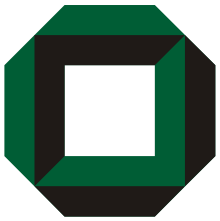


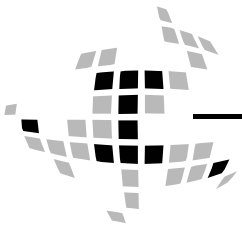
Universität Karlsruhe (TH)

Forschungsuniversität · gegründet 1825

Softwaretechnik 1 Tutorium

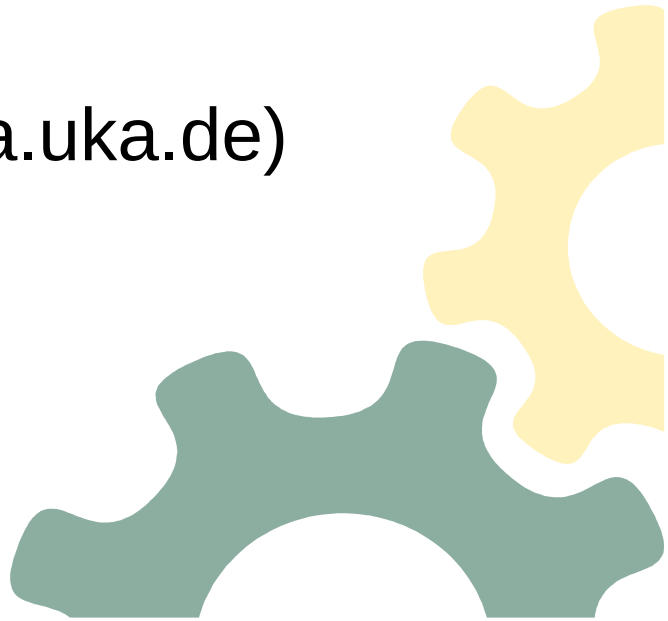
25. Mai 2009

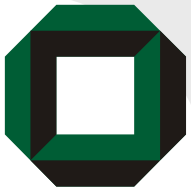
Matthias Thoma (s_thoma@ira.uka.de)



Fakultät für **Informatik**

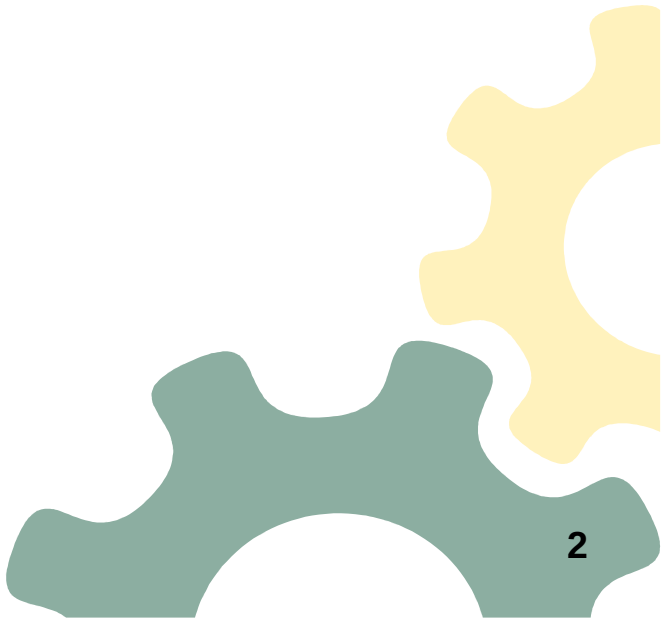
Lehrstuhl für Programmiersysteme

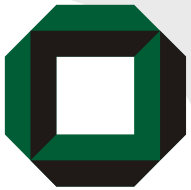




Heute

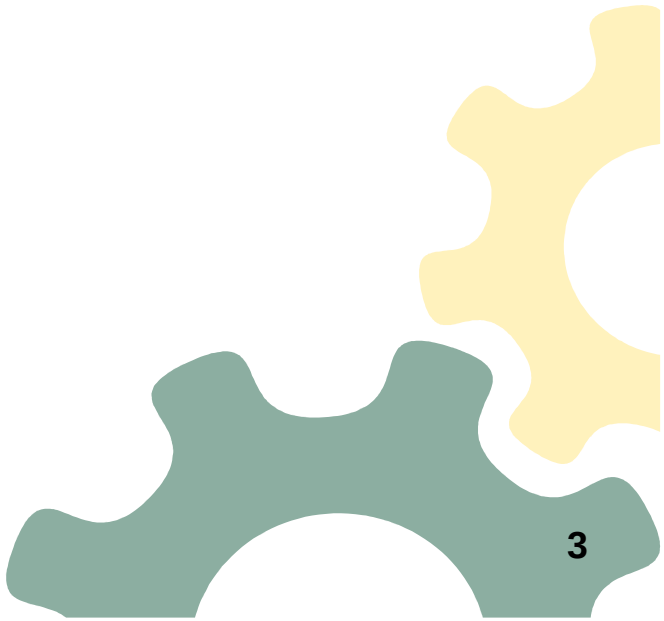
- Übungsblatt Nr. 2
- UML Klassendiagramme
- UML Sequenzendiagramm
- UML Zustandsdiagramme
- UML Aktivätsdiagramme
- Die Benutzt-Relation

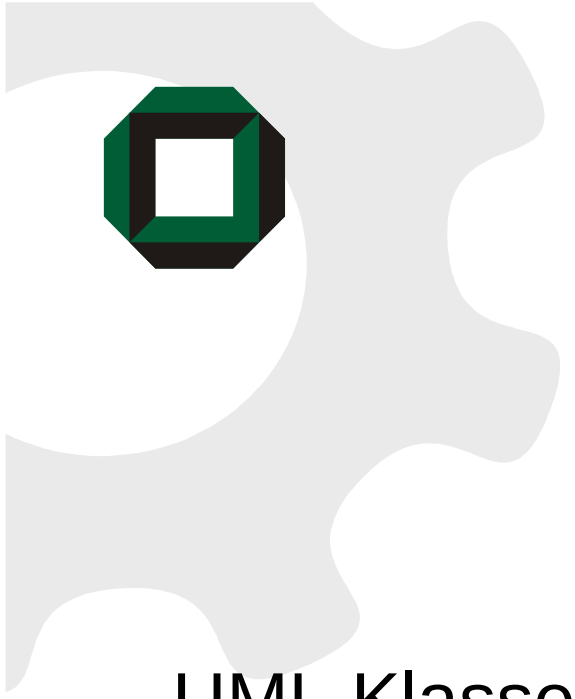




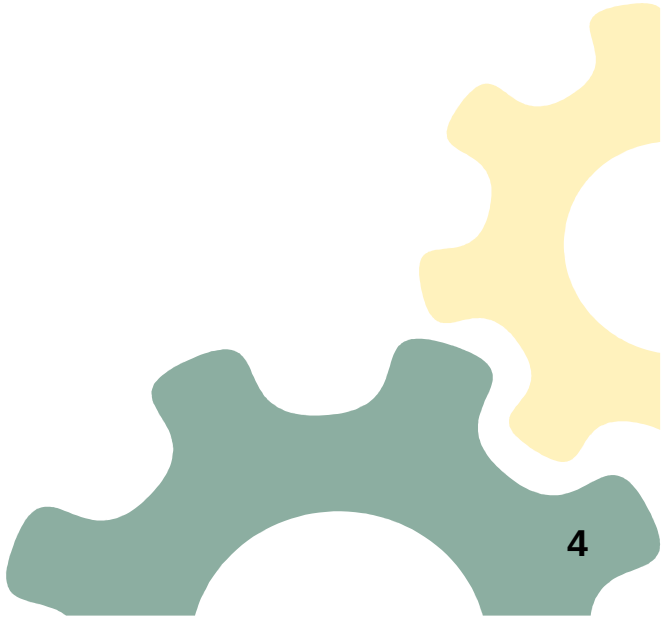
Kommentare zum letzten Übungsblatt

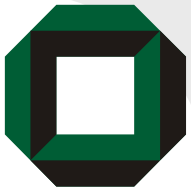
- Die Aufgabentexte genau studieren und einhalten!
- Sinnvoll Argumentieren
- Im Modell bleiben





UML Klassendiagramm

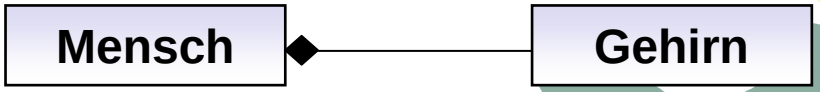




Die Komposition

- Die Komposition bezeichnet eine Teil-ganzes Beziehung.
- Bei einer Komposition hängt die Existenz des Teiles vom Ganzen ab.
- Frage: Wenn ich das Ganze lösche, kann ich dann den Teil mitlöschen?
- Immer 1:x Beziehung!

