Auf zu neuen „Daten“ – Datenbanken, Datenübertragung und Datenschutz im Informatikunterricht

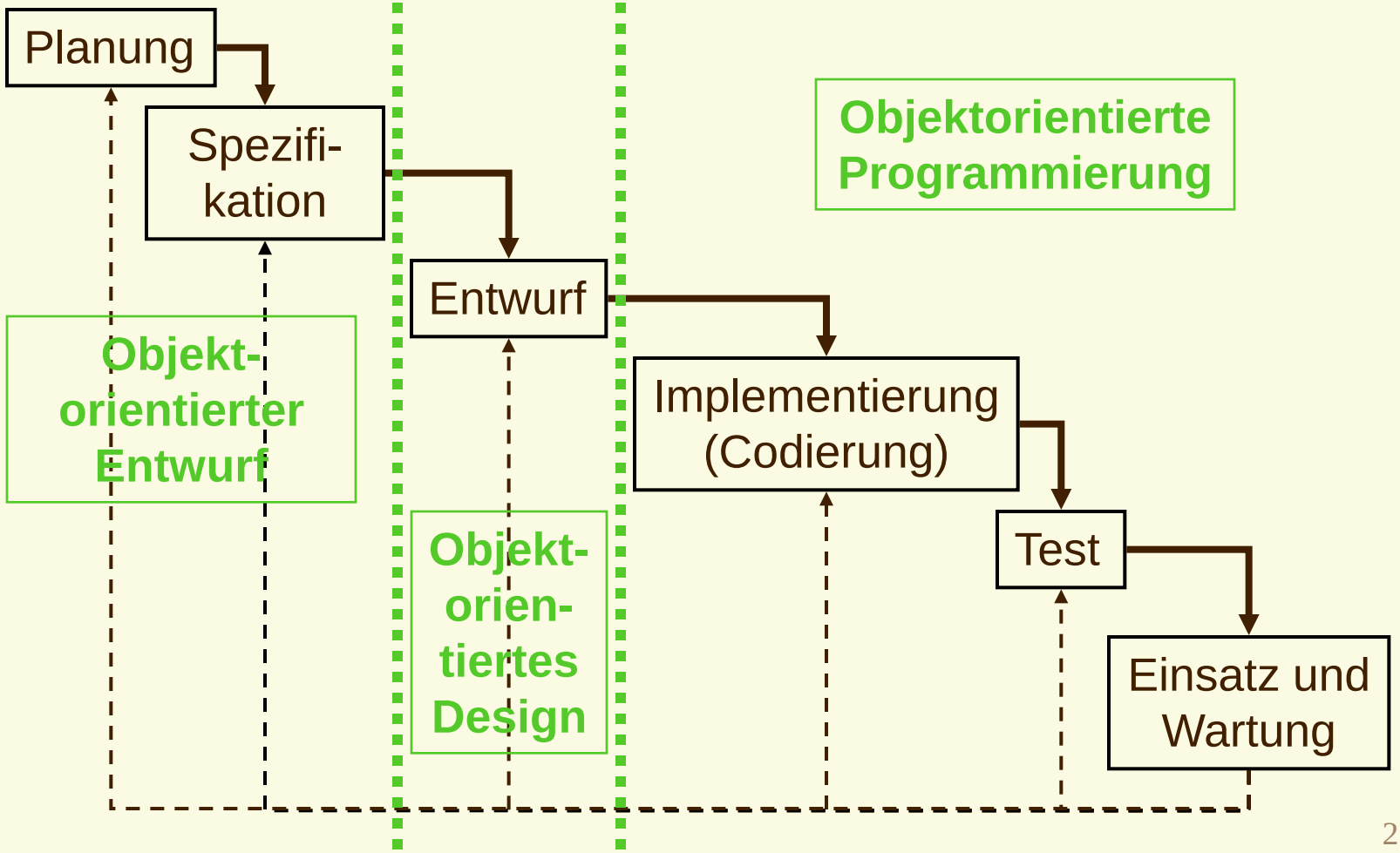
Informatisches Modellieren mit UML

Dr. Henry Herper

Otto-von-Guericke-Universität Magdeburg
Institut für Simulation und Graphik

Lisa-Weiterbildung – 25.09.2007

Softwarelebenszyklus (objektorientiertes Vorgehensmodell)



Objektorientierter Entwurf

Voraussetzung für die Softwareentwurfsphase:

abgeschlossene Analysephase mit Dokumentation der Ergebnisse in einem Pflichtenheft oder Analysemodell

Was leistet das das zu entwickelnde System?

Ergebnis der Softwareentwurfsphase:

Entwicklung des Entwurfsmodells, das in der Implementierung zum Produkt verfeinert und vervollständigt wird

Wie wird das zu entwickelnde System realisiert?

Objektorientierter Softwareentwurf

Ziele des Entwurfs:

- Umsetzung der funktionalen und nichtfunktionalen Anforderungen
- Festlegung der Entwurfsarchitektur und Aufteilung in Subsysteme
- Definition der wesentlichen Klassen
- insgesamt: Erstellen des Entwurfsmodells, einer Abstraktion der Systemimplementierung, so dass die Implementierung eine Verfeinerung des Entwurfs ist.

/Peter Forbrig, Immo O. Kerner, Lehr- und Übungsbuch Softwareentwicklung, Fachbuchverlag Leipzig, 2004/

OOS – Grundlegende Definitionen - Objekt

„Ein **Objekt** ist allgemein ein Gegenstand des Interesses, insbesondere einer Beobachtung, Untersuchung oder Messung. Objekte können Dinge und Begriffe sein.

In der objektorientierten Softwareentwicklung besitzt ein Objekt bestimmte Eigenschaften und reagiert mit einem definierten Verhalten auf seine Umgebung. Außerdem besitzt jedes Objekt eine Identität, die es von allen anderen Objekten unterscheidet.

Die Eigenschaften eines Objektes werden durch dessen Attributwerte ausgedrückt, sein Verhalten durch eine Menge von Methoden.“

/Helmut Balzert; Lehrbuch der Softwaretechnik; Spektrum Verlag, 1996/

OOS – Darstellungsarten von Objekten

Mögliche Darstellungsarten in der Sekundarstufe I:

- Umgangssprache
- MindMap
- Klassenmodell (UML)
- Punktnotation

Festlegungen zur Darstellung von Klassen und Objekten im Unterricht notwendig, wegen

- Vergleichbarkeit von Lösungen
- Einhaltung und Training von Normen

/Steffen Friedrich, Schulinformatik zwischen PISA und Bildungsstandards, Lehrertag Magdeburg, 2007/

OOS – Grundlegende Definitionen - Klasse

„Eine **Klasse** beschreibt eine Sammlung von Objekten mit gleichen Eigenschaften (Attributen), gemeinsamer Funktionalität (Methoden), gemeinsam en Beziehungen zu anderen Objekten und gemeinsamer Semantik.“

/Helmut Balzert; Lehrbuch der Softwaretechnik; Spektrum Verlag, 1996//

OOS – Grundlegende Definitionen – Attribut – Methode - Botschaft

„Die **Attribute** beschreiben die Daten bzw. Eigenschaften einer Klasse. Alle Objekte einer Klasse besitzen dieselben Attribute, jedoch unterschiedliche Attributwerte. Das bedeutet für die Implementation, dass jedes Objekt Speicherplatz für alle seine Attribute erhalten muss.
Eine **Methode** ist ein Algorithmus, der einem Objekt zugeordnet ist und von diesem abgearbeitet werden kann.
Eine **Botschaft** ist eine Nachricht, die den Aufruf einer Methode gleichen Namens zur Folge hat.“

/Helmut Balzert; Lehrbuch der Softwaretechnik; Spektrum Verlag, 1996//

UML – Unified Modeling Language

/Quelle: http://www.omg.org /

UML - Unified Modeling Language

UML – Unified Modeling Language

• mit UML wurde eine **einheitliche Notation** für unterschiedliche Einsatzgebiete geschaffen (z.B. Datenbankanwendungen, Echtzeitsysteme, Graphikprogramme, Workflow-Anwendungen)

• auf objektorientierte Entwicklungen ausgerichtet

• Modellierungssprache, die basierend auf Diagrammen eine definierte Systembeschreibung realisiert

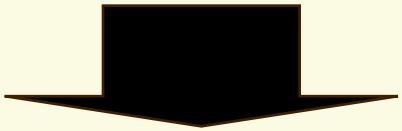
• **Ziel: visuelle Modellierung zur Unterschützung der Entwicklung von Softwaresystemen für alle Projektbeteiligten in allen Projektphasen**

• **UML ist eine Sprache zur Analyse, Spezifikation, Visualisierung, Konstruktion und Dokumentation von Software**

UML - Unified Modeling Language

UML – Unified Modeling Language

Standard zur Gewinnung objektorientierter Spezifikation



Automatische Generierung von Quelltext

Bedingung:
Um vom Modell zum Quellcode und vom Quellcode zum Modell zu kommen, müssen eine **präzise Semantik** und **ein-eindeutige Abbildungsregeln** vorhanden sein.

