# Die Unified Modelling Language (UML) Version 2 - Stichworte

Udo Kelter

05.01.2005

**Zusammenfassung dieses Lehrmoduls**

Die Unified Modelling Language (UML) ist der de-facto-Standard für Modellierungssprachen geworden. Die neue Version 2 weist in vielen Bereichen signifikante Änderungen gegenüber der Version 1.5 auf. Dieses Lehrmodul stellt nach einem Überblick über die Geschichte der UML und den Motiven für die neue Version die Diagrammtypen der Version 2 kurz vor und skizziert die Änderungen gegenüber der Version 1.5.

**Vorausgesetzte Lehrmodule:**

obligatorisch:
- Objektorientierte Modellierung
- Objektorientierter Entwurf
- Zustandsübergangsdiagramme
- Petri-Netze

**Stoffumfang in Vorlesungsdoppelstunden:** 0.7

# Inhaltsverzeichnis

# 1 Einordnung der UML

**UML** ist Abkürzung von *Unified Modelling Language*
Was ist die UML?

- ist eine große Hype, vielfach als die Wunderwaffe schlechthin hochstilisiert (des Kaisers neue Kleider ...)

- ist eine graphische *Notationsform* für Systemmodelle;
  (UM)..L = *language*!
  nicht nur Analyse, sondern auch (bzw. sogar schwerpunktmäßig) Entwurf, incl. Generierung von Programmrümpfen
  stellenweise sogar PiK (Programmieren im Kleinen)

- graphisch → dient zur *Visualisierung* und *Dokumentation* von Softwaresystemen

- ist teilweise Sprache mit "Semantik-Definition"
  d.h. Transformation der Modelle (oder von Modellteilen) in Programmquelltext (Datenstrukturen, Ablaufstrukturen)
  oder Interpretation des Modells
  → "Spezifikationssprache", sofern für Entwurf eingesetzt
  **aber:** ist nicht vollständig, keine Programmiersprache, nicht komplett formalisiert

- ist eine Spezifikation einer SEU (Software-Entwicklungsumgebung): teilweise ja;
  unterstellt leistungsfähige SEU und erläutert z.B. Darstellungsvarianten / Interaktionsformen mit Entwickler

- ist *kein* explizites Vorgehensmodell
  (oder Methode / Pragmatik),
  unterstellt aber ein evolutionäres, oo-typisches Vorgehen

- ist nicht spezialisiert auf ein Anwendungsgebiet, versucht, beliebige Arten von SW / Anwendungsbereichen abzudecken

Stammtischthema: gibt es Modelle, die domänenunabhängig sind?
(Gibt es eine domänenunabhängige Softwaretechnik, die mehr bietet als eine gute Programmiersprache?)

Schwerpunkt bei zur Zeit betrieblichen Informationssystemen und Realzeitanwendungen, hierzu eine Vielzahl ergänzender Standards (Vergleich: Java-Sprachkern - Java Development Kit incl. div. Bibliotheken)

- ist ein Industriestandard: breite Unterstützung, ist OMG-Standard
- ist komplex und kaum noch überschaubar
  (insb. wegen ergänzender Technologien und Standards)
  individuell zu beantwortende Frage: welche Teile der UML sind sinnvoll nutzbar?

## 1.1 Historie der UML

"Methodenkrieg" der oo-Modellierung Ende 198* - Anfang 199*:

- Methode "Object-Oriented Analysis and Design" von Booch [Bo91, Bo94],
- "Object-Oriented Analysis" von Coad/Yourdon [CoY90, CoY91],
- "Object-Oriented Design" von Coad/Yourdon [CoY91a],
- "Object Modelling Technique" (OMT) von Rumbaugh et al. [Ru+91]
- "Object-Oriented Software Engineering" (OOSE) von Jacobson [Ja+92a],

Differenzen zwischen den Methoden:

- Notationsunterschiede (bei gleicher Bedeutung)
- konzeptuelle Differenzen bei der Behandlung gleicher Modellierungsaspekte (teilweise kleinkariertes Gezänk)
- unterschiedliche Abdeckung von Entwicklungsphasen und Modellierungsaspekten

**UML-Versionen:**

`1.0 (1997)` von Booch, Jacobson, Rumbaugh (alle von Rational "aufgekauft");
erster Ansatz zur Vereinigung der Einzelmethoden

`1.1 (1997)` weitere Beteiligte; Hinzunahme der OCL (Object Constraint Language)

`1.2 (1998)` nicht publiziert

`1.3 (1999)` von OMG übernommen;
+ XMI (XML Metadata Interchange)

