Teil IV

Anhang

Anhang A

Die UML Referenz

In diesem Anhang stellen wir die grafische Notation der einzelnen Diagramme noch einmal zusammen. Dabei bedienen wir uns beispielhafter Diagramme, in denen die grafischen Elemente in ihrer Originalgestalt auftreten, und kurzer Kommentare. Wir folgen hier der von uns bevorzugten Übersetzung, geben aber die englischen Originalnamen ebenfalls an. Wir gehen in der gleichen Reihenfolge wie im Teil 1 des Buches vor.

Zu beachten ist ferner, dass die in den Zeichnungen verwendeten Klammern und andere Begrenzer wie `:` oder `/` in dem tatsächlichen Text so zu stehen haben, also terminale Zeichen sind und keine weitere Bedeutung tragen. Die Namen bezeichnen hier immer das Modellelement und sind im normalen Zeichensatz notiert, wohingegen die Verweise auf Kommentare kursiv gedruckt sind.

Im Gegensatz dazu erfolgt die Angabe der genauen Syntax von einigen Beschriftungen in der üblichen EBNF Notation, in der Terminale in " eingeschlossen werden, eckige Klammern optionale Elemente und geschweifte Klammern beliebig häufige Wiederholungen kennzeichnen.

Eine Fülle von Querverweisen macht deutlich, dass die gleichen UML-Modellelemente in verschiedenen Diagrammen vorkommen. Konsequenterweise ist diese Zusammenfassung der UML zusätzlich als Hypertext auf der CD zu finden. Der Modellierer kann auch noch freier Elemente verschiedener Diagramme kombinieren.

In diesem Teil kommen auch einige sonst im Buch nicht erwähnte Elemente aus der Originalbeschreibung der UML vor. Einige Spezialnotationen lassen wir auch hier unter den Tisch fallen. Wir listen ferner die wichtigsten vordefinierten Stereotypen und Einschränkungen auf.

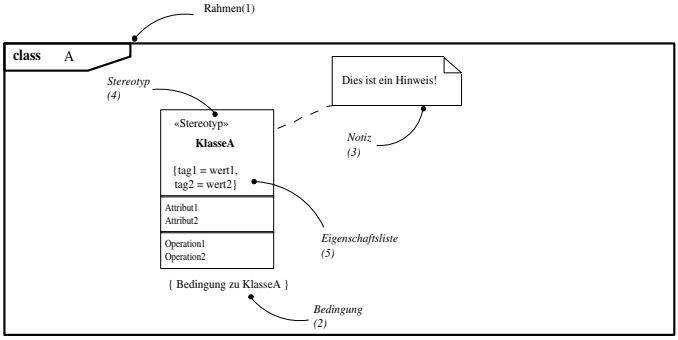

Abbildung A.1. Syntax der gemeinsamen Sprachelemente - hier am Bespiel einer Klasse

A.1 Gemeinsame Elemente aller Diagramme

Beschreibende Elemente können in allen Diagrammen auftreten. Sie spezifizieren ein Modellelement in einem Diagramm präziser.

All:1 Rahmen
Abbildung: A.1
Engl.: Frame
Regeln:
- Jedes Diagramm kann von einem Rahmen umrandet werden. In diesem Rahmen steht am linken oberen Rand die Art (in Fettdruck) und der Name des dargestellten Modells.
- Es gibt folgende Arten
  - use case **uc**
  - activity **act**
  - class **class**
  - interaction **sd**
  - state machine **stm**
  - component **cmp**
  - package **pkg**

All:2 Bedingung, Einschränkung, Constraint
Abbildung: A.1
Engl.: constraint
Regeln:
- Einzelne Modellelemente oder auch Gruppen können mit einer Bedingung versehen werden.
- Die Formulierung der Bedingung kann in OCL erfolgen, ist aber nicht festgelegt.
- Sie wird in geschweifte Klammern eingeschlossen.

- Sie kann in einer Notiz stehen.
- Gestrichelte Linien führen zu den eingeschränkten Elementen.

siehe auch: Notiz (3)
Beschreibung: Seite 73

All:3 Notiz, Kommentar
Abbildung: A.1
Engl.: note
Regeln:
- Eine Notiz ist ein Kommentar zu einem Diagrammelement.

All:4 Stereotyp
Abbildung: A.1
Engl.: stereotype
Regeln:
- Einzelne Modellelemente können durch Stereotypen klassifiziert werden.
- Syntax : ≪Stereotyp≫

Beschreibung: Seite 279

All:5 Eigenschaftsliste, tagged values
Abbildung: A.1
Engl.: tagged values
Regeln:
- Eigenschaftslisten spezifizieren weitere Eigenschaften der Modellelemente.
- Eine Eigenschaft ist ein Schlüssel - Wert Paar.
- Syntax : `tag = value`
- Fehlende Wertangabe entspricht `true`

Beschreibung: Seite 278

A.2 Use-Case-Diagramm

Anwendungsfalldiagramm

Use:1 Akteur
Abbildung: A.2
Engl.: actor
Regeln:
- Jeder Akteur hat einen Namen.
- Ein Akteur kann eine Klasse sein.

siehe auch: Use-Case (2), Assoziation (4), Klasse (1)
Beschreibung: Seite 18

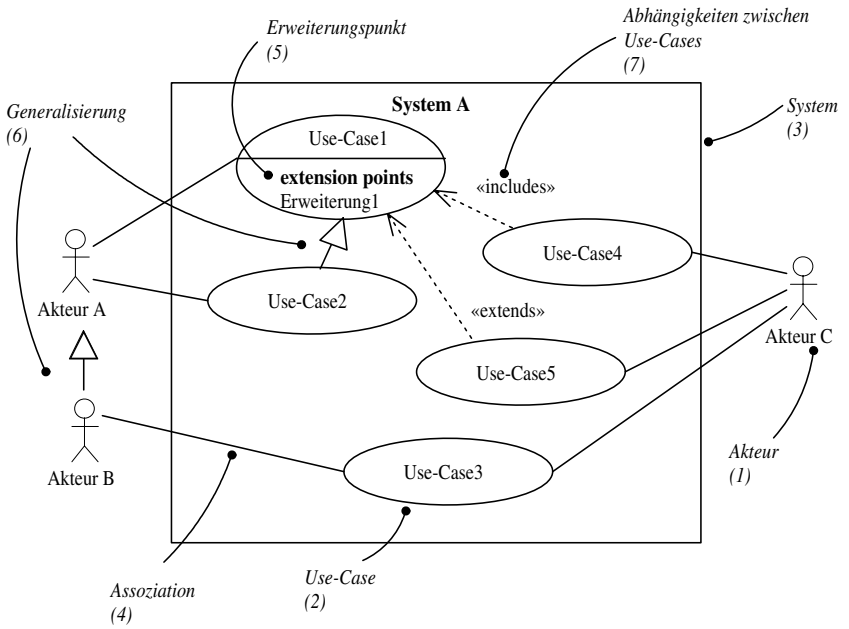

Abbildung A.2. Syntax der Use-Case-Diagramme

Use:2 Use-Case, Anwendungsfall
Abbildung: A.2
Engl.: use case
Regeln:
- Jeder Use-Case hat einen Namen.
- Er kann eine Liste von Erweiterungsstellen enthalten.

siehe auch: Akteur (1), Assoziation (4), Use-Case Abhängigkeit (7)
Beschreibung: Seite 18

Use:3 System
Abbildung: A.2
Engl.: system
Regeln:
- Ein System enthält beliebig viele Use-Cases.
- Ein System kann einen Namen haben.

siehe auch: Use-Case (2)
Beschreibung: Seite 18

Use:4 Assoziation
Abbildung: A.2

Engl.: association
Regeln:
• Assoziationen verbinden Akteure mit Use-Cases.
siehe auch: Use-Case (2), Akteur (1)
Andere Diagramme: Klassendiagramm (Kla:5)
Beschreibung: Seite 18

Use:5 Erweiterungspunkt
Abbildung: A.2
Engl.: extension point
Regeln:
• Jeder Anwendungsfall kann eine Liste von Erweiterungspunkten enthalten.
siehe auch: Use-Case (2), ≪extends≫ -Beziehung (7)
Beschreibung: Seite 23

Use:6 Generalisierung
Abbildung: A.2
Engl.: generalization
Regeln:
• Generalisierung von Use-Cases und Akteuren ist möglich.
siehe auch: Use-Case (2), Akteur (1)
Andere Diagramme: Klassendiagramm (Kla:12)
Beschreibung: Seite 21

Use:7 Use-Case Abhängigkeit
Abbildung: A.2
Engl.: use case relationships
Regeln:
• Use-Cases können mit Abhängigkeitspfeilen miteinandern verbunden werden.
• Es gibt zwei Arten von Abhängigkeiten, die mit entsprechenden Stereotypen gekennzeichnet werden:
  – ≪extends≫ für die Erweiterung eines Use-Cases durch einen anderen
  – ≪includes≫ für die Benutzung eines Use-Cases durch einen anderen
siehe auch: Use-Case (2).
Andere Diagramme: Klassendiagramm (Kla:18), Komponentendiagramm (Kmp:6), Installationsdiagramm (Ins:3), Paketdiagramm (Pak:2)
Beschreibung: Seite 20

A.3 Aktivitätsdiagramm

activity diagram

Aktivitätsdiagramme sind gut geeignet, Vorgänge und Abläufe zu beschreiben. Sie können ferner zur Veranschaulichung von Algorithmen herangezogen werden.

Akt:1 Aktivität
Abbildung: A.3, A.4
Engl.: activity
Regeln:
- Eine Aktivität setzt sich aus mehreren Aktionen zusammen.
- Ein Aktivitätsdiagramme beschreibt eine oder mehrere Aktivitäten.
- Eine Aktivität kann parametrisiert sein.

siehe auch: Aktion 2
Andere Diagramme: Zustandsdiagramm (Zus:2)
Beschreibung: Seite 28

Akt:2 Aktion
Abbildung: A.3
Engl.: action
Regeln:
- Aktionen werden durch Kanten miteinander verbunden.
- Es kann mehrere Eingangs- und Ausgangskanten geben.
- Eine Aktion wird ausgeführt, wenn alle Eingangskanten feuern.
- Ausgangskanten sollten durch disjunkte Wächterbedingungen unterscheidbar sein.

siehe auch: Verzweigung (9) , Kante (4) , Wächterbedingung (7)
Andere Diagramme: Zustandsdiagramm (Zus:2)
Beschreibung: Seite 28

Akt:3 Objekt
Abbildung: A.3
Engl.: ObjectNode
Regeln:
- Der Objektname gibt den Typ, die Klasse an.
- Für ein Objekt kann der aktuelle Zustand angegeben werden.
- Ein Objektname, welcher ein Signal beschreibt, wird in ein pfeilförmiges Rechteck eingeschlossen.

siehe auch: Pin (11); Objektfluss (12)
Andere Diagramme: Sequenzdiagramm (Seq:1), Kommunikationsdiagramm (Kmk:1, Klassendiagramm (Kla:15), Installationsdiagramm (Ins:1)
Beschreibung: Seite 33

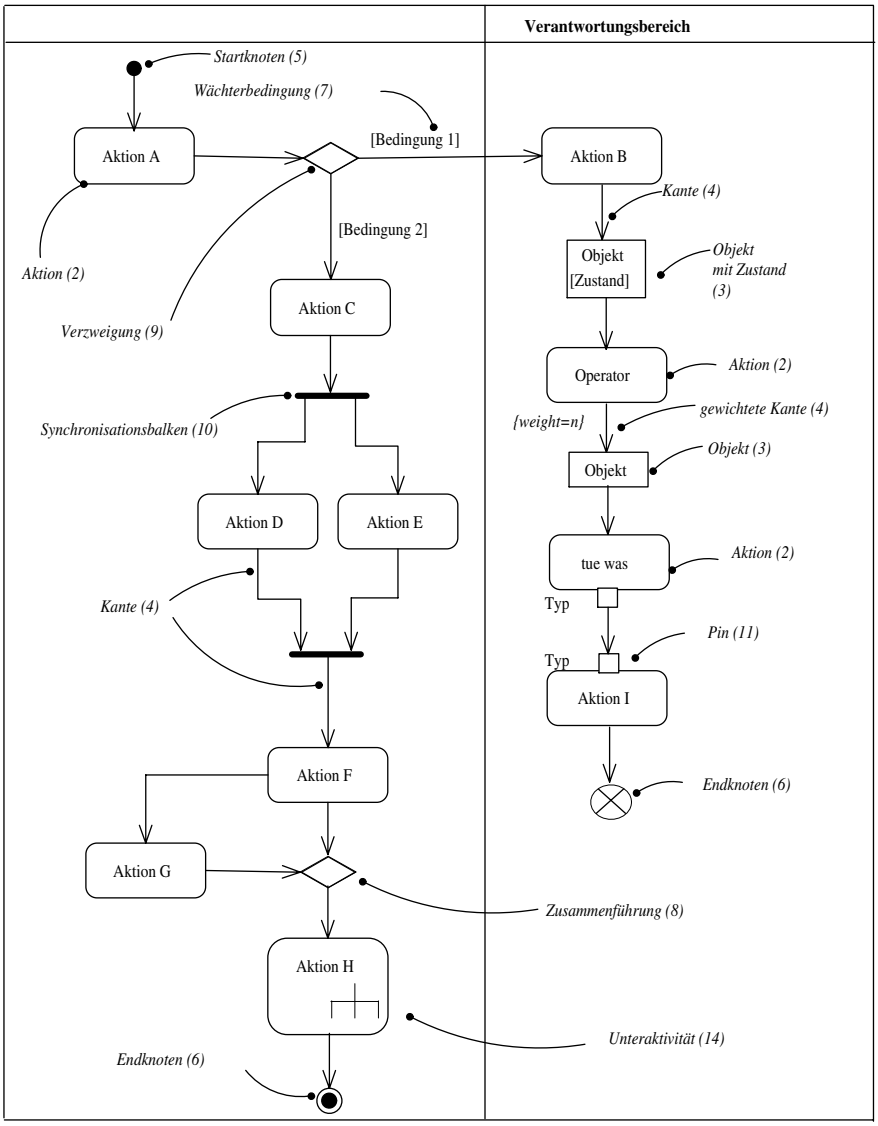

Abbildung A.3. Syntax der Aktivitätsdiagramme 1

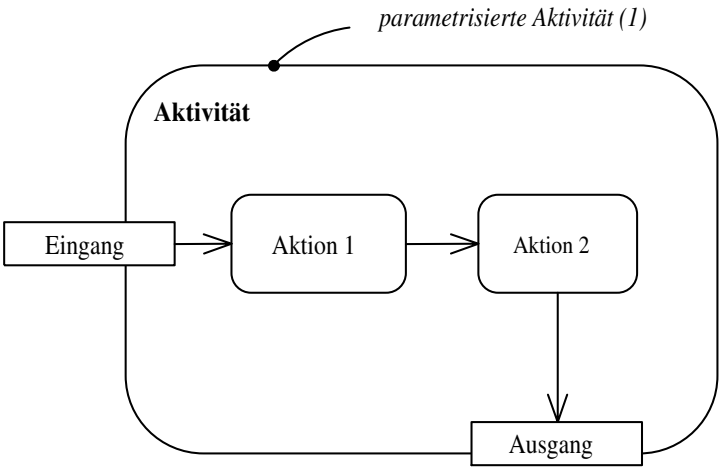

Abbildung A.4. Syntax der Aktivitätsdiagramme 2

Akt:4 Kante, Pfeil
Abbildung: A.3
Engl.: ActivityEdge
Regeln:
- Eine Kante verbindet Aktionen, Objekte oder Aktivitäten.
- Sie kann mit einer Wächterbedingung versehen werden.
- Die Anzahl der transportierten Objekte kann als Gewicht angegeben werden. {`weight=n`}

siehe auch: Aktion (2), Aktivität (1) , Wächterbedingung (7)
Andere Diagramme: Zustandsdiagramm (Zus:3)
Beschreibung: Seite 30

