Software-Engineering

Sebastian Iwanowski
FH Wedel

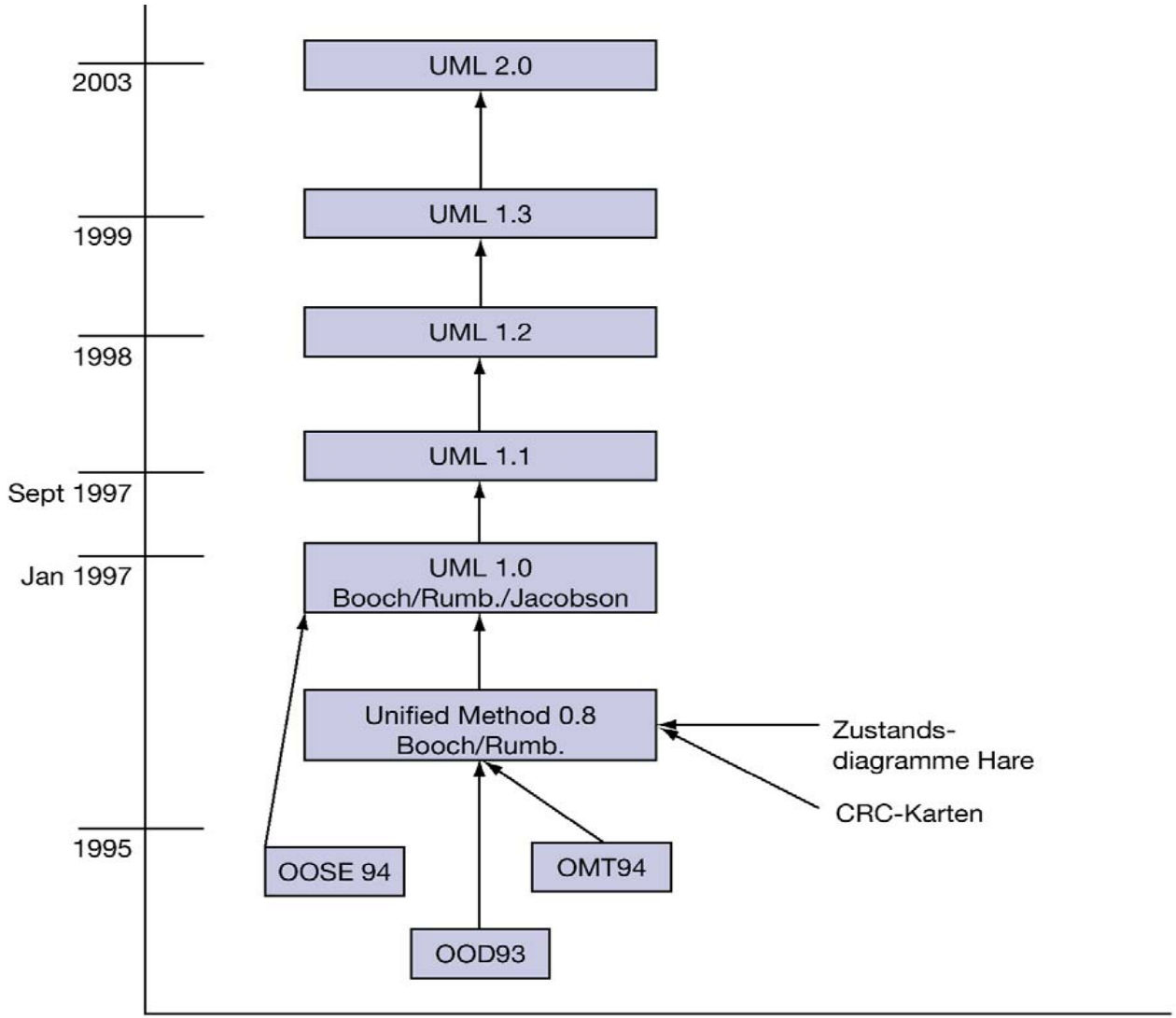
Kapitel 4: Systemanalyse
Teil 3: Der Systemanalysestandard UML

UML: Was ist das ?

UML = Unified Modelling Language

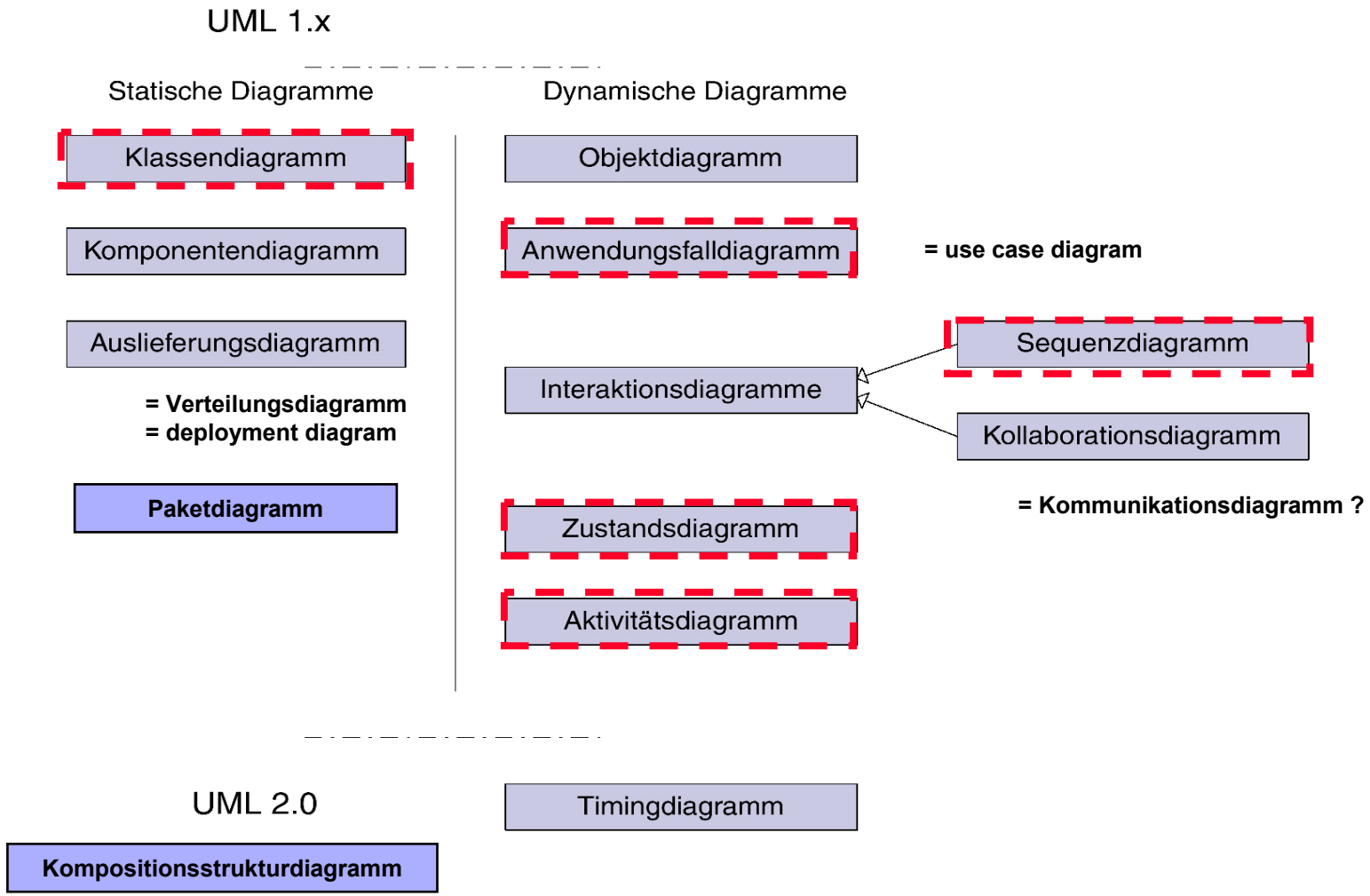
- ist ein Standard, kein Case-Tool (CASE = Computer Aided Software Engineering)
- entworfen von James Rumbaugh, Ivar Jacobsen, Grady Booch
- führt Konzepte zusammen, die ursprünglich unabhängig voneinander entwickelt worden sind
- wird durch das CASE-Tool Rational Rose unterstützt
- Konkurrenzprodukte: Together, etc.
- Kern ist die Beschreibung von Systemen (real und Software) mit objektorientierter Terminologie
- bietet zusätzlich Beschreibungsmöglichkeiten für Prozesse
- Anspruch: Software wird nach Beschreibung in UML automatisch erstellt