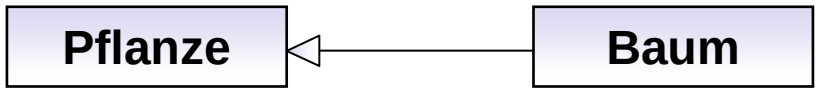
- **Komposition**
  (Aggregation mit einer **zwingenden** Teil-Ganzes-Beziehung)



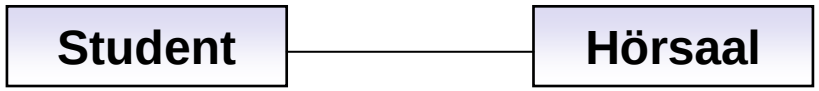
Arten der Beziehungen

In der UML existieren vier Arten von Beziehungen zwischen Klassen:

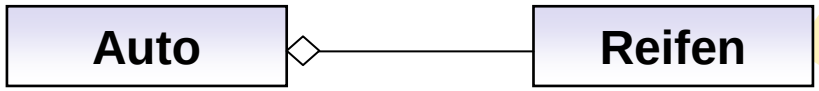
- **Vererbung**
  (Weitergabe von Eigenschaften und Methoden)



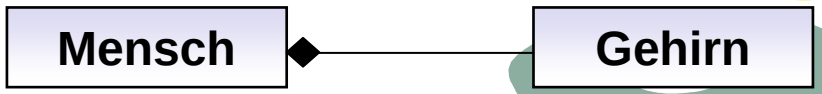
- **Assoziation**
  (Verbindung zwischen zwei gleichwertigen Klassen)



- **Aggregation**
  (Assoziation mit einer Teil-Ganzes-Beziehung)



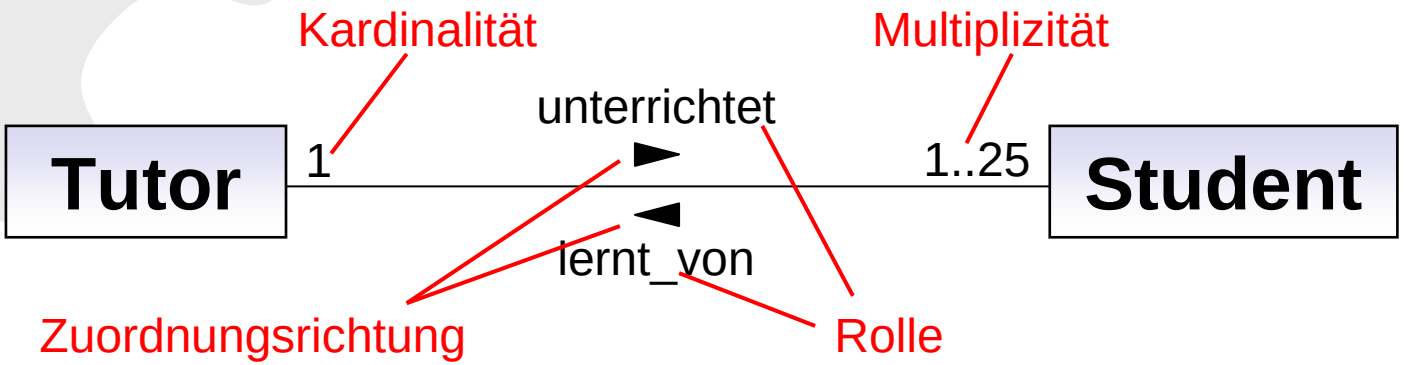
- **Komposition**
  (Aggregation mit einer **zwingenden** Teil-Ganzes-Beziehung)



Folie entnommen: Ingenieurinformatik I, SS 2007, Prof. Dr.-Ing. D. Hartmann, Lehrstuhl für Ingenieurinformatik im Bauwesen, Ruhr-Universität Bochum

Rollen, Kardinalitäten, Multiplizitäten

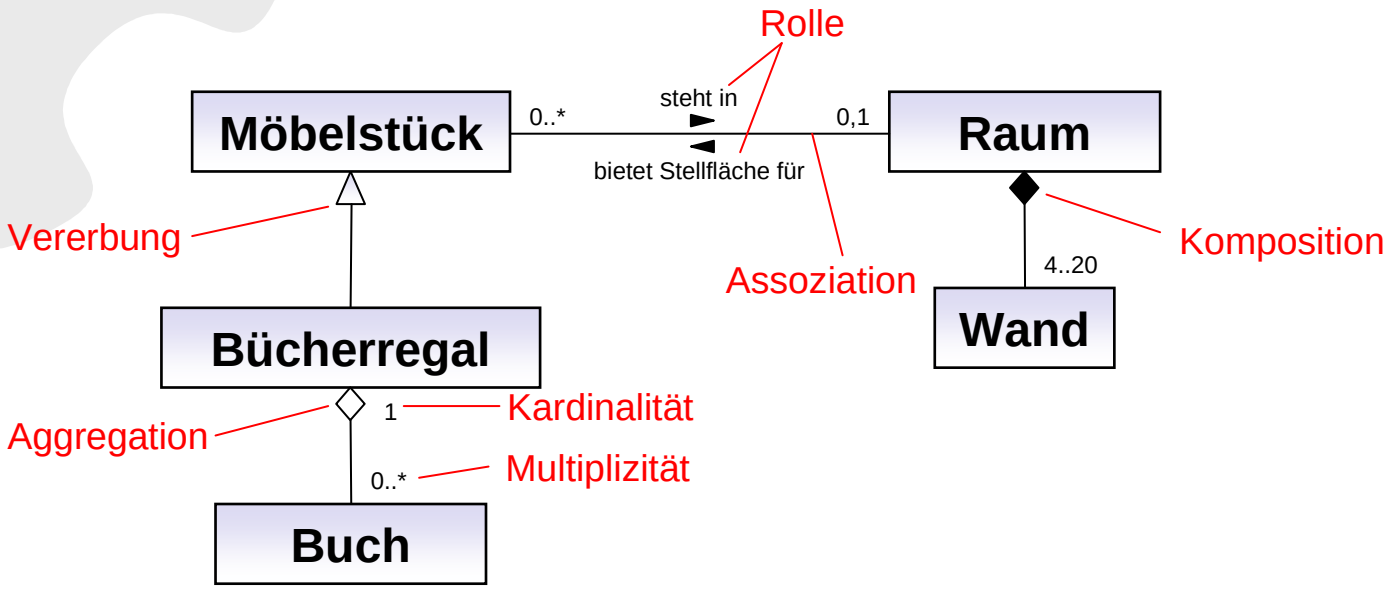
Die drei Beziehungsformen Assoziation, Aggregation und Komposition können im UML-Diagramm durch weitere Informationen spezifiziert werden:



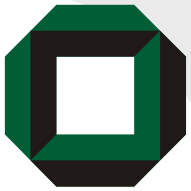
- **Zuordnungsrichtung**: Pfeile legen fest, ob es sich um eine bidirektionale (zwei Pfeile) oder unidirektionale (ein Pfeil) Zuordnung handelt
- **Rolle**: spezifiziert die Art der Beziehung zwischen beiden Klassen
- **Multiplizität**: die Anzahl der Objekte, mit denen eine Assoziation besteht (Beispiel: 0..4, 5..18, 2..*)
- **Kardinalität**: Multiplizität mit fester Anzahl (Beispiel: 1, 5, 23)

Folie entnommen: Ingenieurinformatik I, SS 2007, Prof. Dr.-Ing. D. Hartmann, Lehrstuhl für Ingenieurinformatik im Bauwesen, Ruhr-Universität Bochum

UML Klassendiagramm

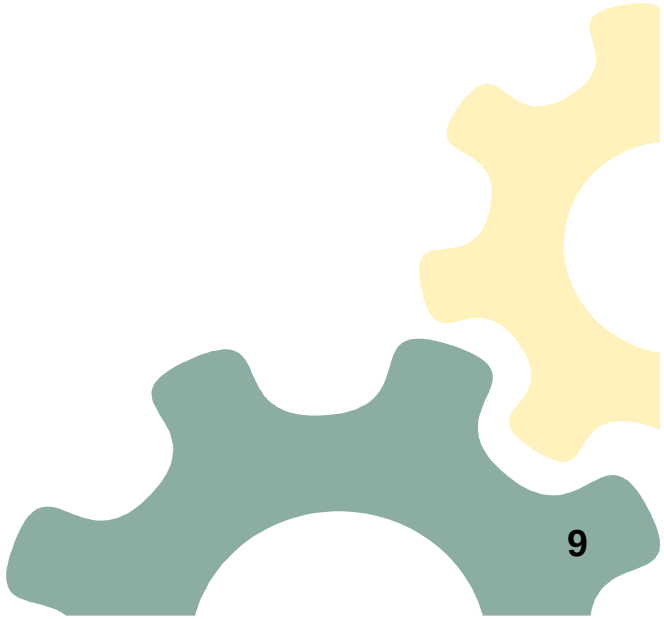


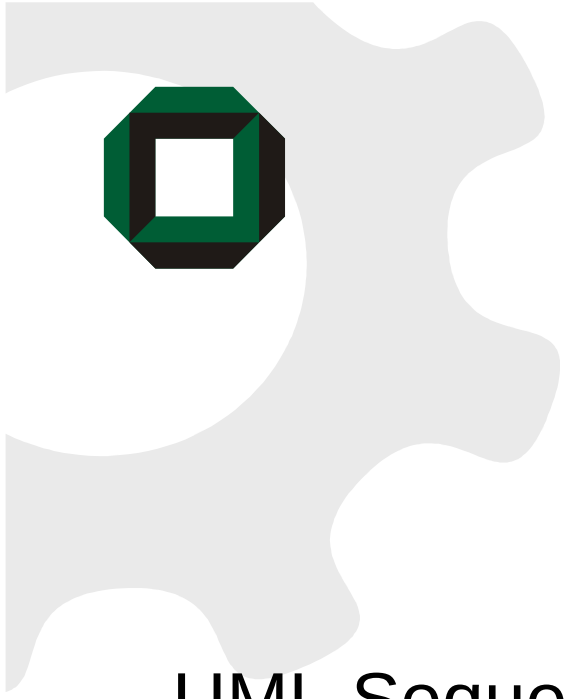
Folie entnommen: Ingenieurinformatik I, SS 2007, Prof. Dr.-Ing. D. Hartmann, Lehrstuhl für Ingenieurinformatik im Bauwesen, Ruhr-Universität Bochum