Akt:5 Startknoten
Abbildung: A.3
Engl.: InitialNode
Regeln:
- Jedes Diagramm kann mehrere Startzustände haben

siehe auch: Endknoten (6)
Andere Diagramme: Zustandsdiagramm (Zus:5)
Beschreibung: Seite 29

Akt:6 Endknoten
Abbildung: A.3
Engl.: ActivityFinal, FlowFinal
Regeln:
- Ein Endknoten mit dem geviertelten Kreis bezeichnet das Ende eines Kontrollflusses, einer mit dem Spiegeleisymbol das der ganzen Aktivität.

siehe auch: Startknoten (5)
Andere Diagramme: Zustandsdiagramm (Zus:6)
Beschreibung: Seite 29

Akt:7 Wächterbedingung
Abbildung: A.3
Engl.: guard condition
Regeln:
- Jede Kante kann mit einer Wächterbedingung versehen werden.
- Die Formulierung der Bedingung kann in OCL erfolgen, ist aber nicht festgelegt.

siehe auch: Aktion (2), Kante (4)
Andere Diagramme: Zustandsdiagramm (Zus:3), Sequenzdiagramm (Seq:2)
Beschreibung: Seite 30

Akt:8 Zusammenführung
Abbildung: A.3
Engl.: MergeNode
Regeln:
- Es gibt mehrere eingehende und eine abgehende Kante.

siehe auch: Verzweigung (9)
Beschreibung: Seite 31

Akt:9 Verzweigung
Abbildung: A.3
Engl.: DecisionNode
Regeln:
- Es gibt eine eingehende und zwei oder mehr abgehende Kanten.
- Alle abgehenden Kanten müssen mit einer unterschiedlichen Wächterbedingung versehen werden.
- `else` fasst alle nicht explizit aufgeführten Bedingungen zusammen.
- Verzweigung und Zusammenführung können zu einem Symbol verschmolzen werden.

siehe auch: Aktion (2) , Kante (4) , Wächterbedingung (7), Zusammenführung (8)
Beschreibung: Seite 31

Akt:10 Synchronisationsbalken
Abbildung: A.3
Engl.: ForkNode, JoinNode
Regeln:
- Ein Synchronisationsbalken kann entweder nur eine eingehende und beliebig viele ausgehende Kanten oder beliebig viele eingehende und nur eine ausgehende Kante haben.
- Er kann mit einer Bedingung versehen werden.

siehe auch: Kante (4)
Andere Diagramme: Zustandsdiagramm (Zus:14)
Beschreibung: Seite 32

Akt:11 Pin
Abbildung: A.3
Engl.: pin
Regeln:
- Der Objektfluss kann auch durch Pins, die getypte Kanäle in Form von kleinen Quadraten bezeichnen, geregelt werden.
- Eingangs- und Ausgangspins können durch Pfeile in den Quadraten gekennzeichnet werden.
- Ein Datenstrom wird durch ausgefüllte Quadrate spezifiziert.
- Mehrere Pins lassen sich zu einem Parametersatz gruppieren.

siehe auch: Objekt (3), Objektfluss (12)
Beschreibung: Seite 35

Akt:12 Objektfluss
Abbildung: A.3
Engl.: ObjectFlow
Regeln:
- Objektfluss wird durch Kanten zwischen Pins oder Objekten beschrieben.
- Bei einem Datenstrom ist die Kantenspitze ausgefüllt.
- Objekte können mit dem Stereotyp ≪centralBuffer≫ als Puffer oder mit ≪dataStore≫ als Speicher ausgezeichnet werden.

siehe auch: Objekt (3), Pin (11)
Beschreibung: Seite 35

Akt:13 Verantwortlichkeitsbereich, (Schwimmbahn)
Abbildung: A.3
Engl.: ActivityPartition
Regeln:
- Eine Schwimmbahn gruppiert Aktivitäten und Kanten.
- Sie sollte einen Namen haben.

- Geschachtelte Verantwortlichkeitsbereiche oder Abgrenzungen in zwei Dimensionen sind möglich.

siehe auch: Aktivität (1), Kante (4)
Beschreibung: Seite 32

Akt:14 Unteraktivität
Abbildung: A.3
Engl.: subactivity
Regeln:
- Eine Unteraktivität ist eine Referenz auf ein anderes Aktivitätsdiagramm.
- Sie wird in einer Aktion ausgeführt.
- Ihre Details können verborgen werden, dann ist als Indikator das Gabelsymbol zu notieren.

siehe auch: Aktion (2), Aktivität (1)
Andere Diagramme: Zustandsdiagramm (Zus:7), Sequenzdiagramm (Seq:9)

Akt:15 Strukturierte Aktivität
Abbildung: A.5
Engl.: StructuredActivityNode
Regeln:
- Eine strukturierte Aktivität ist in einem Kontrollfluss oder Objektfluss direkt aufrufbar.
- Sie kann mit Objekten oder Pins parametrisiert werden.
- Sie dient dazu Knoten mit spezieller Semantik einzuführen.

siehe auch: Fallunterscheidunsaktivität (16), Schleifenaktivität (17), Mengenaktivität (18)

Akt:16 Fallunterscheidunsaktivität
Abbildung: A.5
Engl.: ConditionalNode
Regeln:
- Im `if` und den optionalen `else if` Bereichen stehen Aktionen für Bedingungen. In den `then` und `else` Bereichen stehen die Aktionen, die ausgeführt werden, falls die entsprechende Bedingung erfüllt ist.

siehe auch: Strukturierte Aktivität (15), Schleifenaktivität (17)
Beschreibung: Seite 38

Akt:17 Schleifenaktivität
Abbildung: A.5

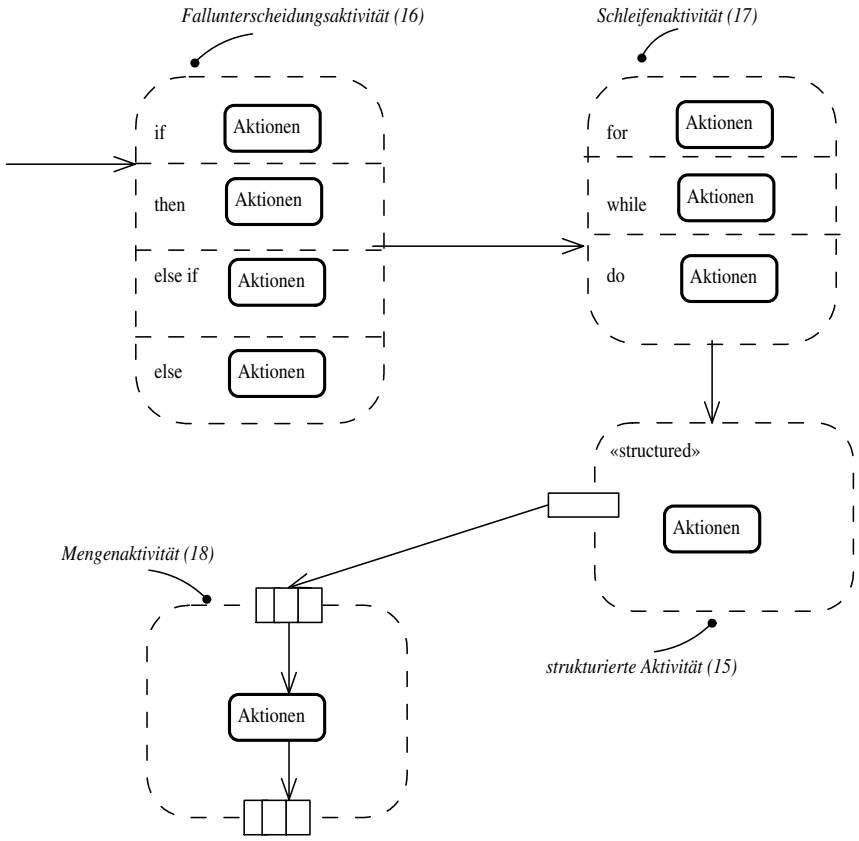

Abbildung A.5. Syntax der Aktivitätsdiagramme 3

Engl.: LoopNode
Regeln:
- Im `for` Bereich steht die Initialisierungsaktion, im `while` Bereich die Schleifenbedingung, im `do` Bereich der Schleifenrumpf

siehe auch: Strukt. Aktivität (15), Fallunterscheidunsaktivität (16)
Beschreibung: Seite 38

Akt:18 Mengenaktivität
Abbildung: A.5
Engl.: ExpansionRegion
Regeln:
- Die Aktionen werden für alle Elemente der durch die Eingangspins eingebrachten Mengen oder Kollektionen ausgeführt.

siehe auch: Strukturierte Aktivität (15)

A.4 Klassendiagramm

class diagram

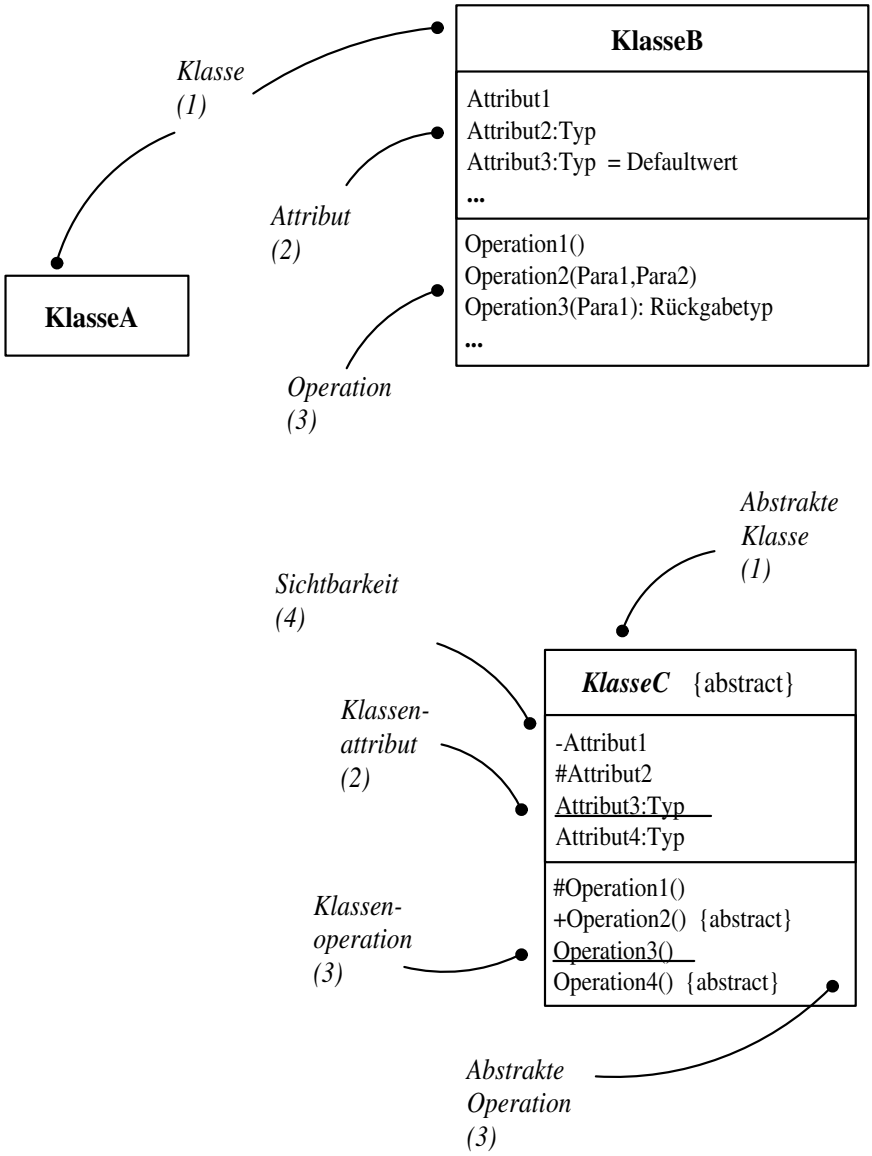

Abbildung A.6. Syntax der Klassendiagramme 1

Kla:1 Klasse

Abbildung: A.6

Engl.: class

Regeln:

- Eine Klasse wird durch Fächer näher beschrieben. Vordefinierte Fächer gibt es für:
  - Name
  - Attribute
  - Operationen
- Eine Klasse hat einen innerhalb des umfassenden Paketes eindeutigen Namen.
- Vor den eigentlichen Klassennamen werden die Namen der umfassenden Pakete gestellt. Syntax:
  Klassenname ::= [Paketname "::"] Klassenname
- Der Klassenname sollte mit einem Großbuchstaben beginnen.
- Der Klassenname einer abstrakte Klasse wird kursiv gedruckt, zusätzlich kann die Klasse mit der Bedingung `{abstract}` versehen werden.
- Stereotypen charakterisieren unter Umständen die Klasse. Es gibt folgende vordefinierte Stereotypen:
  - ≪interface≫ Schnittstelle für einen Teilaspekt
  - ≪implementationClass≫ festgelegte Struktur im Sinne einer Programmiersprache
  - ≪metaclass≫ Exemplare sind Klassen
  - ≪type≫ beschreibt Zustand und Verhalten ohne Implementierung.
  - ≪utility≫ Sammlung von eigenständigen Attributen und Operationen
- In einer Eigenschaftsliste können weitere Eigenschaften z. B. das Persistenzverhalten der Klasse angegeben werden.

siehe auch: Attribut (2), Operation (3), Stereotyp (All:4), Eigenschaftsliste (All:5)

Andere Diagramme: Kompositionsdiagramm (Kps:1), Paketdiagramm (Pak:1)

Beschreibung: Seite 44

Kla:2 Attribut

Abbildung: A.6

Engl.: attribute

Regeln:

- Jedes Attribut hat in der Klasse einen eindeutigen Namen.
- Ist die verwendete Sprache Englisch, so sollte der Attributname mit einem Kleinbuchstaben beginnen.
- Ein Klassenattribut wird unterstrichen dargestellt.
- Syntax:
  Attribut ::= [Sichtbarkeit] Attrname ": " Typ [Multiplizität ["ordered

"]]
["= " Defaultwert] ["{ " Eigenschaften "} "]
siehe auch: Klasse (1), Operation (3), Multiplizität (10)
Beschreibung: Seite 45

Kla:3 Operation
Abbildung: A.6
Engl.: operation
Regeln:
- Jede Operation muss in Kombination mit ihren Typen der Parameterliste und dem Typ des Rückgabewertes in der Klasse eindeutig sein.
- Eine Klassenoperation wird unterstrichen dargestellt.
- Eine abstrakte Operation wird kursiv dargestellt oder mit der Bedingung `{abstract}` versehen.
- Syntax:
  Operation ::= [Sichtbarkeit] Operationsname "(" [Parameterliste] ")" [": " Rückgabetyp] ["{ " Eigenschaften "} "]
  Parameterliste ::= Parameter [", " Parameterliste]
  Parameter ::= ["in " , "inout " , "out "] Name ": " Typ ["= " Defaultwert]

  Der Rückgabetyp kann auch durch einen Parameter namens `return` bestimmt werden.
- Die Eigenschaft `query` bezeichnet eine Operation ohne Zustandsveränderung.
- Die Verwirklichung der Operation durch eine Methode kann in einer Notiz als Kommentar angegeben werden.

