Vorlesung „Modellierung“

Prof. Janis Voigtländer

Wintersemester 2017/18

von Petrinetzen zu UML

Bisher haben wir uns mit Zustandsübergangsdiagrammen und Petrinetzen (sowie deren Erreichbarkeitsgraphen und verwandten Konzepten) beschäftigt.

Dabei handelte es sich um Modellierung von

- dynamischen Aspekten,
- in qualitativer und quantitativer Hinsicht,
- einer breiten Klasse von Systemen,
- unter „white box“-Sicht,
- mit formaler Syntax und Semantik.

Bei UML handelt es sich um eine ganze Familie von Modellierungsmitteln, daher wird einiges anders, teils insbesondere allgemeiner.

von Petrinetzen zu UML

UML beinhaltet Mittel zur

- auch vor allem visuell-grafischer statt textuell-mathematischer Modellierung von
- sowohl statischen als auch dynamischen Aspekten,
- in qualitativer und quantitativer Hinsicht,
- von vor allem (objekt-orientierten) Software-Systemen,
- unter „white box“- oder „black box“-Sicht,
- mit bestenfalls semi-formaler (Syntax und) Semantik,
- zur Verwendung bei Softwareentwicklung „im Großen“ mit strukturierten Entwicklungsprozessen.

UML: Einführung und Objekt-Orientierung

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

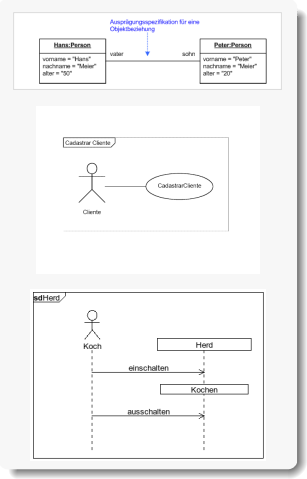
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

UML: Unified Modeling Language

UML = Unified Modeling Language

- Standard-Modellierungssprache für Software Engineering.
- Basiert auf objekt-orientierten Konzepten.
- Sehr umfangreich, enthält viele verschiedene Typen von Modellen.
- Entwickelt von Grady Booch, James Rumbaugh, Ivar Jacobson (1997).



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

UML: Einführung I

Einsatzgebiete von UML im Software-Engineering

- Visualisierung
- Spezifikation
- Konstruktion (z.B. zur Codegenerierung)
- Dokumentation

UML: Einführung II

Vokabular der UML (nach Booch/Rumbaugh/Jacobson):

Dinge

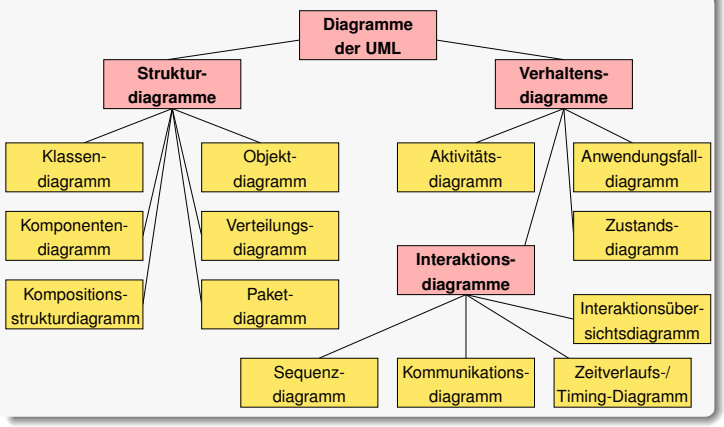
- Strukturen (structural things)
- Verhalten (behavioral things)
- Gruppen (grouping things)
- Annotationen (annotational things)

Beziehungen (relationships)

- Abhängigkeiten (dependencies)
- Assoziationen (associations)
- Generalisierungen (generalizations)
- Realisierungen (realizations)

UML: Einführung III

Diagramme



Wir werden im Folgenden einige dieser Begriffe mit Leben füllen.

UML und Objekt-Orientierung

Grundidee der Objekt-Orientierung

Vereinfacht besteht die Welt aus Objekten, die untereinander in Beziehungen stehen. Diese Sichtweise wird auch auf Modellierung und Softwareentwicklung übertragen.

Etwas genauer . . .

Daten (= Attribute) werden zusammen mit der Funktionalität (= Methoden) in Objekten organisiert bzw. gekapselt.

Jedes Objekt ist in der Lage, Nachrichten (= Methodenaufrufe) zu empfangen, Daten zu verarbeiten und Nachrichten zu senden.

Diese Objekte, bzw. die Objekttypen, können – einmal realisiert – in verschiedenen Kontexten wiederverwendet werden.

UML und Objekt-Orientierung

Geschichte der Objekt-Orientierung

- Entwicklung von objekt-orientierten Programmiersprachen:
  - 60er Jahre: Simula (zur Beschreibung und Simulation von komplexen Mensch-Maschine-Interaktionen)
  - 80er Jahre: C++
  - 90er Jahre: Java
- Verbreitung von objekt-orientierten Entwurfsmethoden:
  - 70er Jahre: Entity-Relationship-Modell
  - 90er Jahre: Vorläufer von UML: OOSE (Object-Oriented Software Engineering), OMT (Object Modeling Technique)
  - Seit 1997: UML
  - Seit 2005: UML 2.0

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

UML und Objekt-Orientierung

Behauptete Vorteile der objekt-orientierten Modellierung und Programmierung:

- Leichte Wiederverwendbarkeit dadurch, dass Daten und Funktionalität zusammen verwaltet werden und es Konzepte zur Modifikation von Verhalten gibt (Stichwort: Vererbung).
- Verträglichkeit mit Nebenläufigkeit und Parallelität: Kontrollfluss kann nebenläufig in verschiedenen Objekten ablaufen und diese können durch Nachrichtenaustausch bzw. Methodenaufrufe miteinander kommunizieren.
- Nähe zur realen Welt: viele Dinge der realen Welt können als Objekte modelliert werden.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

UML und Objekt-Orientierung

Ein Beispiel für die Modellierung von Objekten der realen Welt:

Fahrkartenautomat

- Daten: Fahrziele, Zoneneinteilung, Fahrtkosten
- Funktionalität: Tasten drücken, Preise anzeigen, Münzen einwerfen, Fahrkarten auswerfen

UML und Objekt-Orientierung

Konzepte

- Klasse: definiert einen Typ von Objekten mit bestimmten Arten von Daten und bestimmter Funktionalität.

  Beispiel: die Klasse der VRR-Fahrkartenautomaten
- Objekt: eine Instanz einer Klasse

  Beispiel: der Fahrkartenautomat am Duisburger Hauptbahnhof, Osteingang

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Klassen- und Objektdiagramme

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Klassen- und Objektdiagramme

Wir beginnen mit Klassen- und Objektdiagrammen.

Dabei geht es um statische Modellierung:

- Dinge, ihre Eigenschaften, und Beziehungen zwischen ihnen;
- wie sich der Zustand eines Systems jeweils zusammensetzt, nicht wie/wohin er sich entwickeln kann.

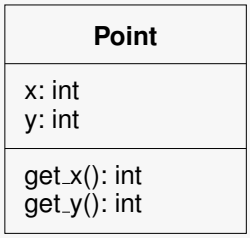
Das klingt weniger spannend als Modellierung des dynamischen Verhaltens eines Systems (etwa mittels Petrinetzen), aber:

- präzise statische Modellierung ist wichtige Hilfe für Implementierung größerer Software-Systeme,
- und erlaubt die Anwendung von anerkannten (aus Erfahrung gewonnenen) Design-Prinzipien, etwa zum angemessenen Einsatz von Vererbung.

Klassen- und Objektdiagramme

Beispiel: Klasse von Punkten mit x-, y-Koordinaten (also zweidimensional) und Operationen zum Auslesen der Koordinaten

Grafische Darstellung einer Klasse



Klassenname

Attribute (evtl. mit Typ)

Operationen/Methoden

Bemerkung: Obwohl ja auch Aktivitäten aufgeführt werden (etwa get_x), handelt es sich dennoch um statische Modellierung. (Warum?)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Attribute & Operationen

Weitere Bemerkungen:

- Bei den Attributen handelt es sich um sogenannte Instanzattribute, das heißt, sie gehören zu den Instanzen einer Klasse (zu den Objekten, nicht zur Klasse selbst).
- Man kann die Sichtbarkeit eines Attributes bzw. einer Methode spezifieren, indem man bestimmte Modifikatoren ($+$, $-$, $\#$, $\sim$) vor den Attribut-/Methodennamen schreibt.
- Attribute haben im Allgemeinen Typen, manchmal auch Vorgabewerte (= initiale Werte). Dies wird dann folgendermaßen notiert: x: int = 0
- Operationen mit Argumenten (und Rückgabewert) werden mit ihren Typen folgendermaßen notiert: add(m: int, n: int): int

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

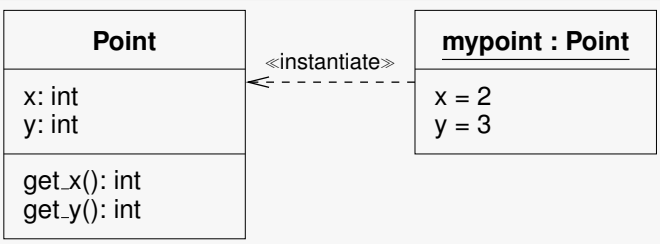
UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Instanziierung

Eine Klasse beschreibt den allgemeinen Aufbau bestimmter Objekte.

Eine Instanz einer Klasse stellt ein konkretes Objekt mit den entsprechenden Werten für die Attribute dar.

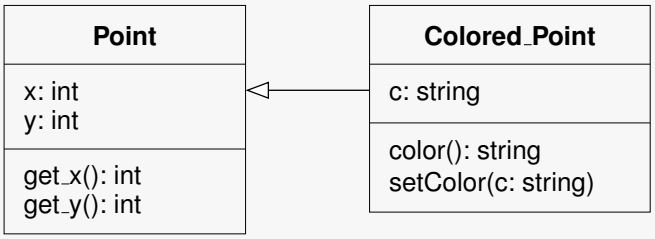
Grafische Darstellung einer Instanz (Objekt) einer Klasse



Generalisierung/Spezialisierung & Polymorphie

Es ist möglich, neue Klassen von bestehenden Klassen erben zu lassen. (Generalisierung/Spezialisierung)

Grafische Darstellung von Vererbung



In einer solchen Situation nennt man **Point** eine Superklasse bzw. Oberklasse und **Colored_Point** eine Subklasse bzw. Unterklasse.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Generalisierung/Spezialisierung & Polymorphie

Bemerkungen:

- Die Subklasse erbt die Attribute, Methoden und Assoziationen der Superklasse, und kann diesen noch weitere hinzufügen. Außerdem kann man in der Subklasse Methoden der Superklasse überschreiben, also durch neue ersetzen.
- Ein Objekt einer Unterklasse kann auch als ein Objekt jeder seiner Oberklassen angesehen werden. $\Rightarrow$ Polymorphie
- Dabei sollte man darauf achten, dass sich das Verhalten eines Programms bei solch einer Substitution nicht ändert. (Liskovs Substitutions-Prinzip)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

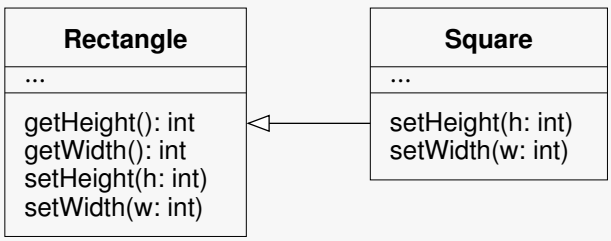
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Generalisierung/Spezialisierung & Polymorphie

Beispiel für mögliche Verletzung des Substitutions-Prinzips

Scheinbar sinnvolle Vererbungsbeziehung:



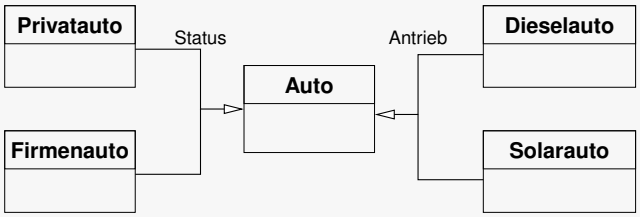
wobei die in der Unterklasse überschriebenen Methoden das Quadratisch-Sein erzwingen.

Was passiert, wenn wir in dem für **Rectangle**s gedachten (und getypten) Code: o.setHeight(10); o.setWidth(20); print(o.getHeight()); ein **Square**-Objekt einsetzen?

Generalisierung/Spezialisierung & Polymorphie

Generalisierungsgruppen

Mitunter können Klassen in unterschiedlicher Weise spezialisiert bzw. unterteilt werden. Einzelne Generalisierungsbeziehungen können dann zu Gruppen zusammengefasst werden.



Dabei wird die jeweilige Generalisierungsgruppe (hier: Status bzw. Antrieb) im Diagramm annotiert.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Generalisierung/Spezialisierung & Polymorphie

Den Generalisierungsgruppen können Eigenschaften (in geschweiften Klammern) zugeordnet werden.

- complete/incomplete:
  - complete: die Generalisierungsgruppe ist vollständig, das heißt, sie überdeckt konzeptionell alle denkbaren Instanzen der Oberklasse.
  - incomplete: die Generalisierungsgruppe ist unvollständig, das heißt, es gibt durch sie nicht erfasste Instanzen.
- overlapping/disjoint:
  - overlapping: es sind Instanzen denkbar, die konzeptionell zu mehr als einer der spezialisierenden Klassen gehören könnten.
  - disjoint: die spezialisierenden Klassen überlappen sich konzeptionell nicht.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

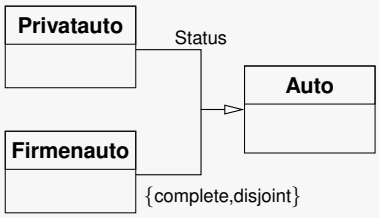
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Generalisierung/Spezialisierung & Polymorphie

Beispiele für die Eigenschaften complete/incomplete und disjoint/overlapping:

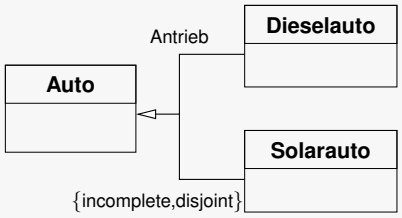
{complete,disjoint}



Hier handelt es sich um eine konzeptionelle Partitionierung der Instanzen der Oberklasse.

Generalisierung/Spezialisierung & Polymorphie

{incomplete,disjoint}



Warum unvollständig? ⇝ Es fehlt z.B. eine Klasse Benzinauto.

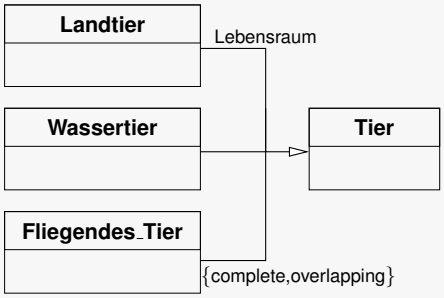
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Generalisierung/Spezialisierung & Polymorphie

{complete,overlapping}



Schildkröten sind sowohl Land- als auch Wassertiere.

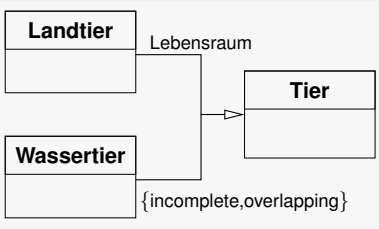
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Generalisierung/Spezialisierung & Polymorphie

{incomplete,overlapping}



Fliegende Tiere fehlen.

