Objektorientierte Modellierung mit der Unified Modeling Language

Michael Neuhold MSc.

Einführung

Inhalt

- Motivation
- Modellbegriff
- Was ist UML?
- Historische Entwicklung
- Diagrammarten im Überblick
- UML im Projektverlauf

Motivation

- “Die Hauptaufgabe der Softwareentwicklung besteht darin, Code zu produzieren”
- “Diagramme sind nichts weiter als Bilder, kein Kunde wird dafür bezahlen”
- “Was Benutzer erwarten, ist Software, die funktioniert”
- Wozu sollte man also modellieren?

Whiscy-Syndrom

“Why isn’t Sam coding yet?”

Motivation - Wozu modellieren?

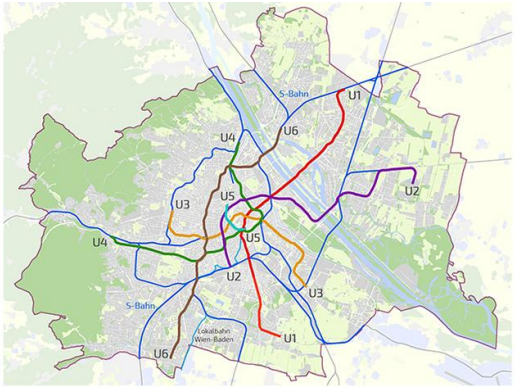
- System: entwerfen, verstehen, visualisieren, dokumentieren, simulieren, überprüfen
- Modellierung ist der Prozess, bei dem ein Modell eines Systems erstellt wird
- Modell ist ein konkretes oder gedankliches
  - Abbild eines vorhandenen Gebilde
  - Vorbild für ein zu schaffendes Gebilde
- Modell sind **Abbildungen** und **Konstruktion** der Realität

Modellbegriff - Abbildungsmerkmal

- Ein **Modell** stellt eine **Abstraktion** eines **Realitätsausschnitts** dar
  - Um Informationen verständlicher darzustellen
  - Um essentielle Aspekte aufzuzeigen
  - Zur Kommunikation (Projektmitarbeiter, Kunden)
  - Um komplexe Architekturen darstellen zu können
- Ein **Diagramm** ist die **grafische Repräsentation** eines Modells bzw. eines Modellausschnitts

Modellbegriff - Verkürzungsmerkmal

- Modelle erfassen meist nicht alle Individuen und Attribute des Originals
  - Bewusste Abstraktion / Verkürzung
- Es wird nur das modelliert, was den Modellschaffenden **wichtig/nützlich/notwendig** erscheint
- Das Modell kann Individuen und Attribute enthalten, die keine Entsprechung im Original haben



Was ist die Unified Modeling Language?

Die Unified Modeling Language ist eine visuelle Sprache zur Spezifikation, Konstruktion und Dokumentation der Artefakte von Software-Systemen.

- Standardisierte, ausdrucksstarke, grafische Modellierungssprache, die auf objektorientierten Konzepten basiert
  - Für den Einsatz in verschiedenen Anwendungsbereichen
  - Zur Formulierung unterschiedlicher Modelltypen
  - Zum Austausch von Modellen

Was bedeutet der Begriff >> Unified <<?

- Unterstützung des gesamten Entwicklungsprozesses
- Flexibilität in Bezug auf Vorgehensmodelle
- Unabhängigkeit von Entwicklungswerkzeugen und -plattformen, sowie Programmiersprachen
- Einsetzbarkeit für verschiedenste Anwendungsbereiche

Was ist die UML nicht?

- Keine Methode oder Vorgehensmodell
  - UML kann in verschiedenen Vorgehensmodellen eingesetzt werden, ist selbst aber kein Softwareentwicklungsprozess
- Kein Entwicklungswerkzeug
  - Werkzeughersteller implementieren UML
- Keine Programmiersprache im herkömmlichen Sinn

Für den Einsatz von UML gilt:

“Wer einen Hammer hat, ist noch lang kein Architekt”

Wer benutzt die UML?

👉Projektmanager

👉Kunde

👉Produktmanager

👉Softwareentwickler

👉Hardwareentwickler

Wie kann die UML angewendet werden?

