method, Version 0.8. Im wesentlichen Vereinigungsmenge von OMT und OOAD. Grundnotation von OMT, mit Details und Methodenelementen von OOAD.

Jacobson stößt dazu und bringt OOSE ein. Das Prinzip des Vereinigens stößt an seine Grenzen. Radikale Vereinfachung (aber auch semantische Verarmung); alle Spezialkonstrukte werden mit Hilfe der neu eingeführten Stereotypen modelliert. Vorläufiger Verzicht auf Einheitsmethode, nur noch Einheitssprache.

1996 Unified Modeling Language (UML), Version 0.9.

Einbezug namhafter Informatikunternehmen (Digital, Microsoft, Oracle, Hewlett-Packard, …); Verfeinerung von Notation, Konzepten und Sprachdefinition. Gleichzeitig wächst der Sprachumfang.

1997	UML Versionen 1.0 und 1.1 erscheinen.
	Einreichung von UML bei der OMG (Object Management Group) als Kandidat für einen geplanten Standard für objektorientierte Modellierungs-sprachen.
Sept. 1997	Die OMG erklärt UML Version 1.1 zum Standard.
1998	UML Version 1.2 erscheint. Enthält Detailänderungen und Fehlerkorrekturen.
	Eine Flut von UML-Büchern bricht über die Informatikwelt herein.
1999	UML Version 1.3.

- Referenzhandbuch: Rumbaugh, Jacobson, Booch (1999)
- Derzeit bestes deutschsprachiges Buch: Oestereich (1998)

2 Grundkonzepte und Elemente der UML

- UML besteht aus einer Sammlung vorwiegend grafischer Sprachen zur Erstellung von Anforderungs- und Entwurfsmodellen aus verschiedenen Perspektiven
- Eine UML-Spezifikation besteht aus einer Sammlung sich ergänzender und teilweise überlappender Modelle
- Im Zentrum steht ein *Klassenmodell*, das den strukturellen Aufbau eines Systems spezifiziert
- Das Klassenmodell beschreibt
  - bei Anforderungsmodellen die Gegenstände der Realität, mit denen das System umgehen muss
  - bei Entwurfs- und Implementierungsmodellen die Klassen der Lösung
- Bei der Modellierung von Anforderungen kommt als zentrales Element das *Anwendungsfallmodell* hinzu, das die Benutzer-System-Interaktion aus Benutzersicht modelliert
- Nach Bedarf beschreiben weitere Modelle zusätzliche Systemsichten, insbesondere dynamisches Verhalten, Funktionalität und Zusammenarbeit

UML unterstützt insgesamt sieben verschiedene Sichten

- Die statische Sicht: Klassen und Objekte, strukturelle Beziehungen
- Die Benutzersicht: Anwendungsfälle
- Die Verhaltenssicht: Zustandsautomaten
- Die Aktivitätssicht: Ablauf von Aktivitäten
- Die Interaktionssicht: Interaktion ausgewählter Objekte
- Die Physische Sicht: Physische Systemstruktur
- Die Gliederungssicht: Portionierung der Modelle in Pakete und Subsysteme

3 Die statische Sicht

Klassen und Objekte

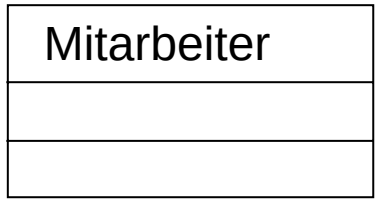
Objekt (object) – Ein individuell erkennbares, von anderen Objekten eindeutig unterscheidbares Element der Realität.

Beispiel: Die konkrete Person Eva Müller, 36 Jahre alt, Dr. oec. publ., Leiterin Fertigung in der Firma AGP, verheiratet, ein Kind, ... wird als *Objekt* modelliert

Gleichartige Objekte werden durch *Klassen* modelliert.

Klasse (class) – eine eindeutig benannte Einheit, welche eine Menge gleichartiger Objekte beschreibt.

Beispiel: Mitarbeiter, Abteilung

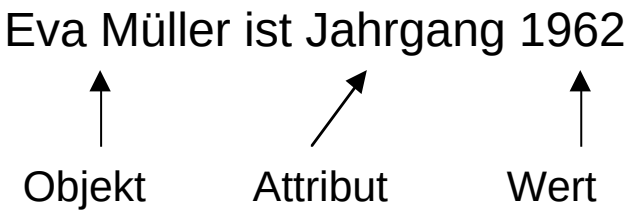


- In UML werden in der Regel Klassen modelliert
- Objekte dienen zur Modellierung besonderer Aspekte, z.B. des Ablaufs einer Operation

Objekte haben Eigenschaften

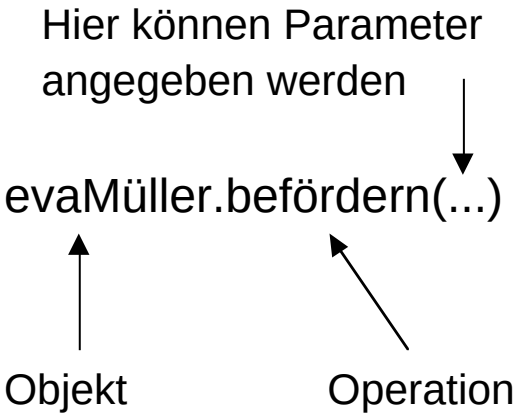
Attribute (attributes) modellieren Merkmale,

- die für jedes Objekt einen *Wert* haben
- deren Werte *keine* eigene *Identität* besitzen

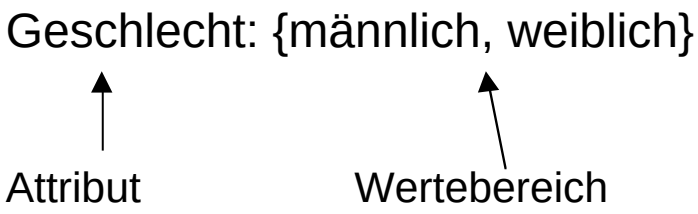


Operationen (operations, services)

- modellieren die Möglichkeiten zur *Bearbeitung* von Objekten
- werden in Anlehnung an Smalltalk und Java auch Methoden (methods) genannt



Wertebereiche (domains) beschreiben die Menge aller möglichen Werte für ein Attribut