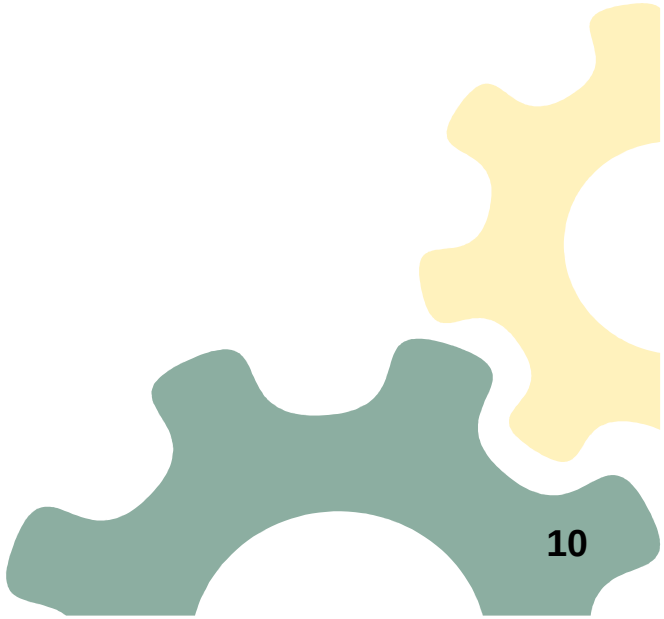
Tipps zum UML Klassendiagramm

- Ist keine Multiziplität angegeben, so ist diese „1“.
- Um in der Klausur Diskussion zu vermeiden ist es besser abstrakte Klassen und Methoden durch <<abstract>> zu kennzeichen, anstatt Sie kursiv zu schreiben.



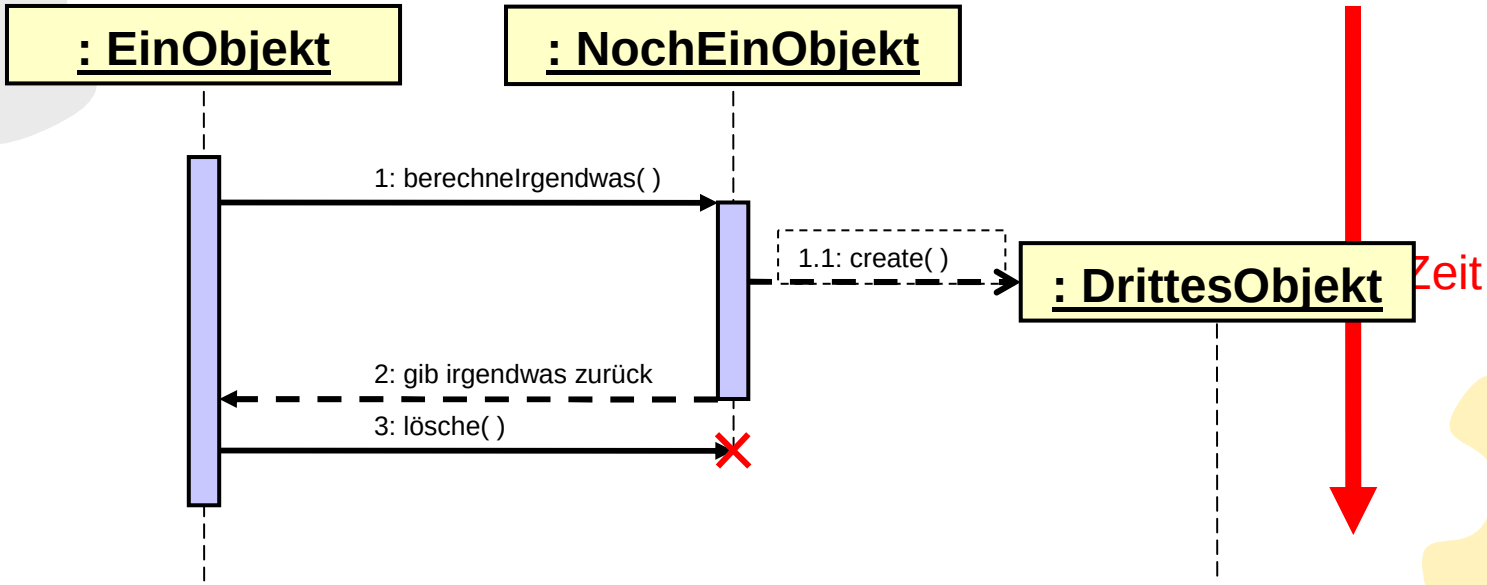


UML Sequenzendiagramm



Sequenzdiagramm

Mit Sequenzdiagrammen lassen sich die zeitlichen Abläufe eines Programms genauer beschreiben.

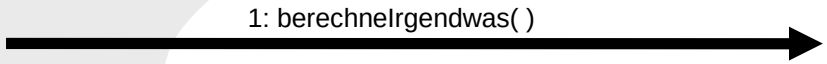


Objekte, die wie im Instanzendiagramm als Rechtecke dargestellt werden.

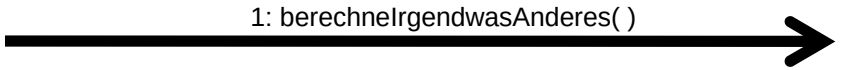
Folie entnommen: Ingenieurinformatik II, WS 2007/08, Prof. Dr.-Ing. D. Hartmann, Lehrstuhl für Ingenieurinformatik im Bauwesen, Ruhr-Universität Bochum

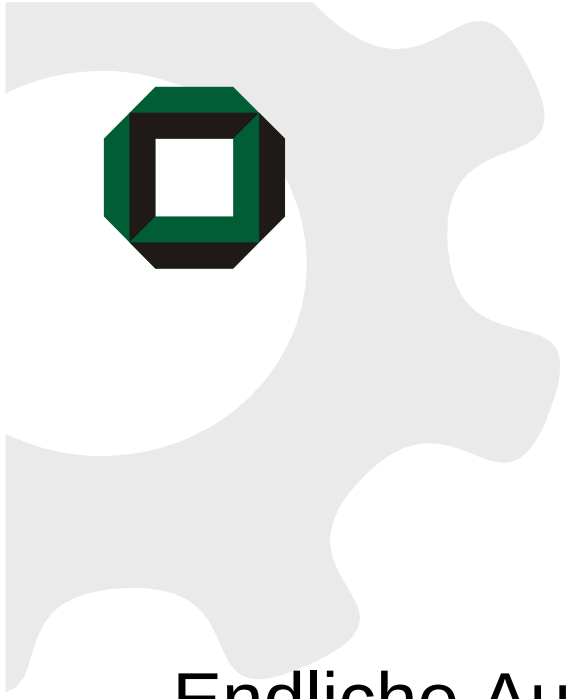
Sychrone vs. Asynchrone Aufrufe

- Synchrone Aufrufe mit ausgefüllter Pfeilspitze

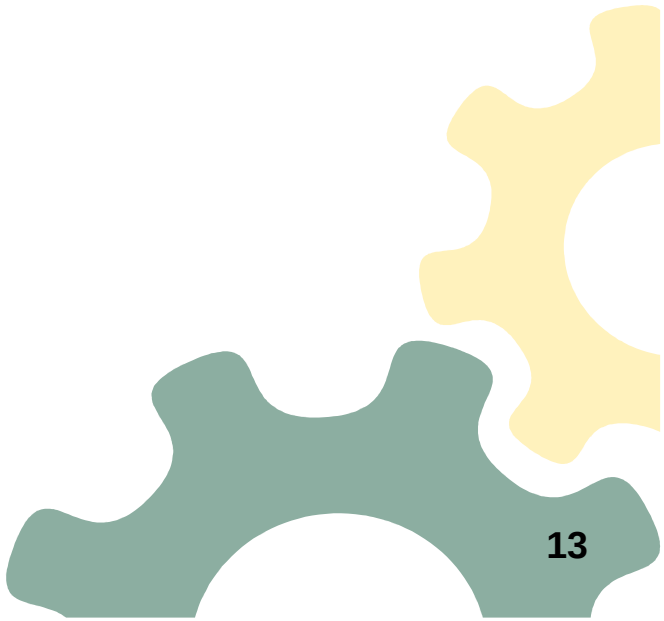


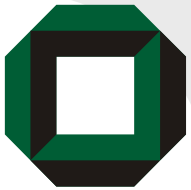
- Asynchrone Aufrufe mit normaler Pfeilspitze