`1.4 (2001)` div. Bereinigungen / Ergänzungen,
von vielen Werkzeugen unterstützt

`1.5 (2003)` div. Bereinigungen

`2.0 (Ende 2004?)` erheblicher Umbau; noch nicht formell verabschiedet

**Schwächen der UML 1.x / Ziele bei der Entwicklung der UML 2.0:**

- "Entrümpelung" um wenig benutzte / spezielle Konstrukte
- mehr Präzision, hierzu: Verbesserung des Metamodells; stärkere Verwendung der OCL
- Verbesserung der Konsistenz zwischen statischen und dynamischen Modellen
- bessere Unterstützung der Komponentenentwicklung auf Basis von J2EE
- bessere Unterstützung von Echtzeitanwendungen
- leistungsfähigere Verhaltensmodellierung (betrifft vor allem Aktivitätsdiagramme und ZÜD)
- bessere Unterstützung von Systemhierarchien / -Zerlegungen

## 1.2 Die Diagrammtypen der UML 2.0 - Übersicht

**Strukturdiagramme:**

1. Klassendiagramm
2. Komponentendiagramm
3. Kompositionsstrukturdiagramm
4. Objektdiagramm
5. Verteilungsdiagramm
6. Paketdiagramm

**Verhaltensdiagramme:**

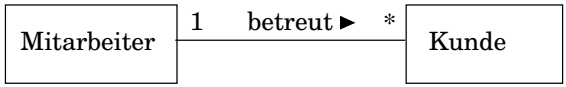
1. Use-Case-Diagramm
2. Aktivitätsdiagramm
3. Zustandsautomat
4. Interaktionsdiagramme
   1. Sequenzdiagramm
   2. Kommunikationsdiagramm
   3. Interaktionsübersichtsdiagramm
   4. Timing-Diagramm

# 2 Kurzbeschreibung der Diagrammtypen

... und Neuerungen in UML 2.0

## 2.1 Klassendiagramm

- Varianten für Analyse und Entwurf/Architektur

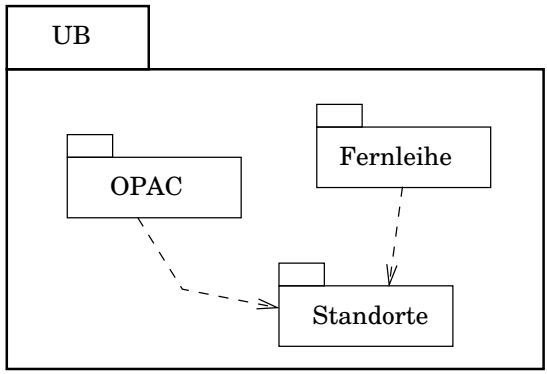



- ist Datenmodell und Funktionsmodell
- Zweck: Überblick über die statische Struktur eines Systems / Beziehungen der Systemteile gewinnen
- dargestellter Inhalt: statische Struktur des Systems (Datentypen und Signaturen der Operationen); Typhierarchien; ggf. Schnittstellen

- Querbeziehungen zu anderen Diagrammtypen: sehr viele, wichtigster Diagrammtyp
- UML 2.0: nur wenige Detailänderungen gegenüber UML 1.x

## 2.2 Paketdiagramm

- ist Modell für die Grobstruktur eines Systems




- Zweck: Überblick über die Grobstruktur eines Systems gewinnen
- dargestellter Inhalt: Hierarchie der Pakete, Namen der enthaltenen Klassen, Benutztbeziehungen;
  bildet Namensräume
- Querbeziehungen zu anderen Diagrammtypen: Klassendiagramme
- UML 2.0: keine wesentliche Änderungen

## 2.3 Objektdiagramm

- ist (im weiteren Sinne) ein Funktionsmodell
- Zweck: Gruppierung von Objekten darstellen, i.d.R. Laufzeitobjekte (vgl. Begriff Kollaboration)
- dargestellter Inhalt: Objekte, deren Beziehungen untereinander, ggf. deren Typ und Attributwerte; auch Anzahl bzw. Mengen von Objekten
  oft anonyme, als Beispiele zu verstehende Objekte

- Querbeziehungen zu anderen Diagrammtypen: Klassendiagramme; "erben" von dort die Liste der zulässigen Attribute und die nicht dargestellten Operationen der Objekttypen
- UML 2.0: keine wesentliche Änderungen

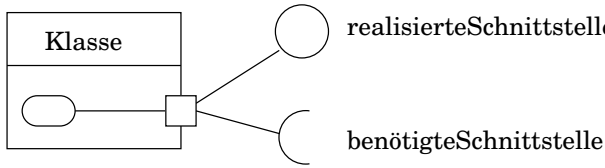
## 2.4 Kompositionsstrukturdiagramm

- komplett neu in UML 2.0
- Zweck: veranschaulichen,
  - wie eine Systemkomponente zusammengesetzt ist
  - wie die Teile mit anderen Systemkomponenten kommunizieren
- dargestellter Inhalt: Laufzeitinstanzen innerhalb verschiedener Systemteile und deren Kommunikationswege