Beziehungen zwischen Klassen

Neben Vererbung (also Generalisierung/Spezialisierung) sind noch andere Beziehungen zwischen Klassen (und letztlich zwischen deren Objekten) in UML darstellbar.

Wir betrachten folgende Arten von Beziehungen:

- Assoziation
- Aggregation
- Komposition

Die Art der Beziehung drückt die jeweilige Stärke der Verbindung aus.

Komposition *ist_stärker_als* Aggregation, und
Aggregation *ist_stärker_als* Assoziation.

Beziehungen zwischen Klassen: Assoziation

Wir beginnen mit der schwächsten Beziehung: der Assoziation.

Assoziation

Es gibt eine Assoziation zwischen den Klassen **A** und **B**, wenn

> es irgendeinen semantischen, nicht-hierarchischen Zusammenhang zwischen den Klassen gibt, der sinnvoll benannt werden kann.

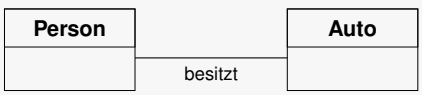
Letztlich drückt eine Assoziation nur aus, dass Objekte der einen Klasse für zumindest einen Teil ihrer Funktionalität eine persistente (abzuspeichernde) Verbindung zu bestimmten Objekten der anderen Klasse brauchen.

Üblicherweise werden Assoziationen durch Referenzen realisiert. Das heißt, eine der Klassen hat ein Attribut vom Typ der anderen Klasse. Diese Attribute werden im Klassendiagramm jedoch nicht explizit angegeben.

Beziehungen zwischen Klassen: Assoziation

Beispiel für eine Assoziation

Eine Person kann ein Auto besitzen.



Obige Angabe schließt bewusst nicht aus, dass eine Person auch mehrere, oder gar kein Auto, besitzen könnte. Oder dass ein Auto von mehreren Personen besessen werden könnte.

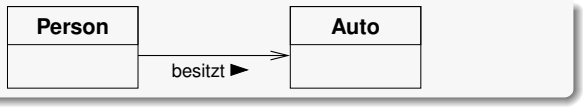
Mögliche Ausprägungen auf der Ebene von Objekten wären also etwa:

1. $\{(person_1, auto_1), (person_2, auto_2), (person_3, auto_3)\}$
2. $\{(person_1, auto_1), (person_1, auto_2), (person_3, auto_3)\}$
3. $\{(person_1, auto_1), (person_1, auto_2), (person_2, auto_2)\}$

Beziehungen zwischen Klassen: Assoziation

Oft wird eine Leserichtung der Assoziation eingeführt:

Separat kann eine Navigationsrichtung eingeführt werden, die beschreibt, welche Klasse ihren Assoziationspartner kennt (und daher seine Methoden aufrufen kann):



Hier hätten also **Person**-Objekte Referenzen auf **Auto**-Objekte. Mengentheoretisch ausgedrückt, zum Beispiel:

2. $person_1 \mapsto \{auto_1, auto_2\}$, $person_2 \mapsto \emptyset$, $person_3 \mapsto \{auto_3\}$

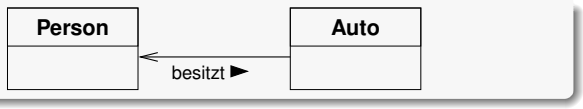
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

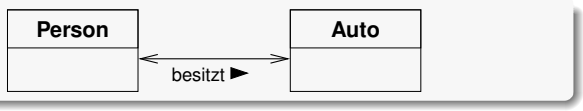
Beziehungen zwischen Klassen: Assoziation

Die Navigationsrichtung kann im Prinzip von der Leserichtung abweichen:



Allerdings ist das ungewöhnlich und deutet auf einen Modellierungsfehler hin.

Hier kennen sich Vertreter beider Klassen gegenseitig:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beziehungen zwischen Klassen: Assoziation

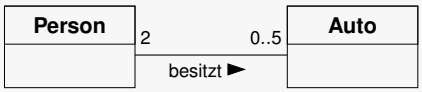
An beiden Enden einer Assoziation können Multiplizitäten in Form von Intervallen *m..n* (oder einfach nur *m* für *m..m*) angegeben werden.

Hier besitzt jede Person bis zu fünf Autos. Und jedes Auto ist im Besitz genau einer Person.

Falls die Multiplizität (am anderen Ende einer navigierbaren Assoziation) größer als Eins möglich ist, muss dies in der Implementierung durch eine Kollektion (Liste, Menge, Array) von Referenzen realisiert werden.

Beziehungen zwischen Klassen: Assoziation

Was bedeuten zum Beispiel folgende Multiplizitäten?



Jedenfalls nicht, dass immer zwei Personen zusammen genau die gleichen Autos besitzen müssen.

Möglich wäre etwa folgende Situation:

$$\{(person_1, auto_1), (person_2, auto_1), (person_1, auto_2), (person_3, auto_2)\}$$

und noch Existenz weiterer Personen, aber nicht weiterer Autos.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

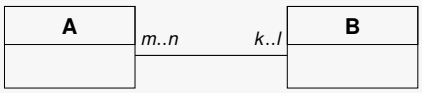
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beziehungen zwischen Klassen: Assoziation

Multiplizitäten allgemein:

In folgendem Diagramm können i Instanzen von **A**, mit $m \leq i \leq n$, mit einer Instanz von **B** assoziiert sein. Umgekehrt können j Instanzen von **B**, mit $k \leq j \leq l$, mit einer Instanz von **A** assoziiert sein.



Falls es keine obere Schranke geben soll, wird ein Stern (= unendlich) verwendet.
Beispielsweise steht $2..*$ für „mindestens zwei“.

Der UML-Standard gibt keine Standardmultiplizität vor.
Für die folgenden Diagramme wird, wenn keine Angabe vorhanden ist, die Multiplizität $0..*$ als Standard angenommen.

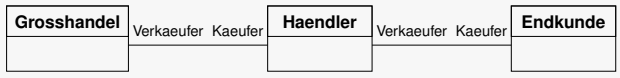
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beziehungen zwischen Klassen: Assoziation

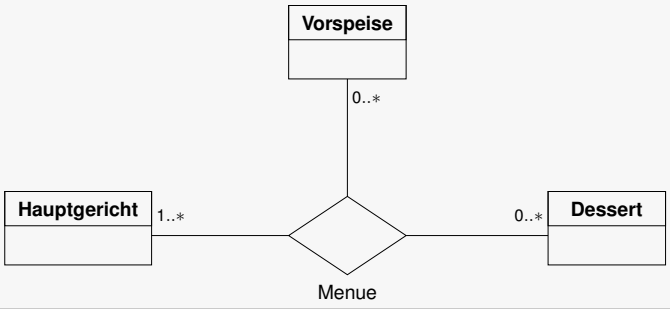
Klassen können in verschiedenen Assoziationen verschiedene Rollen spielen. Rollen werden auch an den Assoziationen notiert (und machen es manchmal überflüssig, die Assoziation selbst zu benennen oder eine Leserichtung anzugeben).



Beziehungen zwischen Klassen: Assoziation

n-äre Assoziation

Neben binären (zweistelligen) Assoziationen gibt es auch n-äre Assoziationen, die eine Beziehung zwischen $n > 2$ Klassen beschreiben.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beziehungen zwischen Klassen: Aggregation

Die nächste Beziehung – Aggregation – ist etwas stärker.

Aggregation

Es gibt eine Aggregation zwischen den Klassen **A** und **B**, wenn

> Instanzen der Klasse **A** Instanzen der Klasse **B** als Teile enthalten. (Ein „Ganzes“ enthält mehrere „Teile“.)

Dabei ist durchaus denkbar, dass ein Teil zu mehreren Ganzen gehört.

Eine explizite Benennung ist oft überflüssig, da aus der Angabe als Aggregation bereits die Natur der Beziehung folgt.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

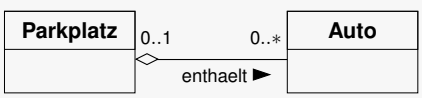
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beziehungen zwischen Klassen: Aggregation

Beispiel für eine Aggregation (mit Benennung)

Ein Parkplatz „enthält“ mehrere Autos.



Jedoch existiert ein Auto nicht nur solange es auf einem Parkplatz steht.

Beziehungen zwischen Klassen: Komposition

Die stärkste Beziehung ist die Komposition.

Komposition

Es gibt eine Komposition zwischen den Klassen **A** und **B**, wenn

Instanzen der Klasse **A** Instanzen der Klasse **B** als Teile enthalten und die Lebenszeit der Teile vom „Ganzen“ kontrolliert wird.

Das heißt, die Teile können (oder müssen sogar) gelöscht werden, sobald die zugehörige Instanz der Klasse **A** gelöscht wird.

Außerdem (oder eigentlich wegen Obigem) darf ein Teil nicht gleichzeitig zu mehr als einem Ganzen gehören.

Auch hier ist eine explizite Benennung oft überflüssig.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

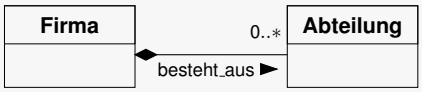
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beziehungen zwischen Klassen: Komposition

Beispiel für eine Komposition (mit Benennung)

Eine Firma besteht aus einer beliebigen Anzahl Abteilungen.



Die Abteilungen existieren nicht mehr, sobald die Firma nicht mehr existiert.

Bemerkung: Bei einer Komposition darf die Multiplizität, die an der schwarzen Raute stünde, nur 0..1 oder 1 sein. Am üblichsten ist 1, dementsprechend wird in dem Fall an diesem Ende gar keine Multiplizität angegeben: jedes Teil gehört zu genau einem Ganzen.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beziehungen zwischen Klassen

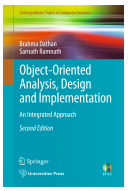
„Merksätze“ (aus: Dathan & Ramnath, Object-Oriented Analysis, Design and Implementation – An Integrated Approach, siehe Literaturfolien zu Beginn des Semesters):

- An association normally represents something that will be stored as part of the data and reflects all links between objects of two classes that may ever exist. It describes a relationship that will exist between instances at run time and has an example.
- . . . , associations should be shown if a class possesses, controls, is connected to, is related to, is a part of, has as parts, is a member of, or has as members some other class in the system.
- . . . association should not be used to denote relationships that: (i) can be drawn as a hierarchy, (ii) stems from a dependency alone, (iii) or relationships whose links will not survive beyond the execution of any particular operation.

Beziehungen zwischen Klassen

„Merksätze“ (aus: Dathan & Ramnath, Object-Oriented Analysis, Design and Implementation – An Integrated Approach, siehe Literaturfolien zu Beginn des Semesters):

- Aggregation is a kind of association where the object of class A is ‘made up of’ objects of class B. This suggests some kind of whole-part relationship between A and B.
- Composition implies that each instance of the part belongs to only one instance of the whole, and that the part cannot exist except as part of the whole.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

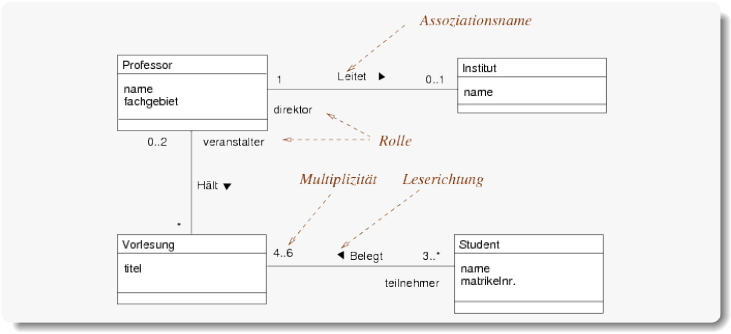
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beziehungen zwischen Klassen

In Klassendiagrammen befinden sich normalerweise nicht nur zwei Klassen mit irgendeiner Beziehung, sondern viele verschiedene Klassen eines Programms oder Moduls, mit ihren diversen Beziehungen untereinander.

Beispiel:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

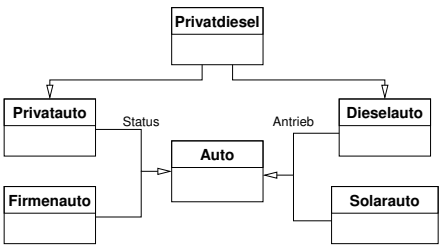
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beziehungen zwischen Klassen

Geeigneter Einsatz von Assoziation / Aggregation / Komposition kann auch Vererbungen überflüssig machen.

Zum Beispiel ist konzeptionell denkbare Mehrfachvererbung:

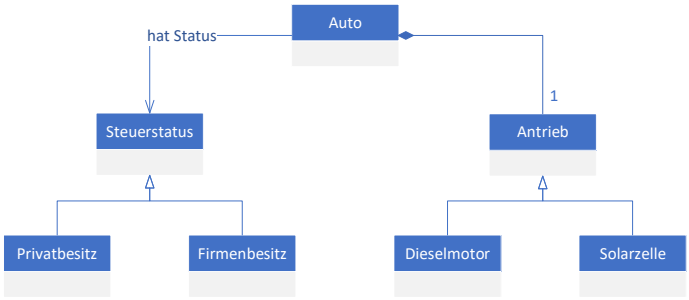


in der Praxis eher problematisch.

Eine alternative Modellierung ist hier möglich . . .

Beziehungen zwischen Klassen

. . . durch Repräsentation der Aspekte „steuerlicher Status“ und „Antrieb“ in eigenen Klassen, und deren Verwendung über Assoziation / Aggregation / Komposition, etwa wie folgt:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Systemmodellierung

Vor Implementierung eines objekt-orientierten Systems stellen sich bei der Modellierung insbesondere folgende Fragen:

- Welche Objekte und Klassen werden benötigt?
- Welche Merkmale haben diese Klassen und welche Beziehungen bestehen zwischen Ihnen?
- Welche Methoden stellen diese Klassen zur Verfügung? Wie wirken diese Methoden zusammen?
- In welchen Zuständen können sich Objekte befinden und welche Nachrichten werden wann an andere Objekte geschickt?

Zur Beantwortung kennt die Literatur bestimmte mehr oder weniger systematische Vorgehensweisen.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beispiel: Modellierung einer Bibliothek

Am Beispiel, Klassen finden: Use Case “Register New Member”

Actions performed	System responses
1. The customer fills out an application form containing the customer’s name, address and phone number, and gives this to the clerk.	
2. The clerk issues a request to add a new member.	
	3. The system asks for data about the new member.
4. The clerk enters the data into the system.	
	5. Reads in the data, and if the member can be added, generates an identification number and remembers information about the member. Informs the clerk whether the member was added and outputs the member’s name, address, phone and id number.
6. The clerk gives the user their identification number.	

Beispiel: Modellierung einer Bibliothek

An Naivität kaum zu übertreffen: Alle Nomen heraussuchen

Actions performed	System responses
1. The **customer** fills out an **application form** containing the **customer's name**, **address** and **phone number**, and gives this to the **clerk**.	
2. The **clerk** issues a **request** to add a new **member**.	
	3. The **system** asks for **data** about the new **member**.
4. The **clerk** enters the **data** into the **system**.	
	5. Reads in the **data**, and if the **member** can be added, generates an **identification number** and remembers **information** about the **member**. Informs the **clerk** whether the **member** was added and outputs the **member's name**, **address**, **phone** and **id number**.
6. The **clerk** gives the **user** their **identification number**.	