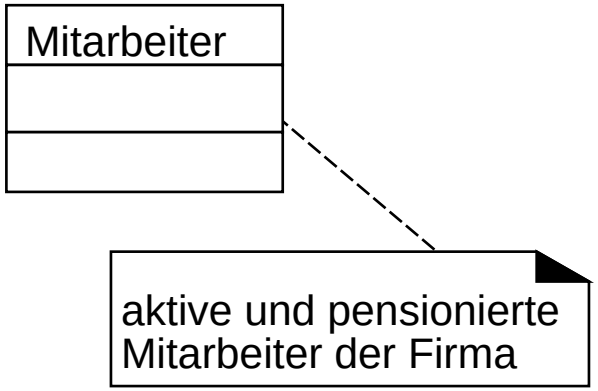
Weitere Eigenschaften

In UML können zu allen Attributen und Operationen

- Kardinalitäten — vornamen [1..5]: String
- Eigenschaftswerte — stammNr: Integer {readonly}
- Zusicherungen — salaer: CHF {salaer > 0}
- und diverses weiteres — --> siehe einschlägige Literatur

angegeben werden

Jede Klasse kann ferner informal mit Notizen kommentiert werden



Notation von Klassen

Als Text mit allen Details:

Mitarbeiter

stammNr: Integer {readonly}

name: String

vornamen [1..5]: String

Geburtsdatum: Date

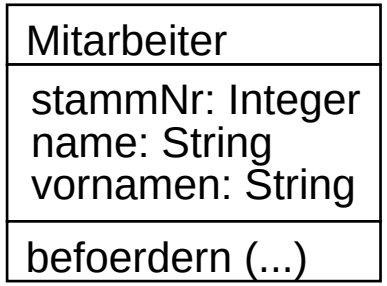
eingetretenAm: Date

...

befoerdern (nach: Stufe, ab: Date)

...

Graphisch mit den wichtigsten Attributen und Operationen:

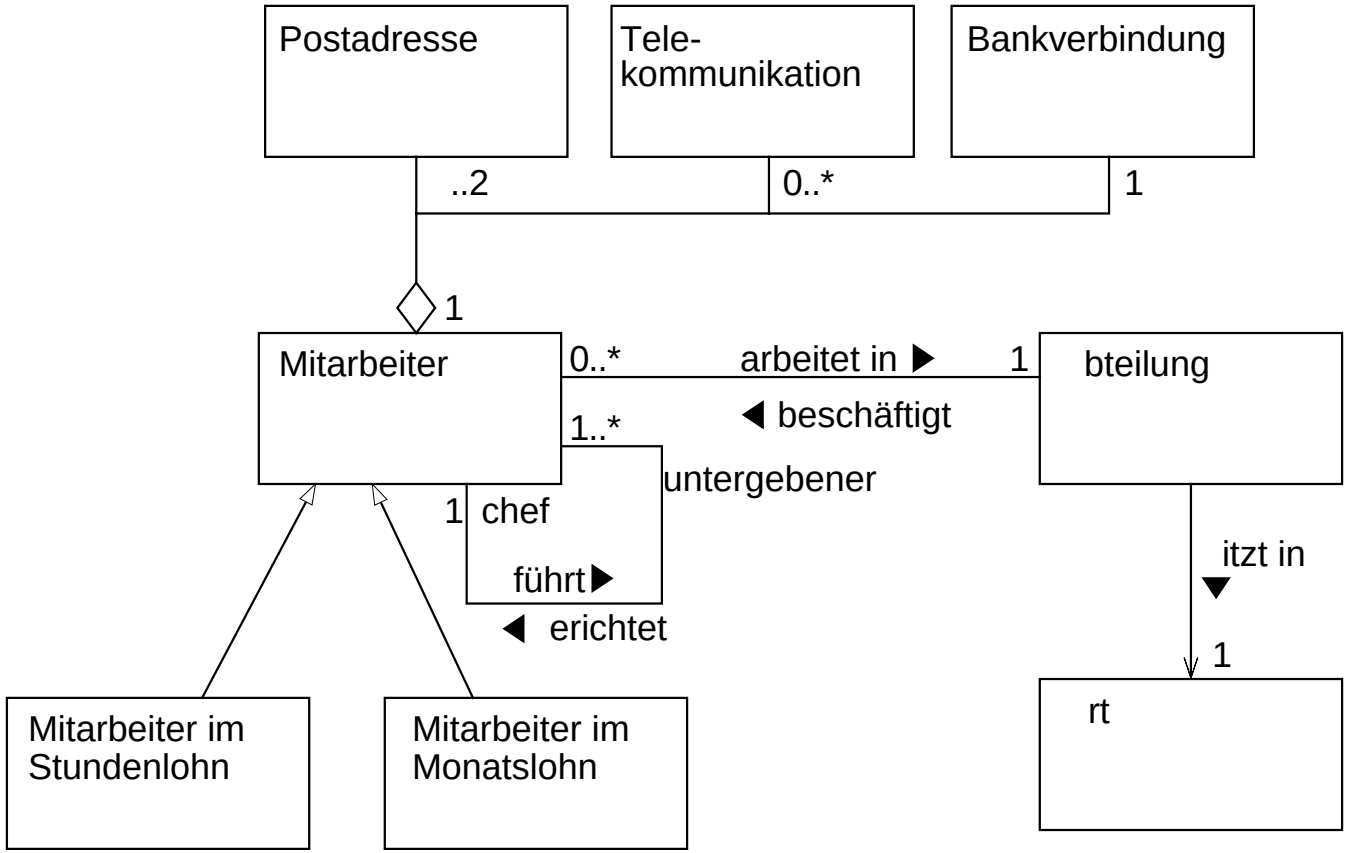


Klassendiagramme

Klassendiagramme beschreiben statische Zusammenhänge zwischen Klassen

- Generalisierung/Spezialisierung (generalization/specialization) – über Vererbung
- Assoziation (association)
- Aggregation (aggregation)
- Abhängigkeit (dependency)