Endliche Automaten



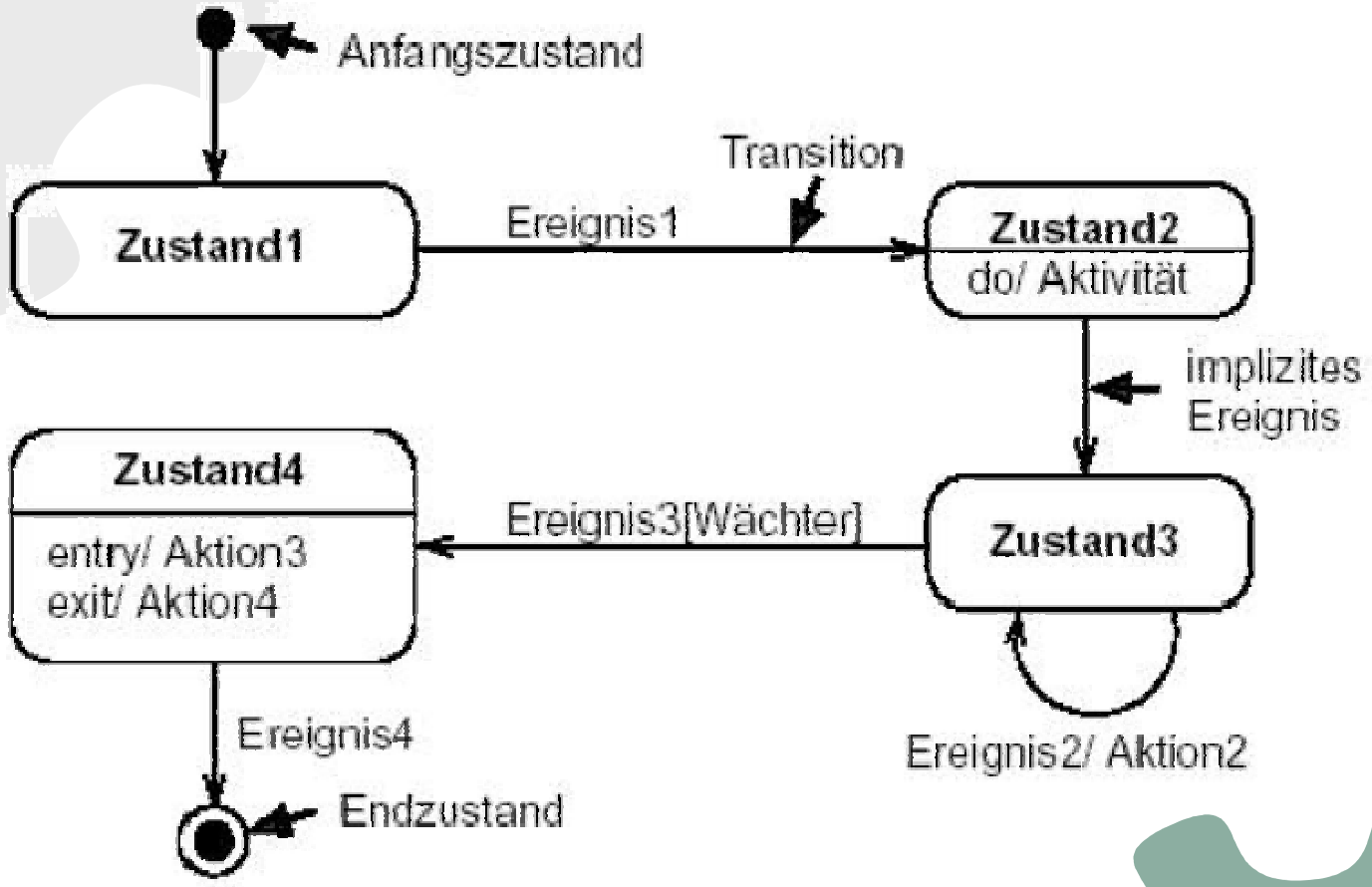


Endliche Automaten in der UML

- Automat
- ein Anfangszustand
- endliche Menge Zustände

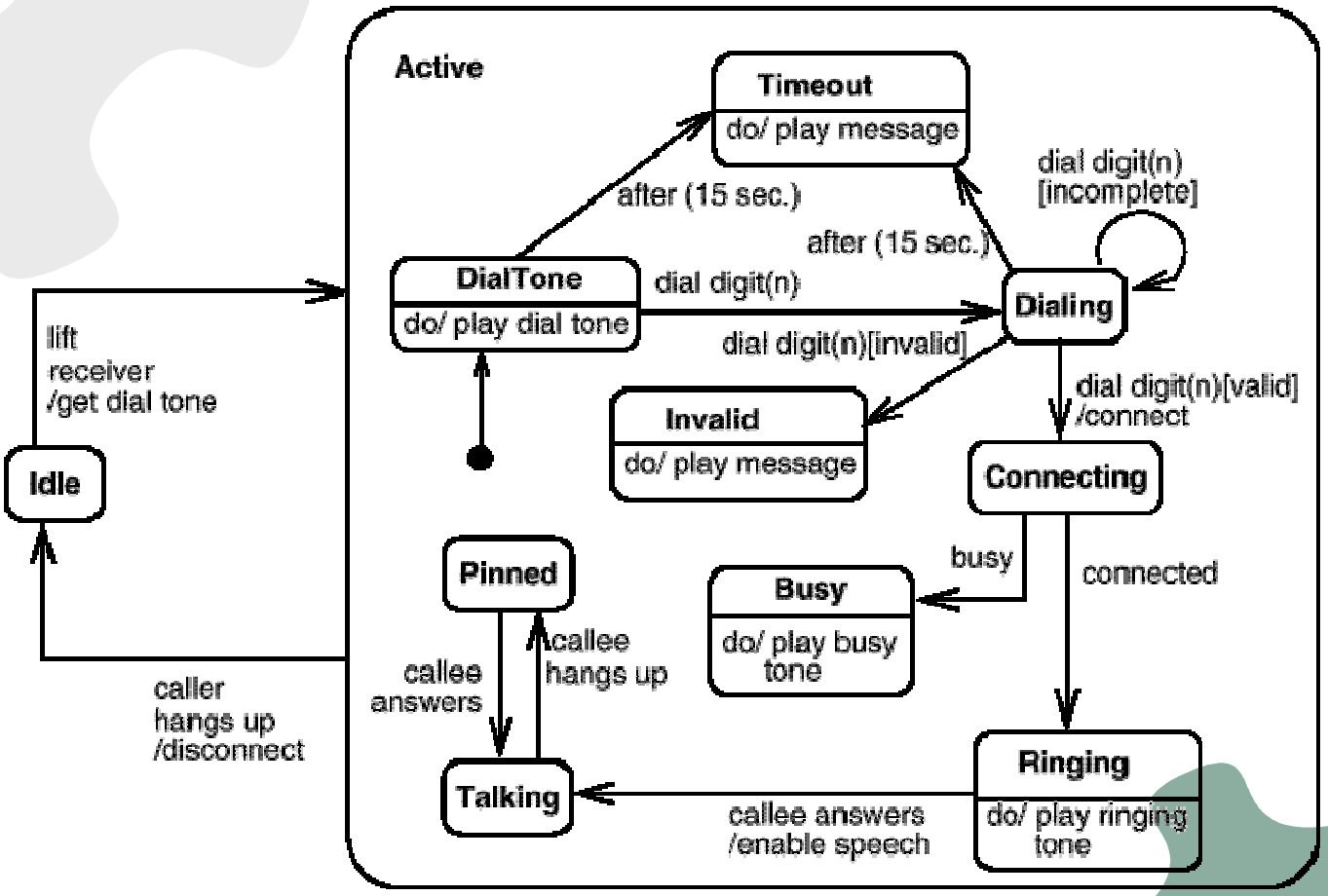
  Aber: Unterzustände mit eigenem Anfangszustand möglich
- endliche Menge Transitionen (Übergänge) durch Ereignisse
- endliche Menge Endzustände

UML Statechart I



UML Statechart II

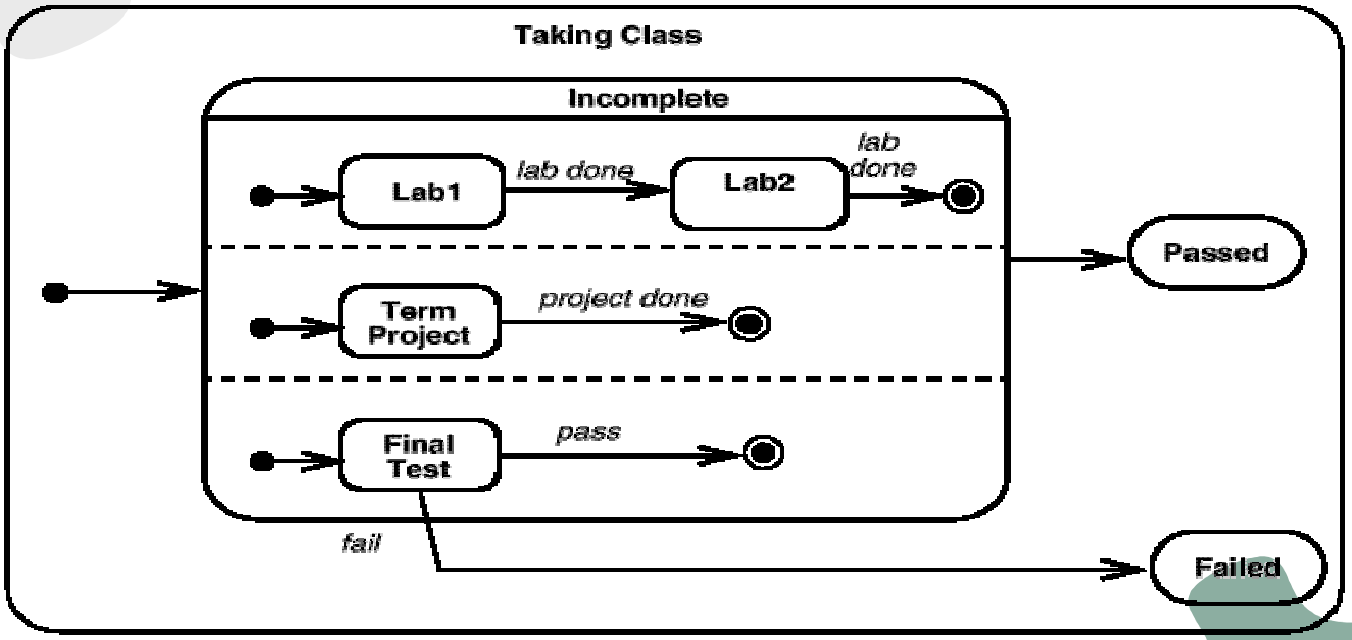
Figure 41. State diagram



UML Statechart III

Nebenläufige Statecharts ermöglichen die Darstellung von Teilsystemen
Beim Betreten werden alle Startzustände betreten
Beim Verlassen werden alle Komponenten verlassen