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beispiel: Modellierung einer Bibliothek

Bereinigung unserer Wortsammlung:

- Als (zusammengesetzte) Einheiten stechen hervor: **member**, **system**
- Bezüglich der anderen vorkommenden Nomen:
  - **customer** – wird ein **member**, ist also ein Synonym
  - **user** – ein weiteres Synonym
  - **application form** – ist ein externes Konstrukt zur Informationsabfrage
  - **request** – nur ein Menüeintrag, wird wie **application form** nicht Teil einer Datenstruktur sein
  - **customer's name**, **address**, **phone number** – Attribute von **member**
  - **clerk** – lediglich ein Akteur, hat keine Repräsentation in Software
  - **identification number** – wird Teil von **member**
  - **data**, **information** – werden als **member** gespeichert

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

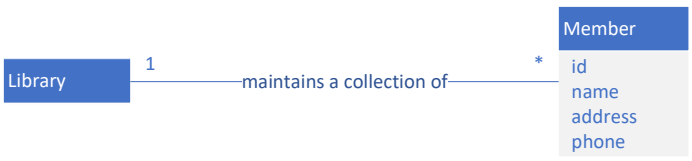
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beispiel: Modellierung einer Bibliothek

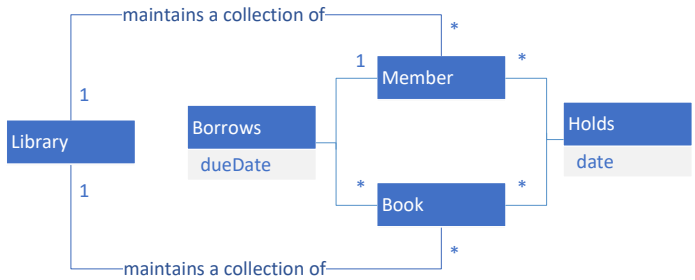
Ermittlung von Beziehungen:

"... [the **system**] remembers **information** about the **member**"



Beispiel: Modellierung einer Bibliothek

Informiert durch weitere Use Cases:



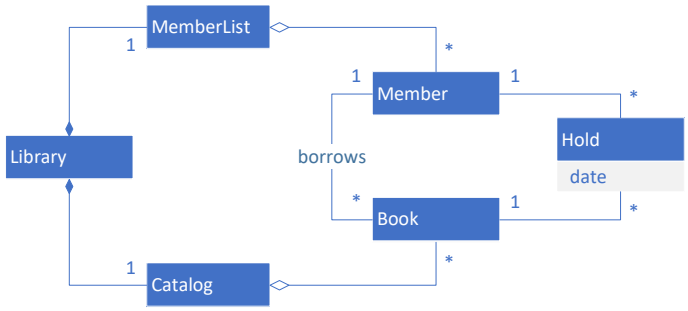
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beispiel: Modellierung einer Bibliothek

Verfeinert für die Implementierung:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beispiel: Modellierung einer Bank

Ein weiteres Beispiel für objekt-orientierte Modellierung:
Wir modellieren eine Bank.

Folgende Anforderungen werden gestellt:

- Eine Bank
  - hat mehrere Kunden
  - und mehrere Angestellte
  - und führt eine Menge von Konten.
- Konten können Giro- oder Sparkonten sein. (Ein Sparkonto wirft höhere Zinsen ab, darf aber nicht ins Minus absinken.)
- Auf den Konten sollen folgende Operationen ausgeführt werden können:
  - Einzahlen
  - Abheben
  - Umbuchen
  - Verzinsen

Beispiel: Modellierung einer Bank

Klasse Bank:

Bank
name: string
neuen_kunden_anlegen() angestellten_einstellen() konten_verzinsen()

Methoden: neuen Kunden anlegen; neuen Angestellten einstellen; alle Konten verzinsen

Außerdem: Eine Bank besteht (in Kompositionsbeziehung) aus einer Menge von Konten, die verschwinden, wenn die Bank verschwindet. Und es gibt Aggregationen mit je einer Menge von Kunden und von Angestellten.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beispiel: Modellierung einer Bank

Klasse Konto:

Konto
nummer: string zins: float kontostand: float
abheben(wert: float) umbuchen_auf(kto: Konto, wert: float einzahlen(wert: float) verzinsen()

Attribute: Kontonummer, Zins, Kontostand

Methoden: einzahlen, abheben, umbuchen eines Betrags auf ein anderes Konto, Konto verzinsen

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

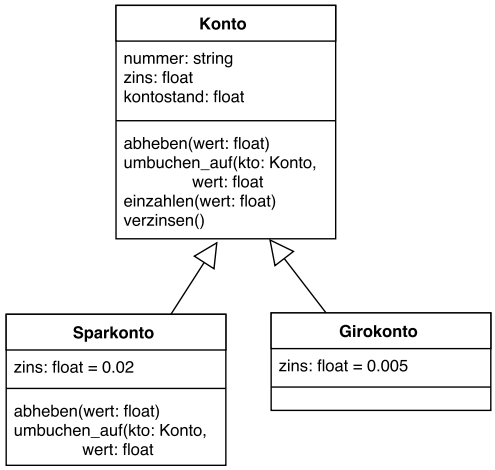
Beispiel: Modellierung einer Bank

Girokonto: Die Klasse Girokonto wird von Konto abgeleitet. Typischerweise ist der Zins bei Girokonten niedriger als bei Sparkonten. Daher wird dieser auf einen sehr niedrigen Anfangswert gesetzt.

Sparkonto: Bei der Klasse Sparkonto muss – wie beim Girokonto – ein bestimmter Anfangswert für den Zins gesetzt werden.

Außerdem: Es muss darauf geachtet werden, dass das Konto nicht ins Minus abgleitet. Dazu werden die entsprechenden Methoden überschrieben. (In der Implementierung muss die Bedingung entsprechend getestet werden.)

Beispiel: Modellierung einer Bank



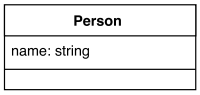
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Beispiel: Modellierung einer Bank

Klasse Person:



Jede Person steht in einer Assoziationsbeziehung mit einer Menge von Konten, die entweder dieser Person gehören (Kunde) oder auf die diese Person Zugriff hat (Angestellter).

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

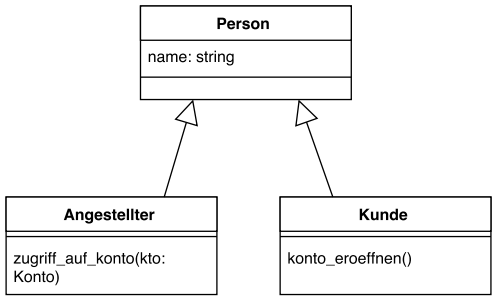
Beispiel: Modellierung einer Bank

Angestellter: Methoden: z.B. Zugriff auf ein Konto erlangen

Kunde: Methoden: z.B. Konto eröffnen (Dabei könnte noch ein Parameter übergeben werden, der beschreibt, ob das Konto ein Giro- oder Sparkonto sein soll und in welcher Währung es geführt werden soll.)

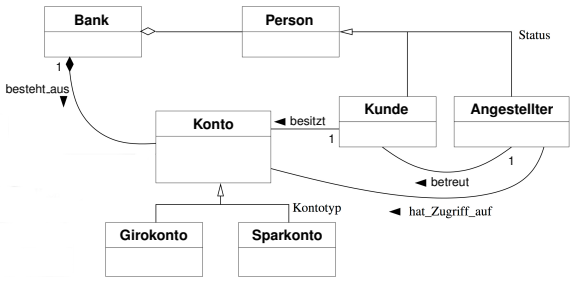
Außerdem: Ein Kunde steht in einer Assoziationsbeziehung mit einem ihn betreuenden Angestellten.

Beispiel: Modellierung einer Bank



Beispiel: Modellierung einer Bank

Insgesamt:



Allerdings gibt es eine Abweichung in diesem Modell vom zuvor Gesagten. Welche?

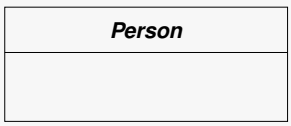
Abstrakte Klassen

Klassen ohne (vollständige) Implementierung nennt man abstrakt. Sie können nicht konkret instanziiert werden.

Abstrakte Klasse

Der Klassenname wird *kursiv* geschrieben.

In dem Bank-Beispiel möchte man etwa nie eine Instanz der Klasse Person bilden, sondern immer nur von Kunde oder Angestellter.

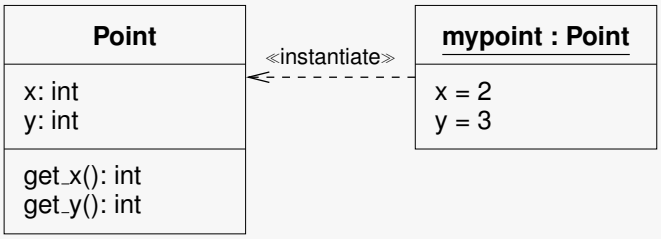


Eine Vererbungsbeziehung zu einer abstrakten Klasse ist allgemein besseres Design als zu einer Implementierungsklasse.

Objektdiagramme

Wir betrachten nun Objektdiagramme. Ein Objekt ist eine Ausprägung einer Klasse.

Klassen und Objekte



Ein Objektdiagramm beschreibt eine Art Momentaufnahme des Systems: eine Menge von Objekten, wie sie zu einem bestimmten Zeitpunkt vorhanden sind, samt ihren Beziehungen zueinander.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Objektdiagramme

Anmerkungen:

- Ein Objekt kann, muss aber nicht, durch einen Namen bezeichnet werden. Es muss aber die Klasse angegeben werden, von der dieses Objekt eine Ausprägung ist (in der Form : Point).
- Die Klassen müssen in dem Diagramm nicht mit abgebildet werden. Es kann aber manchmal angemessen sein, dies zu tun, also sozusagen „Bauplan“ und „Produkt“ gemeinsam darzustellen.
- Nicht alle Attributbelegungen eines Objekts müssen angegeben werden. Daher können durchaus auch Ausprägungen abstrakter Klassen auftauchen.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Objektdiagramme

Links

Ein Link beschreibt eine Beziehung zwischen Objekten. Er ist eine Ausprägung einer auf Klassenebene bestehenden Assoziation (auch Aggregation oder Komposition).

Bemerkungen:

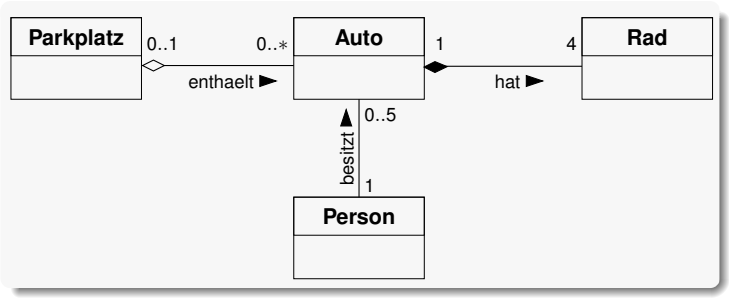
- Links sind nicht mit Multiplizitäten beschriftet, denn ein Link repräsentiert genau eine Beziehung zwischen konkreten Partnern.
- Es ist jedoch darauf zu achten, dass insgesamt die Multiplizitätsbedingungen des Klassendiagramms eingehalten werden. Das heißt, die Anzahlen der Objekte, die miteinander in Beziehung stehen, müssen innerhalb der jeweiligen Schranken sein.

Objektdiagramme

Zurück zu altem Beispiel:

Parkplatz, Autos mit Besitzern, und nun auch noch Räder

Wir betrachten zunächst das (nun erweiterte) Klassendiagramm:



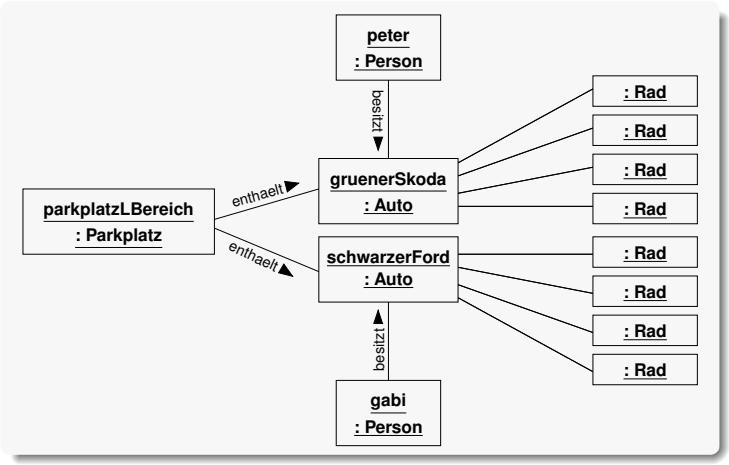
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Objektdiagramme

Objektdiagramm passend zu diesem Klassendiagramm:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

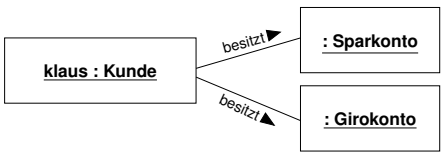
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Objektdiagramme

Achtung: Beziehungen (Assoziationen, Aggregationen, Kompositionen) zwischen Klassen werden vererbt und müssen dann auch entsprechend im Objektdiagramm bei den Ausprägungen der Unterklassen auftauchen.

Beispiel:



Auch Aggregations- und Kompositionssymbole dürfen in Objektdiagrammen auftauchen.

Vererbungsbeziehungen dagegen tauchen in Objektdiagrammen nicht auf.

Aktivitätsdiagramme

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Aktivitätsdiagramme

Wir betrachten nun Aktivitätsdiagramme (activity diagrams). Das sind UML-Diagramme, mit denen man Ablaufpläne, Reihenfolgen von Aktionen, parallele Aktionen, etc. modellieren kann.

Sie werden beispielsweise verwendet, um Geschäftsprozesse (auch Workflow-Prozesse) zu modellieren. Sie können ebenso eingesetzt werden, um Use Cases oder interne Systemprozesse zu beschreiben. Generell: wann immer man Einzelschritte hat, die sich auf gewisse typische Arten zu Gesamtabläufen zusammenfügen.

Aktivitätsdiagramme sind in vielen Aspekten ähnlich zu Petrinetzen. Im Unterschied zu Petrinetzen bieten sie zusätzliche Modellierungsmöglichkeiten, haben jedoch keine vollständige formale Semantik.

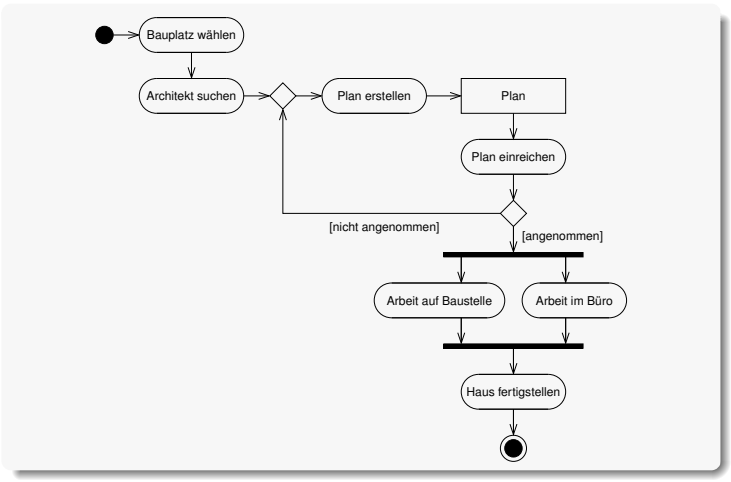
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Aktivitätsdiagramme: Beispiel

Beispiel: Hausbau

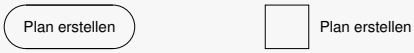


Aktivitätsdiagramme: Elemente

Wir vergleichen im Folgenden Aktivitätsdiagramme und ihre Bestandteile mit Petrinetzen (angelehnt an eine von Störrle aufgestellte Semantik für Teile von Aktivitätsdiagrammen).