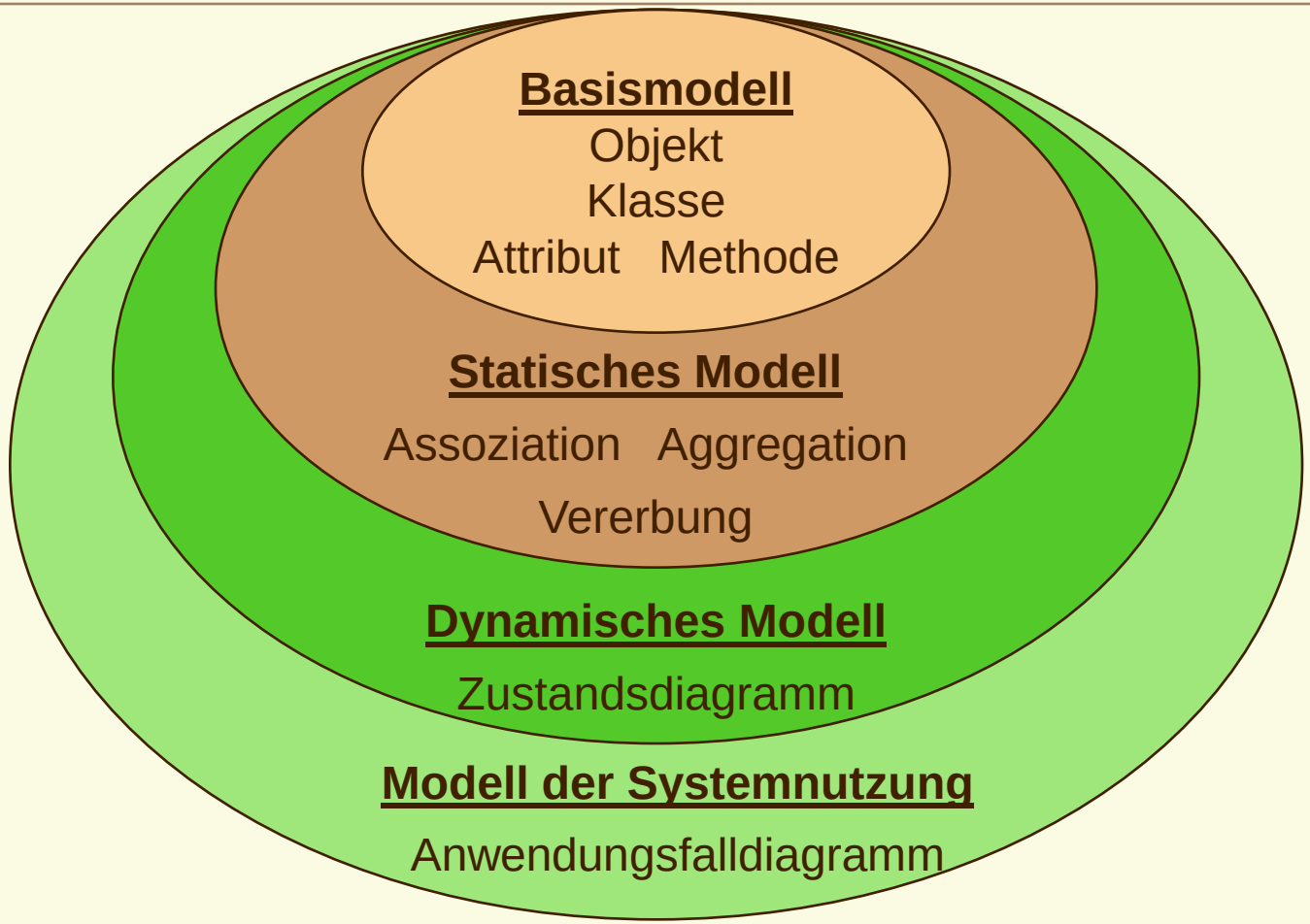
UML - Diagramme

UML stellt eine Menge von Diagrammen zur Verfügung, die verschiedene Sichten auf das System darstellen

➔ Ein und dasselbe UML-Element, das genau einmal im Modell vorkommt, kann auf ganz unterschiedliche Weise für unterschiedliche Sichten und Lesergruppen mehrfach dargestellt werden.

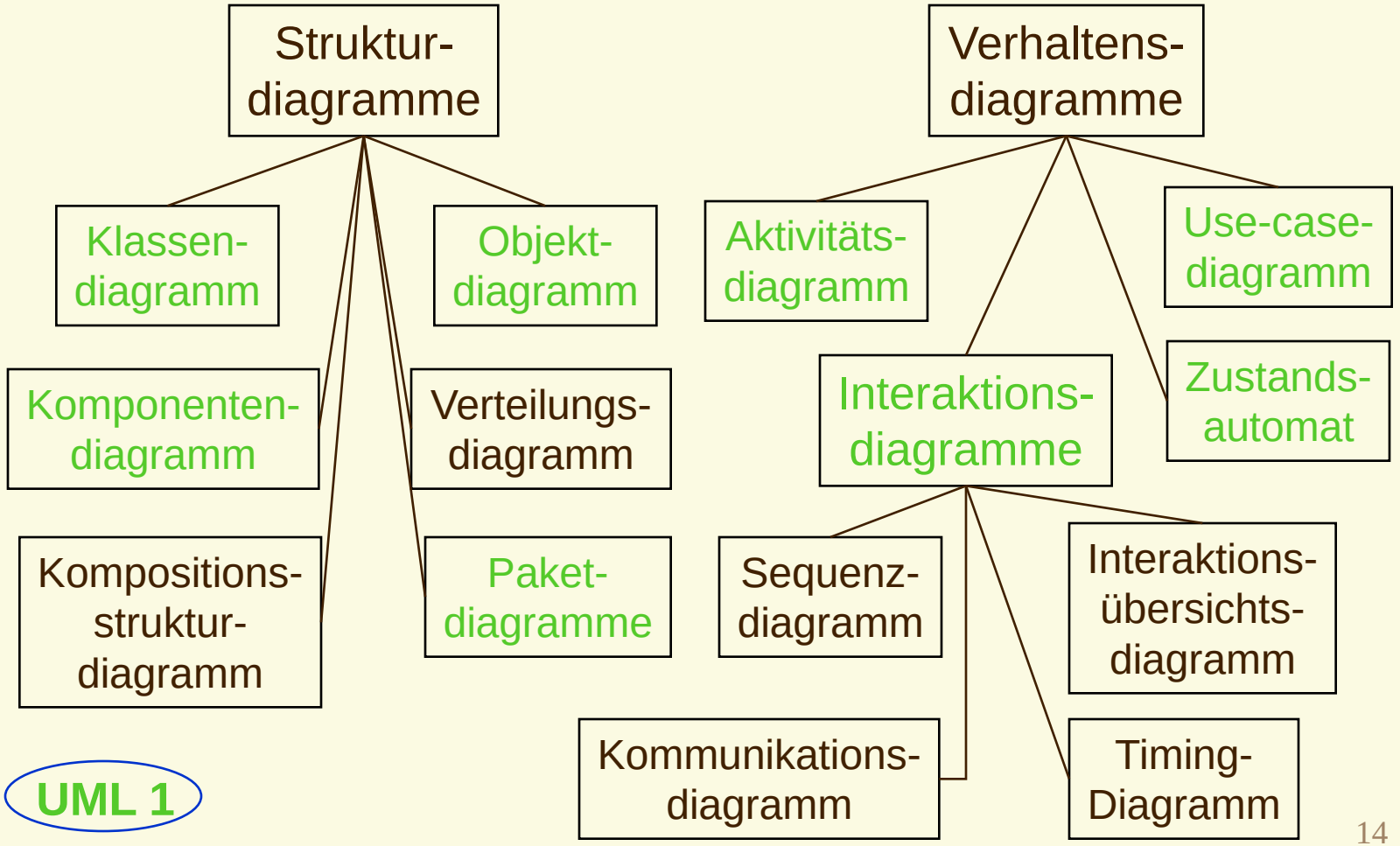
Für **Verhaltensdiagramme** gibt es neben der Möglichkeit der graphischen Darstellung auch die tabellarische Darstellungsform.