Figure 44. Concurrent substates

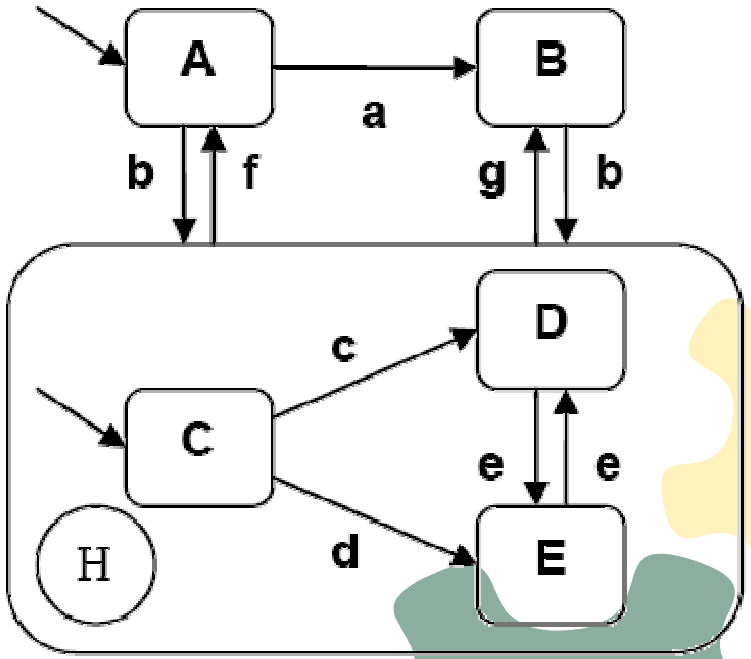


UML Statechart IV

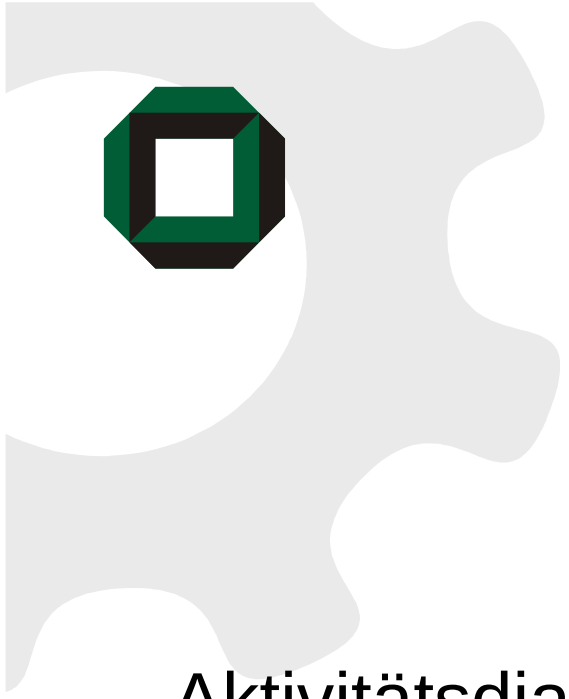
- In welchem Zustand befindet sich der unten stehende Harelautomat nach den folgenden Eingaben? (Der Automat befindet sich zuvor jeweils im Anfangszustand.)

a, b, c, f, b: **D**

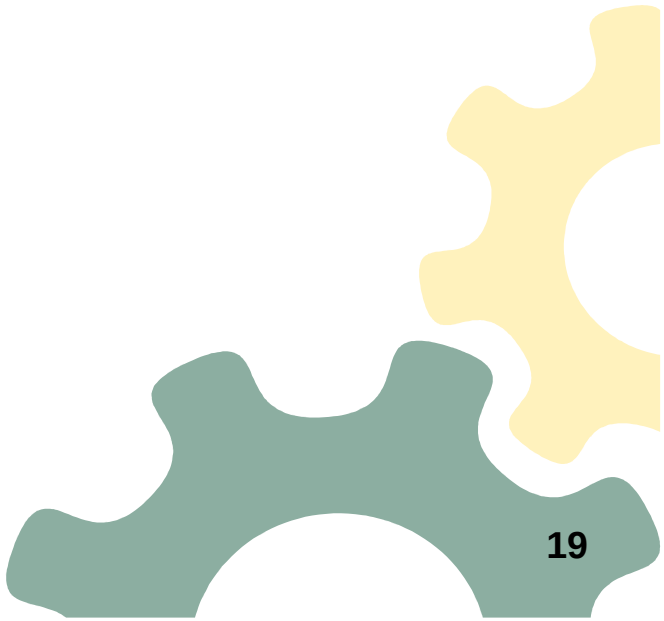
b, d, e, g, e, b: **D**

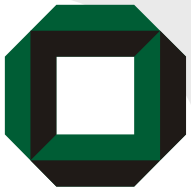


Aufgabe aus der Klausur 2003!



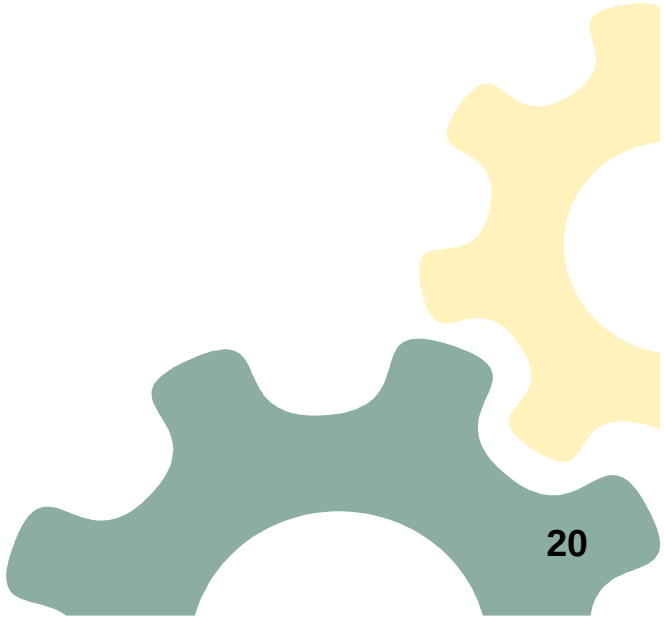
Aktivitätsdiagramme





UML Aktivitätsdiagramm

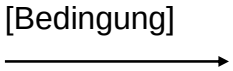
- Wozu: Zur Verhaltensmodellierung
- Ähnliche Notation wie Zustandsdiagramme, aber Fokus auf Aktivitäten (d. h. den Übergängen)
- Eng verwand mit Petri-Netzen



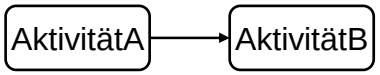
UML Aktivitätsdiagramme

Aktivität

Kontrollfluß

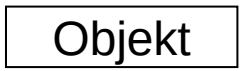


Kontrollfluß, der unter der angegebenen Bedingung gewählt wird.

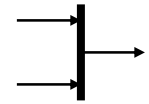


AktivitätB wird im Abschluß an AktivitätA gestartet.

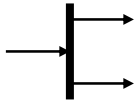
Verzweigunsaktivität



Objektknoten



Synchronisation der Kontrolle (AND) mit Synchronisationsbedingung

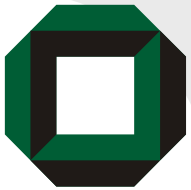


Aufsplitten der Kontrolle (Parallelität)

Startknoten (beliebig viele) möglich

Aktivitätsende (beendet die komplette Aktivität)

Kontrollflußende (beendet nur den Kontrollfluß)



Die wichtigsten Konzepte I

- Aktivität

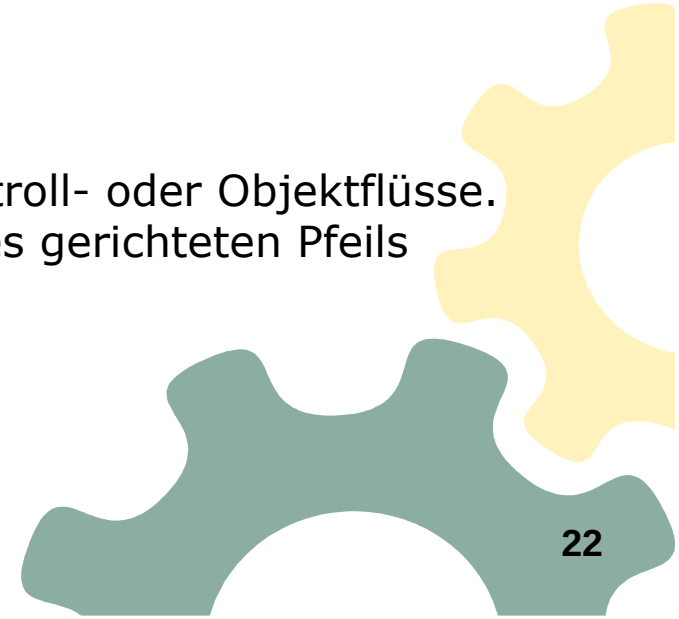
  Eine Aktivität umfasst eine geordnete Folge von Aktivitätsknoten, die durch Aktivitätskanten verbunden sind.