siehe auch: Klasse (1), Attribut (2), Notiz (All:3)
Beschreibung: Seite 46

Kla:4 Sichtbarkeit
Abbildung: A.6
Engl.: visibility
Regeln:
- Die Sichtbarkeit eines Klassenelements wird durch folgende Zeichen geregelt:
- `+`: public
- `~`: package
- `#`: protected
- `-`: private

siehe auch: Klasse (1), Attribut (2), Operation (3)
Andere Diagramme: Paketdiagramm (Pak:3)
Beschreibung: Seite 47

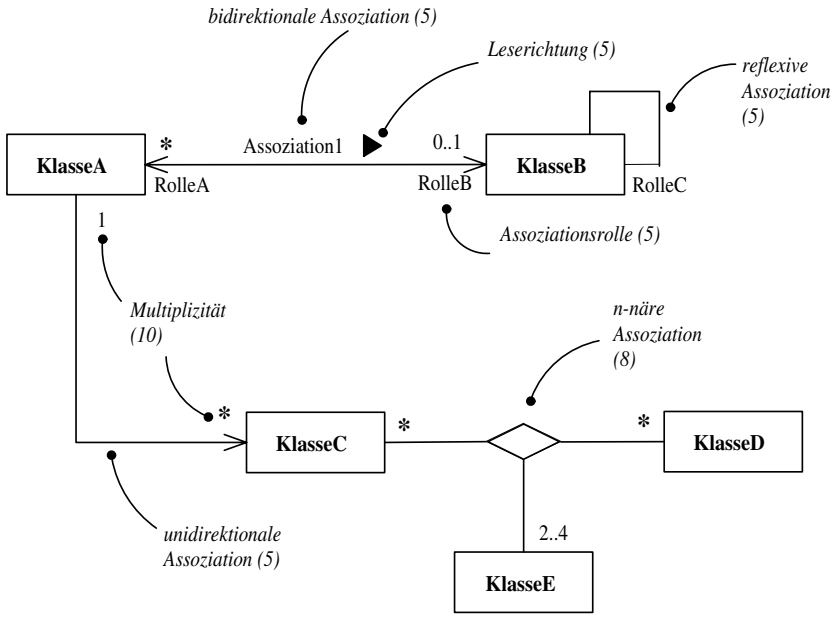

Abbildung A.7. Syntax der Klassendiagramme 2

Kla:5 Assoziation

Abbildung: A.7, A.9
Engl.: association
Regeln:

- Eine (binäre) Assoziation verbindet zwei Klassen miteinander.
- Eine Assoziation kann als navigierbar (mit Pfeilspitze) oder nicht navigierbar (mit Kreuz) gekennzeichnet werden. so entstehen gerichtete Assoziationen.
- Eine reflexive Assoziation verbindet eine Klasse mit sich selbst.
- Für jede Assoziation kann ein Assoziationsname mit optionaler Leserichtung angegeben werden.
- Für jeden Anknüpfpunkt der Assoziation an eine Klasse kann ein Rollenname und die Multiplizität der Klasse in dieser Assoziation angegeben werden.
- Außerdem sind Objektselektoren möglich.
- Die üblichen Zugriffsrechte schränken gegebenenfalls die Sichtbarkeit der Rollennamen ein.
- Ist die Multiplizität größer als 1, so kann die Kollektion als `{ordered}` oder `{unique}` spezifiziert werden.
- Die Bedingung `{xor}` kennzeichnet zwei verschiedene Assoziationen von einer Klasse aus als Alternativen.

siehe auch: Aggregation (6), n-äre Assoziation (8), Assoziationsklasse (9), Klasse (1), Komposition (7), Multiplizität, (10), Objektselektor (11)
Andere Diagramme: Use-Case-Diagramm (Use:4), Kommunikationsdiagramm (Kmk:4)
Beschreibung: Seite 51

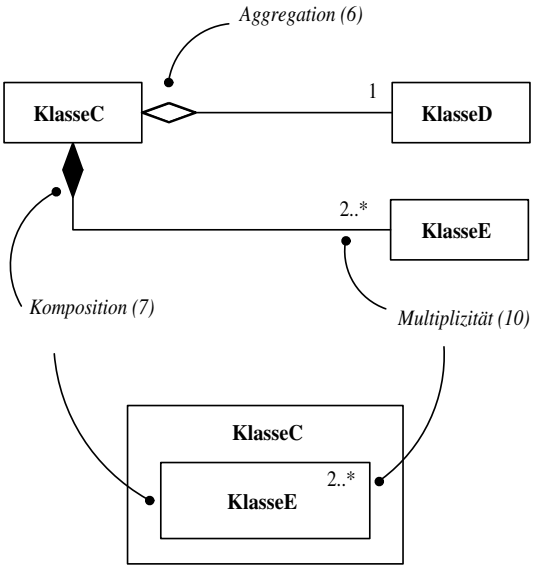

Abbildung A.8. Syntax der Klassendiagramme 3

Kla:6 Aggregation
Abbildung: A.8
Engl.: aggregation
Regeln:
- Eine Aggregation ist eine gerichtete, binäre Assoziation.

siehe auch: Assoziation (5), Komposition (7), Klasse (1)
Beschreibung: Seite 57

Kla:7 Komposition
Abbildung: A.8
Engl.: composition
Regeln:
- Eine Komposition ist eine Aggregation.
- Eine Komposition kann auch durch Einzeichnen der Einzelteile in das Aggregat dargestellt werden.
- In diesem Fall wird die Multiplizität oben rechts notiert.

siehe auch: Aggregation (6), Klasse (1), Multiplizität (10)
Andere Diagramme: Kompositionsdiagramm (Kps:1)
Beschreibung: Seite 57

Kla:8 n-äre Assoziation
Abbildung: A.7
Engl.: n-ary association
Regeln:
- Eine n-äre Assoziation verbindet n Klassen miteinander.
- Die Enden einer n-äre Assoziation können nicht mit Objektselektoren versehen und nicht als Aggregation oder Komposition ausgezeichnet werden.

siehe auch: Assoziation (5), Klasse (1)
Beschreibung: Seite 63

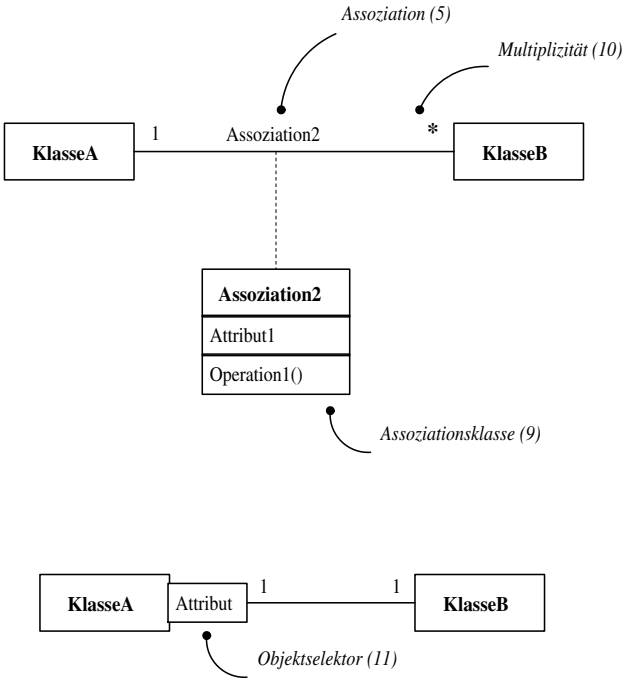

Abbildung A.9. Syntax der Klassendiagramme 4

Kla:9 Assoziationsklasse
Abbildung: A.9

Engl.: association class
Regeln:
- Eine Assoziationsklasse ist eine Klasse als Assoziation.
- Der Name der Assoziationsklasse muss mit dem Namen der Assoziation übereinstimmen.

siehe auch: Assoziation (5), Klasse (1)
Beschreibung: Seite 61

Kla:10 Multiplizität
Abbildung: A.7, A.8, A.9
Engl.: multiplicity
Regeln:
- Syntax: *Multiplizität ::=* "*" | *Intervall*
  Intervall ::= Zahl | *Zahl* ".." *Zahl* | *Zahl* ".." "*"

siehe auch: Assoziation (5), Attribut (2)
Beschreibung: Seite 59

Kla:11 Objektselektor
Abbildung: A.9
Engl.: qualifier
Regeln:
- Ein Objektselektor verbindet ein Ende einer Assoziation mit einer Klasse.
- Er beinhaltet ein Attribut oder eine Liste von Attributen der gegenüberliegenden Klasse, die dort deshalb nicht aufgeführt werden brauchen.

siehe auch: Assoziation (5), Klasse (1).
Beschreibung: Seite 60

Kla:12 Generalisierung
Abbildung: A.10
Engl.: generalization
Regeln:
- Jede Klasse kann beliebig viele Oberklassen und Unterklassen haben.
- Der Graph, der aus den Klassen und Generalisierungspfeilen entsteht, muss kreisfrei sein.
- Die Generalisierungsbeziehungen können unter Angabe eines Kriteriums zu Gruppen zusammengefasst werden.

siehe auch: Klasse (1)
Andere Diagramme: Use-Case-Diagramm (Use:6)
Beschreibung: Seite 65

Kla:13 Interface, Schnittstelle
Abbildung: A.11

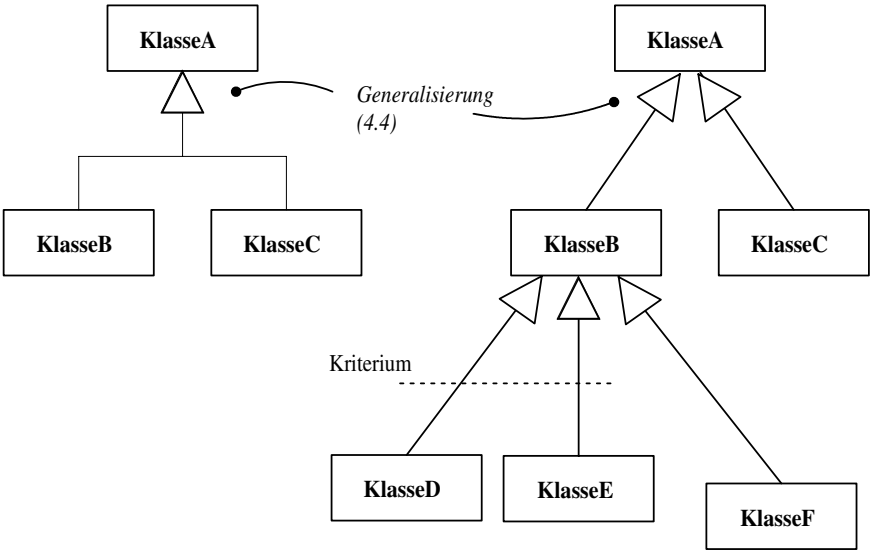

Abbildung A.10. Syntax der Klassendiagramme 5

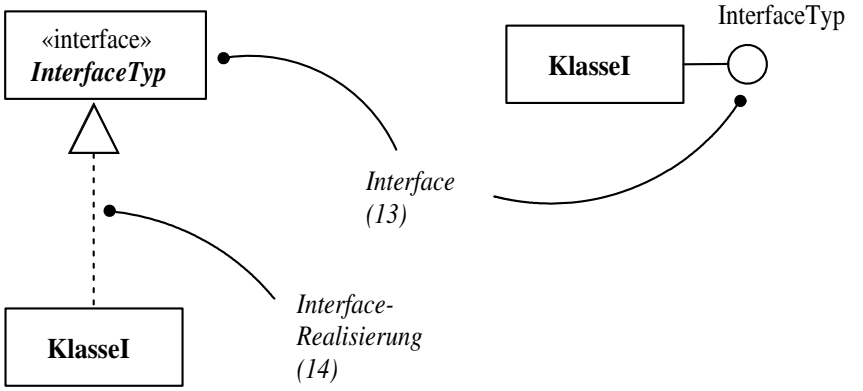

Abbildung A.11. Syntax der Klassendiagramme 6

Engl.: interface
Regeln:
- Ein Interface wird als Klasse mit dem Stereotyp ≪interface≫ dargestellt.
- Ein Interface beinhaltet keine Attribute und nur abstrakte Operationen.

siehe auch: Interface-Realisierung (14), Klasse (1)
Andere Diagramme: Komponentendiagramm (Kmp:2)
Beschreibung: Seite 69

Kla:14 Interface-Realisierung

Abbildung: A.11
Engl.: implemenation of interface
Regeln:
- Eine solche Beziehung verbindet ein Interface mit einer Klasse.
- In der Kurzform wird nur der Name des Interfaces angegeben.

siehe auch: Interface (13) Klasse (1), Abhängigkeit (18)
Andere Diagramme: Komponentendiagramm (Kmp:6)
Beschreibung: Seite 69

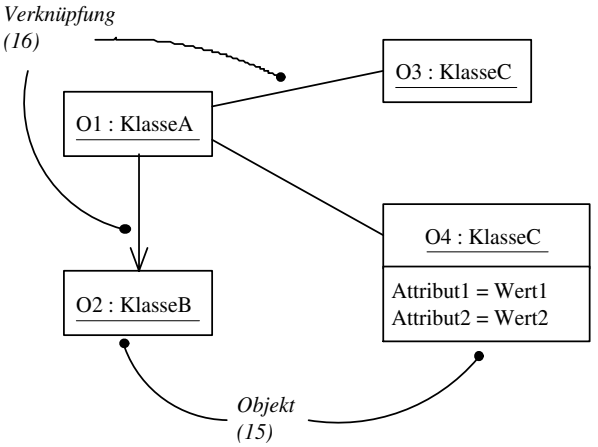

Abbildung A.12. Syntax der Klassendiagramme 7

Kla:15 Objekt

Abbildung: A.12
Engl.: object
Regeln:
- Ein Objekt besteht aus einem Namensfach und einem Attributfach.
- Syntax des kompletten Objektnamens:
  [Objektname] [":" Klassenname] ["[" Zustandsliste "]"]
- Objektname oder Klassenname können fehlen, aber nicht gleichzeitig.
- Gehört ein Objekt mehreren Klassen gleichzeitig an, so werden deren Namen durch Komma getrennt.
- Der Stereotyp der Klasse kann angegeben werden.
- Im Attributfach stehen Wertbelegungen interessanter Attribute.
- Syntax: *Wertbelegung ::= Attributname [":" Typ] "=" Wert*
- Objekte sind durch Links miteinander verbunden.
- Ein zusammengesetztes Objekt (Kompositum) ist eine Instanz einer Komposition.

siehe auch: Link (16) Klasse (1), Komposition (7)
Andere Diagramme: Aktivitätsdiagramm (Akt:3), Sequenzdiagramm (Seq:1), Kommunikationsdiagramm (Kmk:1)
Beschreibung: Seite 49

Kla:16 Link, Verknüpfung
Abbildung: A.12
Engl.: link
Regeln:
- Links verknüpfen Objekte.
- Links können unidirektional oder bidirektional sein.
- Links sind Exemplare von Assoziationen.
- Bei der Assoziation kann als Eigenschaft vermerkt werden, ob Links nur bei der Objekterzeugung geschaltet (`{readOnly}`), nur hinzugefügt (`{addOnly}`) werden dürfen oder frei veränderbar sind.

siehe auch: Objekt (15), Assoziation (5), Eigenschaftsliste (5)
Andere Diagramme: Kommunikationsdiagramm (Kmk:4)
Beschreibung: Seite 51