Objektorientiertes Modellverständnis

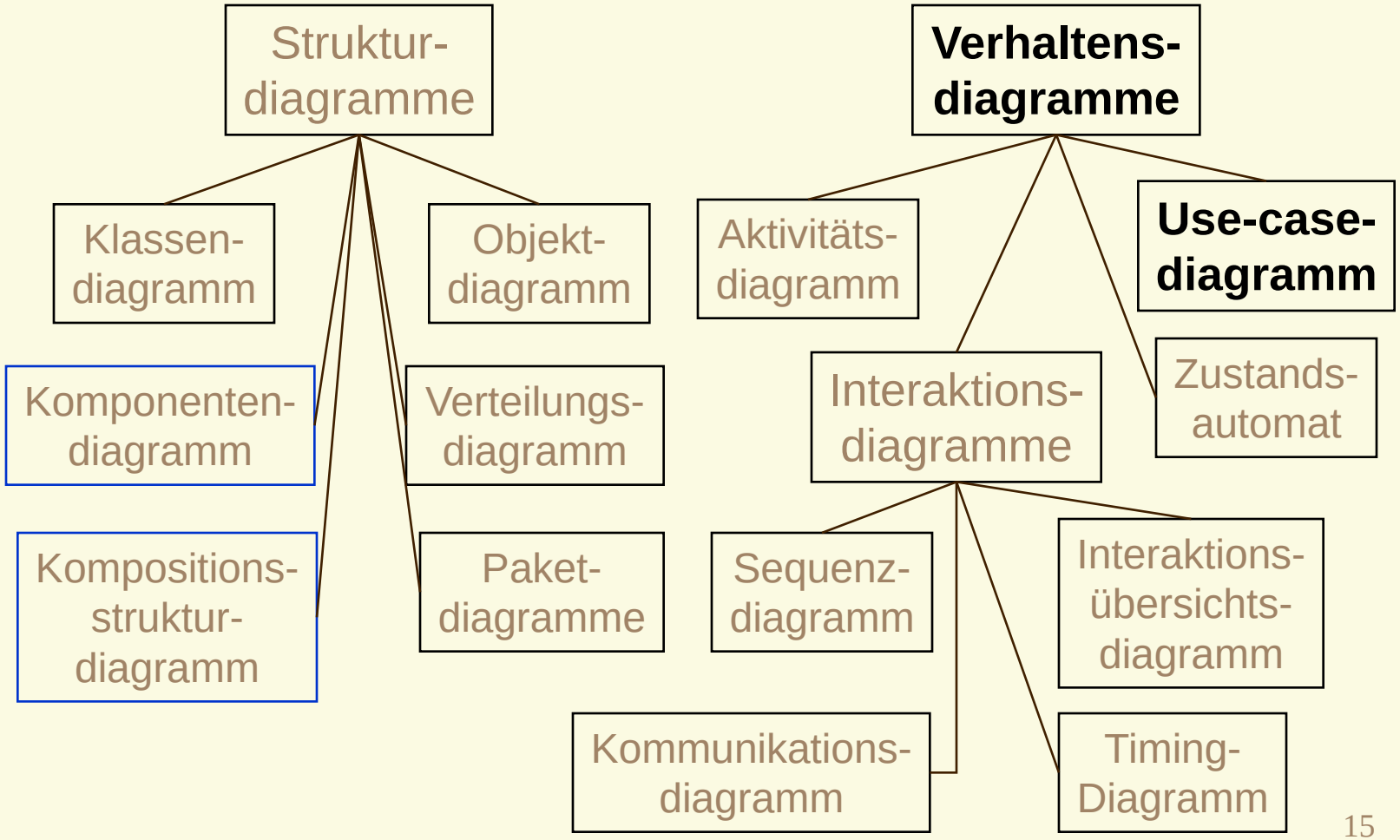


/Quelle: Peter Forbrig, Objektorientierung, LOGIN Heft 128/129, S. 13/

Diagrammtypen der UML 2

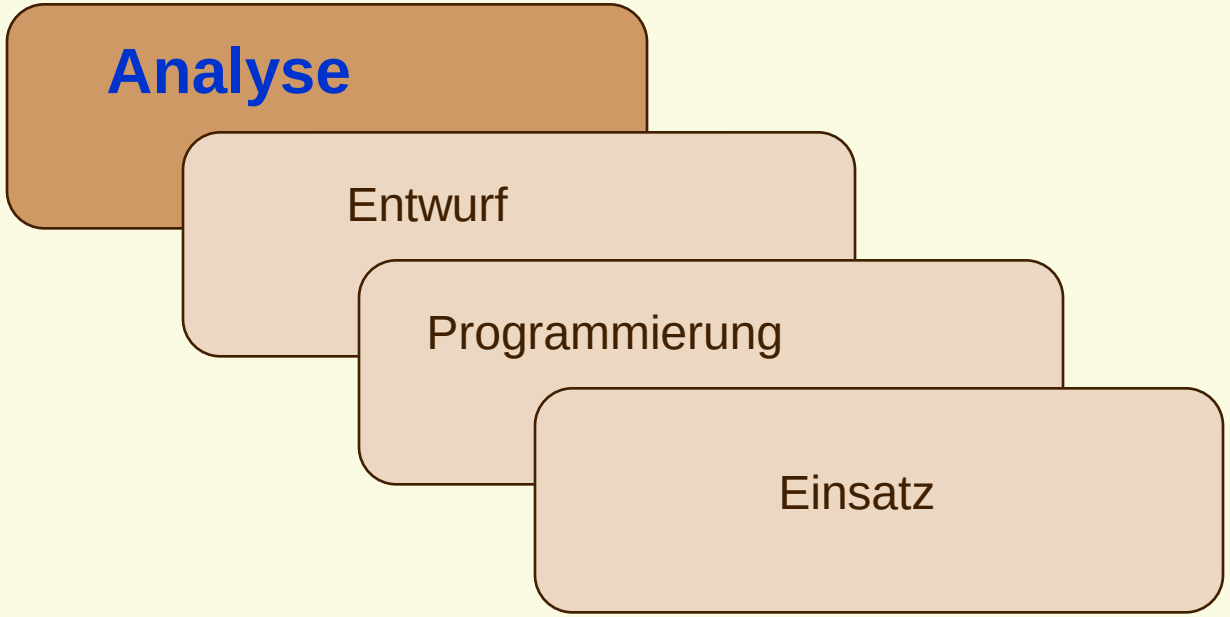


Diagrammtypen der UML 2



Anwendungsfalldiagramme im Softwarelebenszyklus

Was soll mein geplantes System leisten?



Resultat der Anforderungsanalyse sind ein oder mehrere **Anwendungsfalldiagramme**

UML – Anwendungsfälle (Grundbegriffe)

Szenario

„Ein Szenario ist eine spezifische Folge von Aktionen, die zur Verdeutlichung des Verhaltens eines Systems dient.“

konkretes Szenario

„Ein konkretes Szenario ist eine spezifische Folge von Aktionen, die von konkreten Akteuren (Personen oder Systemen) unter konkreten Randbedingungen durchgeführt werden.“

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfälle (Grundbegriffe)

abstraktes Szenario

„Ein abstraktes Szenario ist eine spezifische Folge von Aktionen, die von Akteuren (im Sinne von Rolle) unter abstrakten Randbedingungen durchgeführt werden.“

Anwendungsfall

„Ein Anwendungsfall beschreibt eine Menge von Aktivitäten eines Systems aus der Sicht seiner Akteure, die für diese zu wahrnehmbaren Ereignissen führen. Er wird immer durch einen Akteur ausgelöst.“

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfälle (Use-cases)

Anwendungsfälle sind eine Sammlung von Szenarios über den Systemeinsatz. Jedes Szenario beschreibt eine Sequenz von Schritten. Die solche Sequenzen auslösenden Entitäten werden als **Akteure** bezeichnet. Sie sind externe Kommunikationspartner des Use-Cases.

Die **Summe der Anwendungsfälle** beschreibt den **Projektumfang**.

Die Anwendungsfallanalyse ist kein Werkzeug zum Softwaredesign, sondern dient zur **Beschreibung der Anforderungen an ein Softwaresystem**.

UML – Anwendungsfälle

Der Anwendungsfall ist als **Kommunikationsmittel zwischen Entwickler und potentiellen Anwender** geeignet.