UML: Historie



aus: Wolfgang Zuser / Thomas Grechenik / Monika Köhle: *Software Engineering mit UML und dem Unified Process, Kap. 7*, Pearson Studium 2004

UML: Bestandteile



nach: Wolfgang Zuser / Thomas Grechenik / Monika Köhle: *Software Engineering mit UML und dem Unified Process, Kap. 7*, Pearson Studium 2004

UML: Klassendiagramme

Anmerkungen zu Beziehungen:

- Die allgemeine Beziehungsklasse “Assoziationen” dient zur Beschreibung nicht näher spezifizierte Beziehungen. Eine ungerichtete Assoziation zwischen zwei Klassen beschreibt die Beteiligung beider Klassen an einer Beziehung im Sinne einer ER-Assoziation.

- Die spezielle Beziehung “Aggregation” beschreibt Teilobjekte: Es ist sinnvoll, dass solche Teilobjekte die Werte von Attributen des zusammengesetzten Objektes sind.

- Die spezielle Assoziation “Komposition” bedeutet: Ein Wegfall des zusammengesetzten Objekts hat den Wegfall des Teilobjekts zur Folge.

- Die spezielle Assoziation “Generalisierung” hat die Vererbung der Attribute und Methoden auf die spezialisierten Klassen zur Folge.

UML: Klassendiagramme

Unterschied in UML-Klassendiagrammen zwischen Systemanalyse und Softwareentwurf:

- Richtungspfeile bei allgemeinen Assoziationen sind erst im Softwar*eentwurf* sinnvoll: Sie entsprechen der Erreichbarkeit eines implementierten Objekts von dem anderen Objekt aus.
- Eine gerichtete Assoziation von einer Klasse A nach Klasse B wird genau dann verwendet, wenn ein Attribut oder eine Methode von A die Kenntnis der Klasse B benötigt.

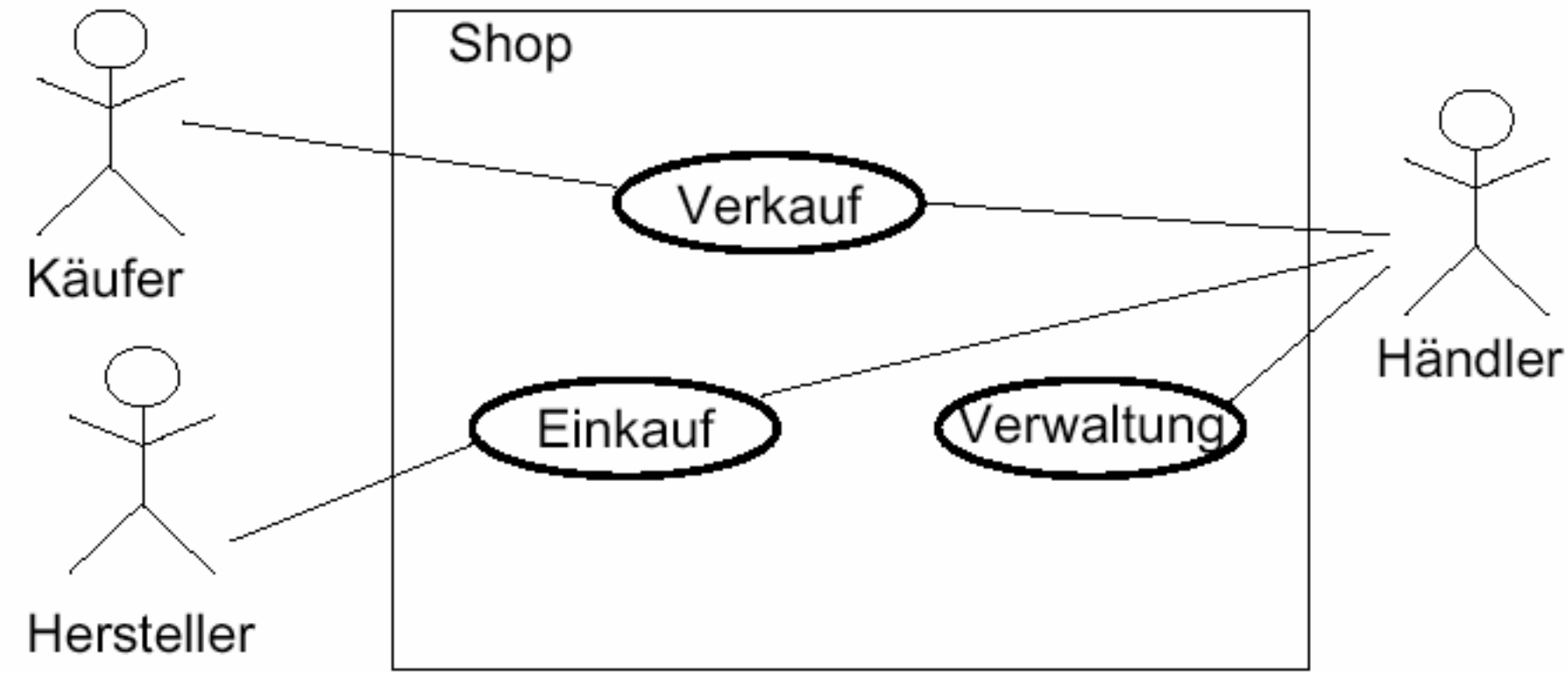
Zusammenfassung:

- In der Systemanalyse sollten grundsätzlich ungerichtete Assoziationen verwendet werden.
- Im Softwareentwurf sollten grundsätzlich gerichtete Assoziationen (uni- oder bidirektional) verwendet werden.
- Bei den Beziehungsklassen Aggregation, Komposition und Generalisierung gibt es keine Unterschiede zwischen Systemanalyse und Softwareentwurf.

UML: Use-Case-Diagramme

zur Beschreibung und Spezifikation von Systemen

1. Beispiel: Shop



UML: Use-Case-Diagramme

Erklärung der Symbole:

Aktor (Strichmännchen):

- Jemand der mit dem System interagiert (d. h. außerhalb des Systems): muss keine natürliche Person sein

Use case (Ellipse):

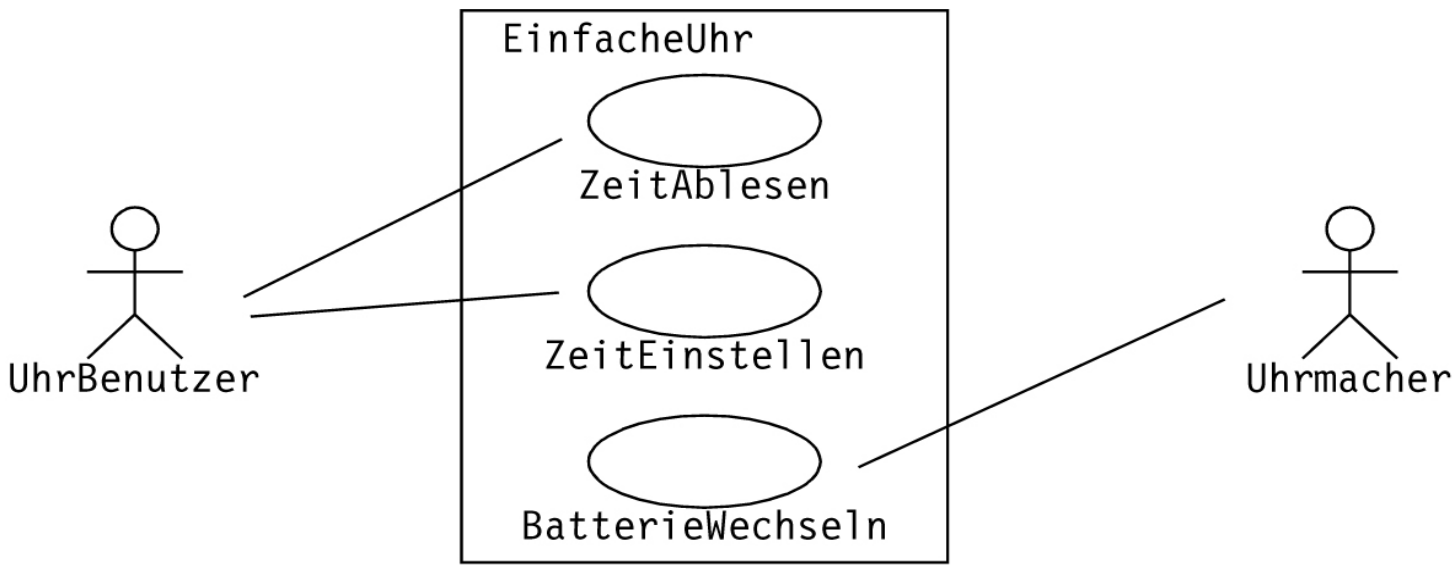
- „Anwendungsfall“: Nach außen sichtbare Teilfunktionalität des Systems
- Kann in nachfolgenden Beschreibungen hierarchisch verfeinert werden
- Zwischen zwei Aktoren muss mindestens ein use case sein !

System (Rechteck):

- Zusammenfassung von use cases
- Unterscheidung von „innen“ und „außen“

UML: Use-Case-Diagramme

2. Beispiel: Einfache Uhr



aus: Bernd Brügge / Allen H. Dutoit: *Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java*, *Kap. 2,* Pearson Studium 2004

UML: Use-Case-Diagramme

Ziele einer Use-Case-Beschreibung:

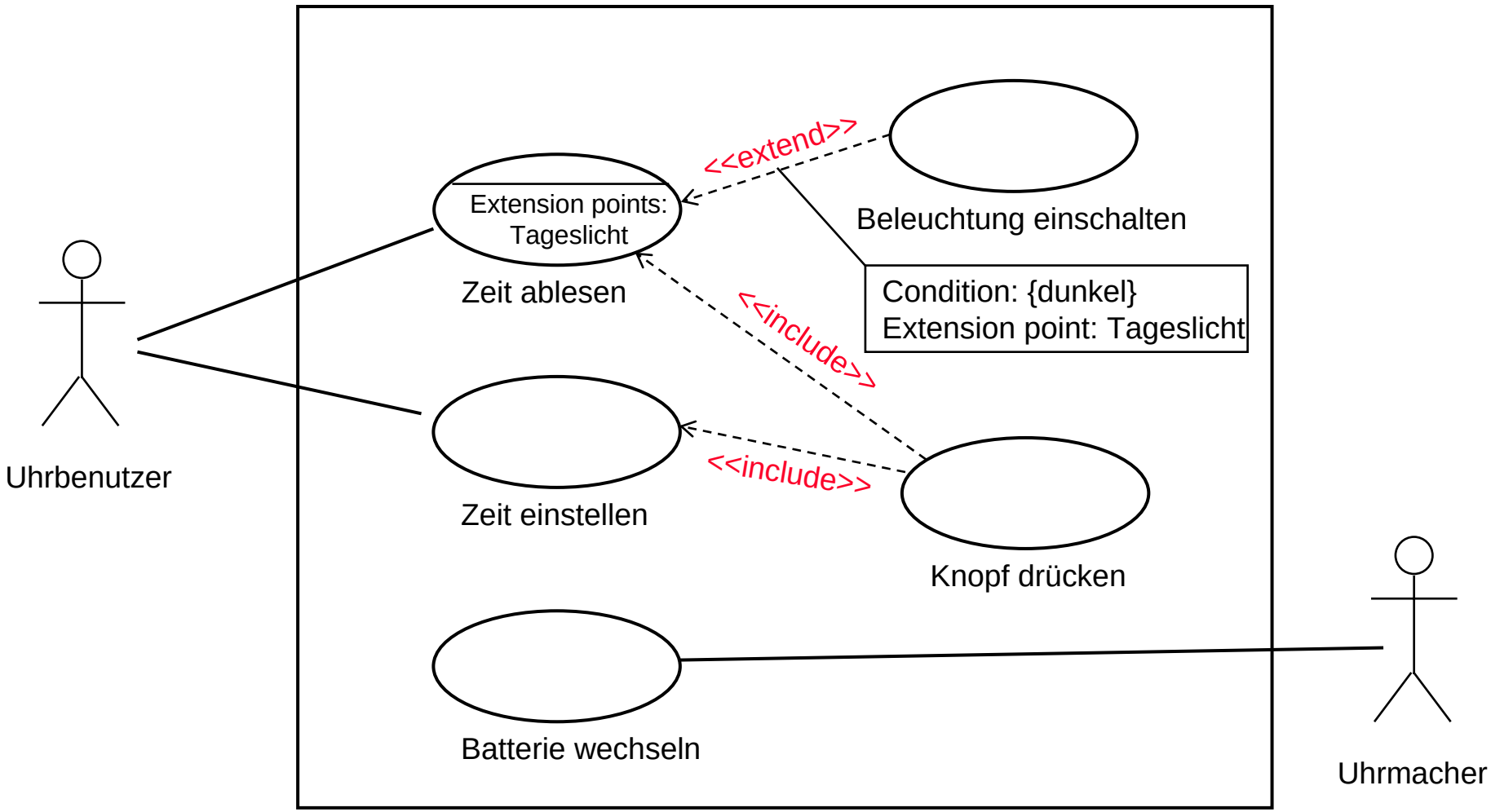
- Zur hierarchischen Beschreibung eines Systems
- Grobe Beschreibung einzelner Transaktionen

Weitere Beschreibungsmöglichkeiten für Beziehungen zwischen use cases:

- “extends”: Beschreibung einer Teilfunktionalität, die aber nicht immer ausgeführt wird (verzweigt sich bei so genannten Extension points des ursprünglichen use cases)
- “includes”: Teilprozess, der von mehreren use cases benutzt werden kann

UML: Use-Case-Diagramme

2. Beispiel (erweitert): Einfache Uhr



UML: Sequenzdiagramme

zur Beschreibung des Ablaufs von Transaktionen und Interaktionen

Transaktion:

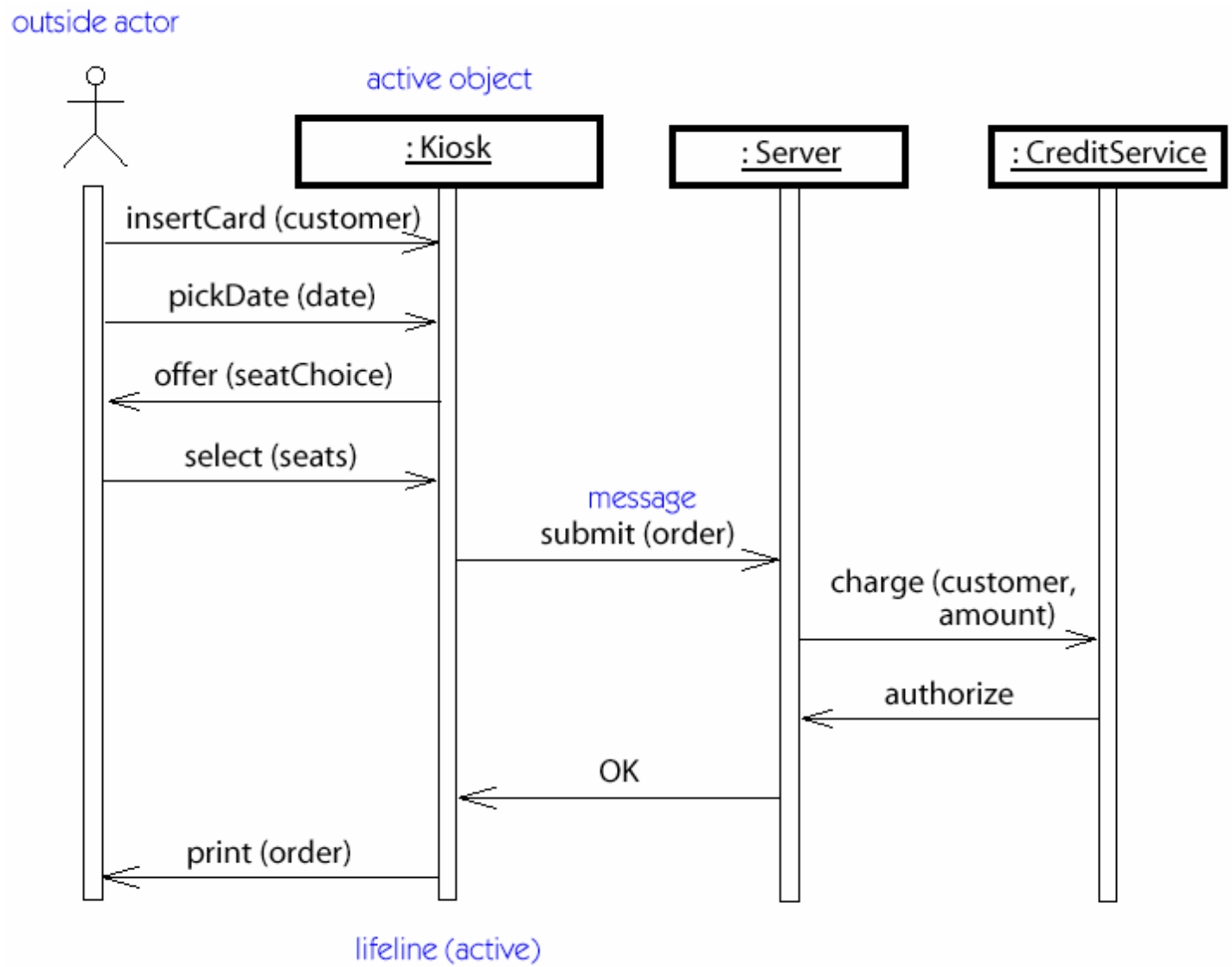
- Menge von Nachrichten, die zwischen einer Gruppe von Objekten ausgetauscht werden, um eine bestimmte Operation oder ein bestimmtes Ergebnis zu veranlassen.
- In vielen Anwendungsfällen müssen Transaktionen unteilbar sein und weitere Anforderungen erfüllen (z.B. im Anwendungsfall Datenbanken)