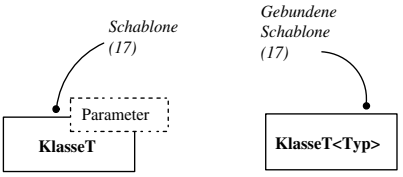

Abbildung A.13. Syntax der Klassendiagramme 8

Kla:17 Schablone
Abbildung: A.13
Engl.: template
Regeln:
- Eine Schablone ist eine parametrisierte Klasse.
- Die Parameter im eingeschobenen Rechteck werden bei der Ausprägung ersetzt.
- Die Ausprägung geschieht durch Angabe von aktuellen Parametern
- oder eine Abhängigkeitsbeziehung mit dem Stereotyp ≪bind≫

siehe auch: Klasse (1), Abhängigkeit (18)
Beschreibung: Seite 71

Kla:18 Abhängigkeit

Abbildung: A.11
Engl.: dependency
Regeln:

- Eine beliebige Beziehung zwischen Modellelementen.
  - Interface und Realisierung
  - Schablone und ausprägendem Typ

siehe auch: Assoziation (5), Schablone (17), Interface (13), Interface-Realisierung (14)
Andere Diagramme: Komponentendiagramm (Kmp:6), Paketdiagramm (Pak:2), Use-Case-Diagramm (Use:7)
Beschreibung: Seite 71

A.5 Sequenzdiagramm

sequence diagram

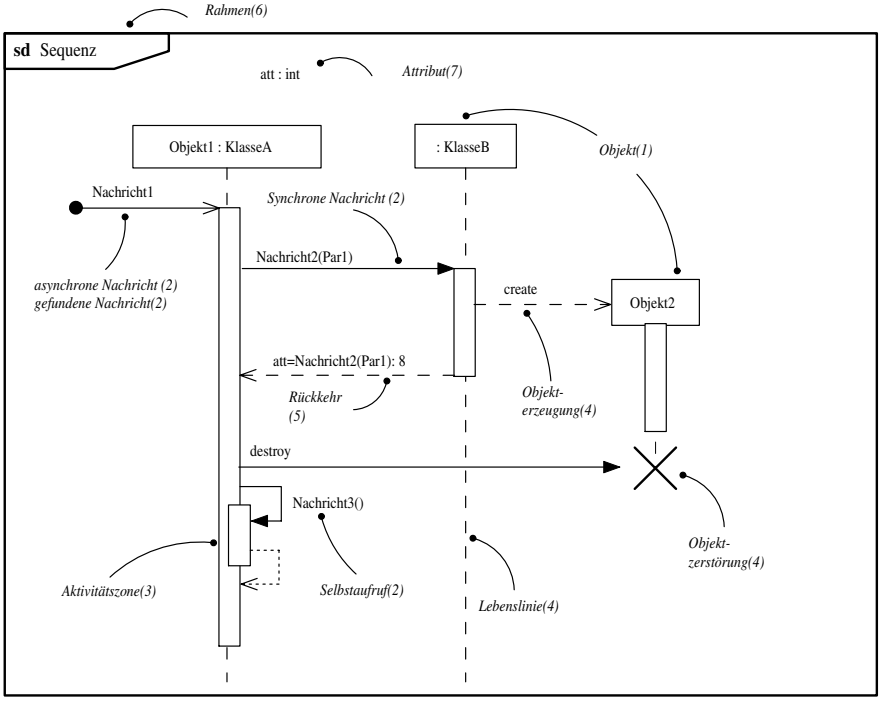

Abbildung A.14. Syntax der Sequenzdiagramme 1

Seq:1 Objekt

Abbildung: A.17, A.14

Engl.: lifeline head

Regeln:

- Objekte müssen bezeichnet werden.
- Syntax für Objektname:
  [Objektname] [Selektor] [": " Klassenname] , nicht leer.
- Der Selektor sucht ein Objekt aus einer Kollektion aus.
- An Stelle der Objekte können auch andere Interaktionspartner, also Daten, Komponenten oder Rechnerknoten stehen.

siehe auch: Objektlebenslinie (4), Objektselektor (11), Nachricht (2), Constraint (2)

Andere Diagramme: Kommunikationsdiagramm (Kmk:1), Kompositionsdiagramm (Kps:2), Aktivitätsdiagramm (Akt:3), Installationsdiagramm (Ins:1), Klassendiagramm (Kla:15)

Beschreibung: Seite 80, Seite 96

Seq:2 Nachricht

Abbildung: A.17, A.14

Engl.: message

Regeln:

- Jede Nachricht muss bezeichnet werden.
- Nachrichten werden von Objekt zu Objekt gesendet.
- Selbstaufrufe des Objekts sind möglich.
- Nachrichtenpfeile verbinden Objektlebenslinien oder Aktivitätszonen.
- Bei der Kreation von Objekten führen sie direkt zum Objektsymbol.
- Für synchrone Nachrichten mit prozeduraler Semantik werden Pfeile mit ausgefüllten Spitzen gezeichnet.
- Rückkehrnachrichten von Prozeduren und solche, die ein Objekt kreieren, werden als gestrichelte Pfeile mit normalen Spitzen gezeichnet ($--\rightarrow$).
- Für asynchrone Nachrichten werden normale Pfeile gezeichnet ($\rightarrow$).
- Verlorene Nachrichten (ohne Empfänger) werden durch einen ausgefüllten Kreis an der Spitze, gefundene (Sender unbekannt) am Ende des Pfeiles modelliert.
- Syntax für Beschriftung der Nachricht:
  Nachricht ::= [Attribut "="] Nachrichtenname ["(" [Parameter { ", " Parameter}] ")"] [":" Rückgabewert]
  Parameter ::= ([Parname "="] Wert) | (Attribut "=" Parname [": " Rückgabewert])
- Nachrichten können mit einer Wächterbedingung versehen werden.

siehe auch: Objekt (1)

Andere Diagramme: Kommunikationsdiagramm (Kmk:2, Kmk:4)

Beschreibung: Seite 81

Seq:3 Aktivitätszone
Abbildung: A.14
Engl.: activation
Regeln:
- Aktivitätszonen können bei direkten oder indirekten Selbstaufrufen des Objekts gestapelt werden.

siehe auch: Objekt (1), Objektlebenslinie (4)
Beschreibung: Seite 81

Seq:4 Objektlebenslinie
Abbildung: A.14
Engl.: lifeline
Regeln:
- Die Objektlebenslinie startet am Objektsymbol.
- Sie repräsentiert das Objekt in der Interaktion.
- Bei der Objekterzeugung beginnt die Objektlebenslinie auf der entsprechenden Position.
- Bei Objektzerstörung endet die Lebenslinie mit dem Zerstörungssymbol.
- Die Objektlebenslinie wird teilweise von Aktivitätszone überdeckt.

siehe auch: Objekt (1), Aktivitätszone (3).
Beschreibung: Seite 80

Seq:5 Rückkehrnachricht
Abbildung: A.14
Engl.: return message
Regeln:
- Nur synchrone Nachrichten können optional mit einer Rückkehrnachricht versehen werden.

siehe auch: Nachricht (2)
Beschreibung: Seite 81

Seq:6 Rahmen
Abbildung: A.14
Engl.: attribute
Regeln:
- Eine Interaktion kann in einen mit **sd** gekennzeichneten Rahmen eingeschlossen werden.

Andere Diagramme: alle (All:1)
Beschreibung: Seite 35

Seq:7 Attribut
Abbildung: A.14

Engl.: attribute
Regeln:
• Eine Interaktion kann ein temporäres Attribut vereinbaren.
siehe auch: Objektlebenslinie (4)
Andere Diagramme: Klassendiagramm (Kla:2)
Beschreibung: Seite 35

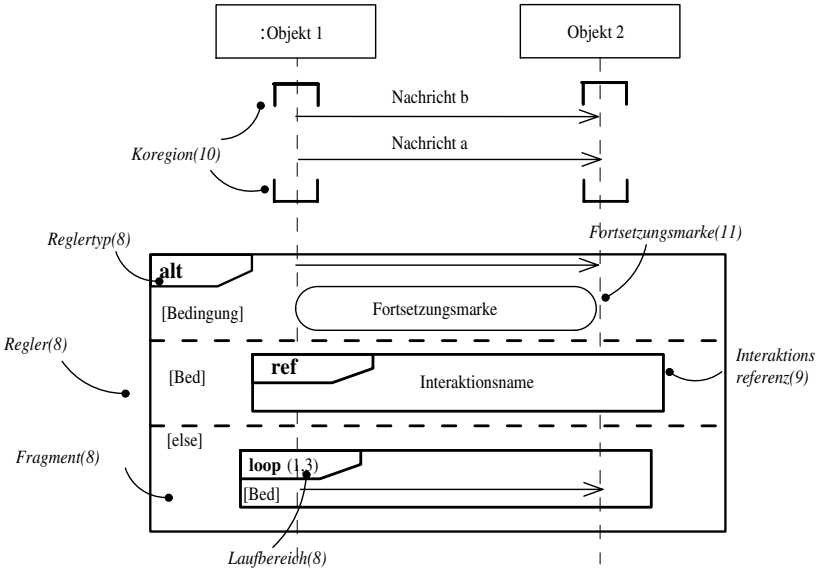

Abbildung A.15. Syntax der Sequenzdiagramme 2

Seq:8 Regler, Kontrolloperatoren
Abbildung: A.15
Engl.: CombinedFragment
Regeln:
• Regler fassen Teile eines Sequenzdiagrammes zusammen und belegen sie mit einer speziellen Semantik.
• Regler werden als Rahmen gezeichnet. Der Name notiert in Fettdruck einen von 12 Typen.
• Der Inhaltsbereich kann in verschiedene horizontale Fragmente aufgeteilt werden.

Typ	#Frag	zus. Syntax	Beschreibung
opt	1	Bedingung	optionale Interaktion
alt	2..*	Bedingungen, [else]	Alternative
loop	1	Laufbereichsgrenzen im Namensfach, Iterationsbedingung	Schleife
break	1	Bedingung	Abbruch der umschließenden Interaktion
par	2..*		nebenläufige Interaktionen, bel. Reihenfolge
seq	2..*		Abfolge gemäß Lebenslinien
strict	2..*		vorgeschriebene Abfolge
critical	1		atomare Interaktion
neg	1		ungültige Interaktion
assert	1		unabdingbare Interaktion
consider	1		wichtig
ignore	1		unwichtig

siehe auch: Rahmen (6), Interaktionsreferenz (9)
Beschreibung: Seite 86

Seq:9 Interaktionsreferenz
Abbildung: A.15
Engl.: interaction reference
Regeln:
- Eine fremde Interaktion kann eingesetzt werden.
- Der Name der referenzierten Interaktion steht in einem Rahmen mit Namen **ref**

siehe auch: Rahmen (6), Regler (8)
Andere Diagramme: Interaktionsübersichtsdiagramm (A.6.3)
Beschreibung: Seite 90

Seq:10 Koregion
Abbildung: A.15
Engl.: Coregion
Regeln:
- Mit eckigen Klammern kann auf den Lebenslinien ein Bereich markiert werden, der in beliebiger Reihenfolge abgearbeitet werden kann.

siehe auch: Regler (par) (8)

Seq:11 Fortsetzungsmarke
Abbildung: A.15
Engl.: Continuation
Regeln:
- Mit ovalen Fortsetzungsmarken lassen sich Fragmente eines Kontrolloperators über mehrere Plätze verteilen.

siehe auch: Regler (alt) (8)

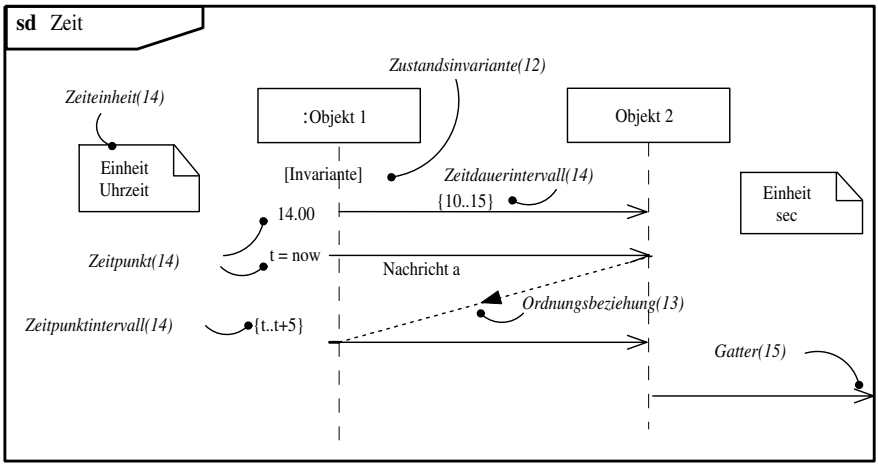

Abbildung A.16. Syntax der Sequenzdiagramme 3

Seq:12 Zustandsinvariante
Abbildung: A.16
Engl.: state invariant
Regeln:
- Lebenslinien können mit Bedingungen die Zustandsinvarianten einer Interaktion ausdrücken.
- Die Bedingung wird direkt, mittels einer Notiz oder in einem abgerundeten Rechteck notiert.

siehe auch: Notiz) (All:3)
Andere Diagramme: Zustandsdiagramm (Zus:1)

Seq:13 Ordnungsbeziehung
Abbildung: A.16
Engl.: GeneralOrdering
Regeln:

- Eine gepunktete Linie mit Pfeilspitze in der Mitte bringt 2 Ereignisse (Start- oder Zielpunkt einer Nachricht) in eine zeitliche Ordnung.

siehe auch: Regler (seq) (8)

Seq:14 Zeitangaben
Abbildung: A.16
Engl.: time constraint, duration constraint
Regeln:
- Zeitpunkte und Zeitdauern können einzeln oder als Intervall an beliebiger Stelle angegeben werden.
- Zeiteinheiten werden in einer Notiz erklärt.
- Die genaue Syntax ist nicht festgelegt.

siehe auch: Objektlebenslinie (4)
Andere Diagramme: Timing-Diagramm (4)

Seq:15 Gatter
Abbildung: A.16
Engl.: gate
Regeln:
- Punkt auf einem Rahmen für ein- oder ausgehende Nachrichten.

siehe auch: Regler (par) (8)

A.6 Weitere Interaktionsdiagramme

interaction diagrams

Die zentralen Begriffe in den Interaktionsdiagrammen wie Objekt und Nachricht wurden bereits bei den Sequenzdiagrammen im vorigen Abschnitt erläutert, sie treten in leicht veränderter Notation auch in den anderen Diagrammen auf.