- Aktivitätsknoten

  Ein Aktivitätsdiagramm besteht aus mehreren Aktivitätsknoten. Es gibt drei Arten von Aktivitätsknoten:

  - Aktion
  - Objektknoten
  - Kontrollknoten

- Aktivitätskanten repräsentieren entweder Kontroll- oder Objektflüsse. Dies wird in Aktivitätsdiagrammen mittels eines gerichteten Pfeils (offene Spitze) dargestellt.



Die wichtigsten Konzepte II

Kanten *zwischen Aktionen* definieren die *Ausführungsreihenfolge* von Aktionen. Sie stehen stellvertretend für die Ausführung einer Aktion.

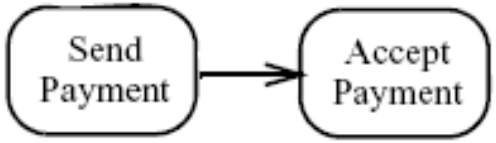


Figure 12.30 - Examples of actions

Die wichtigsten Konzepte III

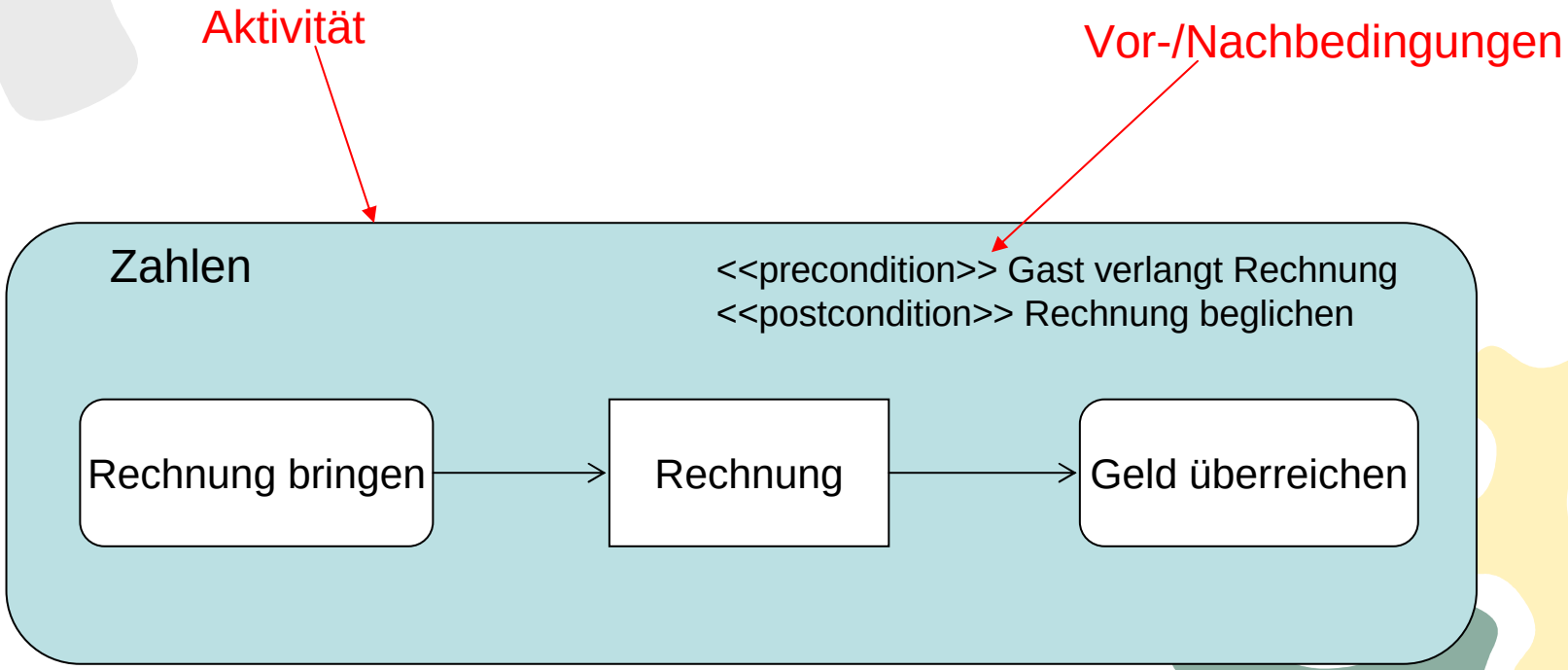
Objektknoten modellieren die Übergabe von Objekten zwischen Aktionen.

Der Transport selbst wird mittels der Aktivitätskante dargestellt, die in diesem Fall den Objektfluß symbolisiert.

Notation: Rechteck in das der Typ des Objektknotens eingetragen wird.

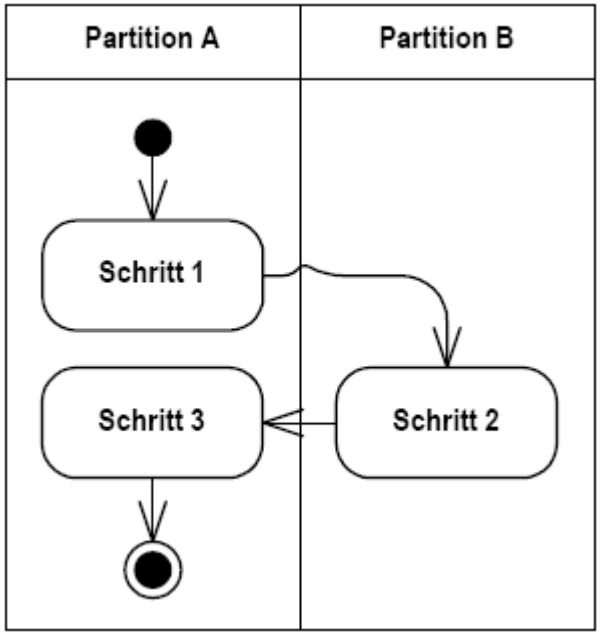
Die wichtigsten Konzepte IV

In Aktivitätsdiagrammen können mehrere Aktivitätsknoten zu Aktivitäten zusammengefasst werden:



Die wichtigsten Konzepte V

- Partitionen werden verwendet, um anzugeben welcher Akteur welche Aktivitäten ausführt.



UML Aktivitätsdiagramm (einfaches Beispiel)

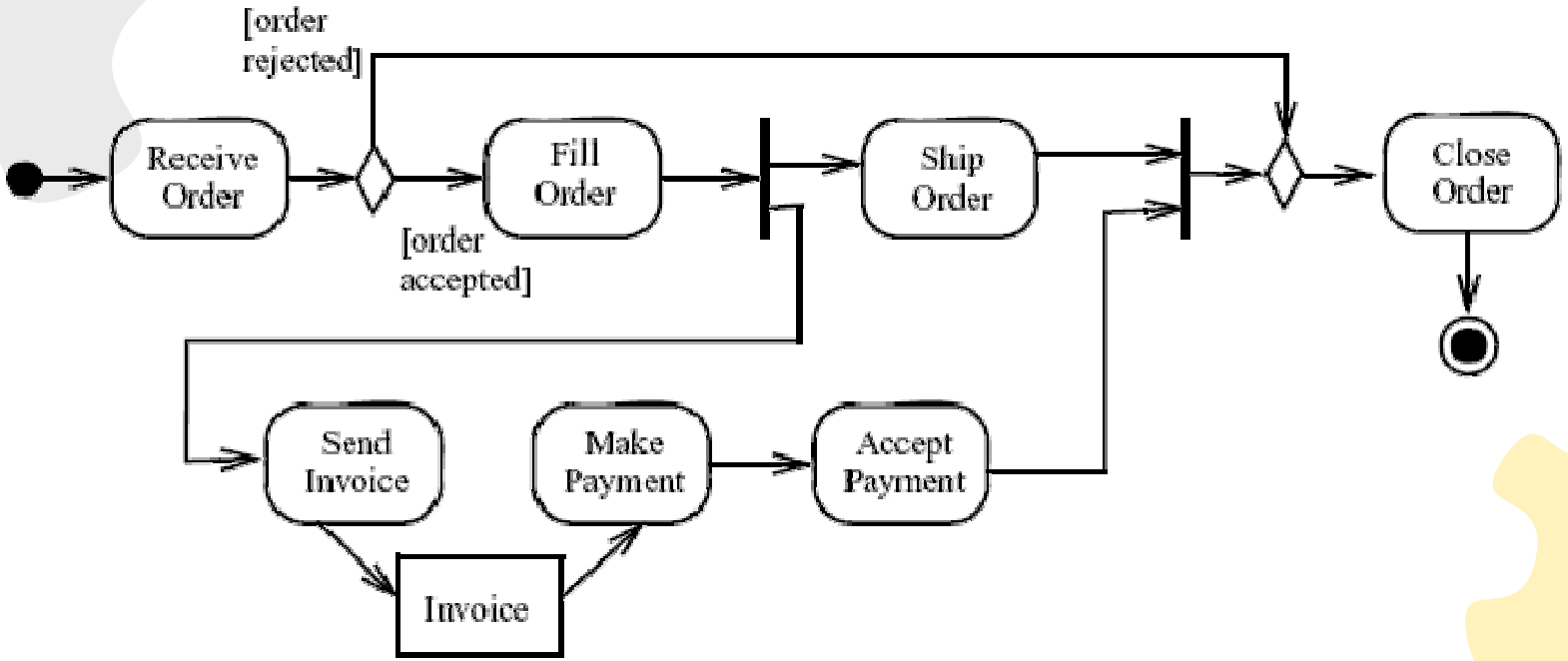


Figure 12.73 - Control node examples (with accompanying actions and control flows)