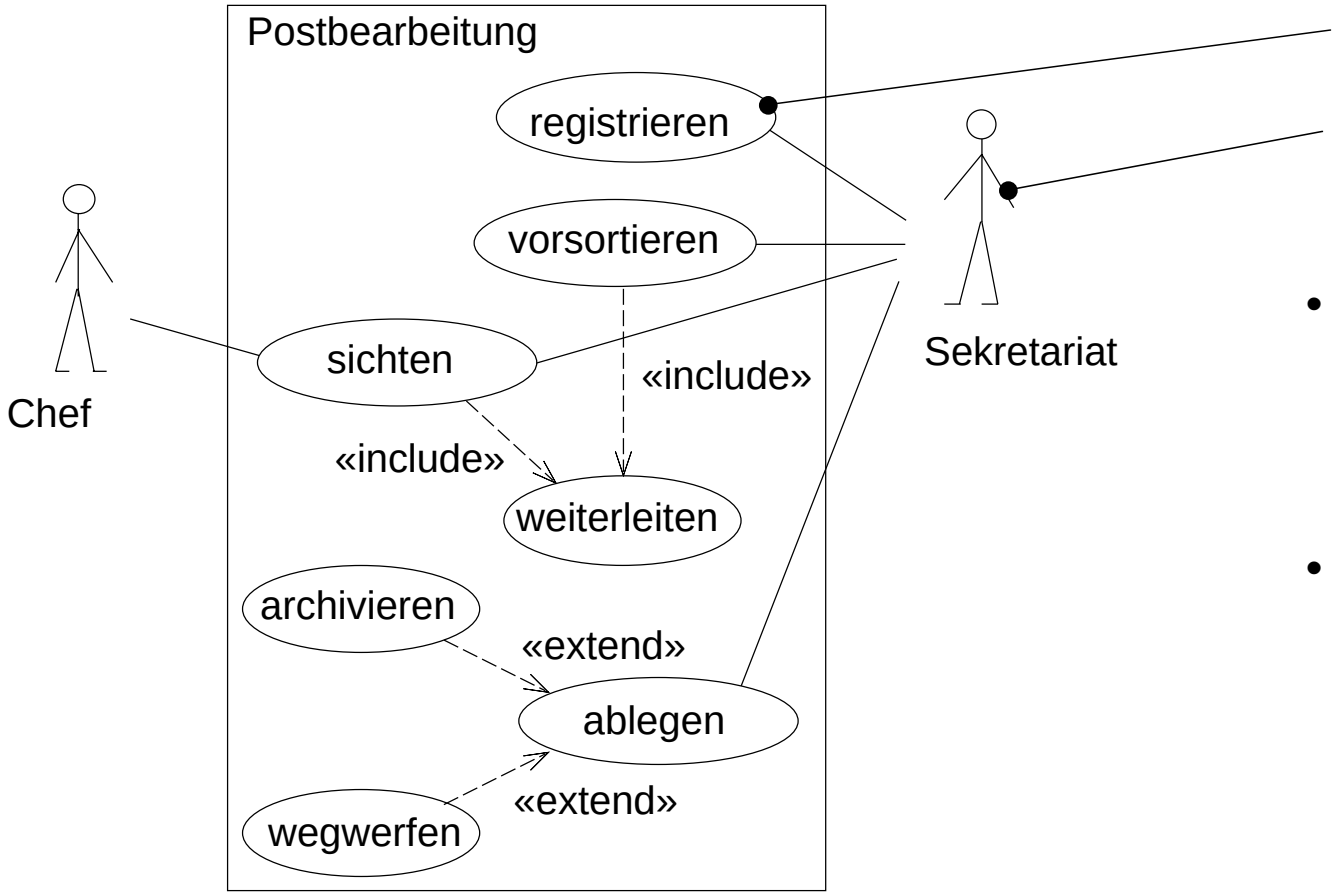
Beispiel eines Klassendiagramms:



4 Die Benutzersicht: Anwendungsfälle

Anwendungsfall (use case) – Eine durch genau einen Akteur angestoßene Folge von Systemereignissen, welche für den Akteur ein Ergebnis produziert und an welchem weitere Akteure teilnehmen können

- modelliert Interaktionen
  - typisch zwischen Anwendern und System
  - oder zwischen Komponenten in Geschäftsvorfällen / Prozessen
  - selten auch zwischen systeminternen Komponenten
- Jeder Anwendungsfall beschreibt eine Klasse möglicher Interaktionen
- Konkrete Interaktionsabläufe werden auch Szenarien genannt
- UML definiert keine Sprache zur Spezifikation von Anwendungsfällen. Meistens wird *strukturierter Text* verwendet
- Die Menge aller Anwendungsfälle wird in einem *Anwendungsfalldiagramm* dargestellt



- Mit «include» werden gemeinsam genutzte Sub-Anwendungsfälle eingebunden
- «extend» dient zur Modellierung von Varianten, die einen Anwendungsfall an definierten Stellen mit unterschiedlicher Funktionalität erweitern

Beispiel eines Anwendungsfalldiagramms

Detailspezifikation eines Anwendungsfalls

- UML lässt beliebige Beschreibungsformen zu
- Gebräuchlich sind Beschreibungen mit
  - Fließtext
  - einer Folge von Schritten in natürlicher Sprache (die heute gebräuchlichste Beschreibungsart)
  - einem Aktiviätsdiagramm (siehe 6)
  - einem Zustandsdiagramm (siehe 5)

Anwendungsfall: Buch ausleihen

...

Normalablauf:

1 BenutzerIn liest ihre Karte ein
System prüft und validiert die Karte

2 BenutzerIn wählt „Ausleihen“
System aktiviert die Ausleihfunktion

3 Benutzerin liest Buchcode ein
System identifiziert das Buch, registriert die Ausleihe, deaktiviert das Diebstahletikett

...

Sonderfälle:

1' Karte ist ungültig: System gibt Karte zurück, beendet Vorgang

...

