Struktur der UML-Spezifikationen

Udo Kelter

28.06.2016

Zusammenfassung dieses Lehrmoduls

Dieses Lehrmodul liefert eine Einführung die Struktur der Version 2.5 der UML-Spezifikationen, in dem die Modellelemente der UML definiert werden. Hierbei werden intensiv Klassendiagramme als Metamodelle eingesetzt. Klassendiagramme werden im UML-Dokument selbstreferentiell als Metamodelle benutzt.

Vorausgesetzte Lehrmodule:

obligatorisch:
- Die Unified Modelling Language (UML) Version 2
- Modellgetriebene Software-Entwicklung
- Metadaten

Stoffumfang in Vorlesungsdoppelstunden: 0.4

Inhaltsverzeichnis

1 UML-Spezifikationen (Version 2.5, 2015)

Hauptdokumente:

[UML2.5] Unified Modeling Language, Version 2.5
http://www.omg.org/spec/UML/2.5/PDF
beschreibt für alle Modellelementtypen:

- deren konzeptuelle Repräsentation selbstreferentiell mit Klassendiagrammen (Metamodelle! Ebene M1)
- ggf. deren graphische Darstellung
- ggf. deren Bedeutung, vor allem durch Prosa

erhebliche Umstellung gegenüber der vorherigen Version 2.4.1 (2011): Zusammenlegung der Superstructure- und Infrastructure-Spezifikation

[MOF2.5] Meta Object Facility Version 2.5
http://www.omg.org/spec/MOF/2.5/PDF
spezifiziert:

- durch Import aus der UML2.5: Grundbegriffe wie Pakete, Klassen usw., insb. alle Modellelemente, die für die UML-Metamodelle gebraucht werden
  werden hier als Meta-Metamodelle benutzt
- technische Aspekte des Modell-Managements: Identifiers, Reflection, ...

[OCL2.3.1] Object Constraint Language
http://www.omg.org/spec/OCL/2.3.1
Sprache, mit der Bedingungen an Objekte in Modellen formuliert werden können

[XMI2.5] XML Metadata Interchange
http://www.omg.org/spec/XMI/2.5
Abbildung von MOF in XMI

2 Ziele der UML-Spezifikation

Struktur der UML-Spezifikation ist kompliziert, weil viele überlappende Dinge zugleich spezifiziert werden:

“repository semantics”:
die UML-Spezifikation spezifiziert in erster Linie wesentliche Aspekte von *Modellierungswerkzeugen* bzw. allg. Systemen, die Modelle verwalten oder handhaben; hier zu spezifizieren:

- konzeptueller Inhalt von Modellen diverser Typen
- teilweise die graphische Darstellung von Modellelementen

“run-time semantics”: UML mutiert immer mehr zu einer graphischen Programmiersprache

- → UML-Spezifikation spezifiziert eine vage operationale Semantik *mancher* Modellelemente, also wie das modellierte System sich verhalten soll;
- Vorsicht: Systemwechsel und Wechsel der Bedeutungsebenen!
- kann man als Beschreibung ansehen, wie Modelle in Code übersetzt oder interpretiert werden sollen.
  Beispiele von Übersetzungen:
  - Übersetzung von Aktivitätsdiagrammen in Quellcode
  - Generierung von Klassenrümpfen

Profile und domain-specific languages (DSL): UML möchte anpaßbar / erweiterbar sein:

- vorhandenen Modelle einschränken und ggf. versehen mit speziellen Bedeutungen
- Erweiterung um neue Modellelemente und neue Diagrammtypen

Benachbarte Probleme:

- Transformation von Modellen in Modelle (z.B. beim MDA-Ansatz)
- Austausch von Modellen zwischen verschiedenen Werkzeugen, Definition von Transportdateiformaten

bedingen alle, den konzeptuellen Inhalt von Modellen und teilweise das Layout zwischen Werkzeugen / Werkzeugfunktionen austauschen zu können

3 Diagramme vs. Modelle

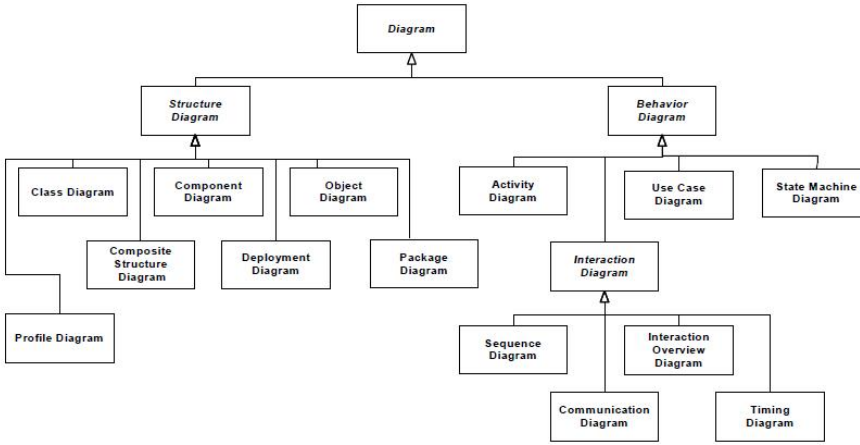
- **Modell** = Gesamtdarstellung des modellierten Systems;
- **Diagramm** = Auswahl ("Zeichenfläche"), die bestimmte Modellelemente und deren Verbindungen (graphisch) zeigt
- gleiches Modellelement kann in mehreren Diagrammen vorkommen (als Spiegelung),
  sogar in unterschiedlichen Darstellungen ("Sichten")
- Diagramme sind für Modelltransformationen irrelevant

Frame Names Annex A: Diagrams: "UML diagrams may have the following kinds of frame names as part of the heading:"

- activity
- class
- component
- deployment
- interaction
- package
- state machine
- use case

Figure A.5 [UML2.5] taxonomy of structure and behavior diagrams:

Diagrammtypen in der Praxis / beim Editieren:



- Diagrammtypen: Festlegung pro Diagrammtyp, welche Arten von Modellelementen darin erlaubt sind
- ausschließliche Verwendung typisierter Diagramme
- Editoren kontrollieren Verwendung der Modellelemente

Diagrammtypen in der UML:

- definiert 13 Diagrammtypen vor (s. Figure A.5), bleibt aber offen für benutzerdefinierte Diagrammtypen
- daher generell keine Restriktionen für die erlaubten Modellelemente in bestimmten Diagrammtypen
- Diagramme sind keine Modellelemente!

