Objektorientierte Analyse
35. Dynamische Modellierung und Szenarioanalyse mit Aktionsdiagrammen

Prof. Dr. rer. nat. Uwe Aßmann

Institut für Software- und Multimediatechnik

Lehrstuhl Softwaretechnologie

Fakultät für Informatik

TU Dresden

Version 19-0.2, 15.06.19

6) Einsatz in der Analyse

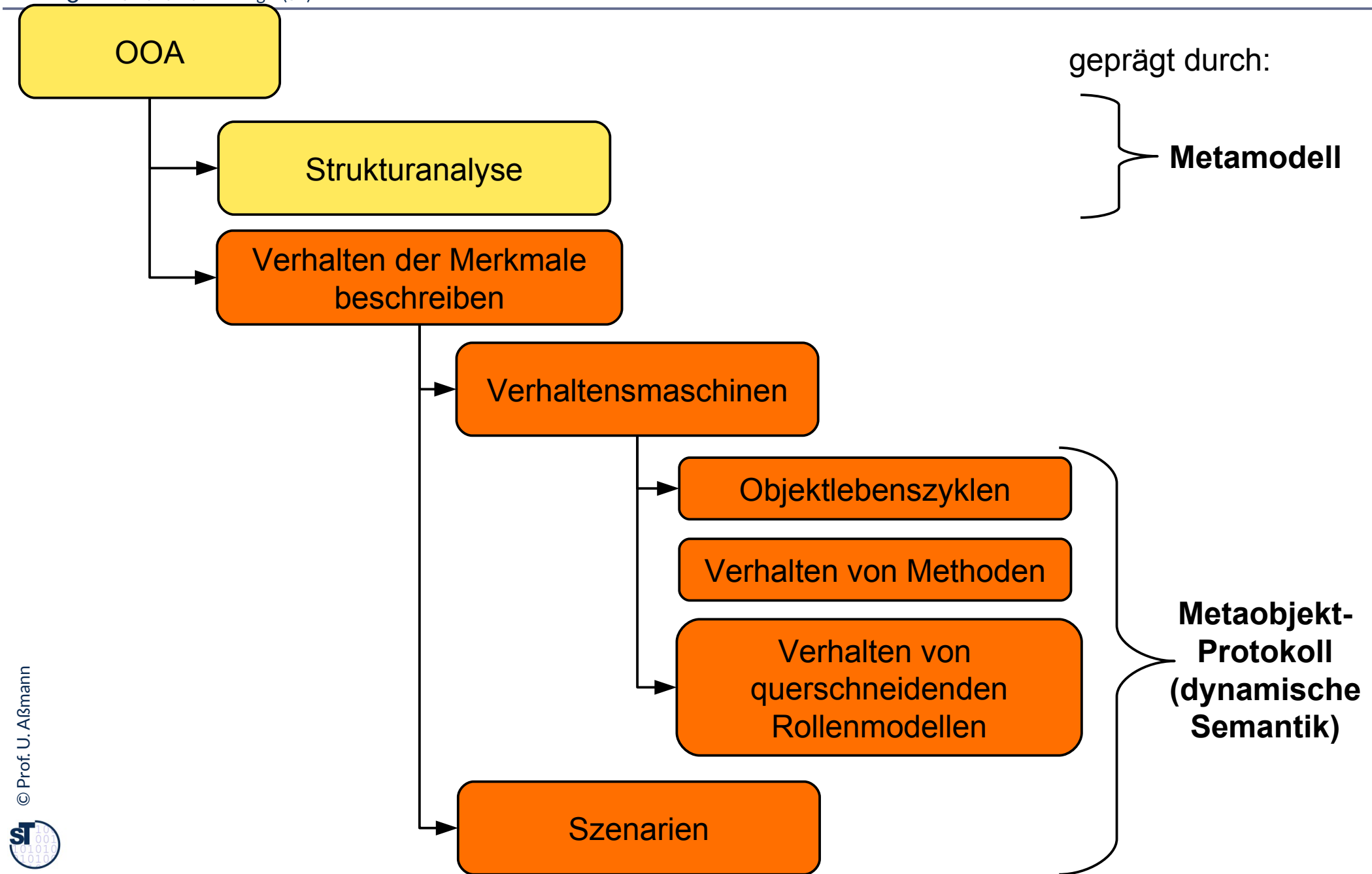
7) Entwurfsmuster State

8) Strukturierte Zustände

Überblick Teil III: Objektorientierte Analyse (OOA)