5 Die Verhaltenssicht: Zustandsautomaten

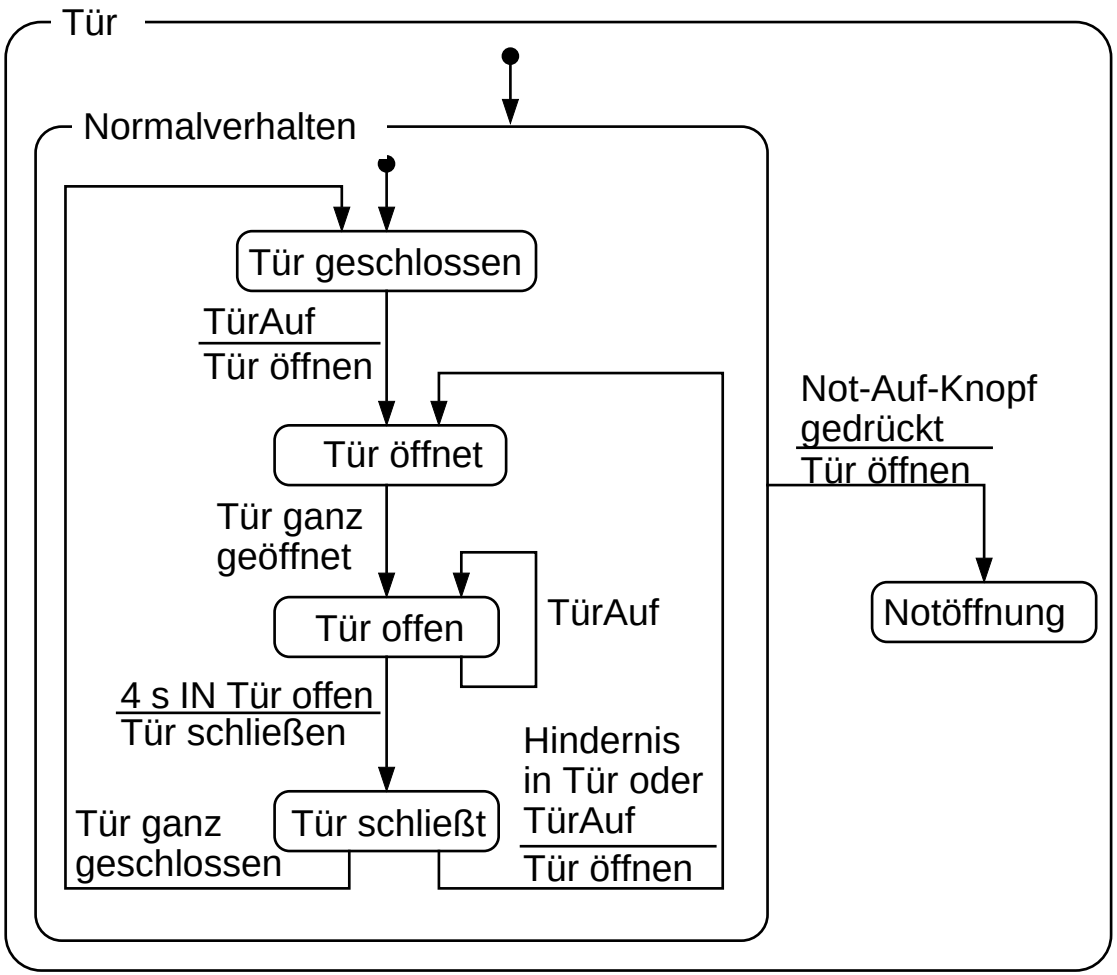
UML modelliert das dynamische zeitliche Verhalten eines Systems mit *Zustandsdiagrammen* (state machine, state diagram)

Ein Zustandsdiagramm modelliert

- mögliche Zustände der Objekte einer Klasse oder eines Teilsystems
- Dynamik des Systemverhaltens: Reaktionen auf Folgen äußerer Ereignisse

Notation und Bedeutungen sind sehr ähnlich wie bei Statecharts (Harel 1988)

Beispiel eines Zustandsdiagramms:



TürAuf =

Karte mit Zutrittsrecht gesteckt
oder
Annäherungssensor hat angesprochen
oder
Öffnungsknopf gedrückt

6 Die Aktivitätssicht: Ablauf von Aktivitäten

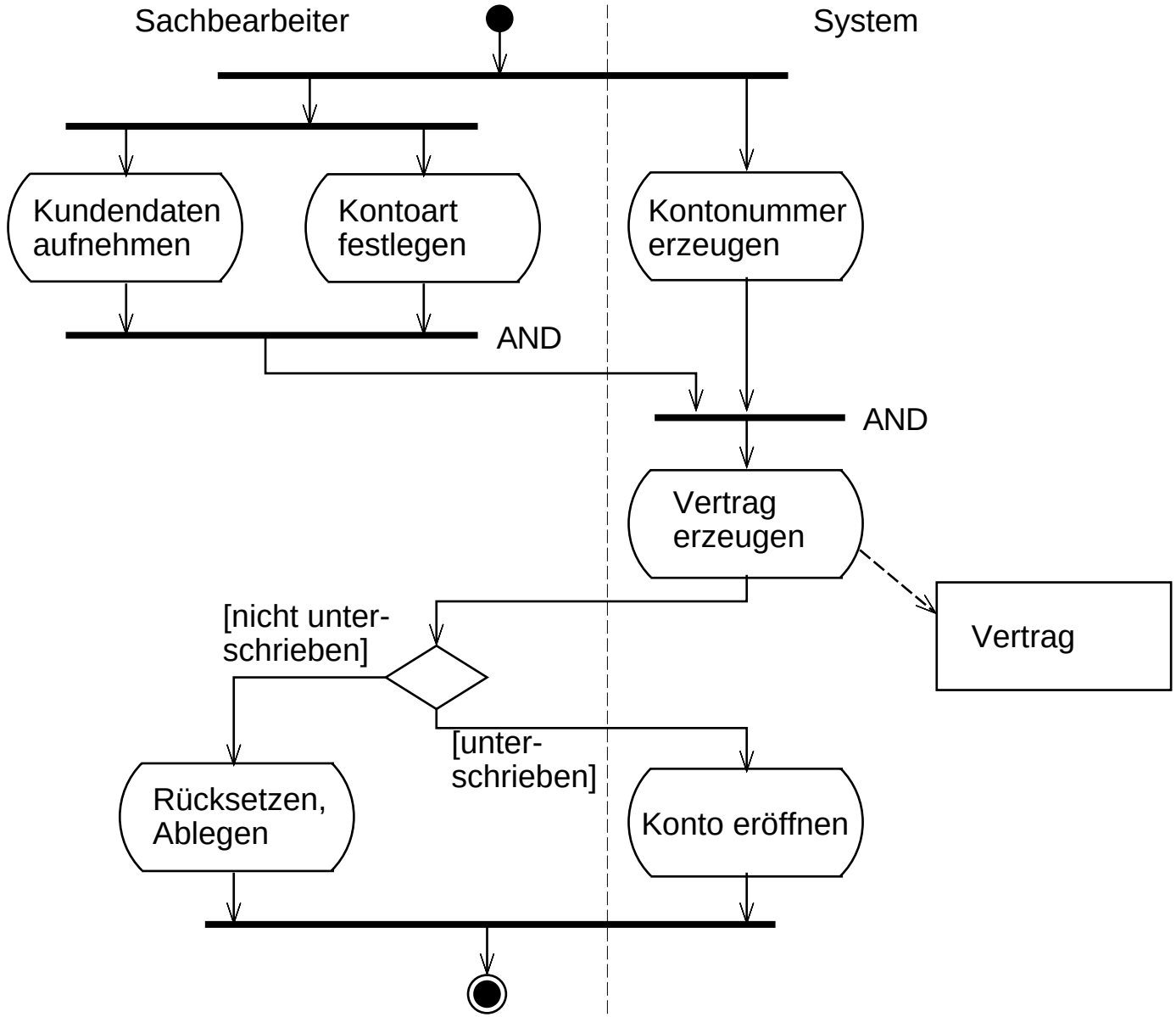
Aktivitätsdiagramme (activity diagrams)