4 Inhaltlicher Aufbau von [UML2.5]

orientiert an Modellelementen, gruppiert nach Themenbereichen, (nicht Diagrammtypen); Auszug aus der Gliederung, jeweils Kapitelnummer- und Bezeichnung und Liste der Themen:

7 Common Structure

Root, Templates, Namespaces, Types and Multiplicity, Constraints, Dependencies

8 Values
Literals, Expressions, Time, Intervals

9 Classification
Classifiers, Classifier Templates, Features, Properties, Operations, Generalization Sets, Instances

10 Simple Classifiers
DataTypes, Signals Interfaces

11 Structured Classifiers
Structured Classifiers, Encapsulated Classifiers, Classes, Associations, Components, Collaborations

12 Packages
Packages, Profiles

13 Common Behavior
Behaviors, Events

14 StateMachines
Behavior StateMachines, StateMachine Redefinition, Protocol-StateMachines

15 Activities
Activities, Control Nodes, Object Nodes, Executable Nodes, Activity Groups

16 Actions
Actions, Invocation Actions, Object Actions, Link End Data, Link Actions, Link Object Actions, Structural Feature Actions, Variable Actions, Accept Event Actions, Structured Actions, Expansion Regions, Other Actions

17 Interactions
Interactions, Lifelines, Messages, Occurrences, Fragments, Interaction Uses, Sequence Diagrams, Communication Diagrams, Interaction Overview Diagrams, Timing Diagrams,

18 UseCases
Use Cases

19 Deployments
Deployments, Artifacts, Nodes

20 InformationFlows
Information Flows

21 Primitive Types

22 Standard Profile
Model, Standard Stereotypes

Annex A: Diagrams

Annex B: UML Diagram Interchange

Annex C: Keywords

Annex D: Tabular Notation for Sequence Diagrams

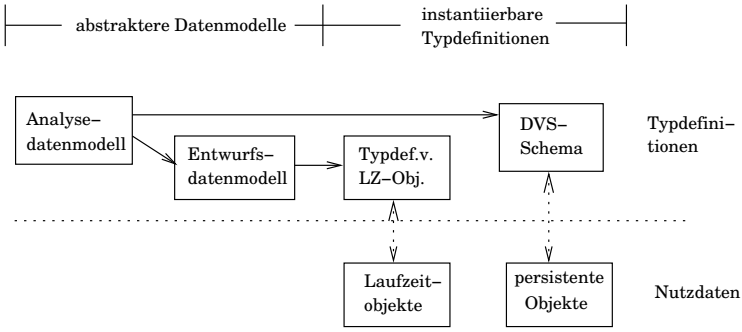
Annex E: XMI Serialization and Schema

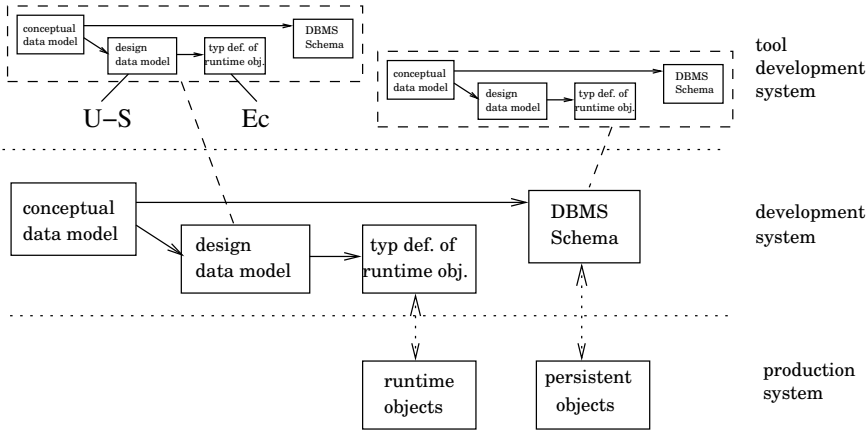
Unterabschnitte (insg. 61) "Abstract Syntax": Konzeptueller Inhalt von Modellelementen wird durch Klassendiagramme (= Metamodelle) dargestellt → Bezug auf MOF und indirekt auf die Anteile von [UML2.5], die Klassendiagramme beschreiben

→ definierte Modellelementtypen sind sowohl für M1-Modelle als tw. auch für M2-Modelle (Metamodelle) gedacht

5 Phasenzugehörigkeit der Metamodelle

Wiederholung aus LM Metamodelle (ST1):





Die UML-Metamodelle sind *keine Analysemodelle, sondern gehören zur (Detail-) Designphase!*

- Assoziationen sind immer gerichtet

  "ungerichtete": s. 6.4.2 Diagram format
- die Metamodelle enthalten sehr viele redundante Daten, z.B. derived unions
- sind teilweise entgegen der Intuition gestaltet, um die Transformation in Programm-Quellcode zu erleichtern.

  Beispiel: Navigierbarkeit von Assoziationen