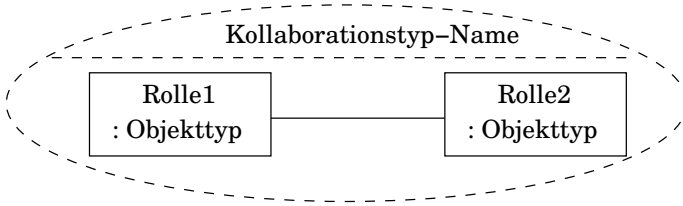
  a) **Teile** (**Parts**): anwendbar für ein beliebiges Modellelement (Classifier); stellt darin enthaltene andere Modellelemente und die jeweilige Anzahl dar

  b) **Port**: repräsentiert Schnittstelle, über die ein Classifier oder ein Teil davon mit seiner Umwelt kommuniziert

  Beispiel:




  c) **Kollaborationstypen**: Rollen von Objekten (eines bestimmten Typs), die in Beziehung zueinander stehen



d) **Kollaborationen**: stellen Anwendungen von Kollaborationstypen mit bestimmten Objekten / in bestimmten Situationen dar

- Querbeziehungen zu anderen Diagrammtypen: zu Klassendiagrammen

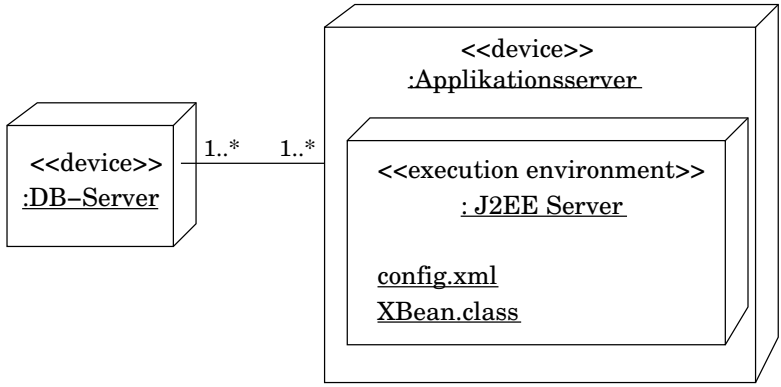
## 2.5 Verteilungsdiagramm

(*deployment diagram*)

- modelliert die Hardware-Umgebung, Ausführungsumgebung und Zuordnung von Komponenten

  nur bei verteilten Systemen sinnvoll
- Zweck: Überblick darüber gewinnen, welche (installierte) Software auf welchem (virtuellen) Prozessor ausgeführt wird und wie die Prozesse miteinander kommunizieren
- dargestellter Inhalt:

  a) **Knoten** in einem Rechnernetz (Prozessoren), dargestellt als flacher Kasten




  Stereotyp «`device`» : reales Gerät / Prozessor

  Stereotyp «`execution environment`» : Softwareumgebung, in der bestimmte "Artefakte" ausgeführt werden können

  Artefakt: Dokument, das als Programm geladen und ausgeführt oder interpretiert werden kann

b) **Kommunikationspfade**

c) **Verteilungsbeziehungen**

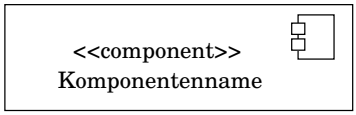
d) **Einsatzspezifikationen**

- Querbeziehungen zu anderen Diagrammtypen: Kompositionsstrukturdiagramm
- in UML 2.0 überarbeitet

## 2.6 Komponentendiagramm

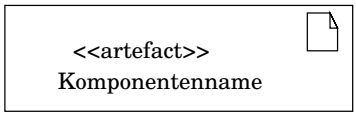
- zeigt Kommunikationsbeziehungen zwischen Komponenten
- Zweck: Bestandteile des Systems als Komponenten darstellen (i.w. nur andere Sicht auf Klassen- und Objektdiagramme)
- dargestellter Inhalt (Modellelemente):

  a) **Komponenten**: realisierte und benötigte Schnittstellen




  ggf. Angaben zur Realisierung durch andere Komponenten und eingesetzte Artefakte

  b) **Artefakte**: physisches Dokument (z.B. Datei), das eine Komponente beinhaltet ("instantiiert"), z.B. `.jar`-Datei




  c) **Abhängigkeiten**: organisatorische Abhängigkeiten
- unterstützt J2EE und .NET
- Querbeziehungen zu anderen Diagrammtypen:
- UML 2.0: erneuert in UML 2.0