Aktionen

Eine Aktion im Aktivitätsdiagramm wird durch ein Rechteck mit abgerundeten Ecken dargestellt. Es entspricht einer Transition eines Petrinetzes.



Intuition: Aktionen stehen für Tätigkeiten, mit Zeitaufwand.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

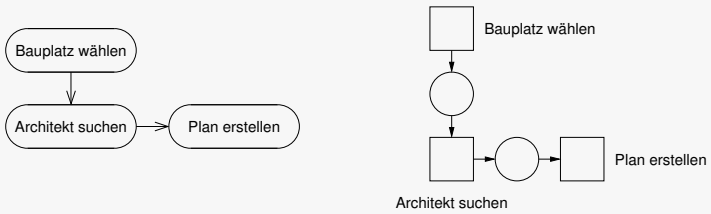
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Aktivitätsdiagramme: Elemente

Kontroll- oder Objektfluss

Der Kontroll- oder Objektfluss zwischen Aktionen (sowie Objektknoten) im Aktivitätsdiagramm, dargestellt durch die Kanten/Pfeile dazwischen, wird in dem entsprechenden Petrinetz mittels Hilfsstellen umgesetzt.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

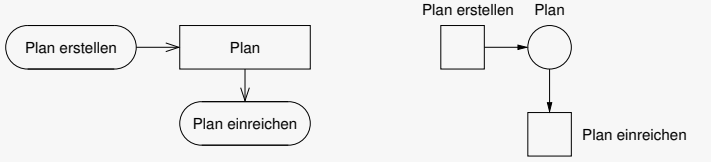
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Aktivitätsdiagramme: Elemente

Objektknoten

Objektknoten im Aktivitätsdiagramm beschreiben Speicher für die Ablage und Übergabe von konkreten Objekten. Sie entsprechen „normalen“ Stellen im Petrinetz, also solchen, die in irgendeiner Weise Ressourcen abbilden (und einen sinnvollen Namen haben).

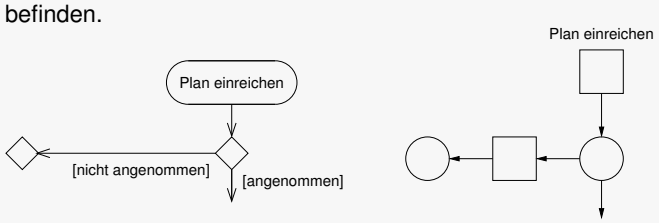


Bemerkung: Objektknoten sind bei Softwaremodellierung in der Regel mit einem Klassennamen beschriftet. Sie können dann (mehrere) Ausprägungen dieser Klasse aufnehmen.

Aktivitätsdiagramme: Elemente

Verzweigungsknoten

Verzweigungsknoten (decision nodes) im Aktivitätsdiagramm beschreiben eine Verzweigung des Kontroll- oder Objektflusses, wobei aus den alternativen Wegen jedes Mal genau einer ausgewählt wird. Sie werden im Petrinetz mittels Hilfsstellen umgesetzt, die sich in der Vorbedingung mehrerer Transitionen befinden.



Bei Bedarf (etwa bei nachfolgenden Verzweigungs- oder Objektknoten) müssen Hilfstransitionen eingeführt werden.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Aktivitätsdiagramme: Elemente

Anmerkungen:

- Wesentlicher Grund für die Einführung von Hilfsstellen und Hilfstransitionen bei der Übersetzung in Petrinetze sind die strukturellen Erfordernisse für ein korrektes Petrinetz (nämlich abwechselndes Auftreten von Stellen und Transitionen).
- Der Kontroll- oder Objektfluss an Verzweigungsknoten wird durch Überwachungsbedingungen (Guards) gesteuert. Sie werden in eckigen Klammern an den ausgehenden Wegen notiert. (Ähnliche Bedingungen, an Transitionen, existieren auch in attributierten Petrinetzen.)
- Die Bedingungen dürfen sich logisch nicht überlappen, und sollten insgesamt alle möglichen Fälle abdecken.
- Während Aktionen einen Zeitaufwand haben, werden Kontrollelemente wie Verzweigungsknoten (und weitere gleich gezeigte) als instantan durchlaufen angesehen.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

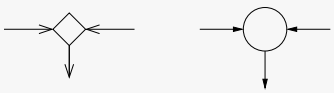
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Aktivitätsdiagramme: Elemente

Verbindungsknoten

Es gibt auch Verbindungsknoten (merge nodes), die alternative Kontroll- oder Objektflüsse zusammenführen. Sie werden im entsprechenden Petrinetz mittels Hilfsstellen umgesetzt, die sich in der Nachbedingung mehrerer Transitionen (gegebenenfalls Hilfstransitionen) befinden.

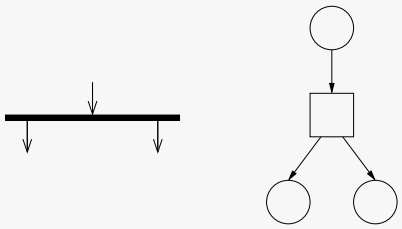


Es gibt auch Knoten (mit gleicher Darstellung), die sowohl Verbindungs- als auch Verzweigungsknoten sind, also mehrere eingehende und mehrere ausgehende Wege haben.

Aktivitätsdiagramme: Elemente

Gabelung (auch: Parallelisierungsknoten)

Eine Gabelung (fork node) im Aktivitätsdiagramm teilt einen Kontroll- oder Objektfluss in mehrere parallele Flüsse auf. Ihr entspricht im Petrinetz eine Transition mit mehreren Stellen (gegebenenfalls Hilfsstellen) in der Nachbedingung.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

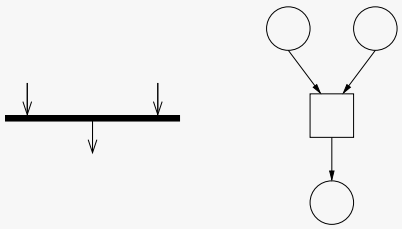
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Aktivitätsdiagramme: Elemente

Vereinigung (auch: Synchronisationsknoten)

Dual dazu gibt es die Vereinigung (join node), die mehrere parallele Kontroll- oder Objektflüsse zusammenführt. Sie wird im entsprechenden Petrinetz durch eine Transition umgesetzt, die mehrere Stellen (ggfs. Hilfsstellen) in der Vorbedingung hat.



Wie Verbindungs- und Verzweigungsknoten kann man manchmal auch Vereinigungen und Gabelungen zu einem Knoten zusammenfassen.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

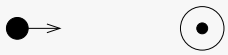
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Aktivitätsdiagramme: Elemente

Startknoten

Ein Startknoten im Aktivitätsdiagramm entspricht einer initial markierten Stelle im Petrinetz. In Startknoten dürfen keine Kanten hineinführen.



Anmerkungen:

- Im übersetzten Petrinetz wird die entsprechende Stelle initial mit so vielen Marken bestückt, wie es ausgehende Kanten vom Startknoten gibt.
- Diese Kanten führen generell nicht zu Objektknoten.

Aktivitätsdiagramme: Elemente

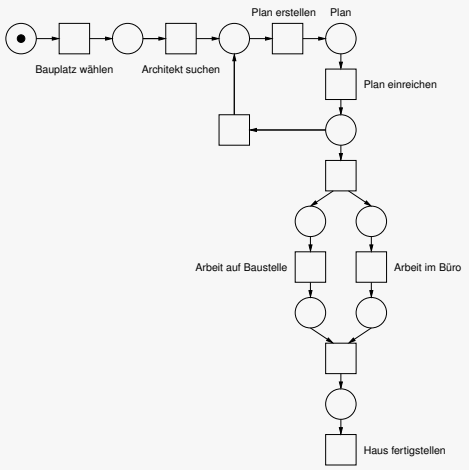
Aktivitätsende

Das Aktivitätsende signalisiert, dass alle Kontroll- und Objektflüsse beendet werden. Es gibt keine allgemeine Entsprechung in Petrinetzen.

Es gibt auch ein Symbol für das Flussende, das nur den in es hineinlaufenden Kontrollfluss beendet.

Aktivitätsdiagramme: Beispiel übersetzt

Übersetztes Beispiel, als Petrinetz:



Aktivitätsdiagramme & Petrinetze

Aktivitätsdiagramme enthalten noch mehr Anleihen aus Petrinetzen. Beispielsweise können auch die Kapazität eines Objektknotens und das Gewicht eines Objektflusses spezifiziert werden.

Dabei beschreibt upperBound die Kapazität (maximal 6 Gerichte dürfen hier gleichzeitig fertig sein) und weight das Gewicht (immer 2 Gerichte werden hier gleichzeitig serviert).

In Aktivitätsdiagrammen darf zusätzlich sogar noch spezifiziert werden, in welcher Reihenfolge die Objekte aus dem Objektknoten genommen werden (unordered, ordered, LIFO, FIFO).

Aktivitätsdiagramme & Petrinetze

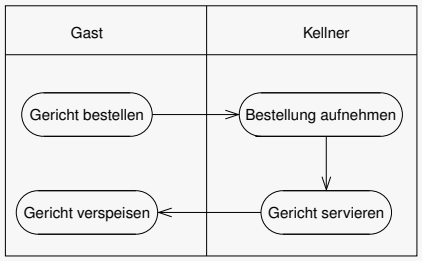
Vorsicht:

- Die Entsprechung zwischen Aktivitätsdiagrammen und Petrinetzen ist nicht immer ganz exakt. Nicht für alle Konzepte gibt es eine direkte Übersetzung.
- Das liegt teilweise auch daran, dass Aktivitätsdiagramme nur ein semi-formales Modelliergungsmittel sind und gar nicht alle Aspekte vollständig spezifiziert sind.

Aktivitätsdiagramme: Strukturierung

Aktivitätsbereiche (activity partitions / swimlanes)

Zur Strukturierung können Elemente gruppiert werden. Dies dient etwa dazu, die Verantwortung für bestimmte Aktionen festzulegen oder räumliche Verteilung auszudrücken.



Dabei können zum Beispiel Objektknoten auch auf einer Grenze zwischen Bereichen liegen, um eine Übergabe auszudrücken.

Aktivitätsdiagramme: Elemente (Fortsetzung)

Weitere Elemente von Aktivitätsdiagrammen:

- Pins: Parameter und Parametersätze für Aktionen
- Senden und Empfang von Signalen
- Kontrollstrukturen: Sprungmarken, Schleifenknoten, Bedingungsknoten
- Unterbrechungsbereiche (interruptible activity region) zur Behandlung von Ausnahmen (Exceptions)
- Expansionsbereiche (expansion region) zur wiederholten Ausführung von Aktivitäten für mehrere übergebene Objekte

Zustandsdiagramme

Zustandsdiagramme

UML-Zustandsdiagramme (state diagrams, auch state machine diagrams oder statecharts genannt), sind eng verwandt mit den bereits eingeführten Zustandsübergangsdiagrammen.

Sie werden eingesetzt, wenn bei der Modellierung der Fokus auf die Zustände und Zustandsübergänge des Systems gelegt werden soll.

Im Gegensatz zu Aktivitätsdiagrammen werden auch weniger die Aktionen eines Systems beschrieben, sondern eher die Reaktionen eines Systems auf seine Umgebung.

Zustandsdiagramme

Anwendungen sind die Modellierung von:

- Protokollen, Komponenten verteilter Systeme
- Benutzungsoberflächen
- eingebetteten Systemen
- ...

Zustandsdiagramme

Zustandsdiagramme wurden 1987 von David Harel unter dem Namen Statecharts eingeführt.

Features, mit denen Zustandsdiagramme ausgestattet sind:

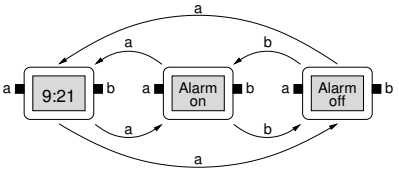
- Zustände und Zustandsübergänge
- hierarchische Verfeinerung von Zuständen
- „Parallelschalten“ durch Regionen
- Historien, um sich früher besuchte Zustände zu merken und in diese zurückzukehren
- Kommunikation über Effekte oder Flags

Viele dieser Ausstattungsmerkmale dienen dazu, Diagramme mit vielen Zuständen und Übergängen übersichtlicher und kompakter zu gestalten.

Zustandsdiagramme: Beispiel

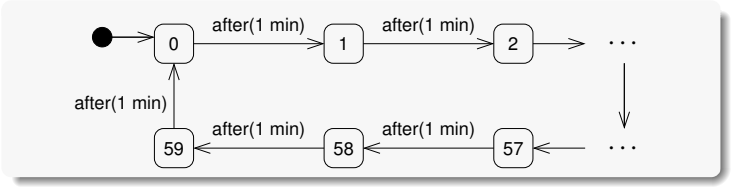
Wir lernen Zustandsdiagramme am Beispiel der Modellierung einer Armbanduhr kennen (stark vereinfacht gegenüber einem Beispiel von Harel aus dem ursprünglichen wissenschaftlichen Artikel).

Die Armbanduhr hat zwei Knöpfe (a, b) und zwei Modi (Zeitanzeige, Alarmeinstellung). Zwischen den Modi wechselt man mit Hilfe von Knopf a. Der Alarm kann an (on) oder aus (off) sein. Zwischen den Alarmzuständen wechselt man mit Hilfe von Knopf b (im entsprechenden Modus). Wenn der Alarm an ist, erzeugt die Uhr zu jeder vollen Stunde einen Piepton.



Zustandsdiagramme: Beispiel

Wir beginnen zunächst mit der Modellierung der Minutenanzeige.



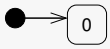
Zustandsdiagramme: Elemente

Zustand

Ein Zustand wird durch ein Rechteck mit abgerundeten Ecken dargestellt.

Startzustand

Der Startzustand wird ähnlich zum Startknoten bei Aktivitätsdiagrammen gekennzeichnet.



Es dürfen in den schwarzen ausgefüllten Kreis keine Kanten hineinführen, und nur genau eine heraus.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

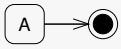
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Elemente

Endzustand

Endzustände werden wie das Aktivitätsende bei Aktivitätsdiagrammen gekennzeichnet.



In Fällen, in denen man (wie in unserem Beispiel) ein System modelliert, das nicht terminieren soll, gibt es keinen Endzustand.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Elemente

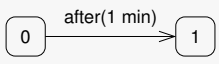
Transition (= Zustandsübergang)

Eine Transition ist eine Kante/Pfeil, beschriftet mit

Ereignis [Bedingung] / Effekt ,

wobei Bedingung und Effekt optional sind.

- Ereignis: Signal oder Nachricht, die die entsprechende Transition auslösen



- Bedingung: Überwachungsbedingung (auch Guard genannt)
- Effekt: Effekt, der durch die Transition ausgelöst wird

In unserem Beispiel (Minutenanzeige) gibt es nur Ereignisse, die eine Zeitspanne angeben, nach der die Transition ausgelöst wird. Allgemein gibt es auch andere Ereignisse, beispielsweise Methodenaufrufe oder durch Broadcast-Effekte.

Zustandsdiagramme: Aktionen

Neben Effekten, die durch Transitionen ausgelöst werden, können in einem Zustand selbst weitere Aktivitäten bei Eintritt, Verweilen oder Verlassen ausgelöst werden.

Diese haben den gleichen Aufbau wie die Beschriftung einer Transition: Ereignis [Bedingung] / Effekt.