→ Anwender soll bereits in die Frühphase der Systementwicklung einbezogen werden (Kundenakzeptanz)

→ Anwendungsfälle können aus der Sicht unterschiedlicher Nutzergruppen unterschiedlich definiert werden

→ Anwendungsfälle können in (Teil-)Anwendungsfälle zerlegt werden und mit der Enthält-Beziehung zusammengefügt werden

UML – Anwendungsfall – Textbeschreibung eines Szenarios

„Ziel:

Was ist das Ziel des Anwendungsfalls?

In zwei bis drei kurzen Sätzen ist das Ziel zu beschreiben, welches mit der Ausführung eines Geschäftsvorfalls (Instanz des Anwendungsfalls) verbunden ist.

Anwendungskontext:

In welchem Kontext tritt der Anwendungsfall auf?

Hier wird beschrieben, in welcher Umgebung die spätere Anwendung genutzt wird.

Bereich:

Was gehört/gehört nicht zum System?

Beschreiben Sie hier die Bereichsgrenzen des Systems.“

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfall – Textbeschreibung eines Szenarios

Niveau:

Welchen Detaillierungsgrad hat der Anwendungsfall?

Beim Niveau kann man komplexes Niveau (1), Aufgabenniveau (2) und Funktionsniveau (3) unterscheiden.

Primärer Akteur:

Wer ist primärer Akteur (Rollenname)?

Hier sind die Akteure zu notieren, die in der Lage sind, einen neuen Geschäftsvorfall auszulösen.

Sekundärer Akteur:

Welche weiteren Akteure werden vom System zur Ausführung des Anwendungsfalls benötigt?

Hier sind die Akteure zu benennen, die in einem Geschäftsprozess involviert sind, ihn aber nicht auslösen können.

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfall – Textbeschreibung eines Szenarios

Betroffene (Stakeholder):

Beschreiben Sie, wer neben den direkten Anwendern durch den Anwendungsfall betroffen ist. Welche Interessen hat der Betroffene am Anwendungsfall?

Bei den Betroffenen handelt es sich um Personen, die keine Akteure sind, also nicht aktiv in den Geschäftsprozess eingreifen, von der Nutzung des Systems aber trotzdem tangiert werden.

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfall – Textbeschreibung eines Szenarios

Vorbedingungen:

Welchen Ausgangszustand der Umgebung benötigt der Anwendungsfall?

Nachbedingung im Erfolgsfall:

Welcher Zustand der Umgebung wird bei erfolgreicher Abarbeitung des Anwendungsfalls garantiert?

Eine kurze Beschreibung des erreichten Zustandes genügt an dieser Stelle.

Nachbedingung in Fehlerfällen:

Welche Zustände werden in Fehlerfällen garantiert?

Einträge sollten mit folgender Struktur erfolgen:

Wenn <Fehlerfall> dann <garantierter Zustand>.

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfall – Textbeschreibung eines Szenarios

Auslöser:

Was ist der Auslöser (Trigger) für den Anwendungsfall?

Hier sind das oder die Ereignisse zu beschreiben, durch die ein neuer Geschäftsvorfall ausgelöst wird.

Interaktionsfolge:

Wie sieht die Interaktionsschrittfolge im Erfolgsfall aus?

Beschreiben Sie die Interaktionsschrittfolge in der Art:

“ Schritt 1 Akteur führt ... aus, System reagiert mit ...,

Schritt 2 Akteur führt ... aus, System ...“.

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfall – Textbeschreibung eines Szenarios

Ausnahmen und Fehlerfälle (Extensions):

Welche Ausnahmen sind möglich?

Schritt# Alternativer Interaktionsschritt

Einträge mit folgender Struktur: <Bedingung für Alternative> : <Aktion oder Name des erweiterten (extending) Anwendungsfalls>

Variationen:

Für welchen Schritt gibt es Variationen?

Falls ein oben genannter Interaktionsschritt in unterschiedlichen Ausprägungen (Variationen) technisch realisiert werden kann, dann wird das hier beschrieben.

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfall – Textbeschreibung eines Szenarios

Enthaltene (Included) Anwendungsfälle:

Aufzählung aller Anwendungsfälle, die in Interaktionsschritten genutzt wurden.

Zusätzliche optionale Informationen:

Risiko:

Welche Auswirkungen hat eine fehlerhafte Ausführung des Anwendungsfalls in Bezug auf das System oder die Organisation?

Ausführungszeit:

Wie lange kann die Ausführung des Anwendungsfalls dauern?

Häufigkeit:

Wie oft wird der Anwendungsfall ausgeführt?

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfall – Textbeschreibung eines Szenarios

Übergeordnete Anwendungsfälle:

In welchen Anwendungsfällen ist der Anwendungsfall enthalten?

Kommunikationskanäle zu den primären Akteuren:

Welche Medien werden zur Kommunikation benutzt (z.B. E-Mail, Telefon; bestimmte Datenformate)?

Kommunikationskanäle zu den sekundären Akteuren:

Welche Medien werden zur Kommunikation benutzt (z.B. E-Mail, Telefon; bestimmte Datenformate)?

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Anwendungsfall – Textbeschreibung eines Szenarios

Offene Punkte:

Über welche offenen Probleme muss noch entschieden werden?

Geplant für:

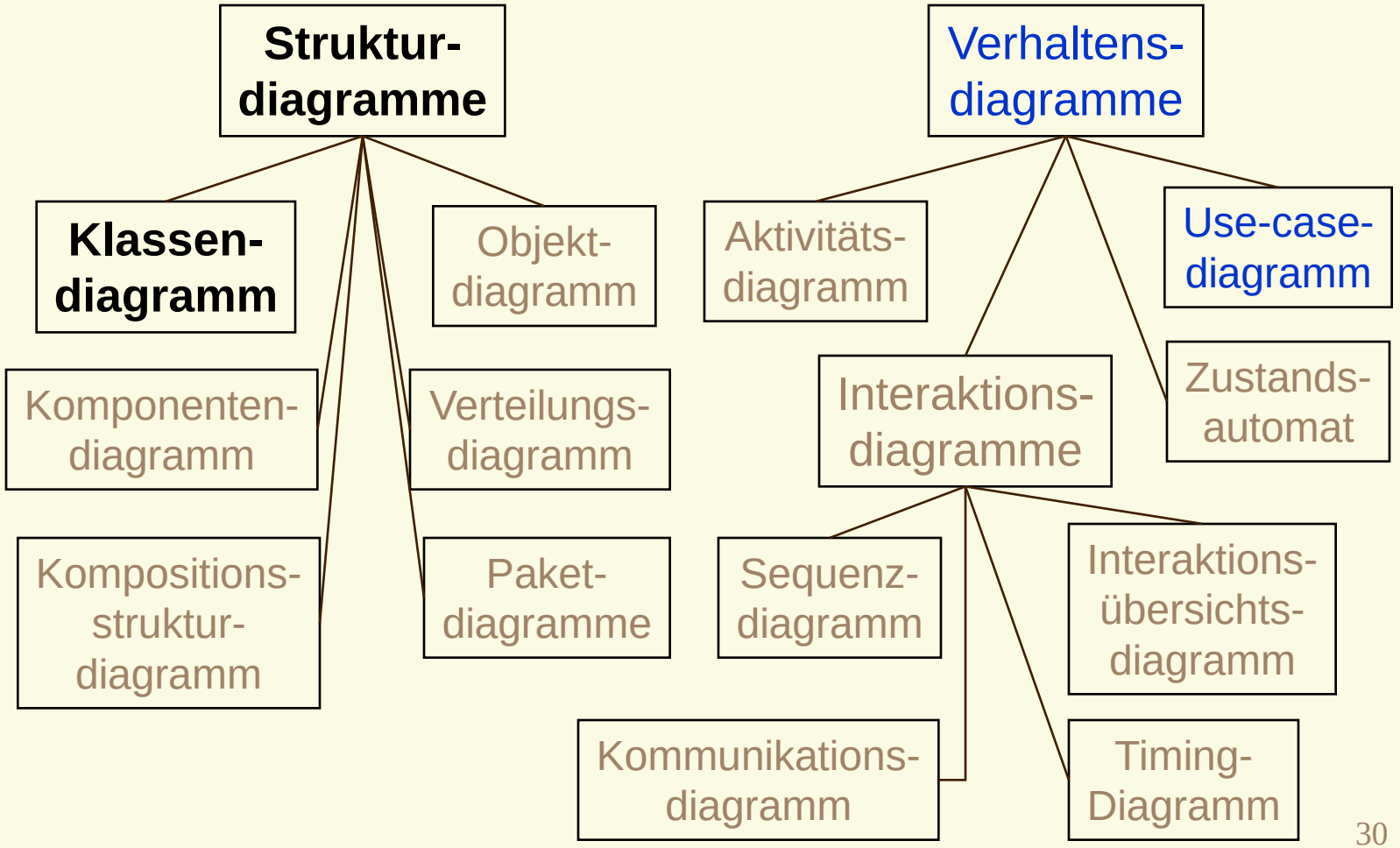
Wann oder in welchem Release wird der Anwendungsfall unterstützt?