## 2.7 Use-Case-Diagramm

- ist Funktionsmodell
- Zweck: Überblick über Systemverhalten aus Anwendersicht gewinnen
- dargestellter Inhalt: Anwendungsfälle; primär textuelle Notation
- Querbeziehungen zu anderen Diagrammtypen: ergänzende Verhaltensdiagramme
- UML 2.0: keine wesentliche Änderungen

## 2.8 Aktivitätsdiagramm

- erhebliche Änderungen gegenüber UML 1.x
  basiert jetzt auf erweiterten Petri-Netzen
- ist Funktionsmodell, teilweise auf der Ebene von Programmiersprachen, d.h. ausführbar, ferner Zustandsmodell
- Zweck: Verhalten detailliert spezifizieren
- dargestellter Inhalt:
  - Programmablaufkonstrukte (Verzweigungen usw.), u.a. ähnlich wie Nassi-Shneidermann-Diagramme
  - Plätze und Transitionen wie bei Petri-Netzen
  - Start-, End- u.a. Pseudozustände
- Querbeziehungen zu anderen Diagrammtypen: viele

→ eigenes Lehrmodul

## 2.9 Zustandsautomat bzw. Zustandsübergangsdiagramm

- ist Zustandsmodell
- Zweck: Überblick über Zustände eines Systems gewinnen
- dargestellter Inhalt: s. Lehrmodul ZÜD
- Querbeziehungen zu anderen Diagrammtypen: Klassendiagramme

- in UML 2.0 überarbeitet:
  - bessere Integration mit statischen Strukturelementen
  - Unterart: Protokollzustandsautomat

## 2.10 Interaktionsdiagramme

1. Sequenzdiagramm: stellt einen Ablauf (Sequenz von Operationsaufrufen) im System dar, an dem bestimmte Objekte beteiligt sind, die i.d.R. durch Beziehungen verbunden sind;
   betont zeitlichen Ablauf (Zeitachse)

   Ergänzungen / Neuerungen in der UML 2.0:
   - ist strukturierbar und zerlegbar (Hierarchie)
   - div. Konzepte, mit denen Kontrollflüsse und Nebenläufigkeit dargestellt werden kann
2. Kommunikationsdiagramm: stellt auch einen Ablauf dar
   betont die Kommunikationspartner / Kommunikationspfade
   hieß in UML 1.3 Kollaborationsdiagramm; keine wesentlichen konzeptuellen Änderungen gegenüber UML 1.3
3. Interaktionsübersichtsdiagramm
   - komplett neu in UML 2.0
   - verwaltet / integriert einzelne Verhaltensdiagramme, stellt übergeordnete Abläufe dar
   - benutzt hierzu gleiche Kontrollkonstrukte wie in Aktivitätsdiagrammen
4. Timing-Diagramm
   - komplett neu in UML 2.0
   - beschreibt Zeitverhalten von Objekten:

     pro Zustand eine horizontale Bahn

     Aktivitätslinie, die von links nach rechts durch die Bahnen läuft, Bahnwechsel durch Ereignisse

     zusätzlich Zeitangaben an den Linienabschnitten (veranschaulicht durch Länge der Linie)

# Literatur

[Bo91] Booch, G.: Object-oriented design with applications; The Benjamin/Cummings Publ. Comp.; 1991

[Bo94] Booch, G.: Object-oriented analysis and design with applications, 2nd edition; The Benjamin/Cummings Publ. Comp.; 1994

[BoRJ99] Booch, Grady; Rumbaugh, James; Jacobson, Ivar: The Unified Modeling Language User Guide; Addison Wesley; 1999

[CoY90] Coad, Peter; Yourdon, Edward: Object-oriented analysis; Yourdon Press, Prentice-Hall, Englewood, New-Jersey; 1990

[CoY91] Coad, Peter; Yourdon, Edward: Object-oriented analysis, 2nd edition; Yourdon Press, Prentice-Hall, Englewood, New-Jersey; 1991

[CoY91a] Coad, Peter; Yourdon, Edward: Object-oriented design; Yourdon Press, Prentice-Hall; 1991

[Ja+92a] Jacobson, I.; Christerson, M.; Jinsson, P.; Övergaard, G.: Object-oriented software engineering - a use case driven approach; Addsion Wesley; 1992

[Ru+91] Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.: Object-oriented modelling and design; Prentice-Hall, Englewood Cliffs, New-Jersey; 1991

[UML99] OMG Unified Modeling Language Specification (draft, Version 1.3 alpha R5, March 1999); OMG; 1999

# Index