1. Überblick Objektorientierte Analyse
   1. (schon gehabt:) Strukturelle Modellierung mit CRC-Karten
2. Strukturelle metamodellgetriebene Modellierung mit UML
   1. Strukturelle metamodellgetriebene Modellierung für das Domänenmodell
   2. Strukturelle Modellierung von komplexen Objekten
   3. Strukturelle Modellierung für Kontextmodell und Top-Level-Architektur
3. Analyse von funktionalen Anforderungen (Verhaltensanalyse)
   1. Funktionale Verfeinerung: Dynamische Modellierung und Szenarienanalyse mit Aktionsdiagrammen
   2. Funktionale querschneidende Verfeinerung: Szenarienanalyse mit Anwendungsfällen, Kollaborationen und Interaktionsdiagrammen
   3. (Funktionale querschneidende Verfeinerung für komplexe Objekte)
4. Beispiel Fallstudie EU-Rent

Schritte der objektorientierten, metamodellgetriebenen Analyse



Punktweise und querschneidende dynamische Verfeinerung

Punktweise funktionale Verfeinerung ist eine funktionale Verfeinerung eines Modellfragmentes (meist Objekt oder Methode), die *punktweise* geschieht, d.h. pro Modellfragment separat durchgeführt wird.

Kap. 34-35

- Ergebnis der Verfeinerung einer Klasse im Strukturmodell:
  - **Lebenszyklus** des komplexen Objektes
  - **Implementierung** einer Methode

Welchen Lebenszyklus durchläuft ein Objekt? Welche Zustände oder Aktivitäten hat eine Methode?

Querschneidende funktionale Verfeinerung ist eine funktionale Verfeinerung *mehrerer* Modellfragmente gleichzeitig, die *querschneidend* geschieht.

Kap. 36

- Damit kann man das Zusammenspiel mehrerer Objekte oder Methoden untersuchen, eine *Szenarienanalyse*, die quasi die Draufsicht auf ein Szenario ermittelt
  - Siehe Kapitel “Szenarienanalyse”

Welches Interaktionsprotokoll durchläuft eine Gruppe von Objekten?

35.1 Einsatzzwecke von Zustandsdiagrammen

Q5: Schritte der Modellierung in Bezug auf Schichten des Systems

	Schichten	Analyse	Entwurf	Feinentwurf	Implementierung
Benutzungsschnittstelle (Boundary)	GUI		Verfeinerung →		
	Controller				
Anwendungslogik (Control)	Kontextmodell	Aufstellung aus Domänenmodell; Erarbeitung System-schnittstellen	stabil	Umsetzen auf jUML	stabil
	Top-Level-Architektur	Verfeinerung des Kontextmodells	stabil	Umsetzen auf jUML	stabil
	Architektur		Ausarbeitung Architektur (PSM)	Einziehen von Plattformabhängigkeiten (PSM) Umsetzen auf jUML Sequentialisierung	Details ausfüllen, Methoden ausprogrammieren
	Tools		Ausarbeitung Tools	Einziehen von Plattformabhängigkeiten (PSM); Umsetzen auf jUML;	Details ausfüllen, Methoden ausprogrammieren
Datenhaltung (Database)	Material	Aufstellung aus Domänenmodell		Einziehen von Plattformabhängigkeiten (PSM); Umsetzen auf jUML;	Details ausfüllen, Methoden ausprogrammieren