Interaktion:

- Der Austausch einer einzelnen Nachricht

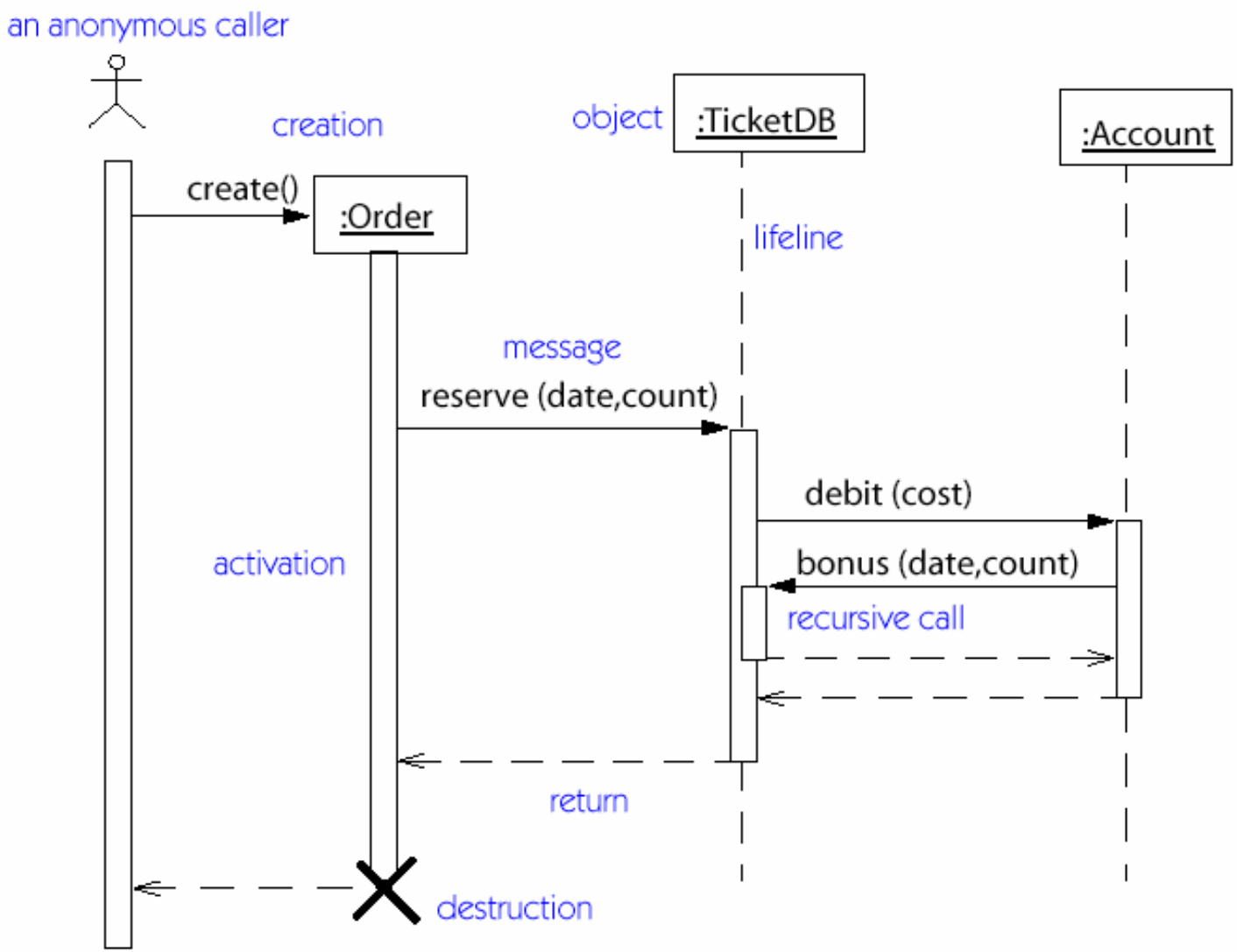
UML: Sequenzdiagramme

1. Beispiel: Veranstaltungsbuchung am Kiosk per Kreditkarte



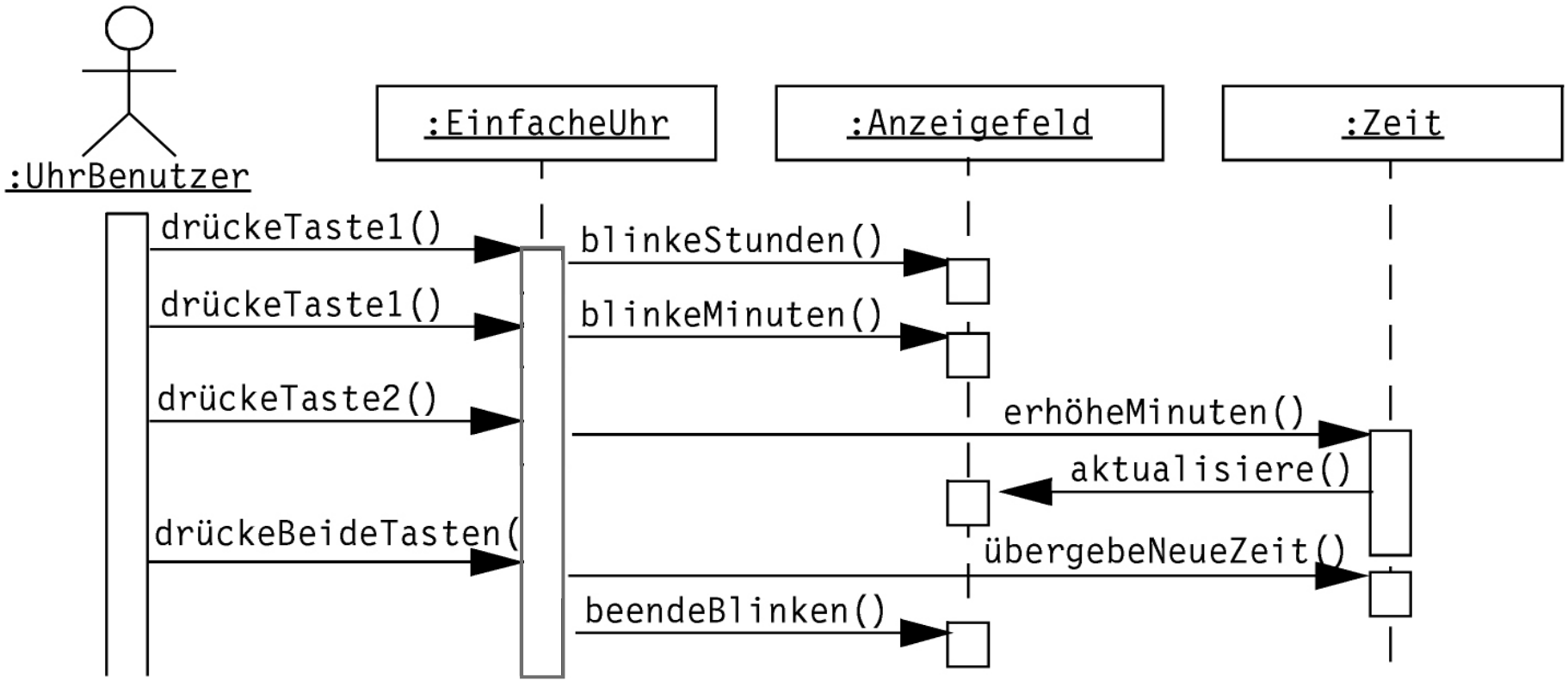
UML: Sequenzdiagramme

2. Beispiel: Fahrkartenbuchung für registrierte Kunden



UML: Sequenzdiagramme

3. Beispiel: Einfache Uhr



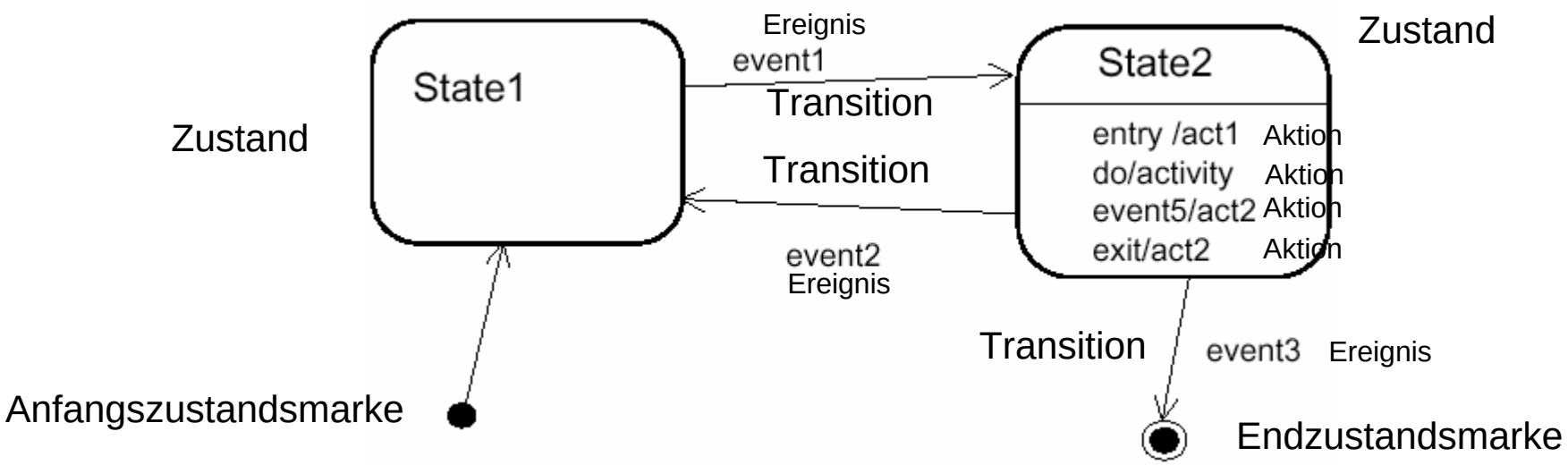
aus: Bernd Brügge / Allen H. Dutoit: *Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java*, *Kap. 2,* Pearson Studium 2004

UML: Zustandsübergangsdiagramme (Statecharts)

zur Beschreibung von dynamischen Objekten des Systems

Zustandsübergangsdiagramme beschreiben:

- die Lebenshistorie von Objekten einer Klasse
- die Ereignisse, die einen Übergang (Transition) von einem Zustand zu einem anderen veranlassen
- die Aktionen, die sich aus diesem Zustandswechsel ergeben



UML: Zustandsübergangsdiagramme (Statecharts)

Klassifizierung der Aktionen:

Generell wird eine Aktion bezeichnet durch: **`Ereignis / Aktion`**

entry-Aktionen

- werden immer ausgeführt beim Betreten des Zustandes

exit-Aktionen

- werden immer ausgeführt beim Verlassen des Zustandes

interne Aktion

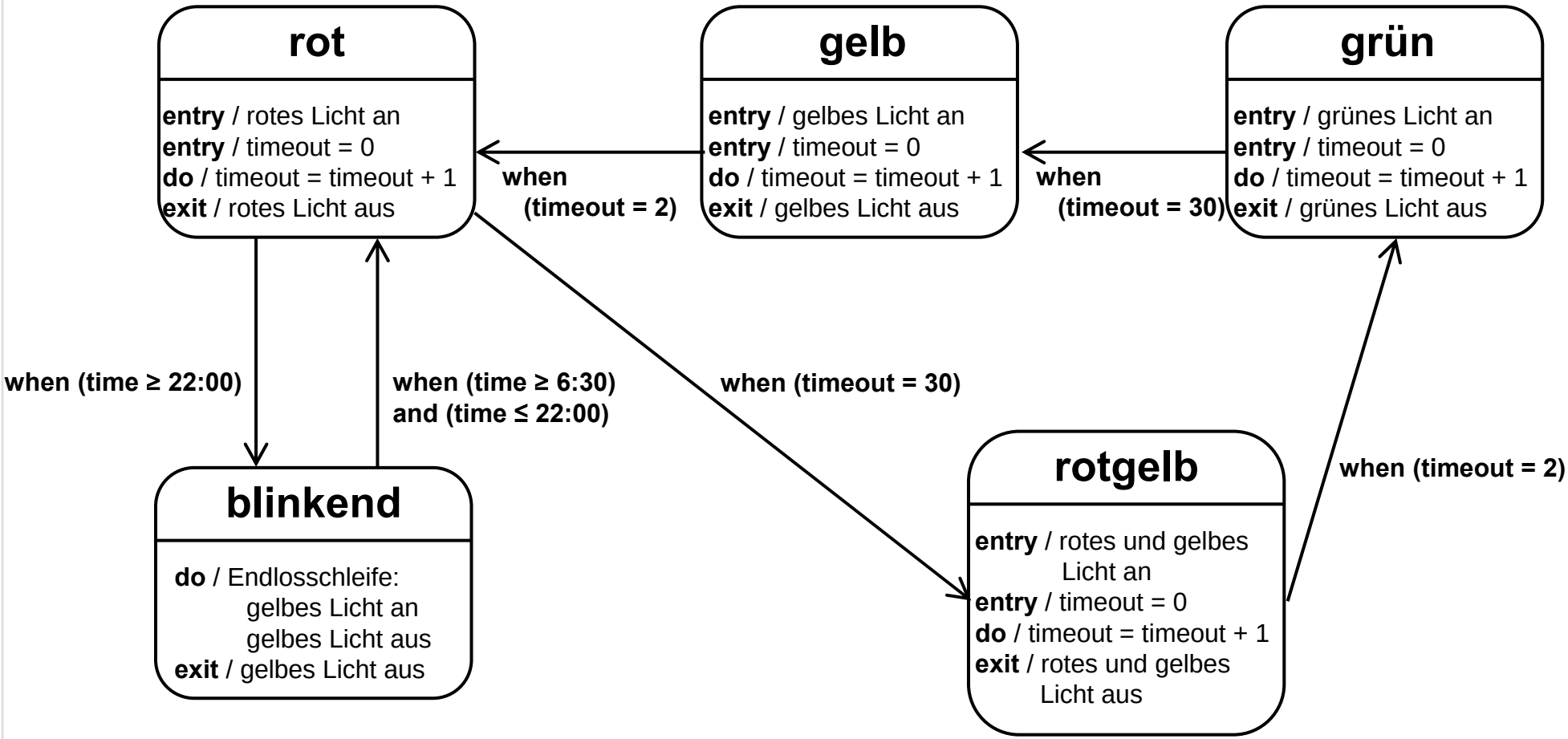
- Aktion, die durch irgendein anderes Ereignis ausgelöst wird, aber nicht den Zustand verlässt.

do-Aktivität

- (längere) Aktion, die ausgeführt wird, während der Zustand andauert
- wird entweder beendet durch vorgesehene Beendigung der Aktion oder durch Auftreten eines Ereignisses, das einen Zustandsübergang verursacht
- entspricht einer internen Aktion, die unmittelbar ausgeführt wird (also ohne spezielles Ereignis)

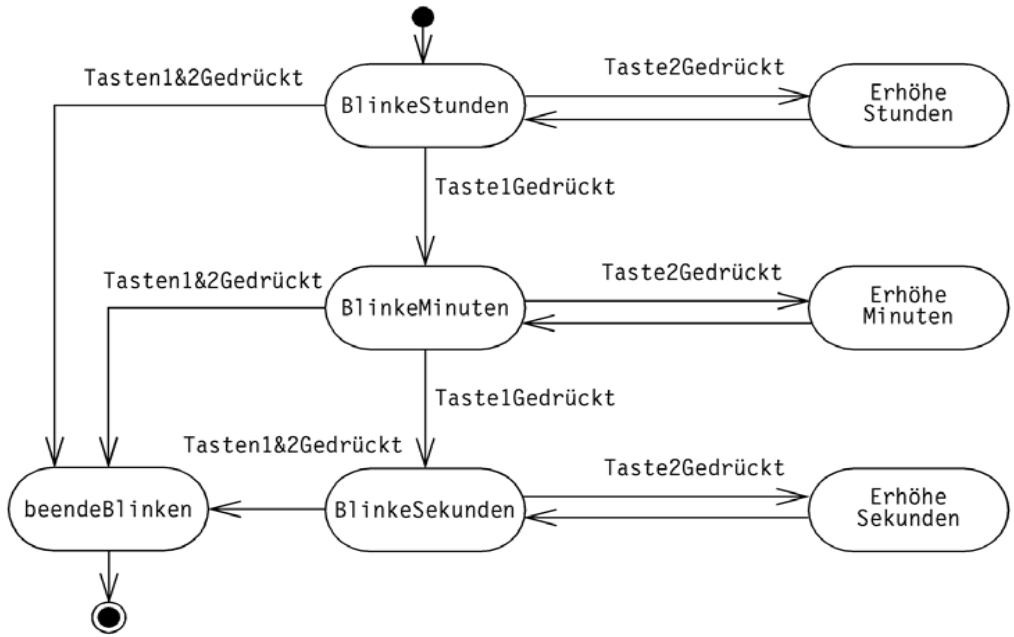
UML: Zustandsübergangsdiagramme (Statecharts)

1. Beispiel: Verkehrsampel



UML: Zustandsübergangsdiagramme (Statecharts)

2. Beispiel: Stellen einer Uhr



Korrigieren Sie die Ungenauigkeiten!

aus: Bernd Brügge / Allen H. Dutoit: *Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java*, *Kap. 2,* Pearson Studium 2004

UML: Aktivitätsdiagramme (Version UML2)

Aktivitätsdiagramme als Alternative zu Zustandsdiagrammen:

- Transitionen werden grundsätzlich durch die Beendigung von Aktionen ausgelöst
- spezielle Notation für parallele Abläufe
- wesentlich mächtigerer Beschreibungsapparat

Anleihen von Petri-Netzen:

- Tokenkonzept
  (potentiell unendlich viele, Tokens werden im Diagramm aber nicht dargestellt)
- Maximalgewichte für Kanten und Maximalkapazitäten für Objekte
- semantische Bedeutung von parallelen Abläufen

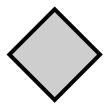
UML: Aktivitätsdiagramme (Version UML2)

Elemente von Aktivitätsdiagrammen:

- **Aktionsknoten:** kleinste ausführbare Funktionseinheit

- **Objektknoten:** Datenbehälter (zum Weitergeben)