A.6.1 Kommunikationsdiagramm

communication diagram

Kmk:1 Objekt
Abbildung: A.17
Engl.: lifeline
Regeln:
- wie im Sequenzdiagramm

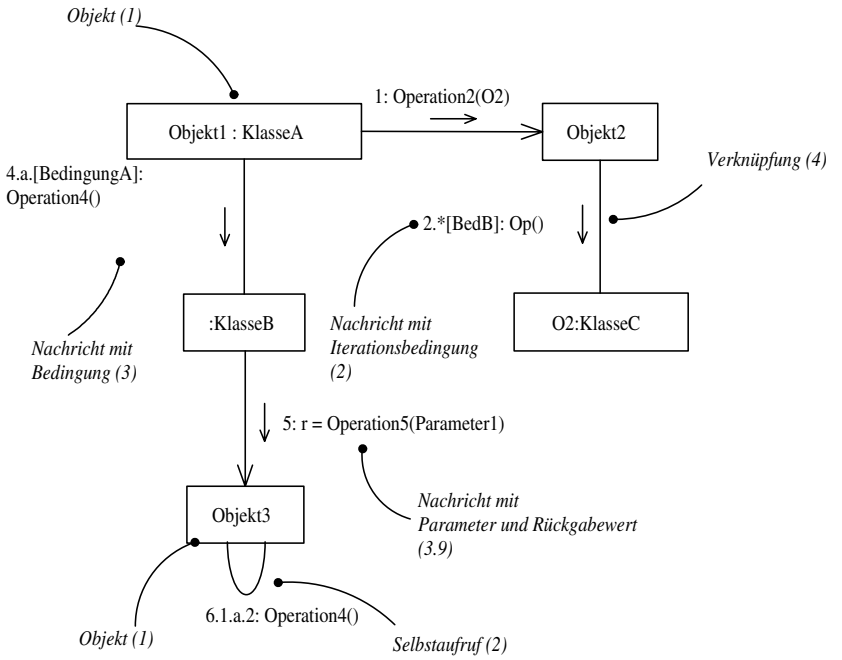

Abbildung A.17. Syntax der Kommunikationsdiagramme 1

Andere Diagramme: Sequenzdiagramm (Seq:1)

Beschreibung: Seite 96

Kmk:2 Nachricht

Abbildung: A.17

Engl.: message

Regeln:

- Jede Nachricht muss bezeichnet werden.
- Nachrichten werden von Objekt zu Objekt gesendet.
- Selbstaufrufe des Objekts sind möglich.
- Nachrichten laufen entlang der Links.
- Ihre Aufrufrichtung wird durch kleine Pfeile angedeutet.
- Sequenznummern regeln die Reihenfolge.
- Syntax für Beschriftung der Nachricht:
  Nachricht ::= Sequenznummer [Rückgabewert "="] Nachrichtenname ["(" Parameterliste ")"]
  Sequenznummer ::= [Ziffern | Buchstaben] ["" "[" Iterationsausdruck "]" | "[" Bedingung "]"] ["." Sequenznummer]* , nicht leer

siehe auch: Link (4), Objekt (1)
Andere Diagramme: Sequenzdiagramm (Seq:2)
Beschreibung: Seite 96

Kmk:3 Wächterbedingung
Abbildung: A.17
Engl.: guard
Regeln:
- Jede Nachricht kann mit einer Wächterbedingung versehen werden.
- Die Formulierung der Bedingung kann in OCL erfolgen, ist aber nicht festgelegt.

siehe auch: Nachricht (2), Bedingung (All:2)
Andere Diagramme: Zustandsdiagramm (Zus:3), Aktivitätsdiagramm (7)
Beschreibung: Seite 85

Kmk:4 Link, Verknüpfung
Abbildung: A.17
Engl.: link
Regeln:
- Links verknüpfen Objekte.
- Links können unidirektional oder bidirektional sein.

siehe auch: Objekt (1), Nachricht (2)
Andere Diagramme: Klassendiagramm (Kla:16)
Beschreibung: Seite 97

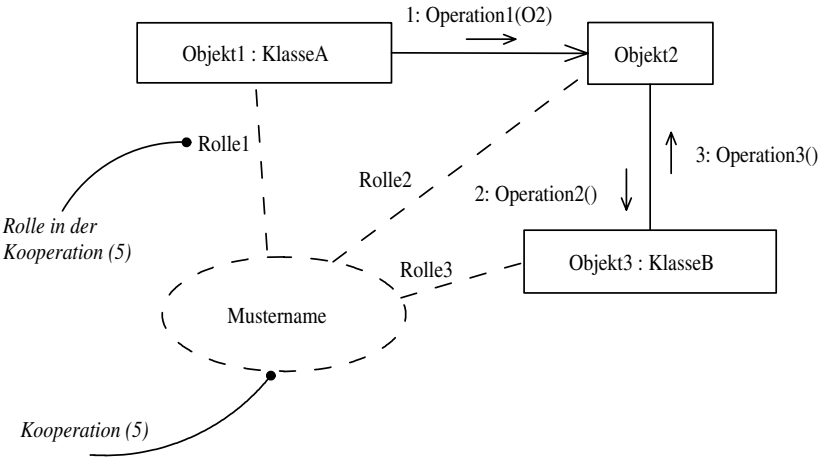

Abbildung A.18. Syntax der Kommunikationsdiagramme 2

Kmk:5 Kooperation
Abbildung: A.18
Engl.: collaboration
Regeln:
- Kooperationen sind benannte Objektgruppen, die zusammenarbeiten.
- Mit einer Kooperation kann der Einsatz eines Entwurfsmusters dokumentiert werden.
- Ihr Name wird in einer gestrichelten Ellipse angegeben, mit der die beteiligten Objekte verbunden sind.
- Die Verbindungslinien sind mit den Rollennamen markiert.

Andere Diagramme: Kooperationsdiagramm (Kop:1)
Beschreibung: Seite 131

A.6.2 Timing-Diagramm

timing diagram

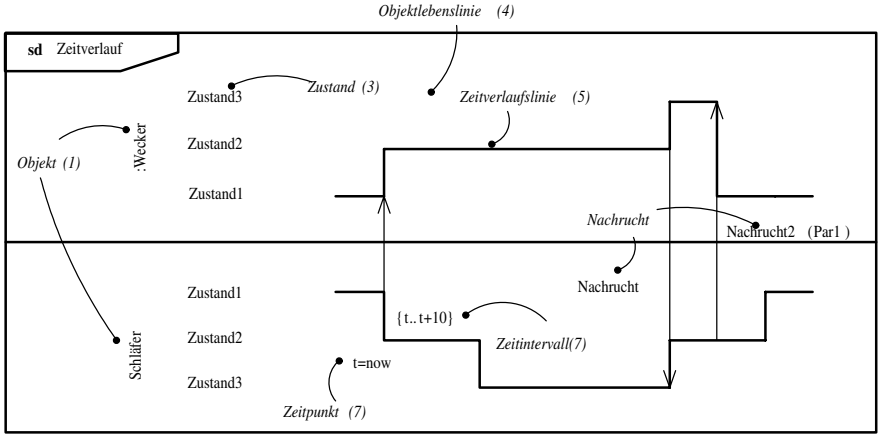

Abbildung A.19. Syntax der Timing-Diagramme 1

Tim:1 Objekt
Abbildung: A.19
Engl.: lifeline
Regeln:
- wie im Sequenzdiagramm
- vertikale Orientierung

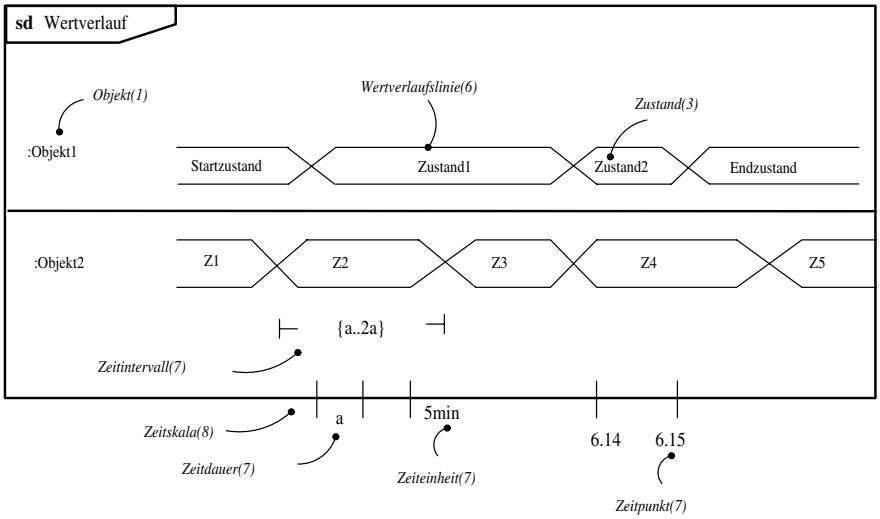

Abbildung A.20. Syntax der Timing-Diagramme 2

Andere Diagramme: Sequenzdiagramm (Seq:1), Aktivitätsdiagramm (Akt:3), Installationsdiagramm (Ins:1), Klassendiagramm (Kla:15)
Beschreibung: Seite 102

Tim:2 Nachricht
Abbildung: A.19
Engl.: message
Regeln:
- wie im Sequenzdiagramm
- vertikale Orientierung

Andere Diagramme: Sequenzdiagramm (Seq:2)
Beschreibung: Seite 81, Seite 96

Tim:3 Zustand
Abbildung: A.19
Engl.: state
Regeln:
- Die Zustände, die ein Objekt in der Interaktion einnimmt, belegen unterschiedliche Ebenen der Objektlebenslinie.
- Der Zustandsname steht links von der Zeitverlaufslinie.
- Wird eine Wertverlaufslinie angegeben belegen die Zustände die sechseckigen Fächer.

siehe auch: Objektlebenslinie (4), Zeitverlaufslinie (5), Wertverlaufslinie (6)

Andere Diagramme: Zustandsdiagramm (Zus:1)
Beschreibung: Seite 102

Tim:4 Objektlebenslinie
Abbildung: A.19
Engl.: lifeline
Regeln:
- Die Objektlebenslinie zeigt den Zeitverlauf der Zustandswechsel des Objekts.
- Sie kann mit Zeit- oder Wertverlaufslinien gezeichnet werden.

siehe auch: Zustand (3), Zeitverlaufslinie (5), Wertverlaufslinie (6)
Beschreibung: Seite 102

Tim:5 Zeitverlaufslinie
Abbildung: A.19
Engl.: state timeline
Regeln:
- Eine Treppenlinie, die die Zustandsebenen verbindet.

siehe auch: Objektlebenslinie (4), Wertverlaufslinie (6)
Beschreibung: Seite 102

Tim:6 Wertverlaufslinie
Abbildung: A.20
Engl.: value lifeline
Regeln:
- Eine parallele Doppellinie, deren Teile beim Zustandswechsel die Position tauschen und so sechseckige Figuren ausbilden.

siehe auch: Objektlebenslinie (4), Wertverlaufslinie (6)
Beschreibung: Seite 102

Tim:7 Zeitangaben
Abbildung: A.19
Engl.: time constraint, duration constraint
Regeln:
- Zeitpunkte und Zeitdauern können einzeln oder als Intervall an beliebiger Stelle angegeben werden.
- Die genaue Syntax ist nicht festgelegt.

siehe auch: Zeitskala (8), Objektlebenslinie (4)
Andere Diagramme: Sequenzdiagramm (14)
Beschreibung: Seite 102

Tim:8 Zeitskala
Abbildung: A.19

Engl.: timing ruler
Regeln:
- Eine Zeitskala legt am Rand des Rahmens die Zeiteinheit fest.
- Diese gilt proportional für den folgenden Teil des Diagramms.

siehe auch: Zeitangaben (7)
Andere Diagramme: Sequenzdiagramm (14)
Beschreibung: Seite 102

A.6.3 Interaktionsübersichtsdiagramm

interaction overview diagram

Ein Interaktionsübersichtsdiagramm ist ein Aktivitätsdiagramm, das an Stelle von Aktivitäten Interaktionen oder vorzugsweise Interaktionsreferenzen verbindet. Die Kontrolloperatoren eines Sequenzdiagramms werden dabei durch Verzweigungen von Kanten abgebildet.

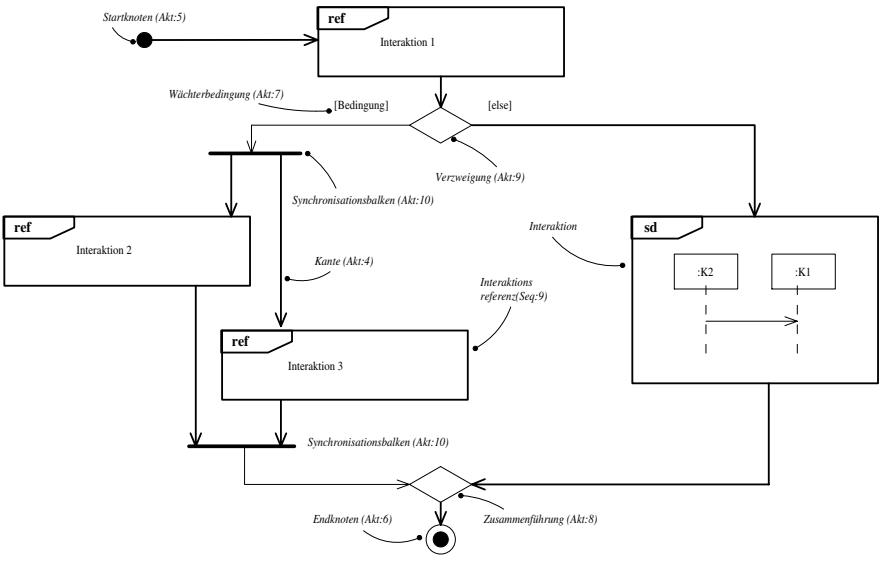

Abbildung A.21. Syntax der Interaktionsübersichtsdiagramme

Wir verweisen direkt auf die Elemente der zitierten Diagramme.

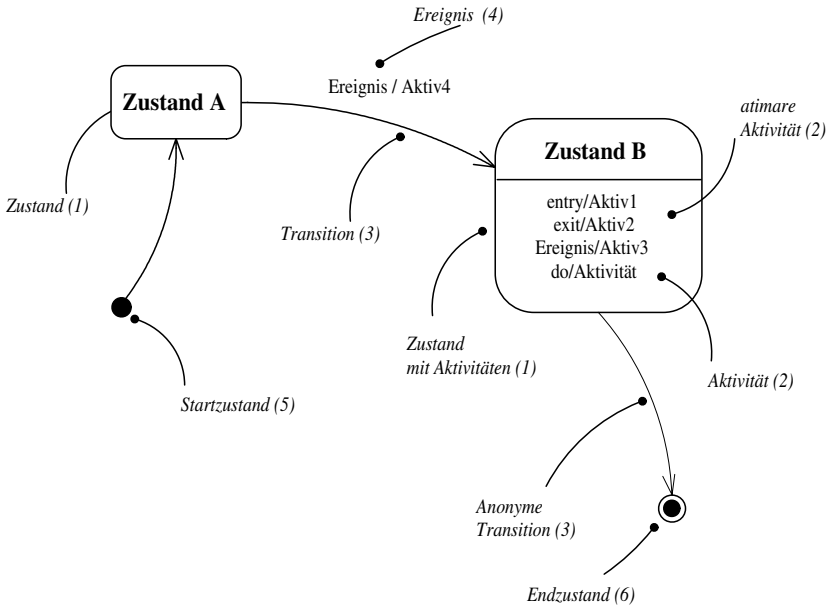

Abbildung A.22. Syntax der Zustandsdiagramme 1

A.7 Zustandsdiagramm

statechart diagram

Zus:1 Zustand
Abbildung: A.22
Engl.: state
Regeln:
- Jeder Zustand wird durch seinen Namen identifiziert.
- Anonyme Zustände gelten als verschieden.
- Zustände können geschachtelt werden.

siehe auch: Transition (3), Hierarchischer Zustand (7), History-Zustand (8)
Andere Diagramme: Aktivitätsdiagramm (Akt:3), Klassendiagramm (Kla:15), Timing-Diagramm (Tim:1)
Beschreibung: Seite 107

Zus:2 Aktivität
Abbildung: A.22
Engl.: activity
Regeln:

- Bei Transitionen werden Aktivitäten bei der Ereignisbeschriftung mit angegeben.
- Für jede Aktivität können Parameter und Rückgabewerte angegeben werden.
  Syntax: *[Objekt "="] Aktivitätsname ["(" Parameterliste ")"]*
- In jedem Zustand kann maximal eine andauernde Aktivität angegeben werden.
- Das Pseudoereignis `do` bewirkt die Ausführung eines anderen Zustandsautomaten als Aktivität in einem Zustand.
- In Zuständen werden Aktivitäten als Ereignis-Aktivitäts-Paar aufgelistet.
  Syntax: *Ereignis "/" Aktivität*
- Für atomare Aktivitäten, die beim Erreichen bzw. Verlassen eines Zustandes ausgeführt werden sollen, sind die Pseudoereignisse `entry` bzw. `exit` definiert.
- Jedes Ereignis oder Pseudoereignis darf maximal einmal in einem Zustand auftauchen.

siehe auch: Transition (3), Ereignis (4). Zustand (1).
Andere Diagramme: Aktivitätsdiagramm (1)
Beschreibung: Seite 111

Zus:3 Transition
Abbildung: A.22
Engl.: transition
Regeln:

- Transitionen verbinden Zustände miteinander.
- Transitionen sind mit Ereignissen beschriftet.
- Für jedes Ereignis darf maximal eine einzige Transition von einem Zustand abgehen.
- Die anonyme Transition ohne Beschriftung sollte nur von Zuständen mit definierten Aktivitäten abgehen.
- Syntax der Beschriftung einer Transition:
  Transition ::= Ereignis [Bedingung] ["/" Aktivitäten]
  Ereignis ::= Ereignisname ["(" Parameterliste ")"]
  Aktivitäten ::= Aktivität [";" Aktivitäten]

siehe auch: Aktivität (2), Ereignis (4). Zustand (1).
Andere Diagramme: Aktivitätsdiagramm (Akt:1)
Beschreibung: Seite 111

Zus:4 Ereignis
Abbildung: A.22
Engl.: trigger
Regeln:

- Ereignisse beschriften Transitionen.