- UML als **Skizze**
  - Informelle und unvollständige Diagramme (Handskizze auf Whiteboards, “Bierdeckel-Entwurf”)
- UML als **Blaupause**
  - Detaillierte Entwurfsdiagramme für
    - Forward Engineering: Code-Generierung aus UML Diagrammen
    - Backward Engineering: vorhandenen Code mit UML darstellen
- UML als **ausführbare Programme**
  - Vollständig ausführbare Spezifikation eines Softwaresystems in UML
    - Erstellung von ausführbaren Modellen (Model Driven Architecture/Development - Code durch Modelle ablösen)

Agile Ansätze betonen UML als Skizze!

Historische Entwicklung (Die Wurzeln von UML)

- Ansätze der ersten Generation
  - **Booch-Methode (G.Booch)**
    - Starker Programmiersprachenbezug (Ada)
    - Modellierung von Echtzeitsystemen
  - **OOSE - Object Oriented Software Engineering (I. Jacobson)**
    - Stark in der Beschreibung von Anforderungen
    - Use Case - orientiert
  - **OMT - Object Modeling Technique (J. Rumbaugh)**
    - Starker Bezug zur Datenmodellierung
    - Erweiterte Entity-Relationship-Diagramme

Historische Entwicklung

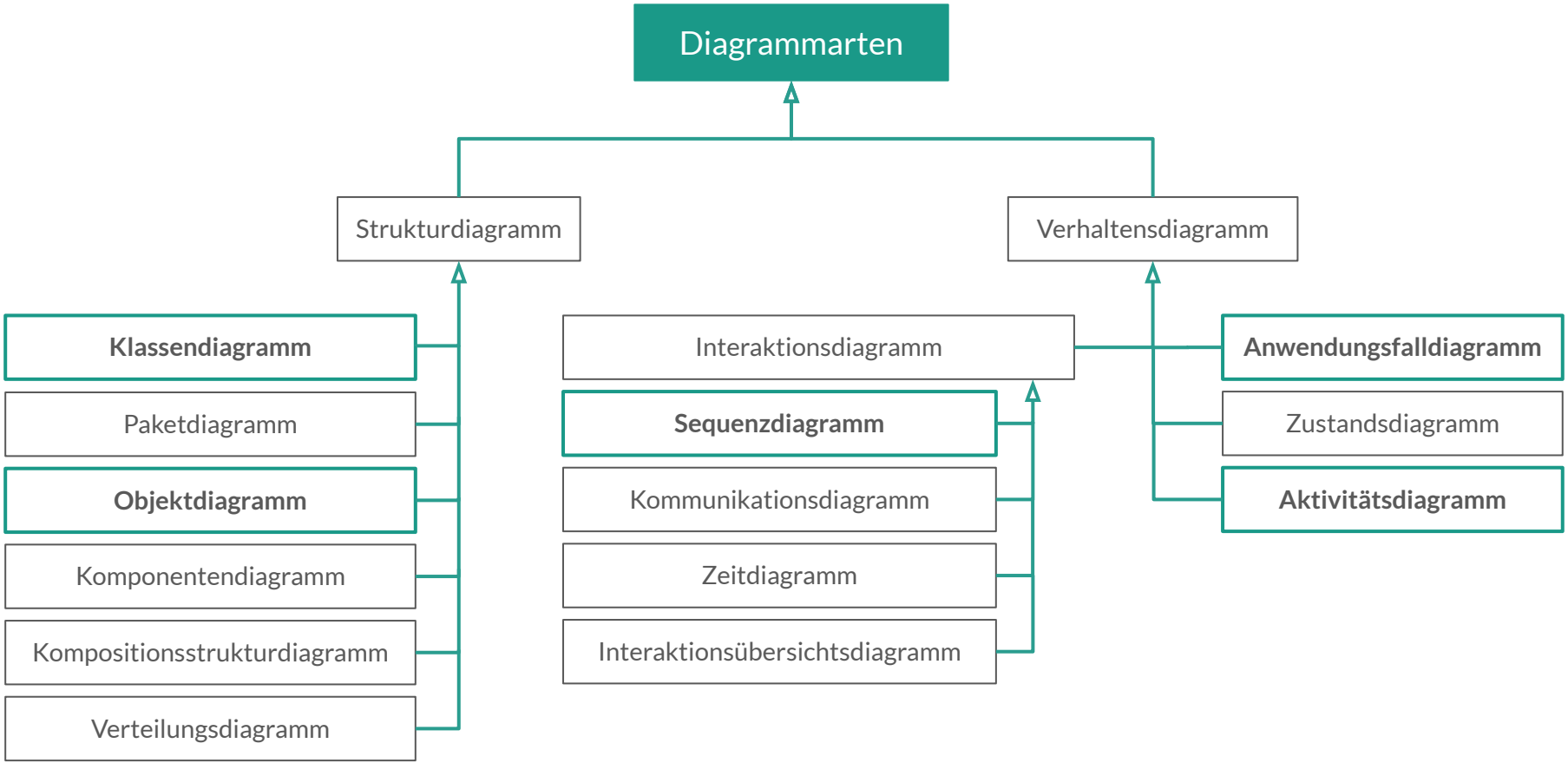
2015	Schärfung der Semantik von Modellelementen	**UML 2.5**
2015		
2005	Fundamentale Neuerungen	**UML 2.x**
2004		
2003	Aktionsmodell	**UML 1.5**
2001	Komponenten, Profile	**UML 1.4**
1999	XMI	**UML 1.3**
1998	OCL	**UML 1.1**
1995		**UML 1**