- **Kontrollknoten**: für Entscheidung und Zusammenführung im Kontrollfluss



- **Synchronisationsbalken**: für nichtdeterministische Verzweigung und Synchronisation

- **Aktivitätsbereiche:** Zusammenfassung von Teildiagrammen zu höheren Einheiten

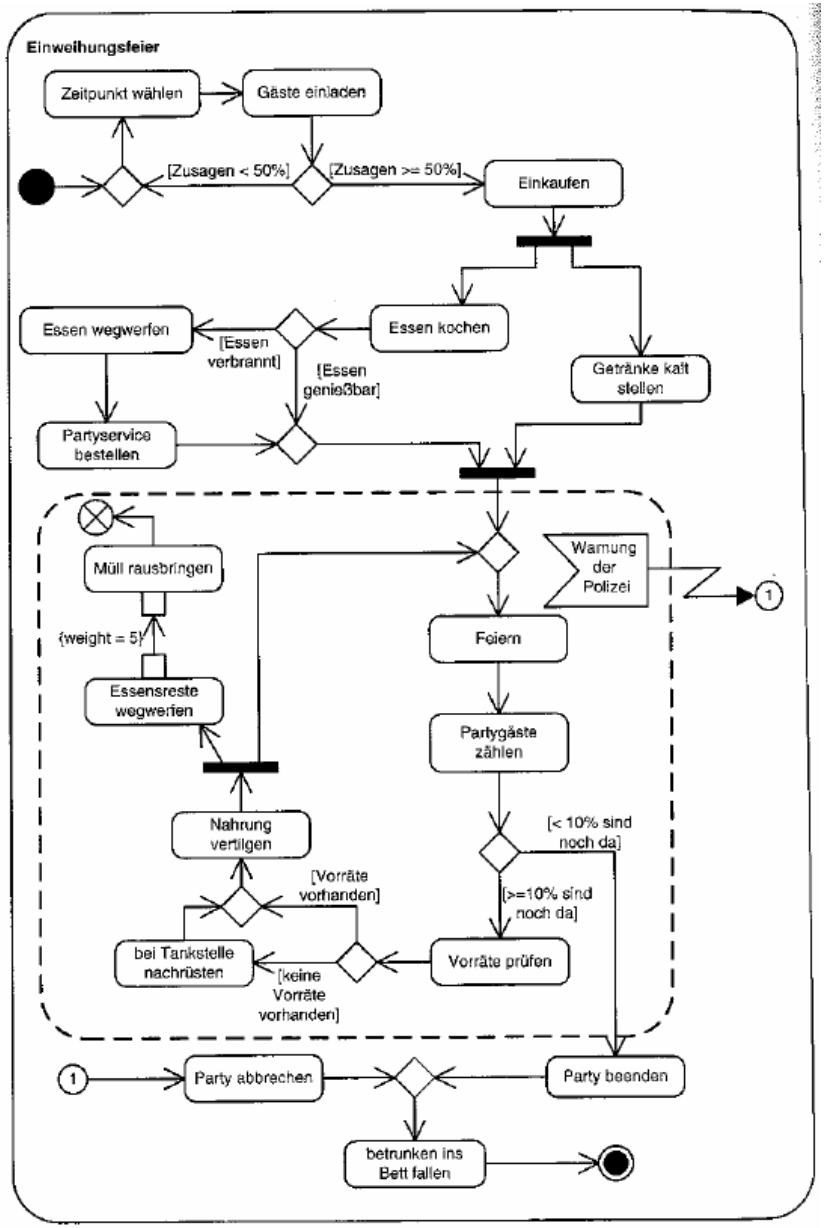
- **Startzustand:** Einsetzpunkt der Aktivität (muss nicht eindeutig sein)

- **Ablaufende:** Ende eines Kontrollflusses (andere können noch aktiv sein)

- **Endzustand:** Ende der gesamten Aktivität

UML: Aktivitätsdiagramme (Version UML2)

Beispiel: Ausgiebige Party



aus: Mario Jeckle et al.: *UML2 glasklar*, Hanser 2004

Zusammenfassung: UML

Strukturdiagramme:

- **Klassendiagramme** mit Beziehungen und Hierarchien:
  objektorientierte Systembeschreibung
- *Objektdiagramme:*
  Ausschnitte der Klassendiagramme mit bestimmten Zuständen
- *Paketdiagramme:*
  Zusammenfassung größerer Einheiten nach logischen Gesichtspunkten
- *Kompositionsstrukturdiagramme:*
  Hierarchische Zusammensetzung von Strukturen
- *Komponentendiagramm:*
  Modulare Zusammensetzung über die Schnittstellen
- *Verteilungsdiagramm: Verteilung der Systemkomponenten zur Laufzeit*

Klassifizierung nach Mario Jeckle

Zusammenfassung: UML

Verhaltensdiagramme:

- **Use Cases**:
  Beschreibung von Anwendungsszenarien (Außensicht auf das System)
- **Sequenzdiagramme**: Beschreibung von Prozessabfolgen
- **Zustandsübergangsdiagramme**: Beschreibung von Zustandsabfolgen
- **Aktivitätsdiagramme**: spezielle Zustands-Aktivitäts-Beschreibungen
- *Kooperationsdiagramme:*
  Prozessorientierter Zusammenhang der verschiedenen Komponenten
- *Timingdiagramm: Exakte Modellierung der Abfolge der Zustände*
- *Interaktionsdiagramm:*
  Generalisierung von Interaktionen (Sequenz- und Kommunikationsdiagramm)

Klassifizierung nach Mario Jeckle

Literaturempfehlungen für UML

Heide Balzert, *UML 2 kompakt*, Spektrum 2005, ISBN 3-8274-1389-3

Bernd Brügge / **Allen H. Dutoit:** *Objektorientierte Softwaretechnik mit UML, Entwurfsmustern und Java*, Pearson Studium 2004, ISBN 3-8273-7082-5

Mario Jeckle / Chris Rupp / Jürgen Hahn / Barabara Zengler / Stefan Queins: *UML2 glasklar*, Hanser 2004, ISBN: 3-446-22575-7
Vortrag zum Buch unter: http://www.jeckle.de/files/giArch.pdf

Jochen Seemann / **Jürgen Wolff von Gudenberg:** *Software-Entwurf mit UML*, Springer-Verlag 2006 (2. Auflage), ISBN 3-540-30949-7

Wolfgang Zuser / **Thomas Grechenik** / **Monika Köhle:** *Software Engineering mit UML und dem Unified Process,* Pearson Studium 2004, ISBN 3-8273-7090-6