- Ereignisse haben Parameter.
- Es gibt vordefinierte Ereignisse (`after`).
- Die Auslösung eines mit `defer` gekennzeichnetem Ereignisses wird verzögert.
- Es gibt die Pseudoereignisse
  - `entry` Zustandsanfang
  - `exit` Zustandsende
  - `do` Zustandsinterne Aktivität

siehe auch: Transition (3), Zustand (1), Aktivität (2)
Beschreibung: Seite 107

Zus:5 Startzustand
Abbildung: A.22
Engl.: start state
Regeln:
- Der Startzustand markiert den Anfang des Zustandsautomaten.
- Jeder Automat darf maximal einen Startzustand haben.

siehe auch: Endzustand (6), History-Zustand (8)
Andere Diagramme: Aktivitätsdiagramm (Akt:5)
Beschreibung: Seite 107

Zus:6 Endzustand
Abbildung: A.22
Engl.: final state
Regeln:
- Ein Endzustand markiert ein Ziel des Zustandsautomaten.
- Jeder Automat darf beliebig viele Endzustände haben.

siehe auch: Startzustand (5), History-Zustand (8)
Andere Diagramme: Aktivitätsdiagramm (Akt:6)
Beschreibung: Seite 107

Zus:7 Hierarchischer Zustand
Abbildung: A.23
Engl.: composite state
Regeln:
- Statt hierarchischer Zustand sagt man auch zusammengesetzter Zustand oder Oberzustand.
- Der Name steht im obersten Fach des Zustandsrechtecks oder in einem Karteireiter.
- Ein Oberzustand beinhaltet beliebig viele Zustände oder Unterzustände.
- Jeder Oberzustand beinhaltet maximal einen Startzustand oder alternativ maximal einen History-Zustand jeder Kategorie.

siehe auch: Transition (3), History-Zustand (8), Zustand (1).
Beschreibung: Seite 114

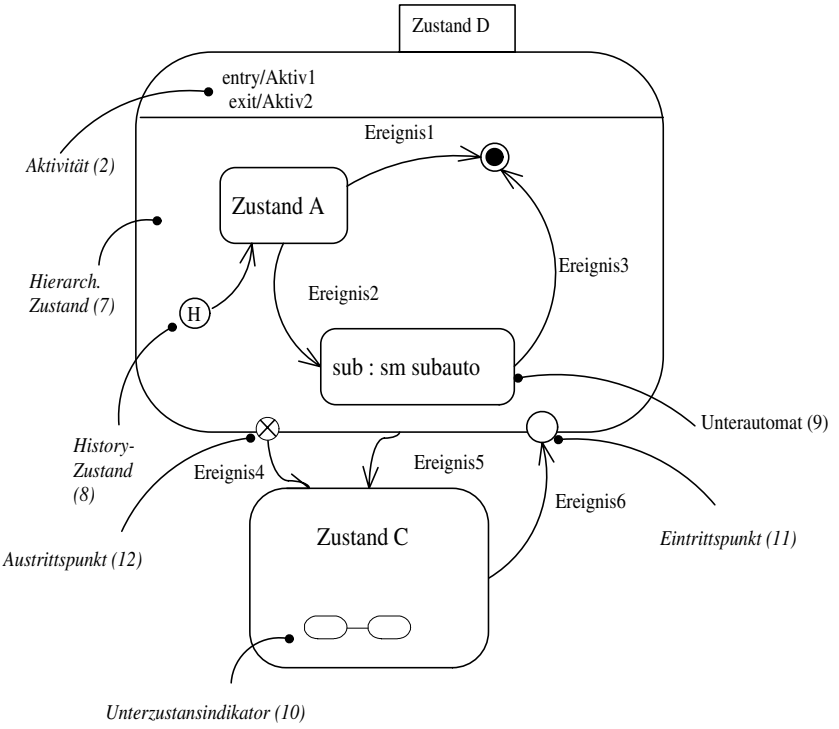

Abbildung A.23. Syntax der Zustandsdiagramme 2

Zus:8 History-Zustand
Abbildung: A.23
Engl.: history state
Regeln:
- Ein Zustand kann einen flachen `H` und einen tiefen `H*` History-Zustand haben.

siehe auch: Hierarchischer Zustand (7), Zustand (1)
Beschreibung: Seite 114

Zus:9 Unterautomat
Abbildung: A.23
Engl.: submachine
Regeln:
- Die Namen der Unterautomatenzustände enthalten nach einem Doppelpunkt den Namen des Unterautomaten.
- Unterautomaten kommunizieren über Ein- und Austrittspunkte.

siehe auch: Hierarchischer Zustand (7)
Beschreibung: Seite 114

Zus:10 Unterzustandsindikator
Abbildung: A.23
Engl.: decomposition indicator icon
Regeln:
- Wenn ein Zustand Unterzustände besitzt, die nicht eingezeichnet werden, so kann der Zustand mit dem Unterzustandsindikator gekennzeichnet werden.

siehe auch: Hierarchischer Zustand (7)
Beschreibung: Seite 114

Zus:11 Eintrittspunkt
Abbildung: A.23
Engl.: entry point
Regeln:
- Eintrittspunkte fassen mehrere in einen hierarchischen Zustand eingehende Transitionen zusammen.
- Alternativ zum Kreis am Zustandsrand kann die Transition mit dem in 2 Halbkreise eingeklammerten "via Eintrittspunktname"beschriftet werden.

siehe auch: Hierarchischer Zustand (7), Austrittspunkt (12)
Beschreibung: Seite 114

Zus:12 Austrittspunkt
Abbildung: A.23
Engl.: exit point
Regeln:
- Austrittspunkte fassen mehrere aus einem hierarchischen Zustand hinaus gehende Transitionen zusammen.
- Alternativ zum geviertelten Kreis am Zustandsrand kann die Transition mit dem in 2 Halbkreise eingeklammerten "via Austrittspunktname"beschriftet werden.

siehe auch: Hierarchischer Zustand (7), Eintrittspunkt (11)
Beschreibung: Seite 114

Zus:13 Konkurrente Zustände
Abbildung: A.24
Engl.: concurrent region
Regeln:
- Ein Zustand kann beliebig viele konkurrente Bereiche enthalten.
- In jedem konkurrenten Bereich läuft ein unabhängiger Zustandsautomat ab.
- Die einzelnen Bereiche können über Synchronisationsbalken aufgespalten und wieder vereinigt werden.

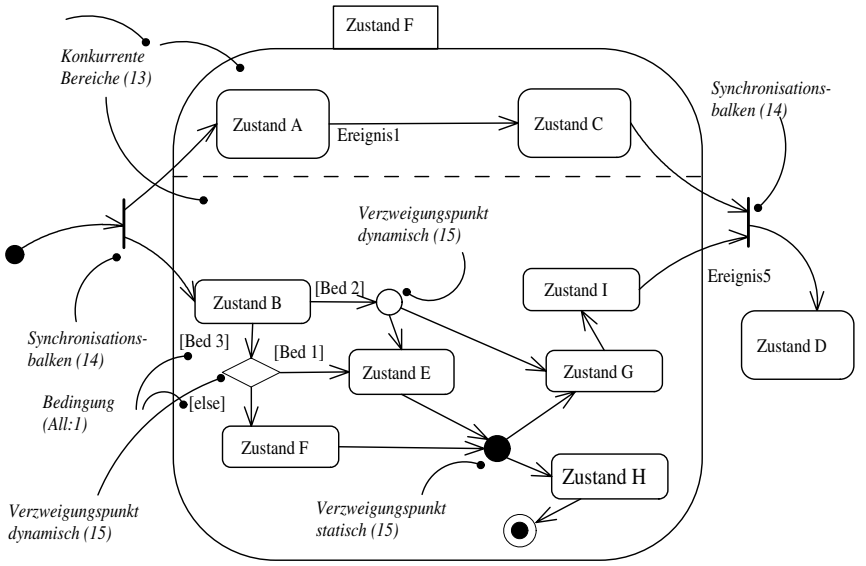

Abbildung A.24. Syntax der Zustandsdiagramme 3

siehe auch: Transition (3). Synchronisationsbalken (14).
Beschreibung: Seite 116

Zus:14 Synchronisationsbalken
Abbildung: A.24
Engl.: synchronization bar
Regeln:
- Ein Synchronisationsbalken kann entweder nur einen eingehenden und beliebig viele ausgehende Transitionen oder beliebig viele eingehende und nur einen ausgehenden Transitionen haben.
- Er kann mit einer Bedingung versehen werden.

siehe auch: Transition (3).
Beschreibung: Seite 117

Zus:15 Verzweigungspunkt
Abbildung: A.24
Engl.: choice, junction
Regeln:
- Eine oder mehr Transitionen können in einem Verzweigungspunkt enden.
- Zwei oder mehr Transitionen gehen von einem Verzweigungspunkt ab.
- Die abgehenden Transitionen müssen mit disjunkten Wächterbedingungen versehen werden.

- Bei einem statischen Verzweigungspunkt (Kreuzung) werden alle Bedingungen vor dem Start der Transition ausgewertet.

siehe auch: Transition (3)
Beschreibung: Seite 109

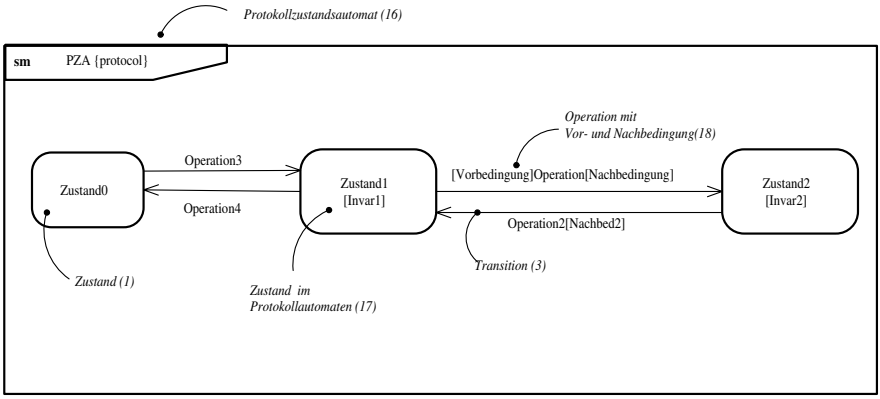

Abbildung A.25. Syntax der Zustandsdiagramme 4

Zus:16 Protokollzustandsautomat
Abbildung: A.25
Engl.: protocol state machine
Regeln:
- beschreibt eine Aufrufreihenfolge, ein Protokoll.
- wird durch `{protocol}` gekennzeichnet.

siehe auch: Transition (3), Zustand im Protokollzustandsautomat (17)
Beschreibung: Seite 118

Zus:17 Zustand im Protokollzustandsautomat
Abbildung: A.25
Engl.: state
Regeln:
- Eine Invariante kann angegeben werden.

siehe auch: Zustand (1), Protokollzustandsautomat (16)
Andere Diagramme: Aktivitätsdiagramm (Akt:3), Klassendiagramm (Kla:15)
Beschreibung: Seite 118

Zus:18 Operation
Abbildung: A.25

Engl.: call trigger
Regeln:
- Operationen beschriften Transitionen.
- Operationen haben Parameter.
- Es können Vor- und Nachbedingungen spezifiziert werden.

siehe auch: Transition (3), Zustand im Protokollzustandsautomat (17)
Beschreibung: Seite 118

A.8 Komponentendiagramme

component, deployment, and composite structure diagrams

Die Elemente der Komponentendiagramme können teilweise auch in anderen Diagrammen, z.B. im Klassendiagramm auftreten. Nach den Installationsdiagrammen folgen die Kompositionsdiagramme und Kooperationsdiagramme, die in UML zu einem Diagramm, dem composite structure diagram zusammen gefasst werden.