Dabei kann Ereignis dann unter anderem folgendes sein:

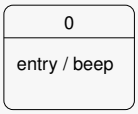
- entry: Der entsprechende Effekt wird bei Eintritt in den Zustand ausgelöst.
- do: Der Effekt ist eine Aktivität, die nach Betreten des Zustands ausgeführt wird und die spätestens dann endet, wenn der Zustand verlassen wird.
- exit: Der entsprechende Effekt wird bei Verlassen des Zustands ausgelöst.

Zustandsdiagramme: Beispiel

Beispiel:

Die Uhr soll zu jeder vollen Stunden ein Signal von sich geben. Daher wird die Aktivität beep bei Eintritt in den Zustand 0 ausgelöst.

Notation:



Zustandsdiagramme: Elemente

Was ist mit der Stundenanzeige?

Diese soll parallel zur Minutenanzeige laufen, das heißt, die Uhr ist eigentlich immer in zwei Zuständen gleichzeitig: ein Zustand für die Stunden, der andere für die Minuten.

Solch eine Situation wird durch Regionen modelliert.

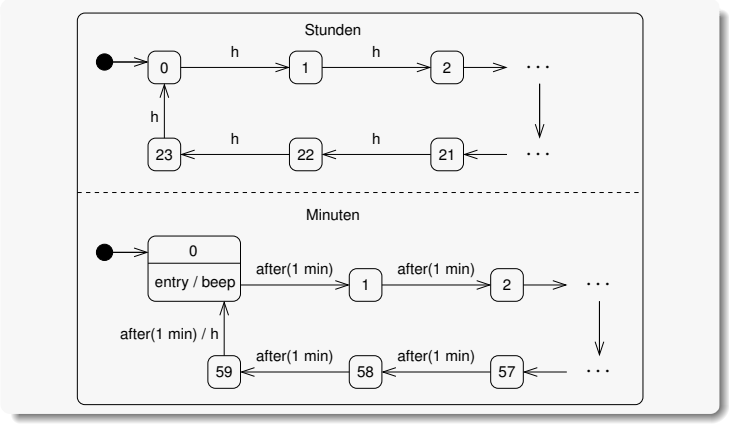
Region

Regionen unterteilen ein Zustandsdiagramm in mindestens zwei Bereiche, die parallel zueinander ausgeführt werden.

Im Beispiel erlaubt uns dies, insgesamt nur $24 + 60 = 84$ Zustände, statt $24 \cdot 60 = 1440$ Zustände, zu zeichnen.

Zustandsdiagramme: Beispiel

So sieht das modifizierte Zustandsdiagramm für die Uhr jetzt aus:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Beispiel

Bemerkungen:

- Es gibt jetzt zwei Startzustände, die beide zu Beginn betreten werden (Uhrzeit: 0:00).

  (Allgemein: höchstens ein Startzustand pro Region.)

- Da Stunden- und Minutenanzeige nicht vollkommen unabhängig voneinander arbeiten, ist eine Synchronisation eingebaut: Die Transition in den Minutenzustand 0 löst einen Effekt h aus (h für „hour“).

  Dieser Effekt dient dann als Ereignis, das den Übergang in den nächsten Stundenzustand verursacht.

  Die beiden Transitionen finden gleichzeitig statt.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Hinweise

Weitere Bemerkungen (für uns nicht so relevant):

- Transitionen verbrauchen im Allgemeinen keine Zeit (im Gegensatz zum Aufenthalt in Zuständen).
- Neben Effekten/Ereignissen, die direkt ausgeführt werden, gibt es auch solche, die zunächst in einer Event Queue (Warteschlange) abgelegt werden. Diese wird dann schrittweise abgearbeitet, wobei die entsprechenden Ereignisse ausgelöst werden. Damit kann asynchrone Kommunikation beschrieben werden.
- Effekte können Broadcasts (= Ausstrahlungen) sein, die überall im Zustandsdiagramm sichtbar sind, oder können alternativ direkte Kommunikation bedeuten (beispielsweise Methodenaufrufe).
  (Die ursprüngliche Statecharts-Semantik von Harel verwendete nur Broadcasts.)

Zustandsdiagramme: Elemente

Nun soll die Möglichkeit hinzugefügt werden, das Stunden-Alarmsignal aus- und wieder einzuschalten.

Dazu führen wir zunächst weitere Zustände (Alarm on, off) ein, zu denen man durch Drücken von a gelangt (und zwischen denen man mit b wechselt).

Problem: Wir bräuchten ziemlich viele mit a beschriftete Transitionen, die von den Stunden-/Minutenzuständen ausgehen!

Dafür bieten Zustandsdiagramme folgende Lösung:

Zusammengesetzte Zustände

Zusammengesetzte Zustände dienen dazu, Hierarchien von Zuständen zu modellieren und dadurch ein- und ausgehende Transitionen zusammenzufassen.

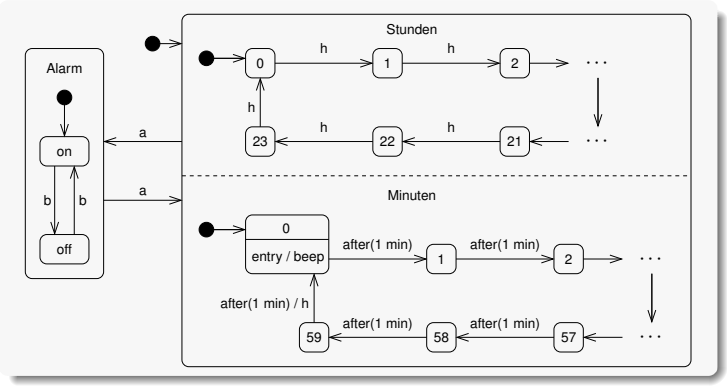
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Beispiel

Damit ergibt sich folgendes Zustandsdiagramm:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

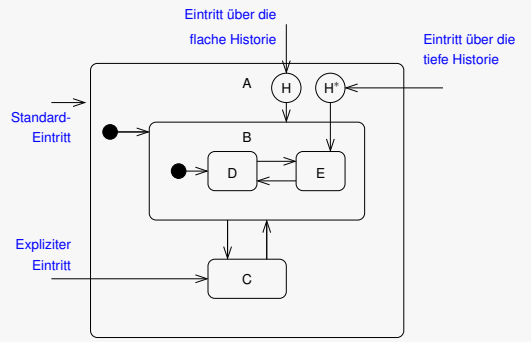
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Eintrittsmöglichkeiten

Um möglichst viel Flexibilität bei der Modellierung zu haben, werden für zusammengesetzte Zustände verschiedene Eintritts- und Austrittsmöglichkeiten bereitgestellt.

Eintrittsmöglichkeiten in einen Zustand (grafisch)



Zustandsdiagramme: Eintrittsmöglichkeiten

Beschreibung der Eintrittsmöglichkeiten:

- Standard-Eintritt: Dabei wird der Startzustand des zusammengesetzten Zustands angesprungen.

  (Fortsetzung bei B, was schließlich zu einer Fortsetzung bei D führt.)

- Expliziter Eintritt: Es wird bei dem explizit angegebenen Folgezustand fortgesetzt.

  (Fortsetzung bei C.)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Eintrittsmöglichkeiten

Beschreibung der Eintrittsmöglichkeiten (Fortsetzung):

- Eintritt über die tiefe Historie: Wurde der zusammengesetzte Zustand bereits früher besucht, so wird der letzte vor dem Verlassen des Gesamtzustands aktive Unterzustand der tiefstmöglichen Ebene betreten.

  (Falls also der zusammengesetzte Zustand A das letzte Mal von D aus verlassen wurde, so wird jetzt wieder bei D fortgesetzt.)

  Falls man noch niemals zuvor diesen zusammengesetzten Zustand betreten hat, so wird der Zustand betreten, der mit der Kante gekennzeichnet ist, die von dem H^*-Knoten ausgeht.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Eintrittsmöglichkeiten

Beschreibung der Eintrittsmöglichkeiten (Fortsetzung):

- Eintritt über die flache Historie: Wurde der zusammengesetzte Zustand bereits früher besucht, so wird der letzte vor dem Verlassen des Gesamtzustands aktive Unterzustand der obersten Ebene betreten.

  (Falls also der zusammengesetzte Zustand A das letzte Mal von E aus verlassen wurde, so wird jetzt bei B fortgesetzt, was letztendlich zu einer Fortsetzung bei D führt.)

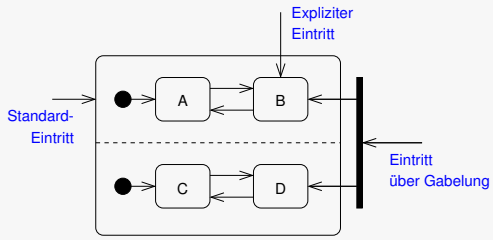
  Falls man noch niemals zuvor diesen zusammengesetzten Zustand betreten hat, so wird analog wie bei der tiefen Historie verfahren.

Außerdem: Eintritt über einen Eintrittspunkt (wird hier nicht behandelt).

Zustandsdiagramme: Eintrittsmöglichkeiten

Wenn ein zusammengesetzter Zustand in mehrere Regionen unterteilt ist, so ergeben sich bei den Eintrittsmöglichkeiten einige Besonderheiten.

Eintrittsmöglichkeiten bei Regionen (grafisch)



Es gäbe zusätzlich auch wieder die Fälle zum Eintritt über flache oder tiefe Historie zu diskutieren, darauf verzichten wir hier jedoch.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Eintrittsmöglichkeiten

Beschreibung der Eintrittsmöglichkeiten bei Regionen:

- Standard-Eintritt: Dabei werden die jeweiligen Startzustände der Regionen angesprungen.

  (Fortsetzung bei A und C.)
- Expliziter Eintritt: Ein Zustand einer Region wird direkt angesprungen. In der anderen Region wird beim Startzustand fortgesetzt.

  (Fortsetzung bei B und C.)
- Eintritt über Gabelung: Die beiden anzuspringenden Zustände in den Regionen werden ähnlich zur Gabelung bei Aktivitätsdiagrammen gekennzeichnet.

  (Fortsetzung bei B und D.)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

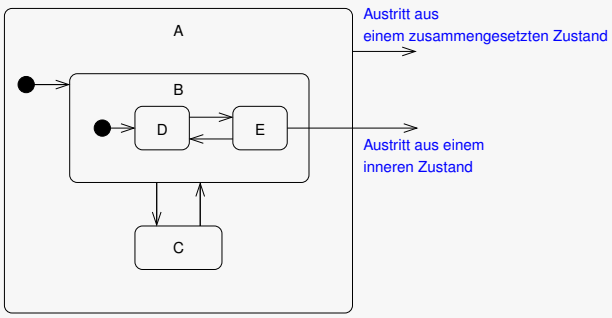
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Austrittsmöglichkeiten

Auch für den Austritt aus zusammengesetzten Zuständen gibt es verschiedene Möglichkeiten.

Austrittsmöglichkeiten aus einem Zustand (grafisch)



Zustandsdiagramme: Austrittsmöglichkeiten

Beschreibung der Austrittsmöglichkeiten:

- Austritt aus einem zusammengesetzten Zustand: Sobald das mit der Transition assoziierte Ereignis stattfindet, wird jeder beliebige Unterzustand verlassen.
- Austritt aus einem inneren Zustand: Die Transition wird nur genommen, wenn man sich gerade im entsprechenden Unterzustand (hier: Zustand E) befindet, und das entsprechende Ereignis stattfindet.

Außerdem: Austritt über einen Austrittspunkt, Endzustand oder Terminator (hier nicht behandelt).

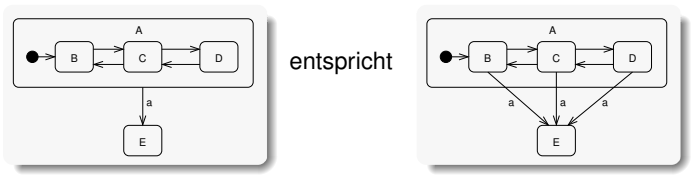
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Austrittsmöglichkeiten

Beispiel zu Austrittsmöglichkeiten:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

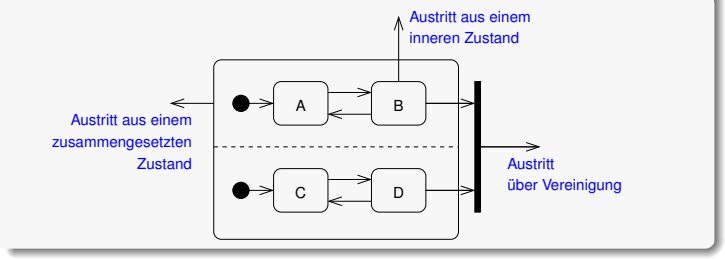
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Austrittsmöglichkeiten

Auch für Austrittsmöglichkeiten müssen wir untersuchen, welche Besonderheiten sich bei Regionen ergeben.

Austrittsmöglichkeiten bei Regionen (grafisch)



Zustandsdiagramme: Austrittsmöglichkeiten

Beschreibung der Austrittsmöglichkeiten bei Regionen:

- Austritt aus einem zusammengesetzten Zustand: Dabei wird der zusammengesetzte Zustand verlassen, egal in welchen Unterzuständen man sich gerade befindet.
- Austritt aus einem inneren Zustand: Der zusammengesetzte Zustand wird nur verlassen, wenn man sich in der entsprechenden Region in dem Zustand befindet, der durch den Pfeil verlassen wird. In den anderen Regionen kann man sich in beliebigen Zuständen befinden.

  (Hier Austritt nur, wenn man sich in der ersten Region in B befindet.)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Austrittsmöglichkeiten

Beschreibung der Austrittsmöglichkeiten bei Regionen (Fortsetzung):

- Austritt über Vereinigung: Der zusammengesetzte Zustand kann nur verlassen werden, wenn man sich in den Regionen in den Zuständen befindet, von denen die Pfeile in die Vereinigung hineinführen.

  (Hier Austritt nur, wenn man sich in B und D befindet.)

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Beispiel

Wieder zurück zur Armbanduhr.

Es gibt weitere Probleme:

Wenn man aus der Alarmeinstellung zurückkehrt, ist die Zeit auf 0:00 zurückgesetzt!

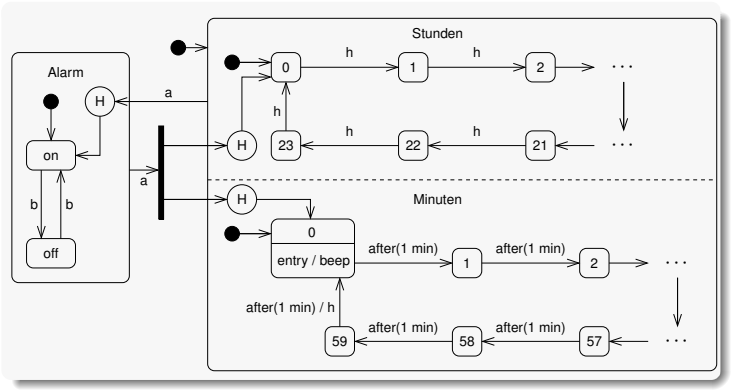
Und umgekehrt, wenn man zur Alarmeinstellung wechselt, wird der Alarm immer auf „on“ gesetzt!

Lösung:

Jeweils Verwendung des Eintritts über die (flache) Historie.

Da man im Fall der Zeitanzeige in zwei Regionen gleichzeitig eintreten muss, verwenden wir dort eine Gabelung.