- beschreiben Abläufe in einem System
- verwenden Aktivitäten, Steuerfluss und Objektzustände als Hauptelemente
- können beim Steuerfluss Fallunterscheidungen und Parallelität modellieren
- können außerdem das Versenden und Empfangen von Signalen modellieren

- Ferner ist die Zuordnung von Verantwortlichkeiten für die einzelnen Aktivitäten möglich.
- Aktivitätsdiagramme unterscheiden sich konzeptionell nur unwesentlich von den traditionellen Programmablaufplänen (flowcharts)

  ➜ Vorsicht: beliebige Spaghetti-Strukturen sind modellierbar

Beispiel eines Aktivitätsdiagramms:



7 Die Interaktionssicht: Interaktion ausgewählter Objekte

In der Interaktionssicht modelliert man

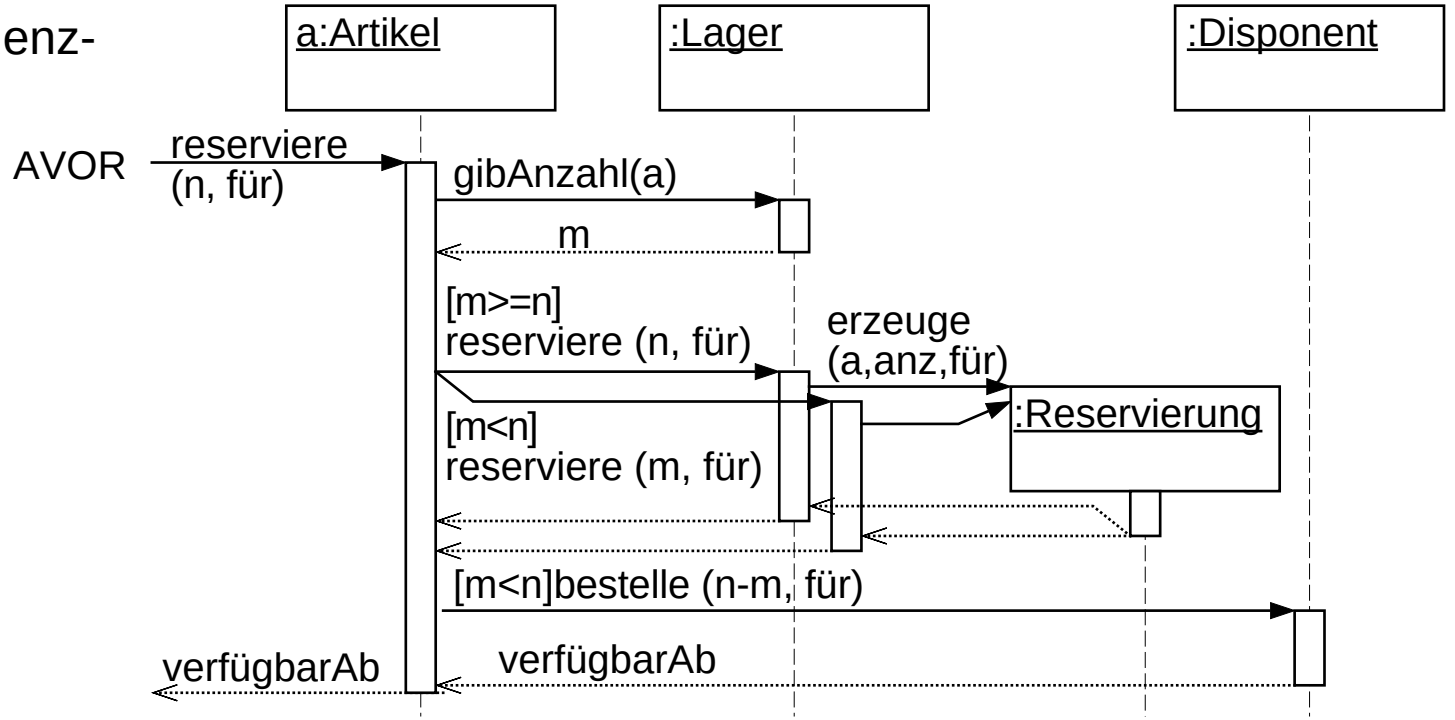
- ausgewählte Objekte und
- deren Interaktion in zeitlicher Reihenfolge
- ⇨ partielle dynamische Aspekte eines Systems

Zwei Arten von Diagrammen stehen zur Verfügung:

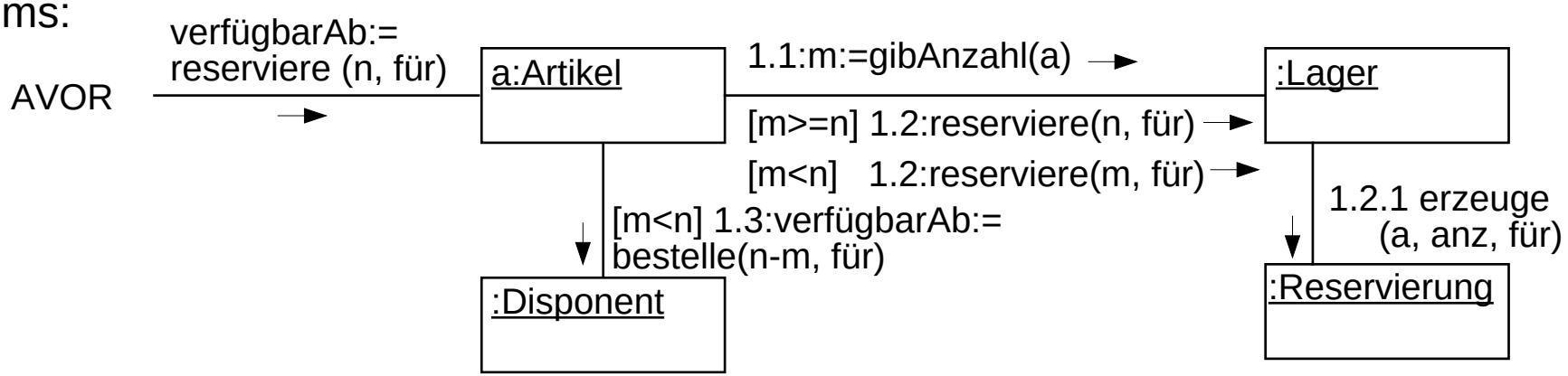
- Sequenzdiagramme (sequence diagrams): Zeitachse ist sichtbar; zeitlicher Ablauf im Vordergrund
- Zusammenarbeitsdiagramme (collaboration diagrams): Objektstruktur und -Aufrufe im Vordergrund