Managementinformation: .

Welche Informationen sind für das Management noch wichtig?

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

Diagrammtypen der UML 2



UML – Analysediagramme

UML-Klassendiagramme beschreiben das statische Verhalten des Systems.

➔ Strukturdiagramme

Die Klassen sollten dabei so modelliert werden, dass sie sich möglichst einfach (automatisch?) in eine objektorientierte Programmiersprache übertragen lassen. Dazu müssen verschiedene Beziehungen zwischen den Klassen definiert werden.

UML – Klassendiagramm

Die Klassendiagramme repräsentieren der Kern der gesamten Modellierungssprache.

Im Rahmen der UML wird eine **Klasse als Typ** interpretiert, dessen **Ausprägungen Objekte** heißen.

Klassendiagramme sollen folgende Frage beantworten:

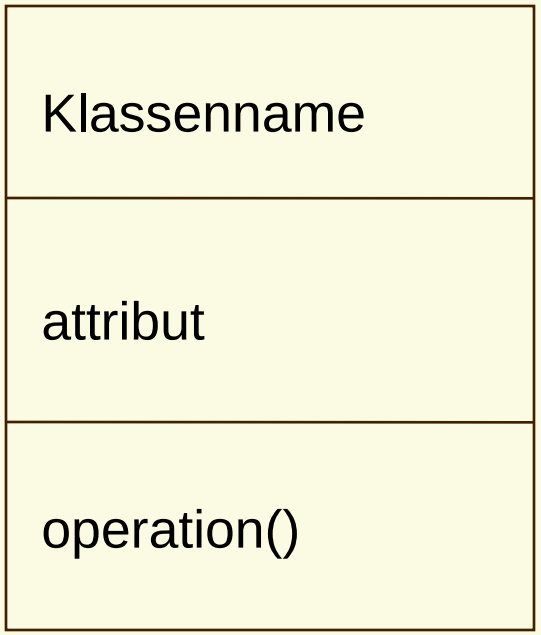
„Wie sind Daten- und Verhalten meines Systems im Detail strukturiert?“

Klassendiagramm: engl. Class Diagramm

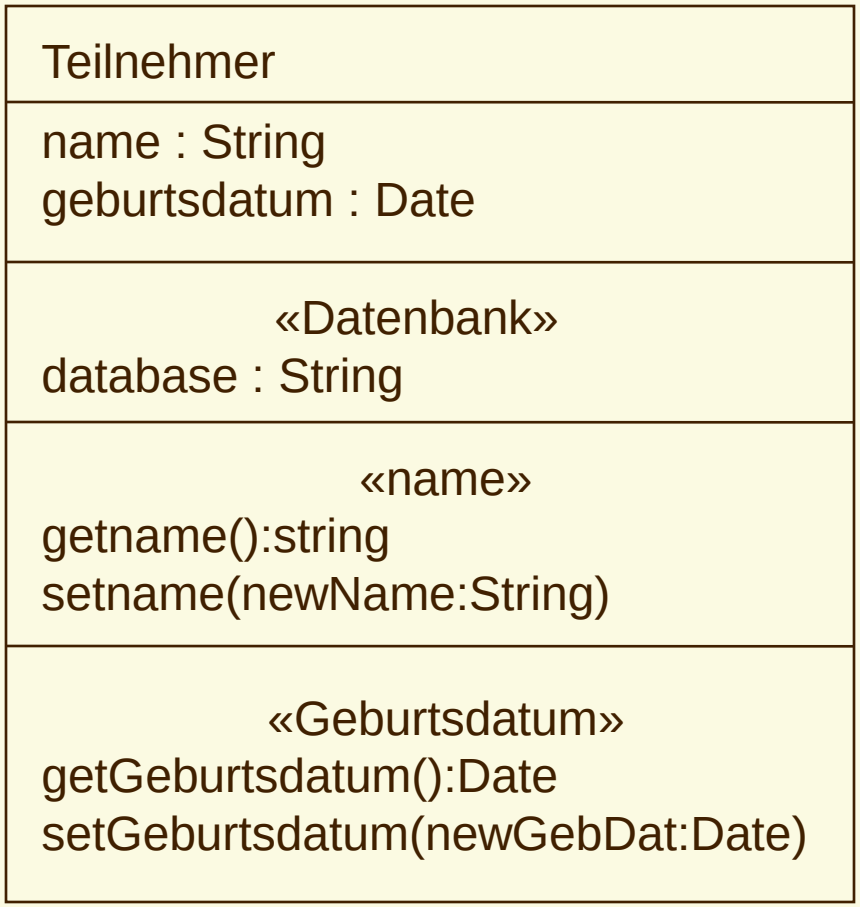
UML - Klassendiagramm

Eine Klasse beschreibt eine Menge von Objekten mit gemeinsamer Semantik, gemeinsamen Eigenschaften und gemeinsamen Verhalten. Sie wird durch das Symbol „Rechteck“ repräsentiert. Das Rechteck teilt sich häufig in 3 Bereiche, für den Klassenamen, die Attribute und die Operationen. Anzahl und Inhalt der Rechteckflächen ist nur für den Klassennamen festgelegt.

Der **Klassenname muss modellweit** bzw. paketweit **eindeutig gewählt werden**.



UML – Klassendiagramm - Beispiele



/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

UML – Modellbeschreibung

Bei der Systemanalyse ist es erforderlich, möglichst viel Wissen über das reale System zu erfassen. Bei dem Aufbau der Klassenstruktur sind folgende Regeln hilfreich:

- aus den Substantiven (in den Anwendungsfällen) werden die Klassen des Modells ausgewählt
- Verben bilden die Operationen der Klassen
- Substantive, die sich auf klassenbildende Substantive beziehen, werden zu Attributen

Problem:

Schaffung einer geeigneten Klassenstruktur

UML – Attribute - Propertys

Attribute repräsentieren die strukturellen Eigenschaften von Klassen. Sie bilden das Datengerüst der Klassen. Zur Laufzeit, bei der Erzeugung von Objekten, werden den Attributen der Objekte konkreten Werten des definierten Typs zugewiesen.

Das Attribut muss nur durch seinen Namen spezifiziert werden, der in der Klasse eindeutig sein muss.

Es soll nicht direkt auf die Attributwerte zugegriffen werden können, sondern nur durch Verwendung der Methoden!

UML – Attribute

Ein nur aus einem Wort bestehender Attributname wird klein geschrieben. Besteht ein Attributname aus mehreren Worten, so werden die Wörter zusammengefügt. Das erste beginnt mit einem Kleinbuchstaben, alle anderen mit einem Großbuchstaben.
Ein Objekt hat für jedes **Attribut** seiner Klasse **einen spezifischen Wert**. Ein Typ kann durch Doppelpunkt getrennt hinter dem Attribut angegeben werden.
Attribute können auch in Form einer Assoziation abgebildet werden.

[sichtbarkeit] [/] name [: typ] [multiplizität]

[=vorgabewert] [{eigenschaftswert}]

UML – Attribute - Sichtbarkeit

[sichtbarkeit] [/] name [: typ] [multiplizität] [=vorgabewert] [{eigenschaftswert}]

Damit wird der Zugriff und die Sichtbarkeit für andere Systemkomponenten festgelegt.

- öffentlich (**public**, Kurzzeichen +) – Jede andere Systemkomponente hat uneingeschränkten Zugriff.
- privat (**private**, Kurzzeichen -) – Nur Ausprägungen der das Attribut beherbergenden Klasse dürfen zugreifen.
- geschützt (**protected**, Kurzzeichen #) – Es dürfen nur Klassen zugreifen, die von der Urprungsklasse erben, bzw. die Klasse selbst.
- paket (**package**, Kurzzeichen ~) – Das Attribut ist für alle Klassen, die sich im selben Paket wie die definierte Klasse befinden, sicht – und zugreifbar.