UML Aktivitätsdiagramm (Endknoten)

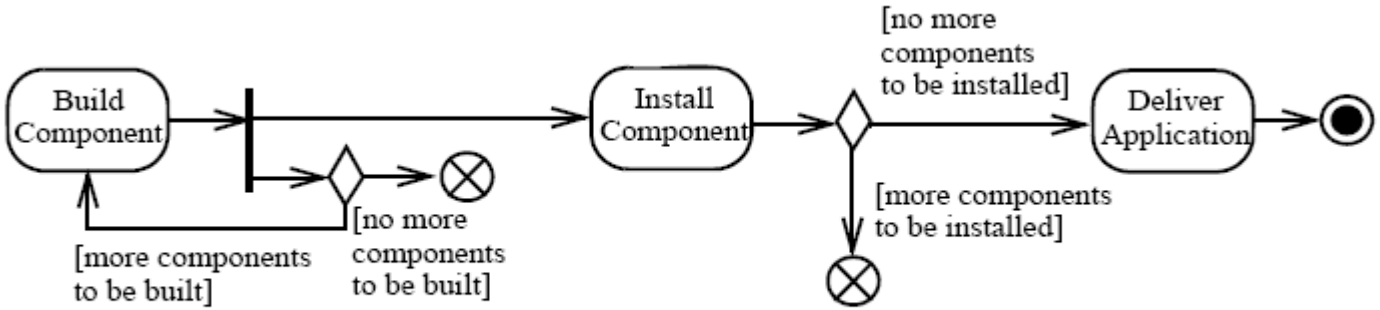
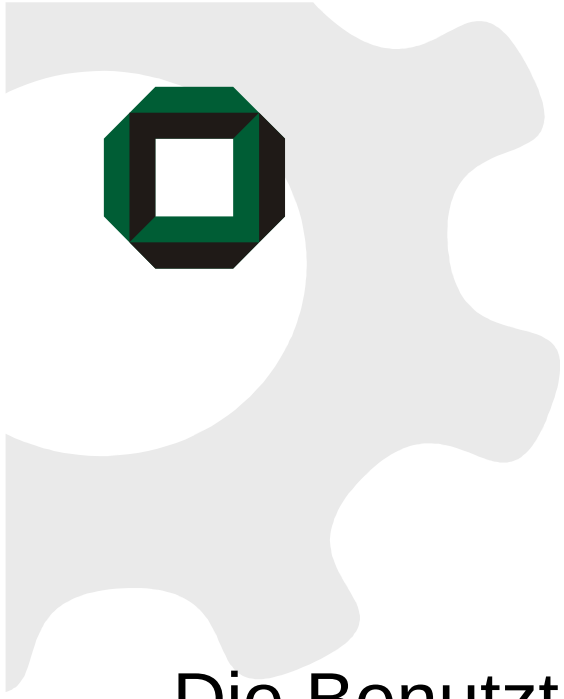
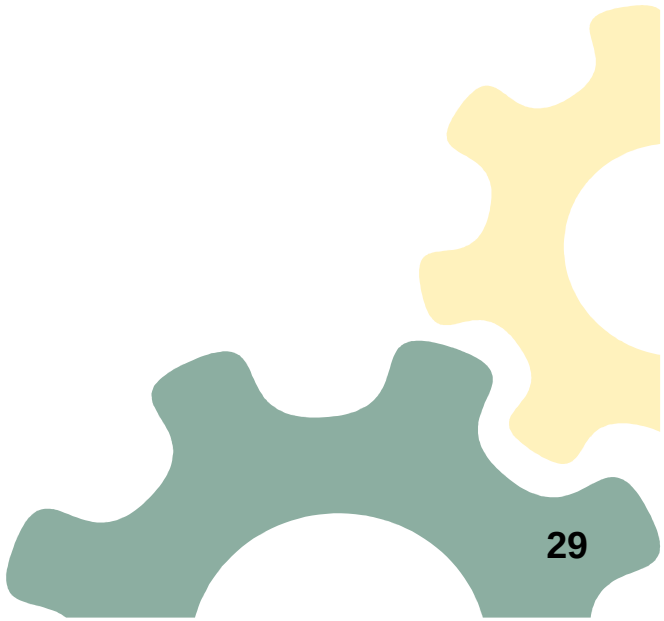
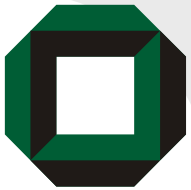


Figure 12.91 - Flow final and activity final example.



Die Benutzt-Relation





Die Benutzt-Relation

- Ein Modul A benutzt ein Modul B, wenn die Korrektheit von A von der Korrektheit von B abhängt!
- Beispiele:

Ein Programm A ruft ein Programm B nicht auf, erwar
vorberechneten Wert in einer gemeinsam genutzten Datei.

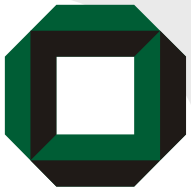
A benutzt B

Ein Modul A nutzt zur Ausgabe von Fehlermeldungen i in Modul B.

A benutzt B nicht

Ein Modul A, das Public-Key Verschlüsselung betreibt, ruft einen Primzahlgenerator in Modul B auf.

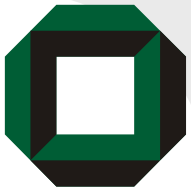
A benutzt B



Die Benutzt-Relation

- Eine Programmkomponente A benutzt eine Programmkomponente B und diese wiederum benutzt eine Programmkomponente C. Demnach erfordert A für den korrekten Ablauf sowohl die Verfügbarkeit einer korrekten Implementierung von B, als auch die Verfügbarkeit einer korrekten Implementierung von C.

Falsch!

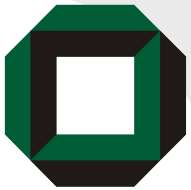


Tipps fürs nächste Übungsblatt

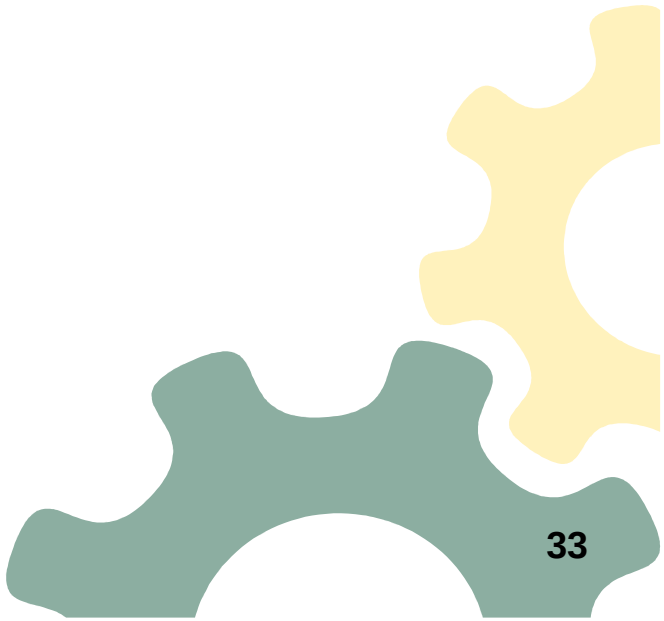
- ListCellRender

```
public Component getListCellRendererComponent(JList list,  Object value,
int index, boolean isSelected, boolean cellHasFocus) {

    // Wichtig ist zu wissen, dass getListCellRendererComponent ein JLabel
    // ist und man sich damit einiges an arbeit ersparen kann...

    JLabel label = (JLabel) super.getListCellRendererComponent(list, value,
index, isSelected, cellHasFocus);

        label.setOpaque(true);

         if (index % 2 == 1) {
           label.setBackground(Color.BLUE);
         } else {
           label.setBackground(Color.RED);
         }
  return label;
}
```