Zustandsdiagramme: Beispiel



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Beispiel

Weiteres Problem: Wenn man längere Zeit in der Alarmanzeige verbringt, so wird in dieser Zeit die Minuten-/Stundenanzeige nicht entsprechend aktualisiert. Denn das Ereignis „after(1 min)“ bezieht sich nur auf die Zeit, die seit dem Eintritt in den entsprechenden Zustand vergangen ist.

Die Zeitanzeige müsste deshalb entsprechend aktualisiert werden. Dieses Problem lösen wir hier nicht.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Beispiel

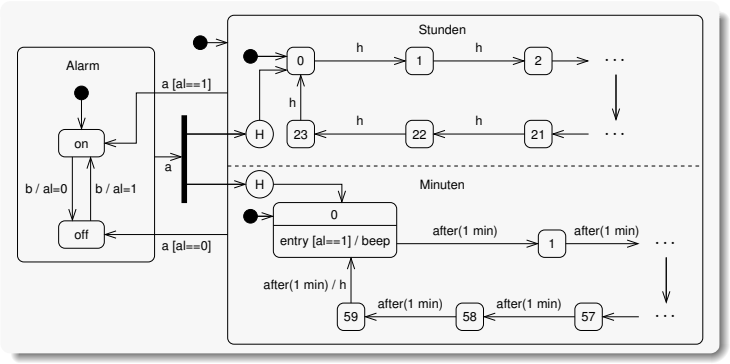
Was fehlt ansonsten noch?

Beim Wechseln zwischen den Alarm-Zuständen (on, off) muss ein Flag (hier al genannt) gesetzt werden, um damit den beep-Effekt zu kontrollieren.

Dieses Flag muss dann mit Hilfe einer Bedingung im Minutenzustand 0 abgefragt werden.

Außerdem betreten wir den Zustand Alarm nun nicht mehr über die flache Historie, sondern fragen mit Hilfe von Bedingungen ab, wie al belegt ist.

Zustandsdiagramme: Beispiel



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Beispiel

Schließlich modellieren wir noch, dass die Batterie der Uhr leer werden kann und gewechselt werden muss.

In diesem Fall will man den zusammengesetzten Zustand nicht über die flache oder tiefe Historie betreten! Es wird also dann tatsächlich die Zeit auf 0:00 zurückgesetzt.

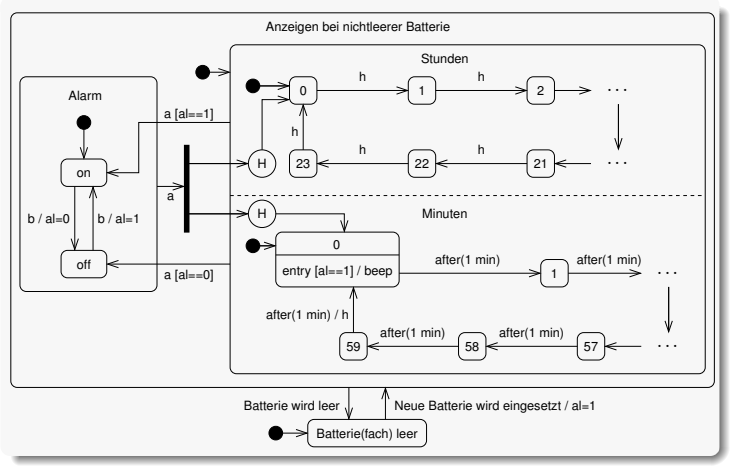
Außerdem setzen wir das Flag al beim Einsetzen der Batterie (zurück) auf den Anfangswert 1.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: Beispiel



Zustandsdiagramme: „Flachklopfen“

Ein großer Teil der Modellierungsmöglichkeiten von Zustandsdiagrammen dient dazu, diese kompakt und übersichtlich zu notieren.

Umgekehrt kann man Zustandsdiagramme oft „flachklopfen“ und zusammengesetzte Zustände auflösen, wodurch man äquivalente Zustandsdiagramme erhält, die die gleichen Übergänge erlauben. Dabei erhält man jedoch im Allgemeinen mehr Zustände und/oder Transitionen.

Wir sehen uns einige Beispiele an (und werden bestimmte Features dabei bewusst nicht betrachten) . . .

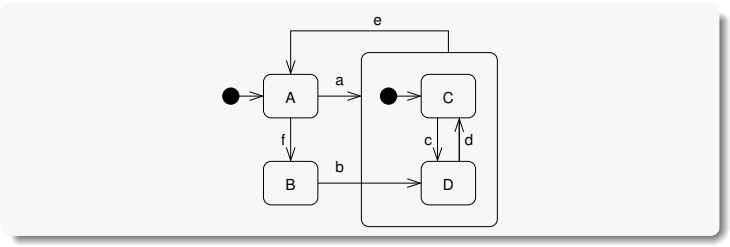
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: „Flachklopfen“

Beispiel 1: Wandeln Sie folgendes Zustandsdiagramm in ein „flaches“ Zustandsdiagramm um:



Charakteristika dieses Beispiels: keine Regionen, keine Historie

Ansatz: einfache Zustände behalten, Eintritte/Austritte an Rand zusammengesetzter Zustände übersetzen, andere Übergänge einfach behalten, und nur äußerster Startzustand bleibt solcher

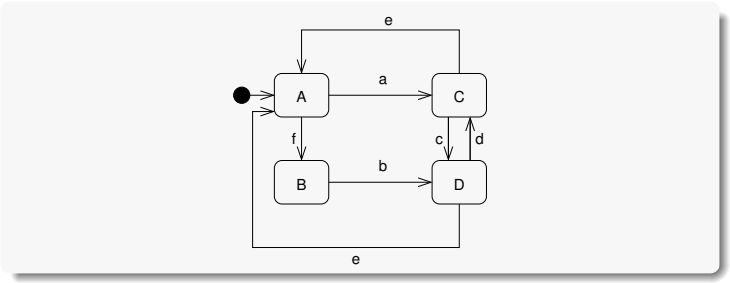
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: „Flachklopfen“

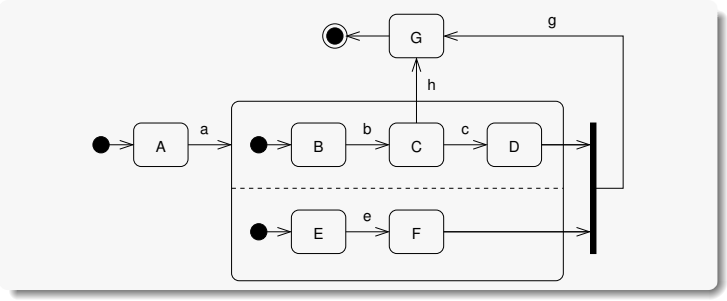
Lösung zu Beispiel 1:



Hauptschritt: Die Transition, die von dem zusammengesetzten Zustand wegführt, wurde durch mehrere Transitionen ersetzt, die von den inneren Zuständen ausgehen.

Zustandsdiagramme: „Flachklopfen“

Beispiel 2: Wandeln Sie folgendes Zustandsdiagramm in ein „flaches“ Zustandsdiagramm um:



Charakteristika dieses Beispiels: Regionen, aber keine Historie

Ansatz außerhalb Regionen wie bisher, zusätzlich: Kreuzprodukt; parallele Übergänge entsprechend den Regionen; Ein-/Austritte an Rand von – aber auch hinein und heraus aus – Regionen übersetzen

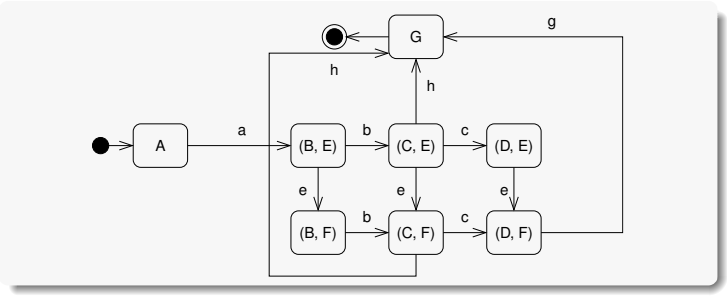
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: „Flachklopfen“

Lösung zu Beispiel 2:



Hauptidee: Kreuzprodukt der Zustandsmengen der Regionen bilden.

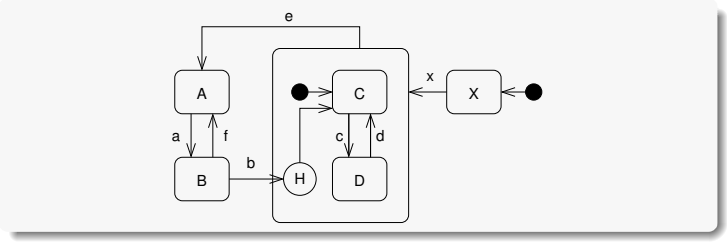
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: „Flachklopfen“

Beispiel 3: Wandeln Sie folgendes Zustandsdiagramm in ein „flaches“ Zustandsdiagramm um:

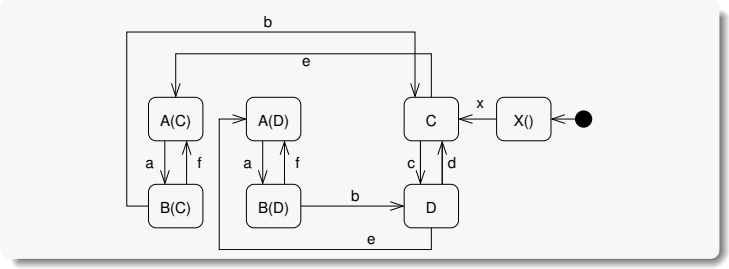


Charakteristika hier: keine Regionen, aber (flache) Historie

Ansatz jetzt: zunächst wie im einfachen Fall, aber für Verlassen eines zusammengesetzten Zustands mit Historien-Knoten gegebenenfalls Kopien äußerer Zustände (und ihrer Übergänge) zum Merken des letzten inneren Zustands, und Verwendung dieser Information bei Wiedereintritt über die Historie

Zustandsdiagramme: „Flachklopfen“

Lösung zu Beispiel 3:



Hauptidee: In den äußeren Zuständen merkt man sich, ob/aus welchem Unterzustand man den zusammengesetzten Zustand zuletzt verlassen hat. Dies führt zu zusätzlichen Zuständen.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Zustandsdiagramme: weitere Features

Weitere Features von UML-Zustandsdiagrammen:

- Unterscheidung zwischen verschiedenen Arten von Ereignissen: call event, signal event, change event, time event, any receive event
- Verzögern und Ignorieren von Ereignissen
- Entscheidungen und Kreuzungen
- Eintritts- und Austrittspunkte, Terminator
- Rahmen und Wiederverwendung von Zustandsdiagrammen

Außerdem: Generalisierung und Spezialisierung von Zustandsdiagrammen

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme

Sequenzdiagramme

Sequenzdiagramme (sequence diagrams) sind die bekanntesten Vertreter von Interaktionsdiagrammen in UML.

Sie dienen dazu, Kommunikation und Interaktion zwischen mehreren Kommunikationspartnern zu modellieren, und beruhen auf dem Basiskonzept der Interaktion:

Interaktion

Eine Interaktion ist das Zusammenspiel von mehreren Kommunikationspartnern.

Typische Beispiele: Versenden von Nachrichten, Datenaustausch, Methodenaufruf

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme

Sequenzdiagramme waren bereits vor Aufnahme in die UML unter dem Namen message sequence charts bekannt.

Im Gegensatz zu Aktivitätsdiagrammen oder Zustandsdiagrammen beschreiben sie im Allgemeinen nicht alle Abläufe eines Systems, sondern nur einen oder mehrere mögliche Abläufe.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

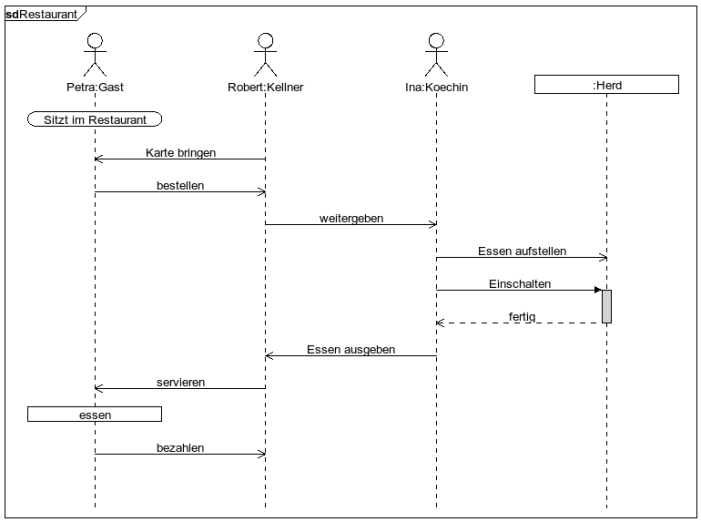
UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme

Sequenzdiagramme beschreiben Interaktionen in zwei Dimensionen:

- von links nach rechts: Anordnung der Kommunikationspartner als Lebenslinien. Oft wird der Partner, der den Ablauf initiiert, ganz links angegeben.
- von oben nach unten: Zeitachse

Sequenzdiagramme: Beispiel (Restaurant)



Sequenzdiagramme: Kommunikationspartner

Kommunikationspartner

Die Kommunikationspartner in einem Sequenzdiagramm werden ähnlich wie in Objektdiagrammen als Rechtecke notiert.

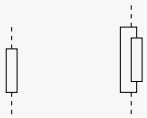
Manchmal werden die Rechtecke auch weggelassen. Menschliche Partner werden auch durch ein Strichmännchen symbolisiert:

Von jedem Kommunikationspartner führt eine gestrichelte Lebenslinie nach unten.

Sequenzdiagramme: Kommunikationspartner

Ausführungsbalken

Aktivitäten eines Kommunikationspartners werden durch sogenannte Ausführungsbalken dargestellt.



Parallele Tätigkeiten eines Kommunikationspartners werden dabei durch übereinander liegende Ausführungsbalken beschrieben (siehe oben rechts).

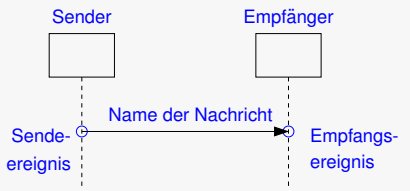
Während die Balken aktive Zeit anzeigen, symbolisieren die gestrichelten Linien passive Zeit.

Sequenzdiagramme: Nachrichten

Nachrichten

Die Nachrichten beschreiben die Kommunikationen bzw. Interaktionen der Kommunikationspartner und werden durch Pfeile dargestellt. Eine Nachricht hat einen Sender und einen Empfänger.

Die Stellen, an denen die Pfeile auf den Lebenslinien auftreffen, nennt man auch Sendeereignis und Empfangsereignis.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: Nachrichten

Synchrone Nachrichten

Bei synchroner Kommunikation warten Sender und Empfänger aufeinander. Der Sender macht erst dann weiter, wenn er weiß, dass der Empfänger die Nachricht erhalten hat. Solche Kommunikation wird unter Verwendung einer schwarzen ausgefüllten Pfeilspitze dargestellt.

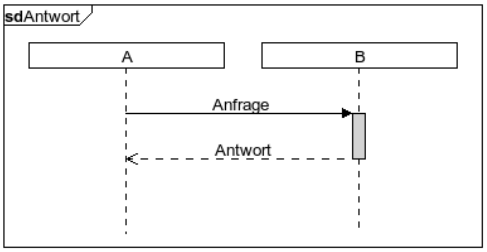
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: Nachrichten

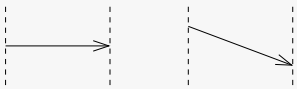
Um zu signalisieren, dass der Empfänger die Nachricht erhalten hat, bzw. die zugehörige Aktion abgeschlossen wurde, wird eine Antwort gesendet. Diese ist als gestrichelter Pfeil dargestellt.