Die beiden Diagrammarten sind logisch äquivalent

Beispiel eines Sequenz-diagramms:



...und eines äquivalenten Zusammenarbeits-diagramms:

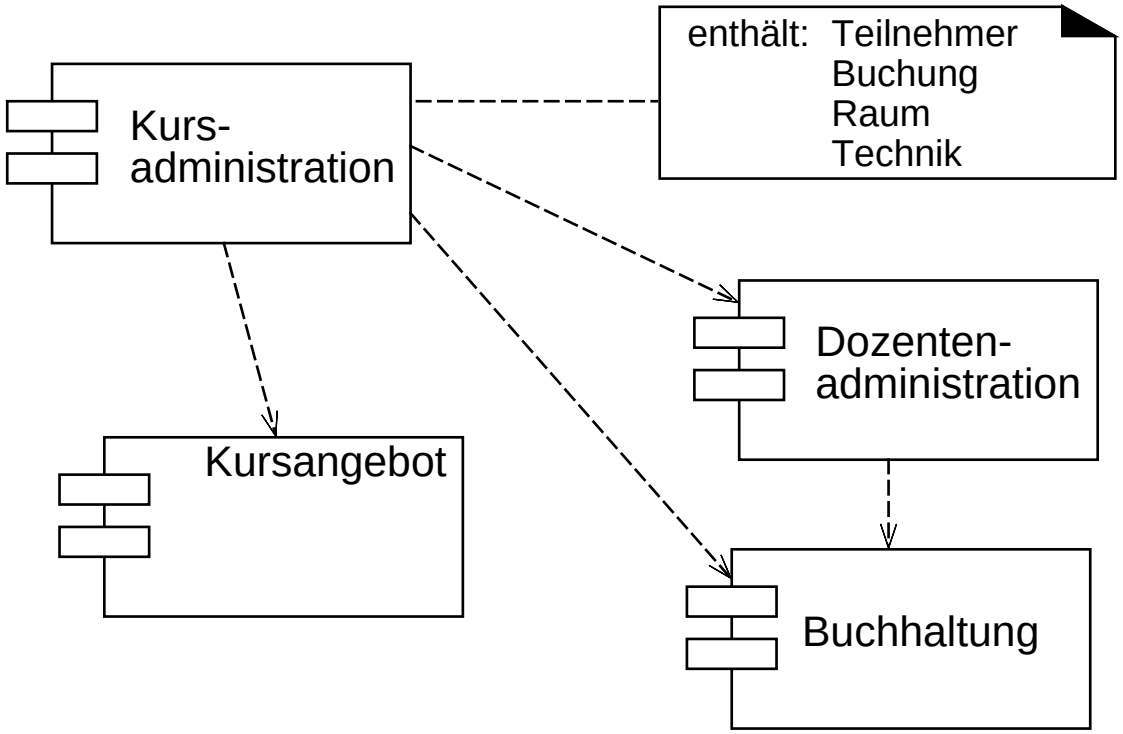


8 Die Physische Sicht: Physische Systemstruktur

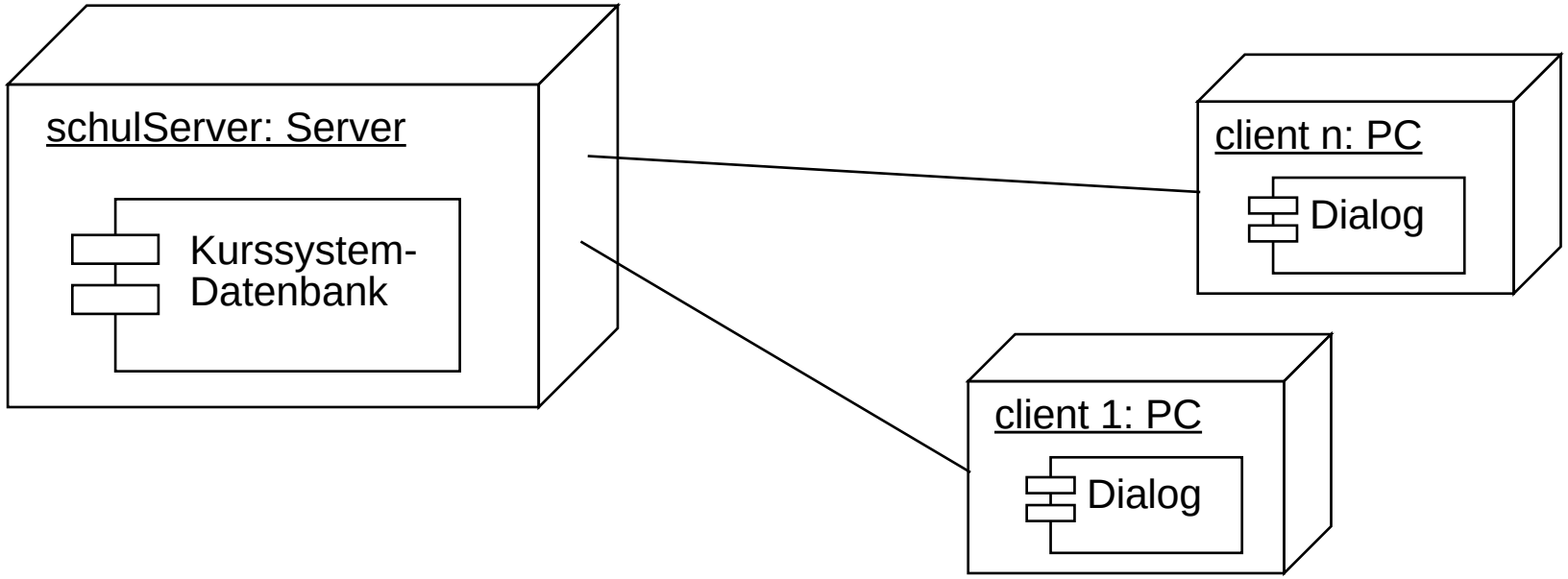
Zur Modellierung der physischen Systemstruktur (primär in Entwurfsmodellen) stellt UML *Komponentendiagramme* und *Verteilungsdiagramme* zur Verfügung:

- Komponentendiagramm (component diagram) – modelliert Komponenten und deren Abhängigkeiten (welche Komponente benutzt welche andere)
- Komponente – Modul mit eigener Identität und wohldefinierter Schnittstelle zum restlichen System)

Beispiel eines Komponentendiagramms



- Verteilungsdiagramm (deployment diagram) – modelliert die Struktur der Hardware und die Verteilung der Software auf Hardware-Komponenten