UML – Attribute

[sichtbarkeit] [/] name [: typ] [multiplizität]
[=vorgabewert] [{eigenschaftswert}]

Das Zeichen / bedeutet, das es ein **abgeleitetes Attribut** ist, dessen Inhalt sich aus anderen im System vorliegenden Daten zur Laufzeit berechnen lassen. Es muss nicht abgespeichert werden.

[sichtbarkeit] [/] name [: typ] [multiplizität]
[=vorgabewert] [{eigenschaftswert}]

Attributname, der aus einer beliebigen Zeichenkette bestehen kann. Leer- und Sonderzeichen sind zugelassen, sollten aber nicht verwendet werden. Der Attributname sollte mit einem Kleinbuchstaben beginnen.

UML – Attribute

[sichtbarkeit] [/] name [: typ] [multiplizität]

[=vorgabewert] [{eigenschaftswert}]

Der **Datentyp des Attributes** kann optional angegeben werden. Im UML-Metamodell sind einige vordefinierten Datentypen vorhanden (Integer, String, Boolean). Erweiterungen sind zulässig.

[sichtbarkeit] [/] name [: typ] [multiplizität]

[=vorgabewert] [{eigenschaftswert}]

Die **Multiplizität** wird in eckigen Klammern eingeschlossen und legt die Ober- und Untergrenze der Anzahl der unter einem Attributnamen ablegbaren Ausprägungen fest.

UML – Attribute

[sichtbarkeit] [/] name [: typ] [multiplizität]

[=vorgabewert] [{eigenschaftswert}]

Mit dem **Vorgabewert** kann ein Wert festgelegt werden, der automatisch für das Attribut gesetzt wird. Es gibt keine Einschränkungen bezüglich des Datentyps.

[sichtbarkeit] [/] name [: typ] [multiplizität]

[=vorgabewert] [{eigenschaftswert}]

Der **Eigenschaftswert** wird in geschweifte Klammern eingeschlossen. Die Eigenschaftswerte können selbst definiert werden bzw. die vordefinierten Eigenschaftswerte können verwendet werden.

UML – Attribute - Beispiele

Waschmaschine
+herstellername +modellbezeichnung -seriennummer -kapazität:integer

Kopierer
+ herstellername + modellbezeichnung
+ seiteKopieren() + tonerWechseln() + seitenHeften() + papierEinlegen() # seitenZählerAktualisieren()

AttMix
att1 : int + att2:int +pi : double = 3,1415 -att3: boolean #att4 : short ~att5 : String = “Test“ {readOnly} att6 : String[0..*] {ordered} /att7

UML – Operationen

Operationen geben an, was eine Klasse tun kann bzw. was man mit der Klasse tun kann. Sie Beschreiben das Verhalten der Objekte. Die Namensgebung und Verwendung der Vorzeichen ist analog zu den Attributen.

Operationen sind die einzige Möglichkeit, Zustandänderungen eines Objektes herbeizuführen.

Operationen werden in der Regel durch Methoden implementiert.

Eine Besonderheit stellen **Klassenoperationen** dar, die die Möglichkeit bieten, Methoden direkt auf der Klasse auszuführen. Sie werden durch Unterstreichen des gesamten Objekteintrages gekennzeichnet. (z.B. Operation zur Erzeugung von Instanzen)

[sichtbarkeit] name (parameterliste) : rückgabetyp

[{eigenschaftswert}]

UML – Operationen

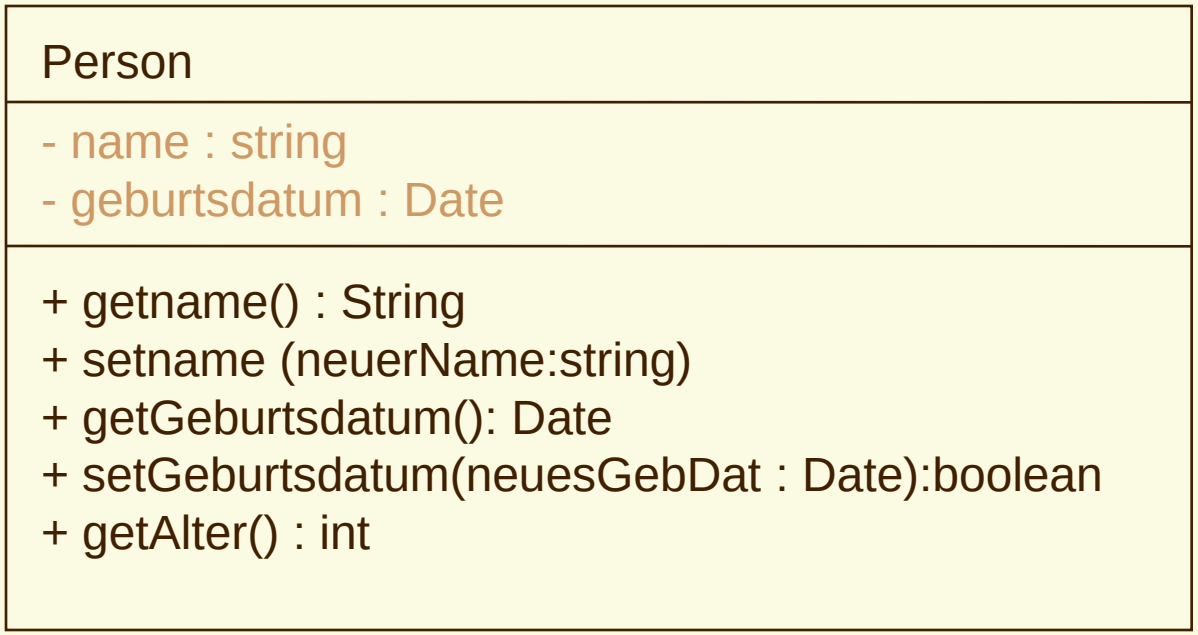
Die Operation beschreibt die Signatur einer sie implementierenden Methode.

Eine Methode ist die Implementierung (Algorithmus bzw. Rumpf) der Operation. Eine Operation darf mehrere Methoden haben (polymorphe Operationen).

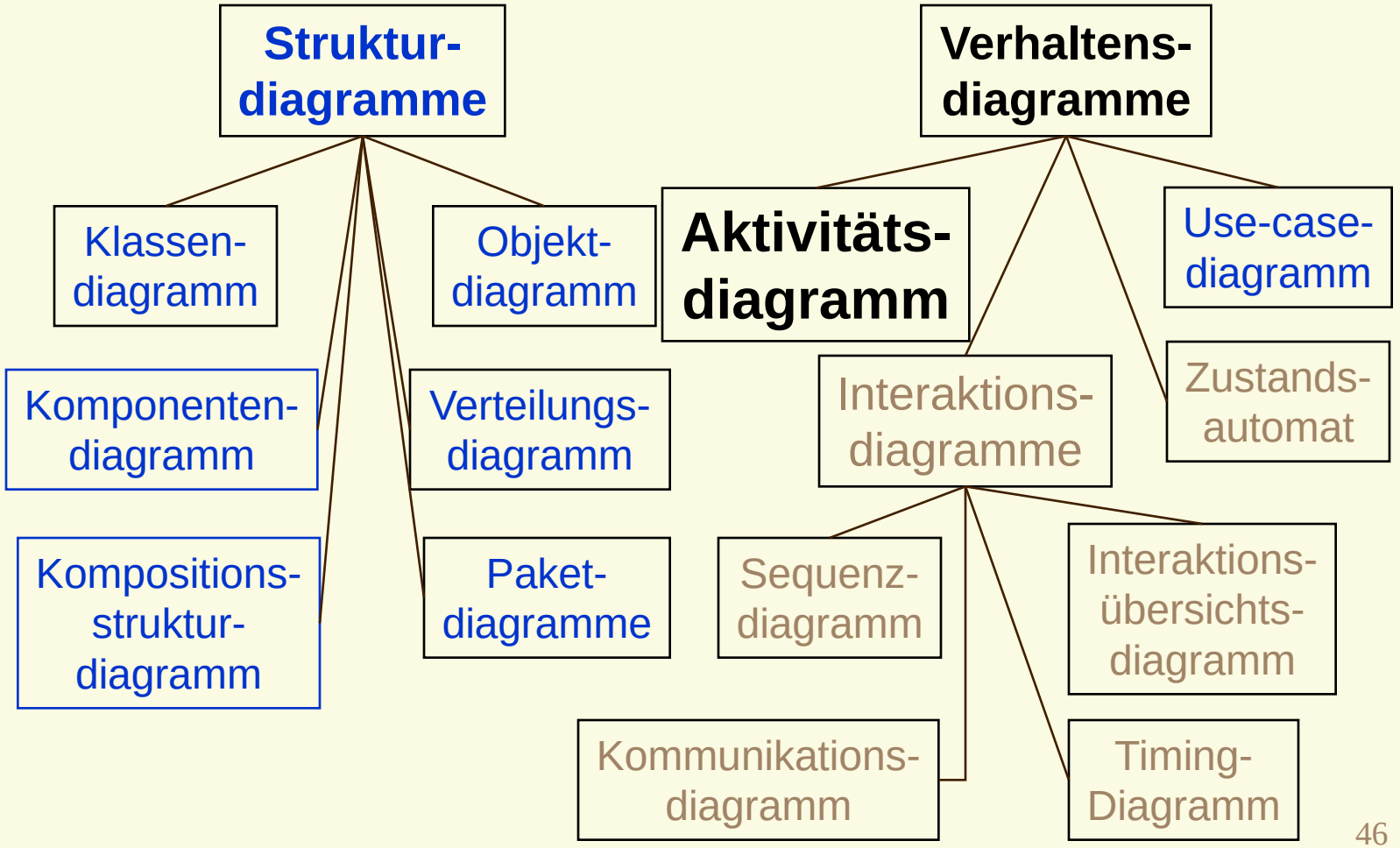
Methoden können z.B.