Sequenzdiagramme: Nachrichten

Asynchrone Nachrichten

Bei asynchroner Kommunikation wartet der Sender nicht darauf, dass der Empfänger die Nachricht erhalten hat. Er arbeitet einfach weiter. Solche Kommunikation wird durch eine einfache Pfeilspitze dargestellt.



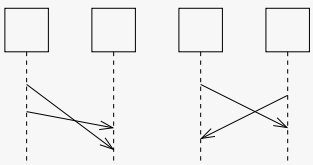
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: Nachrichten

Bei asynchroner Kommunikation (aber nicht bei synchroner Kommunikation) kann auch der Fall eintreten, dass sich Nachrichten überholen oder kreuzen.



Das Überholen von Nachrichten (oben links) ist nur dann nicht möglich, wenn man explizit einen FIFO-Kanal fordert.

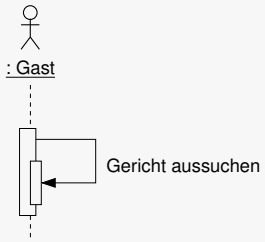
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: Nachrichten

Es ist möglich, eine Nachricht an sich selbst zu schicken.



Sequenzdiagramme: Äquivalenz

Äquivalenz von Sequenzdiagrammen

Zwei Sequenzdiagramme sind äquivalent, wenn sie die gleichen Ereignisse enthalten und die Reihenfolge der Ereignisse identisch ist.

Dabei können die Diagramme durchaus verschieden gezeichnet sein (andere Reihenfolge der Kommunikationspartner, verschiedene Abstände zwischen den Ereignissen, . . .).

Durch die Bestimmung der Reihenfolge der Ereignisse kann die Äquivalenz zweier Diagramme nachgewiesen bzw. widerlegt werden.

Sequenzdiagramme: Interaktions-Operatoren

Bisher haben wir gesehen, wie man mit einem Sequenzdiagramm einen möglichen Ablauf beschreiben kann. In manchen Fällen möchte man jedoch mehrere (vielleicht sogar alle) Abläufe beschreiben.

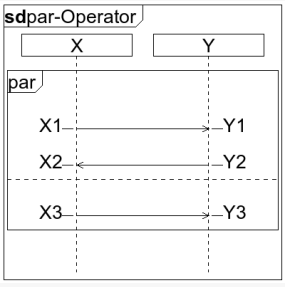
Dazu gibt es die Möglichkeit, kombinierte Fragmente zu verwenden, bei denen mehrere (Interaktions-)Operanden (= Teil-Sequenzdiagramme) mit Hilfe von Interaktions-Operatoren zusammengesetzt werden.

Wir betrachten im Folgenden einige dieser Interaktions-Operatoren.

Sequenzdiagramme: par-Operator

Interaktionsoperator par (Parallelität)

Hier sind die Operanden in beliebiger Reihenfolge ausführbar. Die Reihenfolge der Ereignisse in den Operanden muss aber gewahrt werden. Ansonsten gibt es keine Bedingungen.

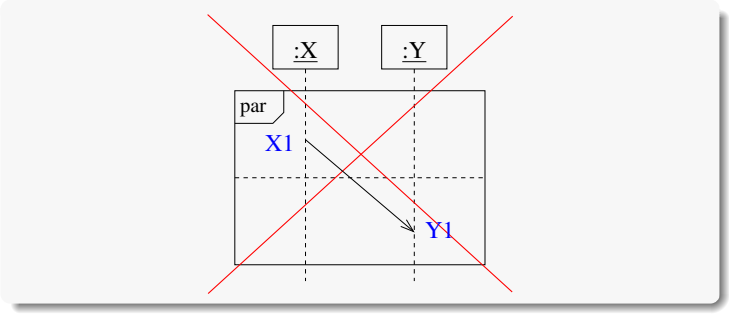


Dabei wird der Operator par links oben in der Ecke angegeben.

Eine gestrichelte waagerechte Linie trennt die verschiedenen Operanden.

Sequenzdiagramme: par-Operator

Nachrichten, die von einem Operanden in einen anderen laufen, sind nicht zugelassen.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: par-Operator

Ordnung auf den Ereignissen:

- $X1 < Y1 < Y2 < X2$ (Operand 1)
- $X3 < Y3$ (Operand 2)

Ansonsten gibt es keine weiteren Einschränkungen.

Dieses Sequenzdiagramm beschreibt insgesamt fünfzehn Abläufe, beispielsweise:

- X1, Y1, X3, Y3, Y2, X2
- X3, X1, Y1, Y2, X2, Y3
- ...

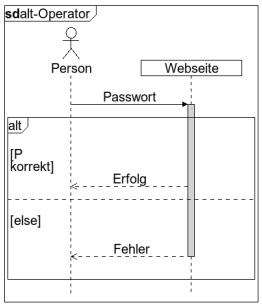
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: alt-Operator

Sequenzdiagramme bieten auch die Möglichkeit, alternative Nachrichtenflüsse darzustellen. Zu diesem Zweck wird der alt-Operator verwendet.
Ein möglicher Anwendungsfall ist die unterschiedliche Rückgabe an eine Person bei der Eingabe eines Passwortes.



Sequenzdiagramme: alt-Operator

Dabei ist es nötig, die Eintrittsfälle für die Alternativen durch Guards zu definieren.

Es ist ebenfalls möglich, eine beliebige Anzahl von Alternativen zu modellieren. Diese sollten aber in einem Kontext stehen und nicht unabhängig voneinander sein. Zudem müssen alle Guards disjunkt sein.

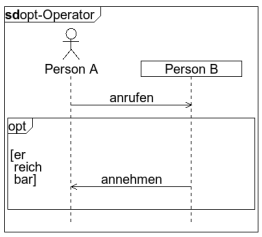
Durch die Verschachtelung der Blöcke ist es aber möglich, auch unabhängige Alternativen zu modellieren.

Es gibt keine Beschränkung der Anzahl Nachrichten, die in einem Block enthalten sein dürfen.

Wichtig: Bei der Nutzung eines alt-Operators müssen alle Fälle abgedeckt sein. Dies kann zum Beispiel durch einen Fall *else* beschrieben werden.

Sequenzdiagramme: opt-Operator

Der opt-Operator dient dazu, optionale Nachrichtenflüsse zu definieren. Im Gegensatz zum bereits vorgestellten alt-Operator ist es damit möglich, diesen Block auch zu überspringen, falls die Nachricht nicht gesendet werden soll.
Ein Beispiel ist ein Telefonanruf, bei dem nur abgehoben wird, wenn die andere Person erreichbar ist.



Sequenzdiagramme: opt-Operator

Auch bei einem opt-Operator ist es nötig, durch Guards die Eintrittsbedingung zu definieren. Ist diese nicht erfüllt, wird der Teil innerhalb des Operators übersprungen.

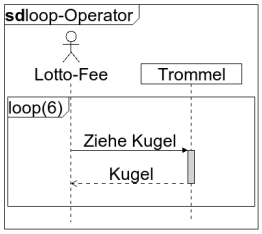
Im Gegensatz zum alt-Operator gibt es keine Möglichkeit, verschiedene Fälle in einem Operator zu definieren. Für mehrere verschiedene Bedingungen muss daher ein eigener opt-Operator eingeführt werden. Innerhalb eines Operators kann ein beliebiger Nachrichtenfluss mit einer beliebigen Anzahl an Nachrichten stattfinden.

Auch dieser Operator kann beliebig verschachtelt und mit anderen Operatoren kombiniert werden.

Sequenzdiagramme: loop-Operator

Der loop-Operator ermöglicht es, einen definierten Nachrichtenfluss zu wiederholen.

Ein Beispiel ist die Ziehung der Lottozahlen *6 aus 49*, bei der die Lottofee 6 mal eine Kugel aus der Trommel zieht.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: loop-Operator

In dem gezeigten Beispiel ist eine feste Anzahl von Abläufen gegeben. Es ist auch möglich, eine minimale und eine maximale Zahl der Form loop(min,max) zu definieren, wobei „*“ für eine beliebige Zahl einschließlich 0 stehen kann. In diesem Fall ist normalerweise zusätzlich ein Guard angegeben, der angibt, wann abgebrochen wird. Ist die min-Anzahl durchlaufen und die Bedingung erfüllt, wird die Schleife verlassen.

Auch der loop-Operator ist nicht in der Lage, verschiedene Fälle zu definieren. Dies geschieht, indem innerhalb die bereits bekannten Operatoren opt und alt verwendet werden.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

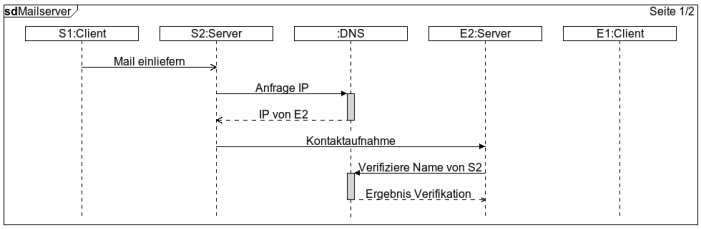
UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: Mailserver (Beispiel)

Beispiel: Wir modellieren ein Protokoll, bei dem ein Client-Rechner S eine E-Mail an einen anderen Client R verschickt.

Weitere Beteiligte sind der Mail-Server von S, der Mail-Server von R und der DNS-Server, der benötigt wird, um Mail-Adressen in die IP-Adressen des entsprechenden Servers umzuwandeln (DNS = domain name system).

Sequenzdiagramme: Mailserver (Beispiel)

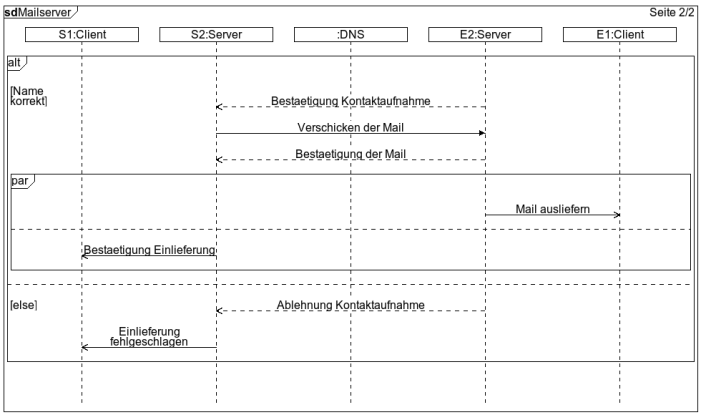


UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: Mailserver (Beispiel)



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme: Mailserver (Beispiel)

Bemerkung: Dieses Sequenzdiagramm ist an den Ablauf im SMTP-Protokoll angelehnt (SMTP = send mail transport protocol), ist jedoch erheblich vereinfacht.

Sequenzdiagramme zu vielen TCP/IP-Netzwerkprotokollen findet man unter:

`http://www.eventhelix.com/Realtimemantra/Networking/`

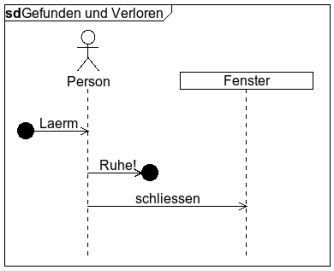
Sequenzdiagramme: weitere Arten von Nachrichten

Es gibt noch einige weitere Arten von Nachrichten:

Gefundene und verlorene Nachrichten

Bei gefundenen und verlorenen Nachrichten werden das Sende- bzw. das Empfangsereignis nicht explizit modelliert. Die Nachrichten tauchen quasi aus der Umgebung auf und verschwinden wieder dorthin.

Solche Nachrichten werden benötigt, wenn die entsprechenden Kommunikationspartner nicht mitmodelliert werden.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

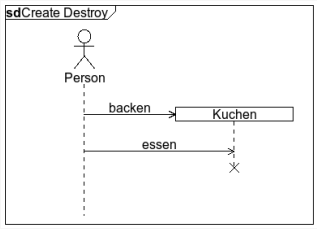
UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Sequenzdiagramme

Erzeugung und Löschung von Objekten

Außerdem kann es passieren, dass Objekte nicht während des ganzen Ablaufs zur Verfügung stehen. Sie können während des Ablaufs dynamisch erzeugt und wieder gelöscht werden.

Dies erfolgt zumeist durch sogenannte Erzeugungs- und Löschnachrichten und wird folgendermaßen dargestellt:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Anwendungsfalldiagramme

Anwendungsfalldiagramme

In frühen Stadien der Entwicklung und bei der Kommunikation mit dem Auftraggeber spielen auch Anwendungsfalldiagramme (engl. use case diagrams) eine große Rolle.

Anwendungsfalldiagramme modellieren die Funktionalität des Systems,

- auf einem hohen Abstraktionsniveau;
- aus der Black-Box-Sicht des Anwenders (das heißt, nur das von außen Sichtbare soll beschrieben werden, nicht die interne Realisierung);
- durch Spezifikation der Schnittstellen.

Die Anwender bzw. Nutzer tauchen als sogenannte Akteure in den Diagrammen auf.

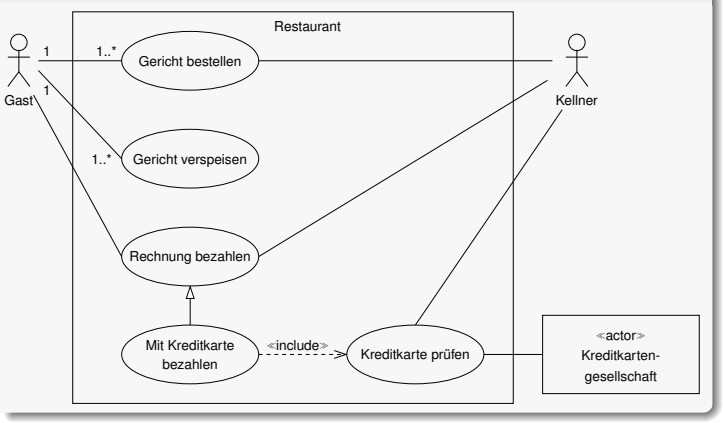
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Anwendungsfalldiagramme

Beispiel: Restaurant



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Anwendungsfalldiagramme

Anwendungsfalldiagramme bestehen aus im Folgenden beschriebenen Komponenten.

Systemgrenze

Die Systemgrenze ist ein Rechteck, das beschreibt, was sich außerhalb und was sich innerhalb des zu erstellenden Systems befindet.

Anwendungsfalldiagramme

Akteur

Ein Akteur ist ein Typ oder eine Rolle, die ein externer Benutzer oder ein externes System während der Interaktion mit dem System einnimmt. Menschliche Akteure werden durch Strichmännchen symbolisiert, andere Akteure durch ein Rechteck, das mit dem Schlüsselwort ≪actor≫ gekennzeichnet ist.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

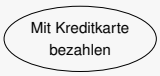
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Anwendungsfalldiagramme

Anwendungsfall

Ein Anwendungsfall ist eine Menge von Interaktionsfolgen, die von dem System bereitgestellt werden und die einen Nutzen für einen oder mehrere Akteure bringen. (Das heißt, es handelt sich dabei um eine Art „Service“ des Systems.) Ein Anwendungsfall wird durch eine Ellipse dargestellt.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Anwendungsfalldiagramme

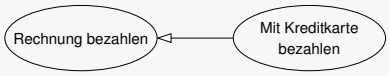
Assoziation

Wie bei Klassendiagrammen gibt es Assoziationen, vor allem zwischen Akteuren und Anwendungsfällen. (Welcher Akteur ist an welchem Anwendungsfall beteiligt?)
Assoziationen dürfen die üblichen Beschriftungen besitzen (Multiplizitäten etc.).