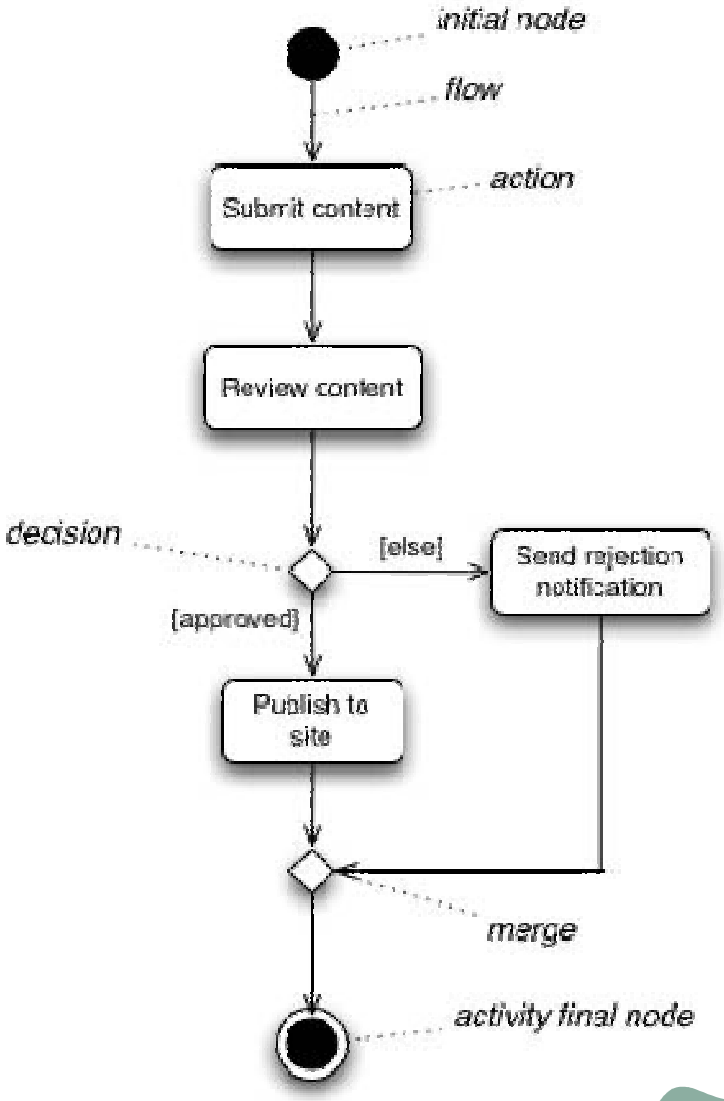
Beispielprogramm auf meiner SWT1-Tut Homepage



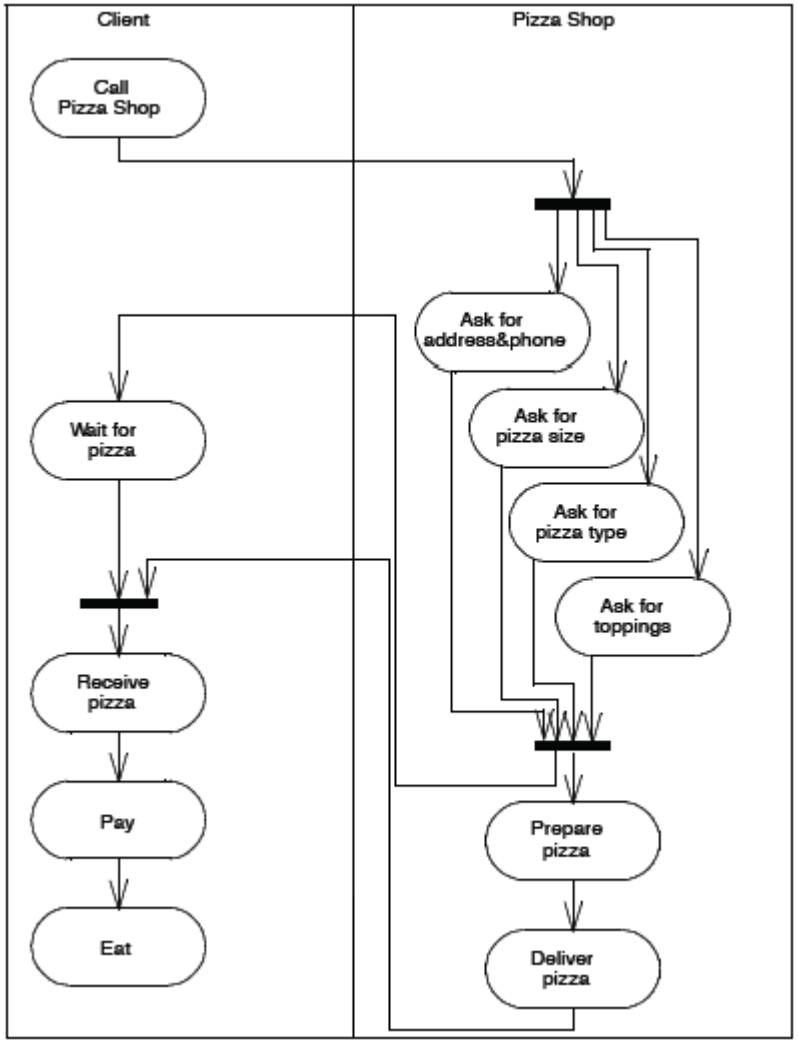
Aufgaben

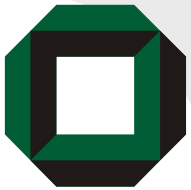


Lösung Aufgabe 1



Lösung Aufgabe 2





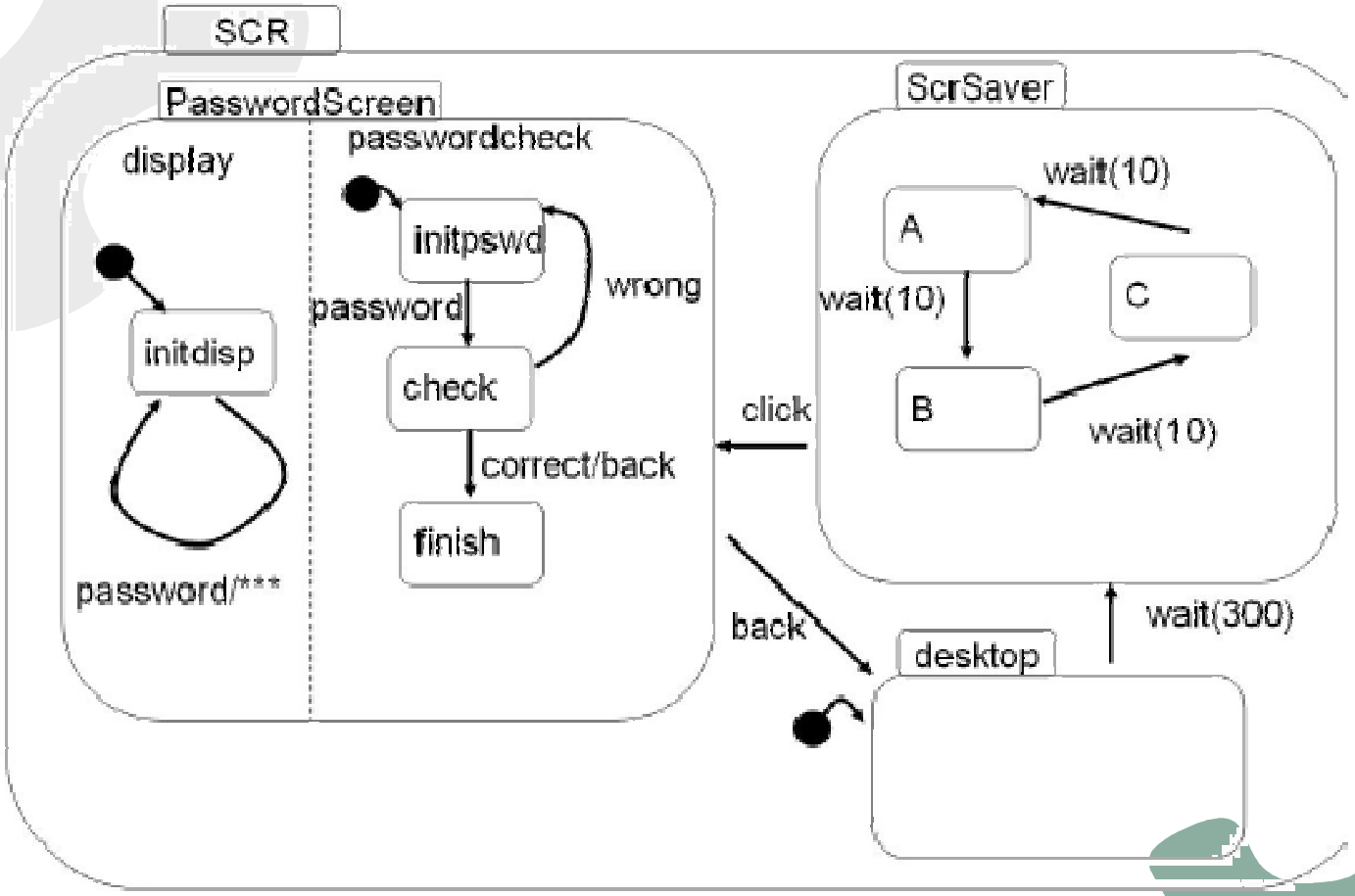
Aufgabe 3

Erzeugen Sie ein Statechart für einen Bildschirmschoner. Zu Beginn zeigt das System den Desktop (Zustand: Desktop) an. Ist das System 5 Minuten inaktiv, so soll in den Zustand Bildschirmschoner gewechselt werden. Innerhalb dieses Zustandes soll für alternierend für jeweils 10 Sekunden die Muster A, B und C gezeigt werden. Nach einem Mausklick soll das System nach einen Passwort fragen. Das System zeigt *** während das Passwort überprüft wird. Sollte das Passwort korrekt sein, wechselt das System wieder in den Desktop Zustand, sonst wird der Passwortbildschirm weiterhin angezeigt.

Hinweise:

- Verwenden sie – nach Möglichkeit – hierarchische Zustände.
- Zeiten werden als Trigger folgendermaßen ausgedrückt:
  wait(Zeitdauer) [Guard] / Ausgabe

Aufgabe 3


SCR
PasswordScreen
display
passwordcheck
initpswd
wrong
password
initdisp
check
correct/back
finish
password/***
click
ScrSaver
wait(10)
A
wait(10)
C
B
wait(10)
back
wait(300)
desktop