A.8.1 Komponentendiagramm

component diagram, structure diagram

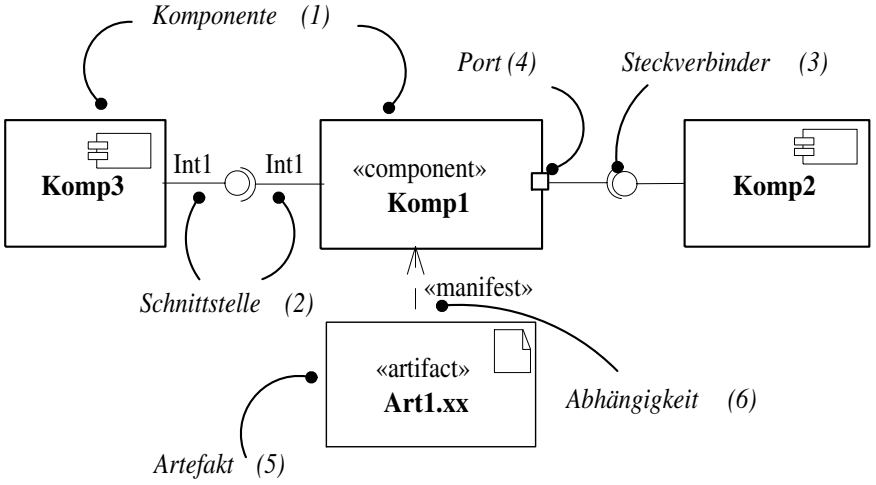

Abbildung A.26. Syntax der Komponentendiagramme 1

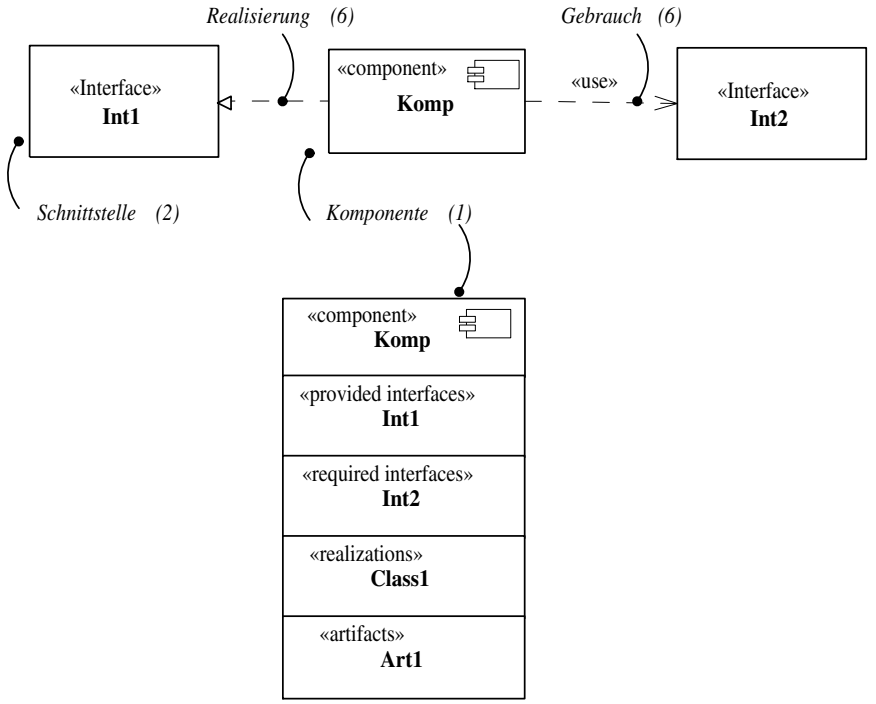

Abbildung A.27. Syntax der Komponentendiagramme 2

Kmp:1 Komponente
Abbildung: A.26, A.27
Engl.: component
Regeln:
- Der Name wird fett geschrieben.
- Komponenten implementieren Schnittstellen oder benötigen Schnittstellen.
- Die Schnittstellen können hinter den Stereotypen ≪provided interfaces≫ bzw. ≪required interfaces≫ aufgelistet werden. Zusätzlich ist die Angabe der realisierenden Klassen oder Artefakte möglich.
- Komponenten werden durch Steckverbinder miteinander gekoppelt.
- Komponenten können durch Stereotypen klassifiziert werden: ≪entity≫, ≪process≫, ≪service≫, ≪implement≫, ≪subsystem≫

siehe auch: Schnittstelle (2), Steckverbinder 3, Artefakt (5)
Andere Diagramme: Installationsdiagramm (Ins:2)
Beschreibung: Seite 122

Kmp:2 Schnittstelle, Interface
Abbildung: A.26, A.27

Engl.: interface
Regeln:
- Die Schnittstelle wird bezeichnet.
- Eine Schnittstelle kann eine Komponente mit einer anderen Komponente über einen Steckverbinder verbinden.

siehe auch: Komponente (1), Steckverbinder (3)
Andere Diagramme: Klassendiagramm (Kla:13)
Beschreibung: Seite 125

Kmp:3 Steckverbinder, Konnektor
Abbildung: A.26
Engl.: assignment connector
Regeln:
- Ein Konnektor verbindet Komponenten über Schnittstellen.

siehe auch: Schnittstelle (2), Komponente (1)
Andere Diagramme: Kompositionsdiagramm (Kps:3)
Beschreibung: Seite 122

Kmp:4 Port, Anschluss
Abbildung: A.26
Engl.: port
Regeln:
- Interaktionspunkt für eine Komponente.
- Komplexe Ports fassen mehrere Schnittstellen zusammen.

siehe auch: Schnittstelle (2)
Andere Diagramme: Kompositionsdiagramm (Kps:4)
Beschreibung: Seite 130

Kmp:5 Artefakt
Abbildung: A.26
Engl.: artifact
Regeln:
- Artefakte beschreiben physische Komponenten.
- Artefakte können durch Stereotypen klassifiziert werden, in UML vorgesehen sind: ≪executable≫ , ≪file≫ , ≪library≫ , ≪source≫

siehe auch: Komponente (1)
Andere Diagramme: Installationsdiagramm (Ins:2)
Beschreibung: Seite 124

Kmp:6 Abhängigkeit
Abbildung: A.26, A.27
Engl.: dependency
Regeln:

- Abhängigkeiten verbinden Komponenten mit anderen Elementen, z.B. Gebrauch von Schnittstellen (Stereotyp ≪use≫), Realisierung oder Implementierung durch Klassen (dreieckige Spitze) oder Manifestation von Artefakten (Stereotyp ≪manifest≫).

siehe auch: Komponente (1), Schnittstelle (2)
Andere Diagramme: Klassendiagramm (Kla:18), Paketdiagramm (2), Use-Case-Diagramm (Use:7), Installationsdiagramm (Ins:3)
Beschreibung: Seite 124

A.8.2 Installationsdiagramm

deployment diagram

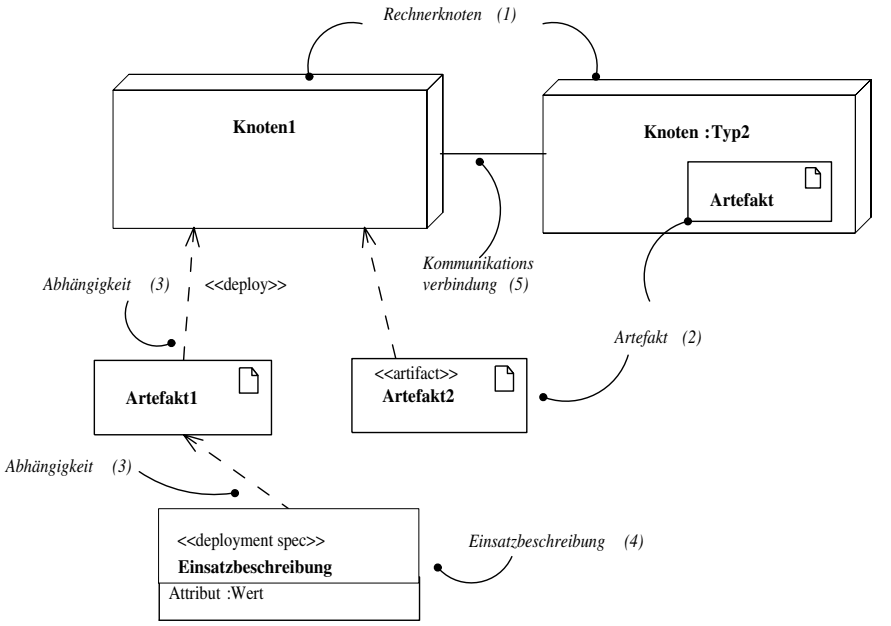

Abbildung A.28. Syntax der Installationsdiagramme

Ins:1 Rechnerknoten
Abbildung: A.28
Engl.: node
Regeln:

- Ein Rechnerknoten hat einen Namen und einen Typnamen. Beide können fehlen, jedoch nicht gleichzeitig.

- Rechnerknoten werden mit Kommunikationsverbindungen miteinander verknüpft.
- Rechnerknoten beinhalten beliebig viele Artefakte und Objekte.
- Rechnerknoten können geschachtelt werden.
- Es gibtfür Geräte und Ausführungsumgebungen eigene Stereotypen, ≪device≫ bzw. ≪execution env≫

siehe auch: Artefakt (2), Kommunikationsverbindung (5)
Andere Diagramme: Komponentendiagramme (Kmp:1)
Beschreibung: Seite 126

Ins:2 Artefakt
Abbildung: A.28
Engl.: artifact
Regeln:
- siehe Artefakt im Komponentendiagramm

Andere Diagramme: Komponentendiagramm (Kmp:5)
Beschreibung: Seite 124

Ins:3 Abhängigkeit
Abbildung: A.28
Engl.: dependency
Regeln:
- Eine spezielle Abhängigkeit (≪deploy≫) verbindet ein Artefakt mit dem Rechnerknoten, auf dem es installiert ist.

siehe auch: Artefakt (2)
Andere Diagramme: Klassendiagramm (Kla:18), Paketdiagramm (2), Use-Case-Diagramm (Use:7), Komponentendiagramm (Kmp:6
Beschreibung: Seite 124

Ins:4 Einsatzbeschreibung
Abbildung: A.28
Engl.: deployment specification
Regeln:
- Ein Objekt das nähere Einzelheiten über den Einsatz (die Installation) eines Artefaktes enthält.

siehe auch: Abhängigkeit 3
Beschreibung: Seite 126

Ins:5 Kommunikationsverbindung
Abbildung: A.28
Engl.: communication association
Regeln:
- Kommunikationsverbindungen verknüpfen Rechnerknoten

- Kommunikationsverbindungen können durch Stereotypen wie ≪TCP/IP≫ oder ≪CORBA≫ charakterisiert werden.

siehe auch: Rechnerknoten (1)
Beschreibung: Seite 126

A.8.3 Kompositionsdiagramm

composite structure diagram

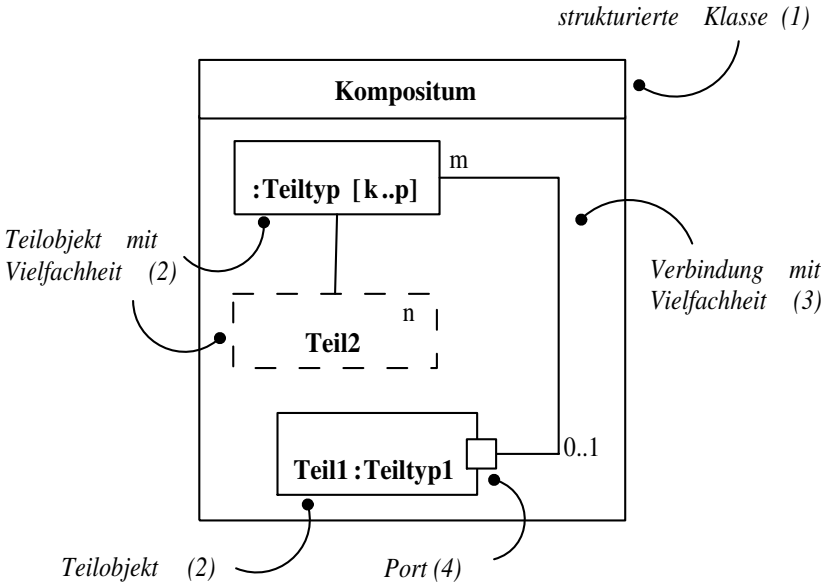

Abbildung A.29. Syntax der Kompositionsdiagramme

Kps:1 strukturierte Klasse
Abbildung: A.29
Engl.: composite structure
Regeln:
- Der Name wird fett geschrieben, er steht im Namensfach.
- Die Komposition ist im 2. Fach beschrieben.
- Auch Komponenten können aus Teilen zusammengesetzt werden.

siehe auch: Verbindung (3)
Andere Diagramme: Klassendiagramm (Kla:1,Kla:7), Komponentendiagramm (Kmp:1)
Beschreibung: Seite 127

Kps:2 Teilobjekt
Abbildung: A.29
Engl.: part
Regeln:
- Der Name wird fett geschrieben, er besteht aus dem Namen des Teilobjekts und dessen Typnamen.
- Die Vielfachheit steht im Kasten oben rechts oder in eckigen Klammern nach dem Typnamen.
- Ein gestrichelter Rand deutet an, dass das Teilobjekt nicht im Besitz des Kompositums ist.

siehe auch: Verbindung (3)
Andere Diagramme: Komponentendiagramm (Kmp:1)
Beschreibung: Seite 127

Kps:3 Verbindung
Abbildung: A.29
Engl.: connector
Regeln:
- Eine Verbindung verbindet Teilobjekte über Ports oder direkt.
- Die Vielfachheit beschränkt die Anzahl der Teile.

siehe auch: Teilobjekt (2), Port (4)
Andere Diagramme: Komponentendiagramm (Kmp:3), Klassendiagramm (Kla:16)
Beschreibung: Seite 127

Kps:4 Port, Anschluss
Abbildung: A.29
Engl.: port
Regeln:
- Anschlusspunkt für andere Teilobjekte.

siehe auch: Teilobjekt (2)
Andere Diagramme: Komponentendiagramm (Kmp:4)
Beschreibung: Seite 130

A.8.4 Kooperationsdiagramm

composite structure diagram

Kop:1 Kooperation
Abbildung: A.30
Engl.: collaboration
Regeln:
- In einer gestrichelter Ellipse wird das Zusammenspiel von Teilobjekten in verschiedenen Rollen beschrieben.

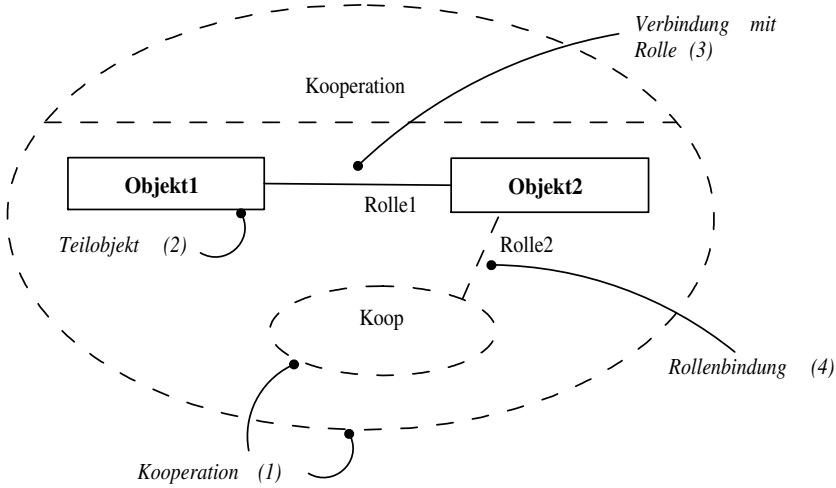

Abbildung A.30. Syntax der Kooperationsdiagramme

siehe auch: Teilobjekt (2)
Andere Diagramme: Klassendiagramm (Kla:15), Kommunikationsdiagramm (Kmk:5), Kompositionsdiagramm (Kps:1)
Beschreibung: Seite 132

Kop:2 Teilobjekt
Abbildung: A.30
Engl.: part
Regeln:
- siehe Kompositionsdiagramm

Andere Diagramme: Kompositionsdiagramm (Kps:2)

Kop:3 Verbindung
Abbildung: A.30
Engl.: connector
Regeln:
- siehe Kompositionsdiagramm
- Zusätzlich wird der Rollenname angegeben.

Andere Diagramme: Kompositionsdiagramm (Kps:3)

Kop:4 Rollenbindung
Abbildung: A.30
Engl.: role binding
Regeln:
- Eine Rollenbindung verknüpft mit einer gestrichelter Line eine Kooperation und deren Teilobjekt.

- Der Rollenname wird angegeben.