Anwendungsfalldiagramme

Generalisierung/Spezialisierung

Anwendungsfälle können andere Anwendungsfälle spezialisieren, das heißt, sie können von ihnen erben. Dabei werden – wie bei Klassen – auch alle Assoziationen geerbt.



Auch Spezialisierungsbeziehungen zwischen Akteuren sind möglich.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

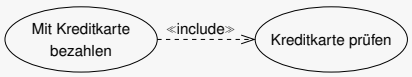
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Anwendungsfalldiagramme

Include-Beziehung

Bei einer Include-Beziehung wird modelliert, dass ein Anwendungsfall die Funktionalität eines anderen Anwendungsfalles auf jeden Fall beinhaltet. Das heißt, der zweite Anwendungsfall wird immer als eine Art „Unterprozedur“ aufgerufen.



Es gibt auch sogenannte Extend-Beziehungen, bei denen ein Anwendungsfall nur unter bestimmten Bedingungen in einen anderen Anwendungsfall eingebunden wird.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Anwendungsfalldiagramme

Bemerkungen:

- Anwendungsfalldiagramme sollten nicht zu viele Details enthalten.
- Sie sind ein einfaches Mittel, um Anwenderwünsche zu diskutieren, und sollten nur eine grobe Sicht auf die Funktionalität des Systems darstellen.
- Bei Bedarf müssen bestimmte Anwendungsfälle dann noch textuell oder mit Hilfe anderer UML-Diagramme genauer beschrieben werden.

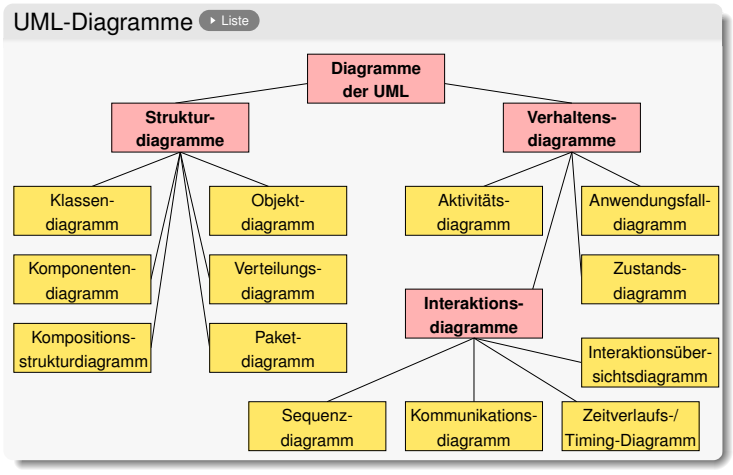
Weitere UML-Diagramme

Überblick über weitere UML-Diagramme

Wir schließen die Vorlesung mit einem kurzen Überblick über die noch fehlenden Typen von UML-Diagrammen ab.

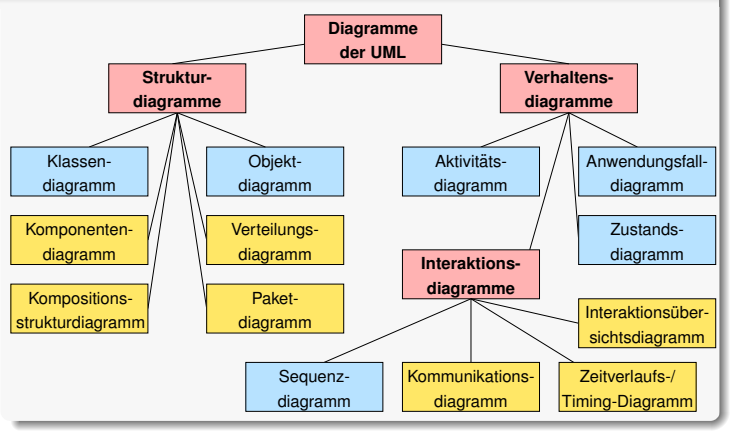
Zuletzt gibt es dann noch ein paar Abschlussbemerkungen.

Überblick über weitere UML-Diagramme



Überblick über weitere UML-Diagramme

UML-Diagramme (blau: bereits behandelte Diagramme)



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Überblick über weitere UML-Diagramme

Wir beginnen zunächst mit den drei noch fehlenden Arten von Interaktionsdiagrammen:

- Kommunikationsdiagramme
- Timing-Diagramme bzw. Zeitverlaufsdiagramme
- Interaktionsübersichtsdiagramme

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Kommunikationsdiagramme

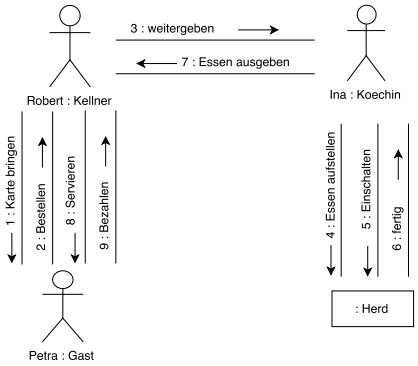
Kommunikationsdiagramme enthalten dieselbe Information wie Sequenzdiagramme, werden jedoch anders dargestellt.

Während bei Sequenzdiagrammen der Fokus eher auf dem zeitlichen Ablauf liegt, heben Kommunikationsdiagramme eher die Kommunikationsbeziehungen der Teilnehmer hervor.

Kommunikationsdiagramme gehören – genau wie Sequenzdiagramme – zur Klasse der Interaktionsdiagramme.

Kommunikationsdiagramme

Das Sequenzdiagramm Restaurantbesuch ▸ Diagramm wird folgendermaßen als Kommunikationsdiagramm dargestellt:



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Kommunikationsdiagramme

Dabei werden die Kommunikationspartner wie bisher durch Rechtecke oder Strichmännchen dargestellt. Es wird jedoch kein zeitlicher Ablauf mehr dargestellt.

Die Interaktionen bzw. Nachrichten werden durch Linien notiert, an denen die Namen der Nachrichten und die Senderichtung ($\rightarrow$) stehen.

Die Nummerierung der Nachrichten (1, 2, 3, . . .) gibt die Reihenfolge an. Durch Buchstaben hinter den Nummern (2a, 2b, . . .) beschreibt man parallele Nachrichten, die in beliebiger Reihenfolge angeordnet sein können.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

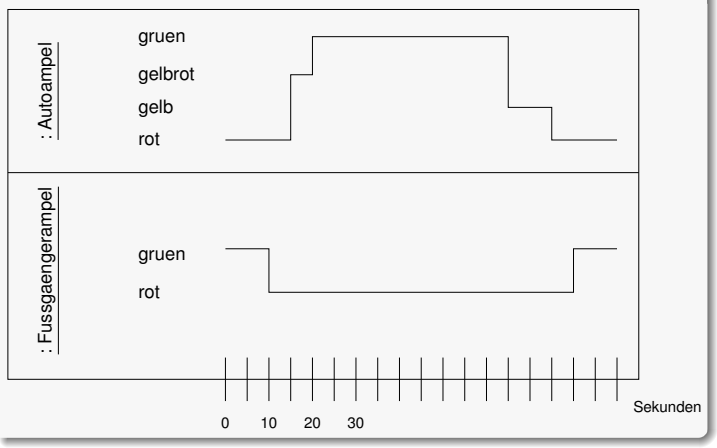
Timing-Diagramme

Timing-Diagramme sind in der Elektrotechnik weit verbreitet und zeigen an, zu welchem Zeitpunkt welcher Kommunikationspartner welchen Zustand einnimmt.

Dabei wird von links nach rechts (horizontal) die Zeit aufgetragen und von oben nach unten werden die Kommunikationspartner und deren Zustände aufgetragen.

Timing-Diagramme

Beispiel: Ampelschaltung



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Interaktionsübersichtsdiagramme

Interaktionsübersichtsdiagramme sind im Wesentlichen Aktivitätsdiagramme, die – anstatt der Aktionen – hierarchisch weitere Interaktionsdiagramme (Sequenzdiagramme, Kommunikationsdiagramme, Timing-Diagramme) enthalten können.

Sie werden eingesetzt, wenn es eine größere Menge verschiedener Arten von Aktionen gibt, über die man sonst nur schwer den Überblick behalten kann.

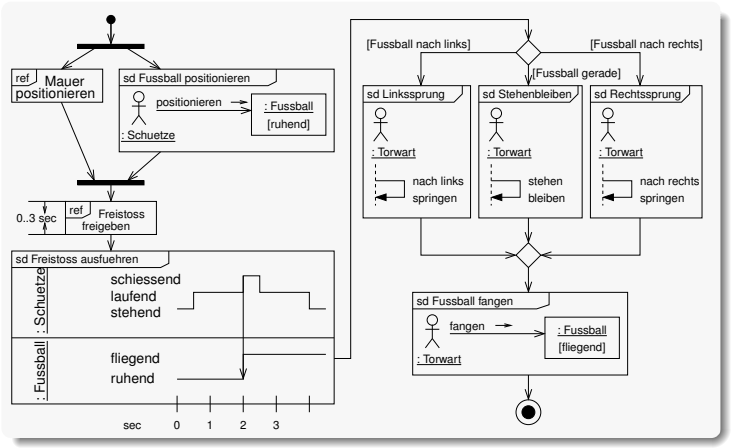
Als Beispiel betrachten wir ein Interaktionsübersichtsdiagramm, das den Ablauf eines Freistoßes in einem Fußballspiel modelliert.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Interaktionsübersichtsdiagramme



Interaktionsübersichtsdiagramme

Bemerkungen:

- Interaktionsreferenzen (Schlüsselwort ref) werden verwendet, um an anderer Stelle definierte Interaktionsdiagramme wiederzuverwenden.
- Anstatt der Aktionen wie in Aktivitätsdiagrammen werden ganze Interaktionsdiagramme verwendet, die alle sd als Diagrammtyp für Interaktionsdiagramme erhalten.

Überblick über weitere UML-Diagramme

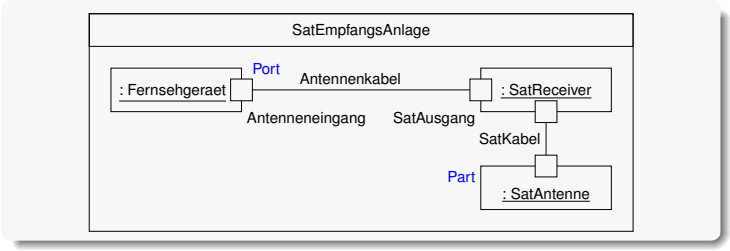
Wir sehen uns nun noch (ganz kurz) die fehlenden vier Typen von Strukturdiagrammen an. ▸ UML-Übersicht

- Kompositionsstrukturdiagramme
- Paketdiagramme
- Verteilungsdiagramme
- Komponentendiagramme

Im weitesten Sinne dienen sie alle dazu, die (übergeordnete) Struktur bzw. Architektur eines Systems darzustellen.

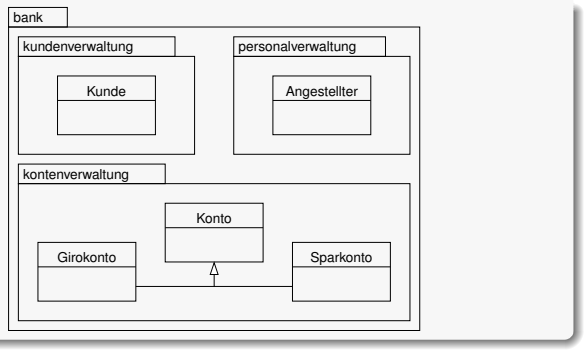
Kompositionsstrukturdiagramme

Kompositionsstrukturdiagramme (engl. composite structure diagrams) beschreiben die interne Struktur von Komponenten (z.B. Klassen) in einer White-Box-Darstellung. Instanzen von durch Komposition verbundenen Klassen werden dabei als sogenannte Parts innerhalb der Klasse dargestellt. Ports, dargestellt als kleine Quadrate, sind Verbindungsstellen zwischen Parts und beinhalten Schnittstellen.



Paketdiagramme

Ein großes Softwaresystem muss in Pakete bzw. Module gegliedert werden. Paketdiagramme (engl. package diagrams) beschreiben die statische Struktur eines großen Systems durch Zusammenfassen von Klassen in Paketen.



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

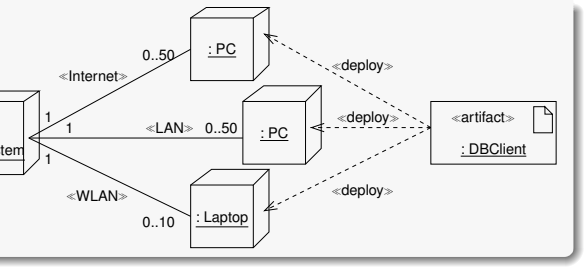
Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Verteilungsdiagramme

Verteilungsdiagramme (engl. deployment diagrams) beschreiben die Hardware-Komponenten, die in einem System benutzt werden, und wie diese in Beziehung stehen.

Sie können auch konkrete physische Informationseinheiten (Dateien, etc.) enthalten, die als Artefakte bezeichnet werden.



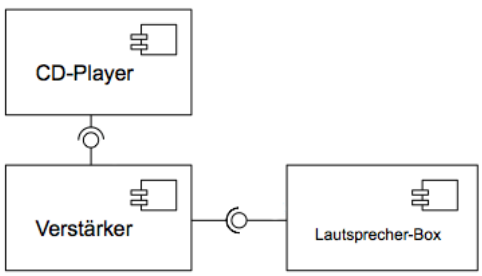
UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Komponentendiagramme

Ein Komponentendiagramm (engl. component diagram) beschreibt im Gegensatz dazu die Software-Architektur eines Systems. Es beantwortet die Fragen, wie Klassen zur Laufzeit zu größeren Komponenten zusammengefasst werden, und welche Schnittstellen (Services) angeboten und genutzt werden.



Abschließende Bemerkungen

UML als semi-formale Modellierungssprache

UML ist eine semi-formale Modellierungssprache, das heißt, nicht bei jedem Sprachelement gibt es vollständige Einigkeit über die Bedeutung.

Trotz dieser Kritik ist die UML ein großer Fortschritt gegenüber früher genutzten Modellierungssprachen, da sie vereinheitlichte Diagramme bereitstellt, die von jedem Beteiligten am Softwareentwicklungsprozess verstanden werden können.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Abschließende Bemerkungen

Konsistenz von Modellen

Bei der Beschreibung eines großen Systems durch mehrere Modelle ist auch darauf zu achten, dass die Modelle untereinander konsistent sind. (Beispielsweise: Klassennamen, die in einem Sequenzdiagramm verwendet werden, tauchen auch im Klassendiagramm auf.)

Es gibt Ansätze, solche Konsistenz (halb-automatisch) zu erreichen, indem man sogenannte Modelltransformationen durchführt und verschiedene Diagramme ineinander übersetzt.

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Modellierung
WS 17/18

UML
Einführung & OO
Klassen- und Objektdiagramme
Aktivitätsdiagramme
Zustandsdiagramme
Weitere UML-Diagramme

Abschließende Bemerkungen

Es gibt noch viele Aspekte in der UML, die wir nicht betrachtet haben.

Weitergehende Informationen über UML gibt es in zahllosen Büchern (siehe Literaturliste) und auf den Seiten der OMG (Object Management Group):

```
http://www.omg.org/technology/documents/formal
/uml.htm
```

Und natürlich gibt es neben UML und Petrinetzen noch zahlreiche andere Modellierungssprachen!