- die Attributwerte des Objektes verändern,
- Objekte erzeugen oder zerstören,
- Rückgabewerte definieren, die nach Beendigung der Methodenausführung dem Aufrufer übermittelt werden.

UML – Operationen - Beispiele



Diagrammtypen der UML 2



UML - Aktivitätsdiagramm

Mit Hilfe von Aktivitätsdiagrammen können Abläufe modelliert werden. Die Anwendung erstreckt sich von der Visualisierung von Anwendungsfällen oder Operationen bis hin zu komplexen Geschäftsabläufen. Nebenläufigkeiten können dargestellt werden.

„Wie realisiert ein System ein bestimmtes Verhalten?“

Eine Aktion im Sinne der UML 2 spezifiziert die Menge der potentiellen Abläufen, die in der Realität unter bestimmten Randbedingungen ablaufen.
Das Konzept der Aktivitätsdiagramme wurde in UML 2 vollständig überarbeitet und erweitert.

Aktivitätsdiagramm

Aktivitätsdiagramme dienen zur Visualisierung von Abläufen zu unterschiedlichen Planungszeitpunkten, mit unterschiedlichen Detailliertheitsgraden. Sie legen die Regeln fest, die alle möglichen Abläufe beschreiben. Aktivitätsdiagramme sind **Grafen, die aus Knoten und Kanten bestehen**. Die Kanten repräsentieren Steuer- und Objektflüsse, Die Knoten repräsentieren Aktivitäten, Aktionen, Objekte und Kontrollelemente zur Ablaufsteuerung.

Es sind verschiedene Abstraktionsniveaus möglich.

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

Tokenkonzept

Zur Veranschaulichung der Abläufe in einem Aktivitätsdiagramm wurde das **Tokenkonzept** eingeführt. Token (Marken) repräsentieren den Ablauf im Aktivitätsdiagramm. Sie werden in UML nicht visualisiert, sie dienen zu logischen Ablauferklärungen.

Token lösen einzelne Aktionen aus. Er verbleibt so lange in der Aktion bis diese beendet ist und er von einer Folgeaktion aufgenommen werden kann. Diese werden als Kontrolltoken bezeichnet.

Eine zweite Tokenart sind Datentoken. Diese besitzen neben der Steuerungsfunktion auch die Möglichkeit, Daten zu speichern, die in den durchlaufenen Knoten verändert werden können.

Tokenkonzept

Token können Verzweigungen und Vereinigungen durchlaufen. Weiterhin ist die Aufteilung eines Ablaufes in nebenläufige Zweige möglich. Die folgenden Aktionen werden parallel abgearbeitet. Dazu werden die Token an „Splitting-Knoten“ vervielfacht. Diese können dann unterschiedliche Folgeaktionen abarbeiten. Beim Zusammenführen paralleler Abläufe, werden alle Token so lange an dem Synchronisationsknoten gesammelt, bis alle Token der jeweiligen UND-Zweige angekommen sind.

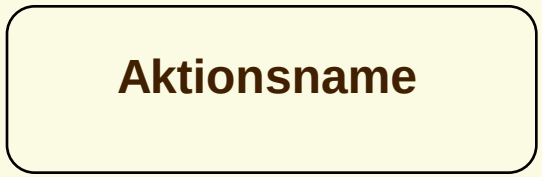
Aktion

Eine **Aktion** (Action) steht für den Aufruf eines Verhaltens oder die Bearbeitung von Daten, die innerhalb der Aktivität nicht weiter zerlegt wird. Eine Aktion ist damit ein Einzelschritt, der zur Beschreibung des Verhaltens (der Aktivität) beiträgt.

Die **Summe aller Aktionen realisiert die Aktivität**.

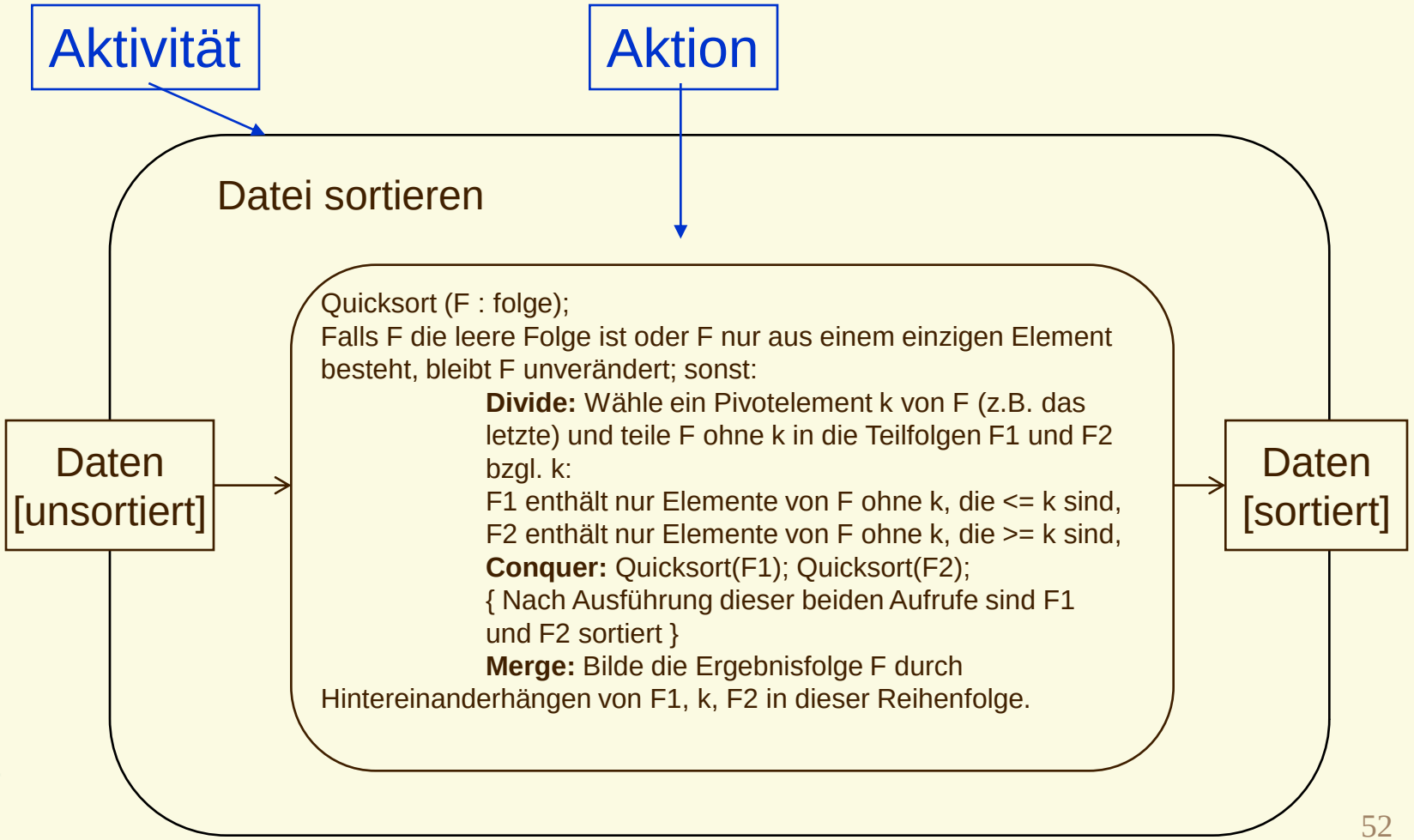
Eine **Aktion ist nicht parametrisiert**.