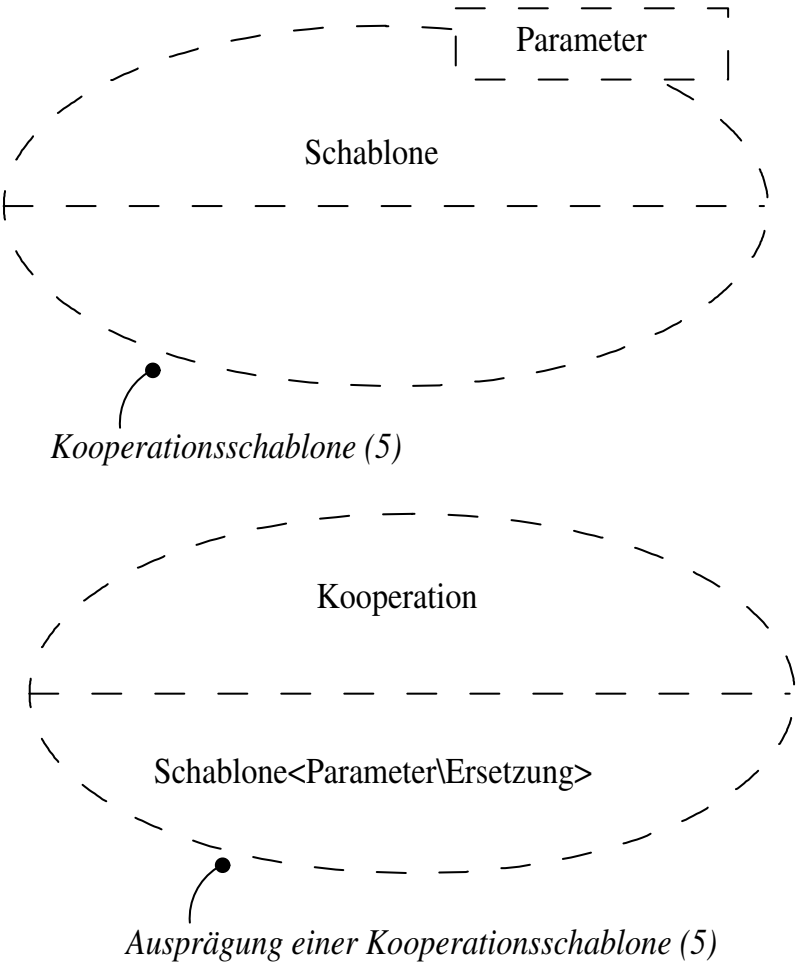

Abbildung A.31. Syntax der Kooperationsdiagramme 2

Kop:5 Kooperationsschablone
Abbildung: A.31
Engl.: collaboration template
Regeln:
- Der oder die Parameter der Kooperation werden in einem gestrichelter Rechteck notiert.

- Bei der Ausprägung werden die Parameter Ersetzungen in spitzen Klammern an den Schablonennamen angehängt.

siehe auch: Kooperation (1)
Andere Diagramme: Klassendiagramm (Kla:17)
Beschreibung: Seite 132

A.9 Paketdiagramm

package diagram

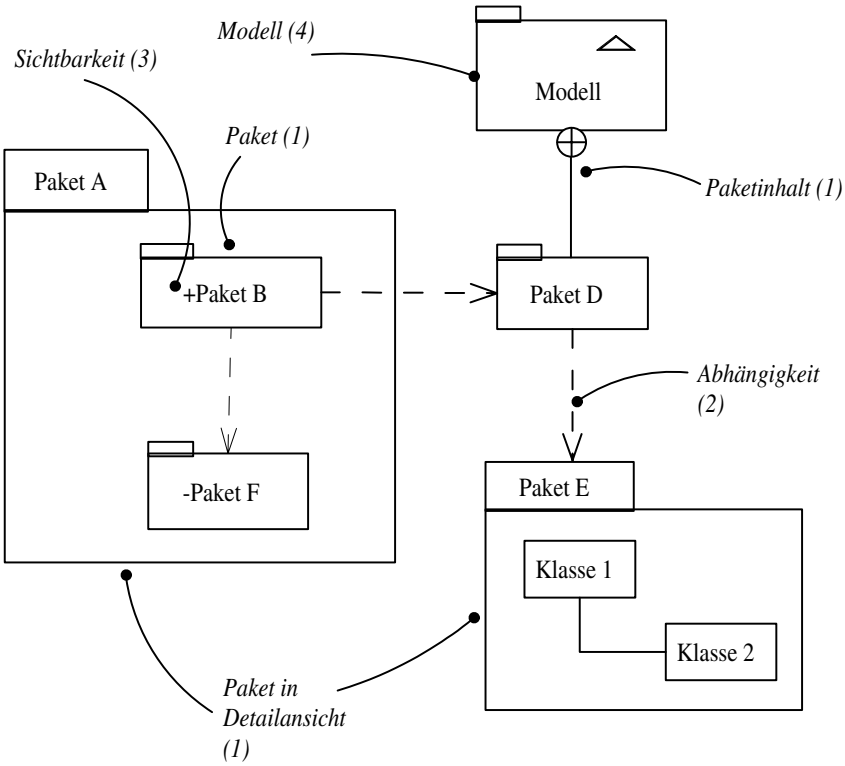

Abbildung A.32. Syntax der Paketdiagramme 1

Pak:1 Paket
Abbildung: A.32
Engl.: package
Regeln:

- Ein Paket hat einen Namen.
- Pakete können unter anderem folgende Modellelemente enthalten:
  - Pakete
  - Klassen
  - Interfaces
  - Objekte
  - Akteure
  - Use-Cases
  - Generalisierungen
  - Assoziationen
  - Links
  - Abhängigkeitsbeziehungen
- Jedes Modellelement ist in höchstens einem Paket enthalten.
- Der Inhalt eines Paketes kann direkt in den Kasten gezeichnet oder durch eine baumartige Notation mit einem geviertelten Kreis als Anker angegeben werden.
- Der Paketname wird dem des enthaltenen Modellelements vorangestellt und mit ":: " abgetrennt.

siehe auch: Abhängigkeitsbeziehungen (2)
Beschreibung: Seite 138

Pak:2 Abhängigkeit (zwischen Paketen)
Abbildung: A.32
Engl.: dependency
Regeln:
- Abhängigkeitsbeziehungen verbinden zwei Pakete miteinander.
- Zur Beschreibung von Abhängigkeitsbeziehungen gibt es vordefinierte Stereotypen wie ≪import≫ , ≪access≫ und ≪merge≫

siehe auch: Paket (1)
Andere Diagramme: Implementierungsdiagramme (6), Use-Case Diagramm (7), Klassendiagramm (Kla:18)
Beschreibung: Seite 138

Pak:3 Sichtbarkeit
Abbildung: A.32
Engl.: visibility
Regeln:
- Die Sichtbarkeit eines Elementes in einem Paket wird durch folgende Zeichen vor dem Namen des Elementes gekennzeichnet:
- +: public
- −: private

siehe auch: Paket (1)
Andere Diagramme: Klassendiagramm (Kla:4)
Beschreibung: Seite 140

Pak:4 Modell

Abbildung: A.32
Engl.: model
Regeln:
- Ein Modell ist als grafischer Stereotyp für Pakete definiert.

siehe auch: Paket (1)
Beschreibung: Seite 144

A.10 Die OCL-Syntax

Die OCL Syntax wird hier nur verkürzt wiedergegeben, um einen Überblick über die möglichen Ausdrücke zu erhalten.

```
constraint := contextDeclaration
              (stereotype ":" expression)*
contextDeclaration := "context"
                      (clasifierContext | operationContext)
classifierContext := <typeName>
operationContext := <typeName> "::" <name>
                    "(" formalParameterList? ")"
                    ( ":" <typeName> )?
formalParameterList := formalParameter (";" formalParameter)*
formalParameter := <name> ":" <typeName>
stereotype := "inv" | "pre" | "post"

Expression := VariableExp | LiteralExp | IteratorExp |
              LetExp | OclMessageExp | IfExp
VariableExp := simpleName
LiteralExpCS := pathName "::" simpleEnumName |
                CollectionLiteralExp | PrimitiveTupelLit
CollectionLiteralExp :=
( "Set"| "Bag"| "Sequence"| "Collection"| "OrderedSet")
 CollectionRange ( "," CollectionRange )*
IteratorExp := Expression "->" (simpleName|"iterate")
"(" (VariableDeclaration,((","|";") VariableDeclaration)? "|" )?
    Expression")"
IteratorExp := Expression "." simpleName ("("arguments?")")?
LetExp := "let" VariableDeclaration ("," VariableDeclaration)*
OclMessageExp := Expression ("^^"|"^") simpleName
 "(" OclMessageArg ("," OclMessageArg)*")"
IfExp := "if" Expression "then" Expression
  "else" Expression "endif"
```

A.11 Vordefinierte Stereotypen

Stereotyp:	betrifft:
≪access≫	Pakete
≪call≫	Abhängigkeit
≪create≫	Nachricht
≪create≫	Abhängigkeit
≪derive≫	Abhängigkeit
≪destroy≫	Nachricht
≪document≫	Komponente
≪executable≫	Komponente
≪file≫	Komponente
≪framework≫	Pakete
≪friend≫	Abhängigkeit zwischen Klassen
≪global≫	Verknüpfung
≪implement≫	Komponente
≪implementation≫	Generalisierung
≪implementationClass≫	Klasse
≪import≫	Abhängigkeit zwischen Paketen
≪instantiate≫	Abhängigkeit
≪invariant≫	Bedingungen
≪library≫	Komponente
≪metaclass≫	Klasse
≪metamodel≫	Modell
≪modelLibrary≫	Modell
≪postcondition≫	Bedingung
≪precondition≫	Bedingung
≪process≫	Klasse
≪realization≫	Klasse
≪realize≫	Abhängigkeit
≪refine≫	Abhängigkeit
≪requirement≫	Kommentar
≪responsibility≫	Kommentar
≪script≫	Komponente
≪send≫	Abhängigkeit
≪systemModel≫	Modell

→

Stereotyp:	betrifft:
≪trace≫	Abhängigkeit
≪type≫	Klasse
≪utility≫	Klasse

A.12 Vordefinierte Bedingungen

Bedingung	betrifft:
{association}	Verknüpfung
{complete}	Generalisierung
{destroyed}	Objekt
{disjoint}	Generalisierung
{global}	Verknüpfung
{implicit}	Assoziation
{incomplete}	Generalisierung
{local}	Verknüpfung
{new}	Objekt
{overlapping}	Generalisierung
{parameter}	Verknüpfung
{self}	Verknüpfung
{transient}	Objekt
{xor}	Assoziation

Anhang B

Inhalt der CD-ROM

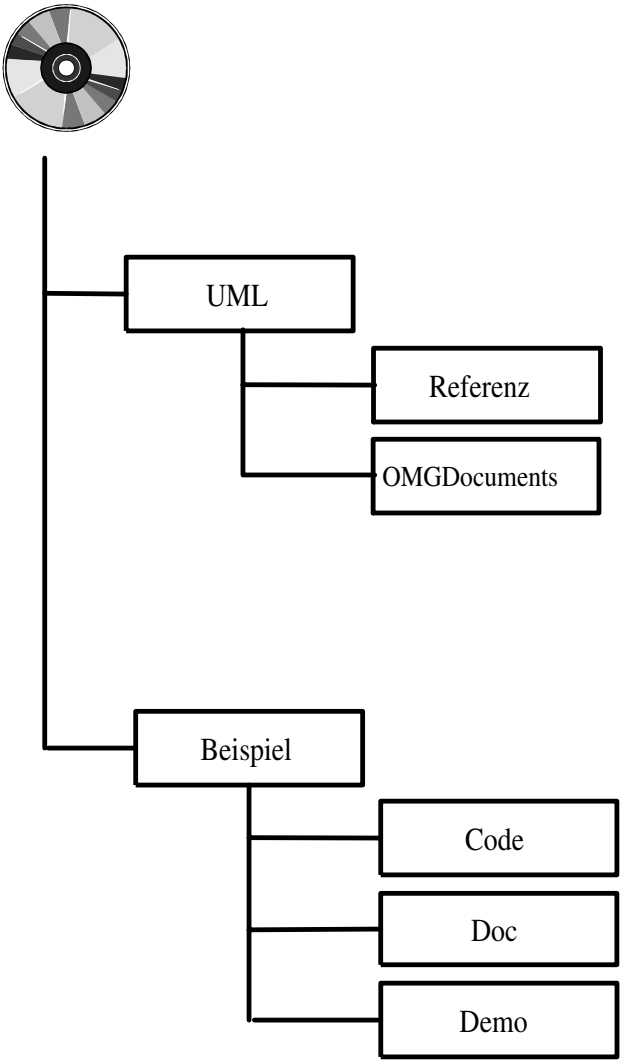

Abbildung B.1. Verzeichnisse auf der CD-ROM

Literatur

[1] C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

[2] K. Beck and W. Cunningham. *A Laboratory For Teaching Object-Oriented Thinking*, volume 24. SIGPLAN Notices, 1989.

[3] G. Booch. *Object-Oriented Analysis and Design with Applications*. Benjamin/Cummings, Redwood City/CA, 1994.

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-orientierte Software Architektur*. Addison-Wesley-Longman, Bonn, 1998.

[5] P. Coad and E. Yourdon. *Object-Oriented Analysis*. Yourdon Press, Prentice-Hall, Englewood Cliffs/NJ, 1991.

[6] P. Coad and E. Yourdon. *Object-Oriented Design*. Yourdon Press, Prentice-Hall, Englewood Cliffs/NJ, 1991.

[7] D. Flanagan. *Java in a Nutshell*. O'Reilly, Sebastopol/CA, 1996.

[8] M. Fowler. *Analyse Muster*. Addison-Wesley, Bonn, 1999.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading/MAS, 1995.

[10] IBM Object-Oriented Technology Center. *Developing Object-Oriented Software, An Experience-Based Approach*. Prentice-Hall, Englewood Cliffs/NJ, 1997.

[11] I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. *Object-Oriented Software-Engineering*. Addison-Wesley, Reading/MAS, 1992.

[12] M. Jeckle, C. Rupp, J. Hahn, B. Zengler, S. Queins *UML 2 glasklar*. Hanser, München, 2004.

[13] C. Larman. *Applying UML and Patterns*. Prentice-Hall, Upper Saddle River/NJ, 1998.

[14] D. Lea. *Concurrent Programming in Java - Design Principles and Patterns*. Addison-Wesley, Reading/MAS, 1997.

[15] J. Martin and J. Odell. *Object-Oriented Analysis and Design*. Prentice-Hall, Englewood Cliffs/NJ, 1992.

[16] B. Meyer. *Objektorientierte Software-Entwicklung*. Hanser, München, 1988.

[17] G. Müller-Ettrich. *Objektorientierte Prozessmodelle*. Addison-Wesley, Bonn, 1999.

[18] Object Management Group. *Model Driven Architecture*. www.omg.org/mda .

[19] Object Management Group. *OCL 2.0 Specification* . www.omg.org, ptc/05-06-06.

[20] Object Management Group. *UML Specification 1.1*. www.omg.org, 1997.

[21] Object Management Group. *Unified Modeling Language Superstructure, version 2.0*. www.omg.org, formal/05-07-04 2005.
[22] Object Management Group. *Unified Modeling Language Infrastructure, version 2.0*. www.omg.org, ptc/04-10-14 2004.
[23] Rational Software Inc. *UML Specification 1.1*. www.rational.com, 1997.
[24] Rational Software Inc. *Rational Objectory Process*. www.rational.com/CD-ROM, 1999.
[25] Rational Software Inc. *UML Specification 1.3 (draft)*. www.rational.com, 1999.
[26] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs/NJ, 1991.
[27] S. Shlaer and S. Mellor. *Object Lifecycles: Modeling the World in States*. Prentice-Hall, Englewood Cliffs/NJ, 1991.
[28] R. Wirfs-Brock, B. Wilkerson, and L. Wiener. *Designing Object-Oriented Software*. Prentice-Hall, Englewood Cliffs/NJ, 1990.

Index