Beispiel eines Verteilungsdiagramms

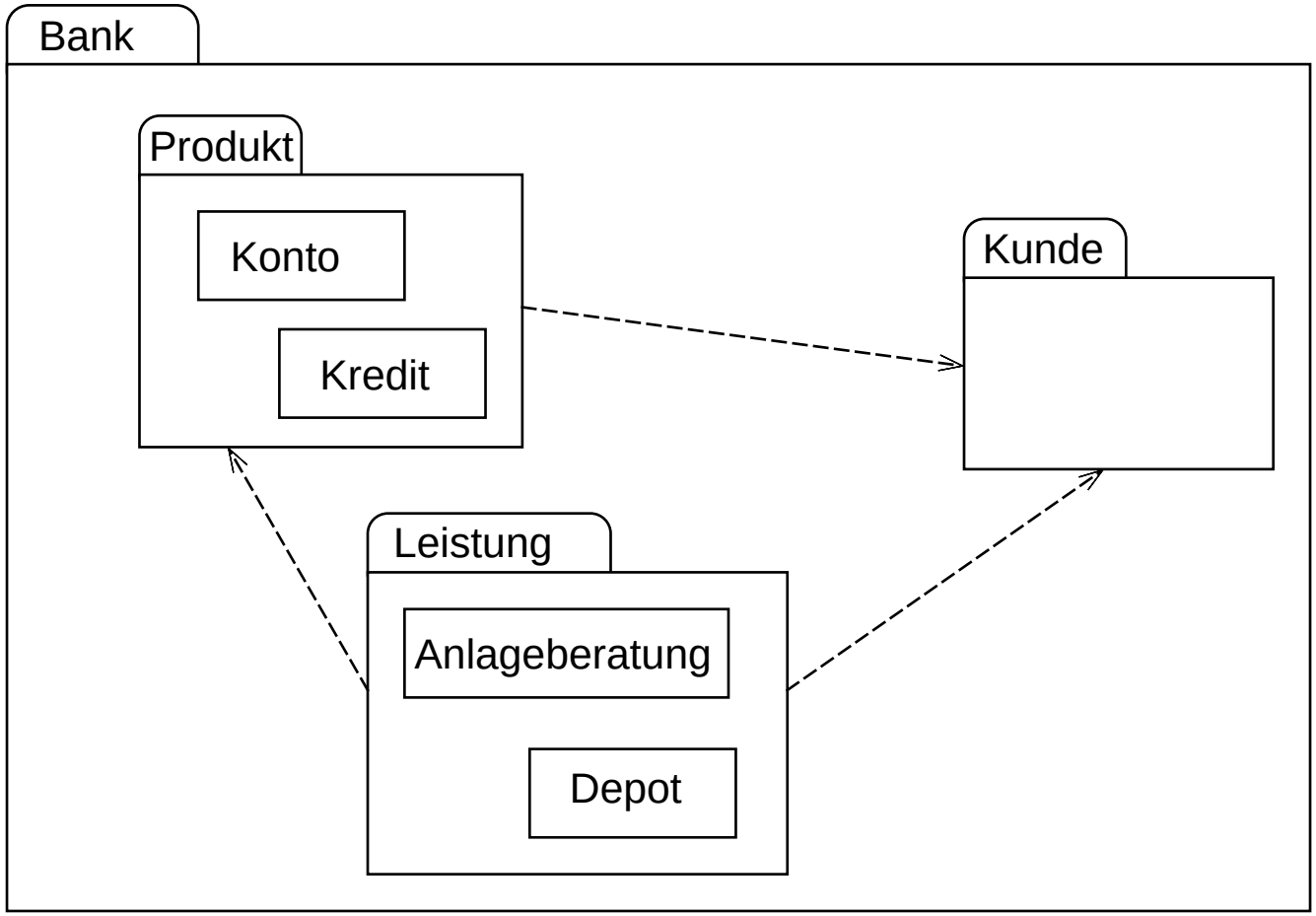
9 Die Gliederungssicht

Große UML-Modelle werden mit Hilfe von Paketen gegliedert

Paket (package) – Behälter für beliebige Modellelemente

- Definiert einen Namensraum (innerhalb des Pakets müssen Namen eindeutig sein)
- Jedes Modellelement gehört in genau ein Paket
- In jedem Paket sind Modellelemente aus anderen Paketen referenzierbar

  Wird ein Element aus Paket A in Paket B referenziert, so wird B von A abhängig („B benutzt ein Element aus A“)



Beispiel eines Paketdiagramms

10 OCL (Object Constraint Language)

Eine formale Sprache zur Formulierung zusätzlicher Modelleigenschaften, z.B.

- Zusicherungen über Zusammenhänge zwischen verschiedenen Modellelementen
- Invarianten, d.h. Eigenschaften von Objekten einer Klasse, die sich nie ändern dürfen
- Voraussetzungen für die Ausführung von Operationen
- Ergebniszusicherungen für das Resultat von Operationen

Beispiele:

Mitarbeiter

self.beruf = Fahrer implies self.alter > = 18

sqrt(argument: Real) {argument >=0}

11 Erweiterungsmechanismen in der UML

- UML versteht sich als offener Standard
- Andere Sprachen/Modelle können mit UML kombiniert werden
- UML selbst kann benutzerdefiniert erweitert oder gar modifiziert werden
- Zwei Erweiterungsmechanismen
  - Standardisierte Annotationen (Tagged Values)
  - Stereotypen

11.1 Standardisierte Annotationen (Tagged Values)

- Bestehen aus einem Schlüsselwort (tag)
- und einem Wert dazu (value)

- Eine Reihe von tags sind vordefiniert
- Jeder Benutzer kann beliebige eigene tags definieren

Beispiele

Eine vordefinierte Annotation für Klassen:

class ticket_DB
    {**location**=host
    persistence=persistent}
...