Die Visualisierung erfolgt über ein Rechteck mit abgerundeten Ecken.

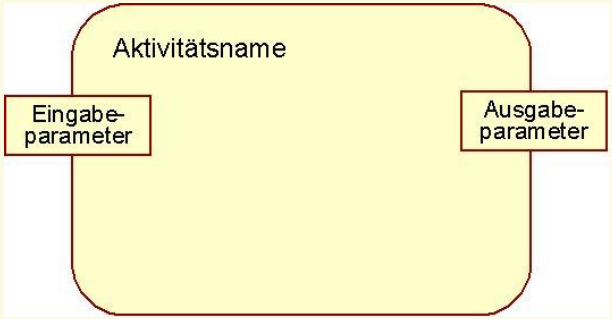


„Eine Aktion ist ein fundamentales Element der Berechnung, das Eingabedaten in Ausgabedaten umwandelt.“ /FORBRIG07/

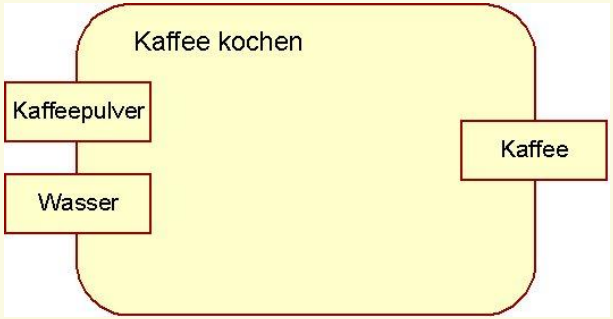
Aktivitäten und Aktionen



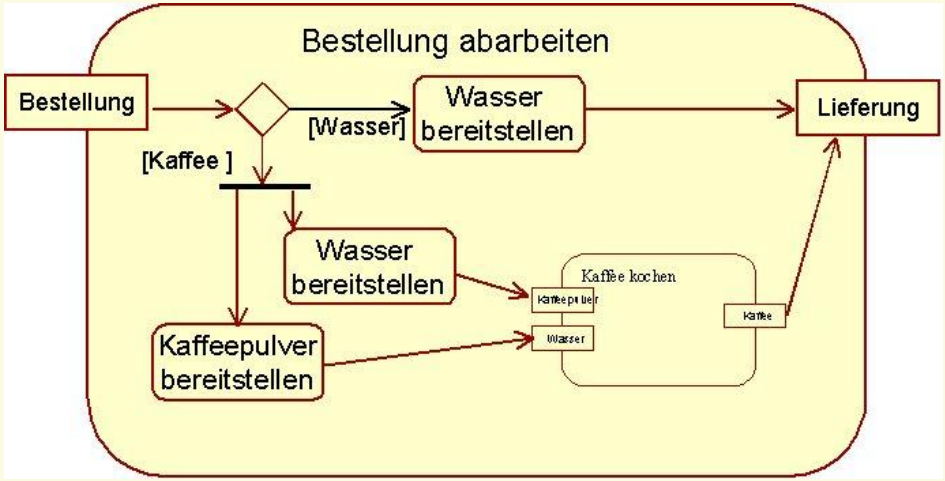
Aktivitäten



Notationsform für Aktivitäten



Beispiel: Aktivität Kaffee kochen



Beispiel: Aktivität Bestellung abarbeiten

/Quelle: Peter Forbrig, Objektorientierte Softwareentwicklung mit UML, Hanser-Verlag, 2007/

Aktivitäten

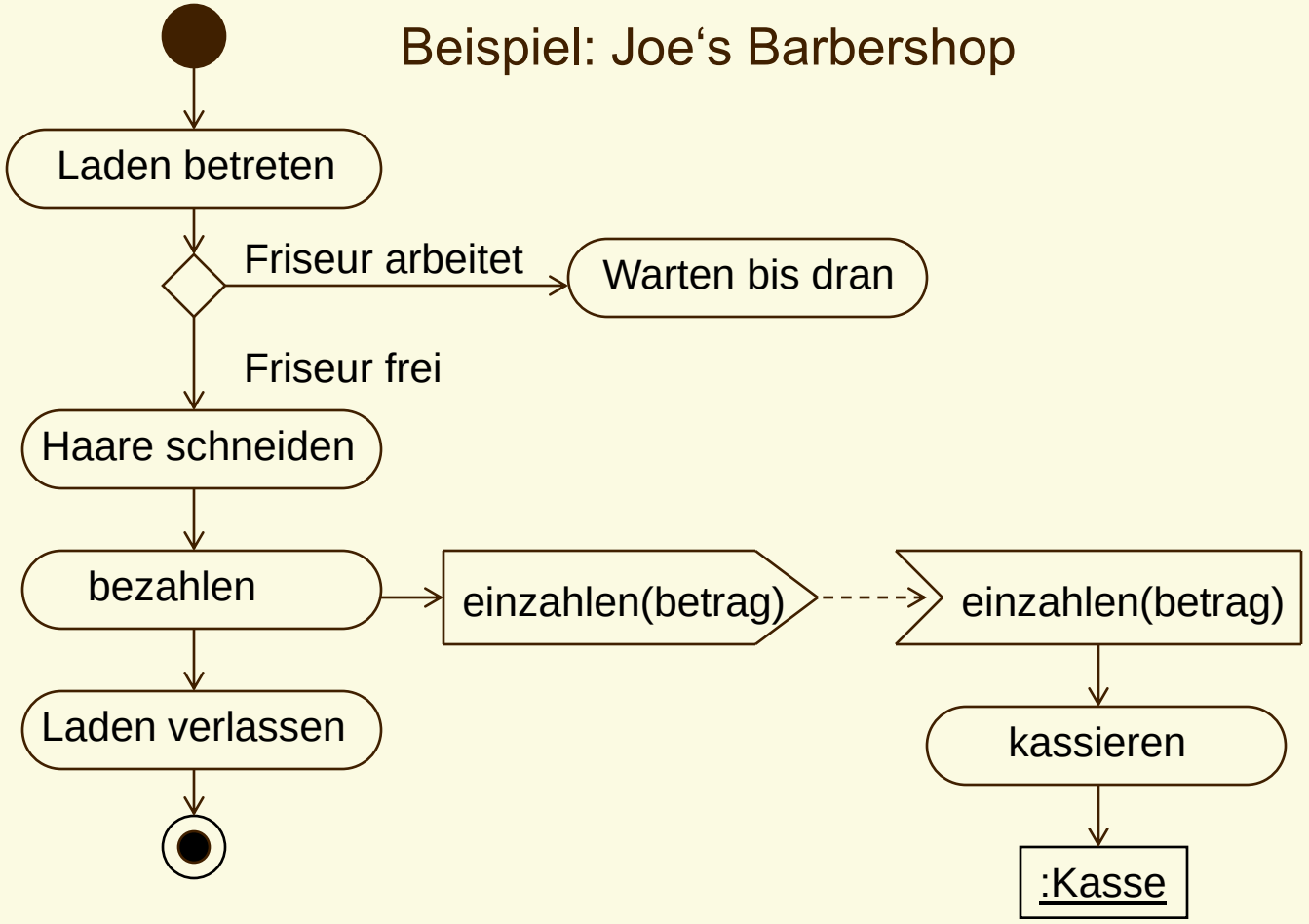
Aktivitäten können synchron und asynchron ablaufen.

Bei einem **synchronen Aufruf** wird der Ablauf so lange an der rufenden Aktion blockiert, bis die geschachtelte Aktivität komplett beendet ist. Die gerufene Aktivität dar einen Parameter zurückliefern.

Wir eine Aktivität **asynchron aufgerufen**, so ist die rufende Aktion beendet, sobald die Aktivität gestartet ist. Die Abläufe werden dabei parallelisiert.

Es ist keine Visualisierung für die Unterscheidung synchroner und asynchroner Aktivitäten vorgesehen. Es sollte eine entsprechende Notiz angegeben werden.

UML – Aktivitätsdiagramme



Schlussfolgerungen

Die Informatikausbildung in der Schule ist kein UML-Zertifizierungskurs, aber die UML-Notation ist geeignet, eine formalisierte Beschreibung von Modellen zur objektorientierten Modellierung zu unterstützen.

Die Verwendung von UML ermöglicht, moderne Entwicklungssysteme mit UML Schnittstellen im Unterricht einzusetzen.