→ **Model Driven Architecture / Development**

Entstand durch Unified Method 0.8

- **Diverse Methoden**
- **OOA/D Booch**
- **OMT Rumbaugh**
- **OOSE Jacobson**

Diagrammarten im Überblick


Diagrammarten
Strukturdiagramm
Verhaltensdiagramm
Klassendiagramm
Paketdiagramm
Objektdiagramm
Komponentendiagramm
Kompositionsstrukturdiagramm
Verteilungsdiagramm
Interaktionsdiagramm
Sequenzdiagramm
Kommunikationsdiagramm
Zeitdiagramm
Interaktionsübersichtsdiagramm
Anwendungsfalldiagramm
Zustandsdiagramm
Aktivitätsdiagramm

Diagrammarten im Überblick - Strukturdiagramme

- **Klassendiagramm**
  - Aus welchen Klassen und Schnittstellen besteht ein System und wie stehen diese untereinander in Beziehung?
  - Beschreibt den strukturellen Aspekt eines Systems zur Design-Zeit
  - Normalerweise unverzichtbar!

- **Objektdiagramm**
  - Welche innere Struktur besitzt ein System zu einem bestimmten Zeitpunkt zur Laufzeit?
  - Zeigt Momentaufnahme der Objekte und deren Beziehung, die zur Laufzeit in einem System herrschen

Diagrammarten im Überblick - Verhaltensdiagramme

- **Anwendungsfalldiagramm**
  - Was leistet das System für die Umwelt (Nachbarsysteme, Stakeholder)?
  - Beschreibt die Funktionalität des zu entwickelten Systems aus Benutzersicht
  - Implementierungsunabhängige Sicht (Use-Cases)

- **Aktivitätsdiagramm**
  - Wie läuft ein bestimmter fluss-orientierter Prozess oder ein Algorithmus ab?
  - Zeigt Abläufe mit Bedingungen, Schleifen, Verzweigungen (prozedurale Verarbeitungsaspekte).
  - Darstellung von Daten- und Kontrollfluss

Diagrammarten im Überblick - Verhaltensdiagramme

- **Sequenzdiagramm**
  - Wer tauscht mit wem welche Informationen in welcher Reihenfolge aus?
  - Beschreibt komplexe Interaktionen zwischen Objekten in bestimmten Rollen, um eine konkrete Aufgabe zu erfüllen.
  - Fokus: Zeit als eigenen Dimension

- **Zustandsdiagramm**
  - Welche Zustände kann ein Objekt, eine Schnittstelle, ein Use-Case, … bei welchen Ereignissen annehmen?
  - Beschreibt das erlaubte Verhalten von Modellelementen in Form von Zuständen und Zustandsübergängen

Diagrammarten im Überblick

- Meist werden nur **ausgewählte Diagrammarten** zur Modellierung verwendet - abhängig von
  - Modellierungszweck (Analyse-, Entwurfs-, Implementierungssicht)
  - Charakteristika des Problembereichs
  - “Vorlieben” des Modellierers für bestimmten Formalismus
  - …

- Flexibilität bei der Zuordnung von Diagrammarten zu Stufen des Entwicklungsprozesses
  - Manche Diagrammarten “eher” geeignet für frühe Phasen (zb Anwendungsfalldiagramm)
  - Manche Diagrammarten “eher” geeignet für spätere Phasen (zb Verteilungsdiagramm)
  - Schrittweise Verfeinerung, unterschiedlicher Detaillierungsgrad