Selbstdefinierte Annotationen:

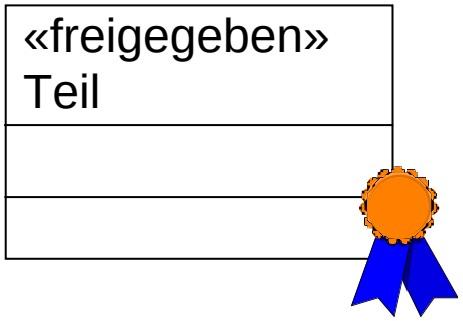
Autor=Glinz erstellt=1999-09-15 Termin=1999-09-16

11.2 Stereotypen

Stereotyp – Ein Mechanismus für benutzerdefinierte Erweiterung, Präzisierung oder Redefinition von Elementen einer Sprache (ohne Veränderung des Metamodells der Sprache)

- *Dekorative Stereotypen* variieren die äußere Darstellung (Notation) einer Sprache
  Anwendung: Schaffung benutzerdefinierter Symbole
- *Deskriptive Stereotypen* erweitern eine Sprache rein syntaktisch
  Anwendung: Annotationen (analog zu tagged values), Sekundärklassifikation
- *Restriktive Stereotypen* präzisieren bestehende Sprachkonstrukte oder schaffen neue
  Anwendung: Hinzufügen fehlender Sprachkonstrukte, Überlagerung bestehender Konstrukte mit zusätzlichen Bedeutungen
- *Redefinierende Stereotypen* verändern die Bedeutung von Sprachkonstrukten
  Anwendung: Definition einer neuen Sprache auf der Grundlage einer bestehenden Sprache

Beispiel:

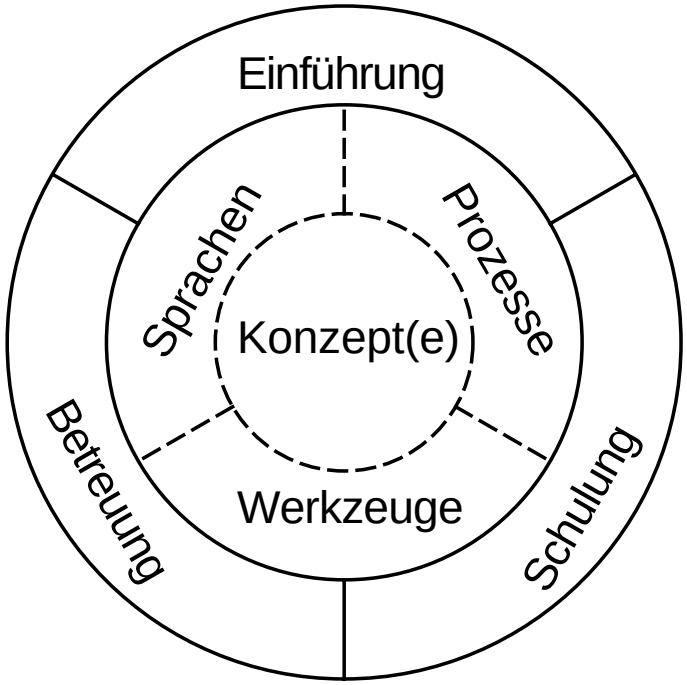


stereotype freigegeben
{
host Class;
properties Date datum; String visum
restrictions visum **in** listeDerFreigabeberechtigten }

12 Bewertung von UML

12.1 Der Stellenwert von UML

- Eine Sprache ist nur *ein* Element im Software Engineering
- UML wird in den nächsten Jahren die Landschaft der Modellierungssprachen dominieren
- UML ist definitiv nicht die endgültige Modellierungssprache



12.2 Stärken und Schwächen von UML

+ umfassend
+ Industriestandard
+ Großes Angebot an Büchern, Unterlagen und Schulung
+ (nach und nach) durch alle großen Werkzeuganbieter unterstützt
+ offener, erweiterbarer Standard

– erhebliche konzeptionelle Schwächen: fehlende Systemdekomposition, (zu) viele Konstrukte ohne klare Bedeutung, Spezifikation des globalen Verhaltens ungeklärt, Modellierung des Systemkontextes / Einbettung vorhandener Modellteile schwierig
– Anything goes; öffnet der Beliebigkeit Tür und Tor
– Großer Sprachumfang
– UML-Modelle sind Sammlungen von Einzelmodellen: Konsistenzprobleme, Problem des Zusammensuchens relevanter Information
– Gefahr der erneuten Sprachverwirrung durch unkontrollierte und undisziplinierte Erweiterungen

Literatur

Auswahl von Büchern zur UML (Stand 1999)

Oestereich, B. (1998). *Objektorientierte Softwareentwicklung*. R. Oldenbourg Verlag München.

Übersichtlich gegliedert, klar und anschaulich geschrieben. Gute Beschreibung der UML-Diagramme und der Notation. Inklusive Kapitel über OCL. Grundlage ist die UML 1.2.

Rumbaugh, J., Jacobson, I., Booch, G. (1999). *The Unified Modeling Language Reference Manual*. Reading, Mass., etc.: Addison-Wesley. [In deutscher Übersetzung: Das UML Benutzerhandbuch, Bonn: Addison-Wesley Longman, 1999]

Referenzhandbuch der Sprache.

Booch, G., Jacobson, I., Rumbaugh, J. (1999). *The Unified Modeling Language User Guide.* Reading, Mass., etc.: Addison-Wesley.

Tutorium für den Gebrauch von UML.

Muller, P.-A. (1998). *Instant UML.* Wrox Press Inc.

Klare Gliederung, gute Beschreibung der Notation. Zusätzlich geht der Autor auf die Anbindung an Java, DBMS, C ... ein. Es fehlt eine Kurzübersicht über die Notation sowie eine Erläuterung von OCL.

OMG (1999). *OMG Unified Modeling Language Specification Version 1.3.* OMG document ad/99-06-08.

ftp://ftp.omg.org/pub/docs/ad/99-06-08.pdf

Sprachdefinition von UML 1.3.

Weitere Literatur

Berner, S., M. Glinz, S. Joos (1999). A Classification of Stereotypes for Object-Oriented Modeling Languages. *Proceedings of the Second International Conference on the Unified Modeling Language*, Fort Collins, Oct 1999. Berlin, etc. Springer.

Booch, G. (1994). *Object Oriented Analysis and Design with Applications*. Second Edition. Redwood City, Ca.: Benjamin/Cummings.

Coad, P., E. Yourdon (1991). *Object-Oriented-Analysis*. Prentice-Hall, Englewood Cliffs, N.J. [auf Deutsch: Objektorientierte Analyse, 1994 im gleichen Verlag]

Harel, D. (1988). On Visual Formalisms. *Communications of the ACM* **31**, 5 (May 1988). 514-530.

Jacobson, I., M. Christerson, P. Jonsson, and G. Övergaard (1992). *Object Oriented Software Engineering: A Use Case Driven Approach*. Amsterdam, etc.: Addison-Wesley.

Oestereich, B. (1998). *Objektorientierte Softwareentwicklung*. R. Oldenbourg Verlag München.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, W. Lorensen (1991). *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, N.J.

Rumbaugh, J., Jacobson, I., Booch, G. (1999). *The Unified Modeling Language Reference Manual*. Reading, Mass., etc.: Addison-Wesley.