Zustandsdiagramme in der Analyse

- **Einsatz von allgemeinen Zustandsmodellen:** Anwendungsfälle können mit Zustandsmodellen verfeinert werden (Szenarienanalyse)
  - die Aktion aus dem Anwendungsfall kann als Aktion einer Verhaltensmaschine aufgefasst werden
  - Achtung: dann ist die Verhaltensmaschine noch nicht einer Klasse zugeordnet
- **Einsatz von Steuerungsmaschinen** (Objektlebenszyklen)
  - Für komplexe Objekte kann eine Steuerungsmaschine angegeben werden
  - Auch für alle Unterobjekte des komplexen Objektes
- **Einsatz von Protokollmaschinen**
  - Zur Modellierung von Geschäftsprozessen in Geschäftssoftwaresystemen
  - Modellierung von Protokollen für Anschlüssen (ports) von UML-Komponenten im Kontextmodell
  - Ihr Einsatz in der Szenarienanalyse ist nicht möglich:
    - Es ist nicht möglich, eine Aktion aus einem Anwendungsfall als Ereignis einer Protokollmaschine aufzufassen und damit Szenarienanalyse zu betreiben
    - da die Protokollmaschine nur Aufrufreihenfolgen beschreibt, aber keine Verfeinerungen von Aktionen zulässt

Zustandsdiagramme im Entwurf

- Zustandsmodelle (ohne Objektzuordnung) sind im Entwurf Objekten/Klassen zuzuordnen, da alle Zustandsdiagramme zu Objektlebenszyklen werden müssen
  - Aus den Zustandsmodellen entstehen also Steuerungsmaschinen
- **Steuerungsmaschinen**
  - können Verhalten, d.h., Implementierungen von beliebigen Klassen spezifizieren (***Objektlebenszyklen***, *white-box object life cycle)*
  - können als technische Steuerungsmaschinen Implementierungen von technischen Geräten beschreiben (***Gerätelebenszyklus***)
- **Protokollmaschinen**
  - können gültige **Aufrufreihenfolgen** an **(Protokolle** von**)** Objekte beschreiben und zur Ableitung von Vertragsprüfern für das defensive Programmieren eingesetzt werden (*black-box object life cycle)*
  - Einsatz bei eingebetteter Software, die hohe Zuverlässigkeit bieten muss

Einsatzzwecke für Zustandsmodelle

	Anwendungsfall-Lebenszyklus	Allgemeiner Objektlebenszyklus (OLC)	Steuerung (technischer Geräte)
Verhaltensmaschine (behavioral state machine)	Zustandsmodell: Szenario-Analyse: Verhalten von Objekten im Kontextmodell und Top-Level-Architektur	Steuerungsmaschine: Verhaltens-beschreibung in Analyse Entwurf Implementierung (white-box OLC)	Technische Steuerungsmaschine: Verhaltens-beschreibung in Analyse Entwurf Implementierung (white-box OLC)
Protokollmaschine (protocol state machine)	Zur Darstellung von Geschäftsprozessen	Vertragsprüfung in Analyse Entwurf Implementierung (black-box OLC)	<< nicht möglich >>

Verwendung von UML-Zustandsmodellen

Verhaltens-Maschinen (Transduktoren):

- Beschreiben das *Verhalten (Implementierung) eines Systems*
  - z. B. die *Steuerung* eines Systems der realen Welt, zum Steuern von Systemen, eingebettete Systeme etc.
  - Ereignisse sind Signale der Umgebung oder anderer Systemteile
- Reaktion in gegebenem Zustand auf ein bestimmtes Signal:
  - neuer Zustand
  - **ausgelöste Aktion** (wie im Zustandsmodell spezifiziert)
- Zustandsmodelle definieren die *Reaktion* des gesteuerten Systems auf mögliche Ereignisse, d.h. geben eine Implementierung an

Protokoll-Maschinen (Akzeptoren, Prüfmaschinen):

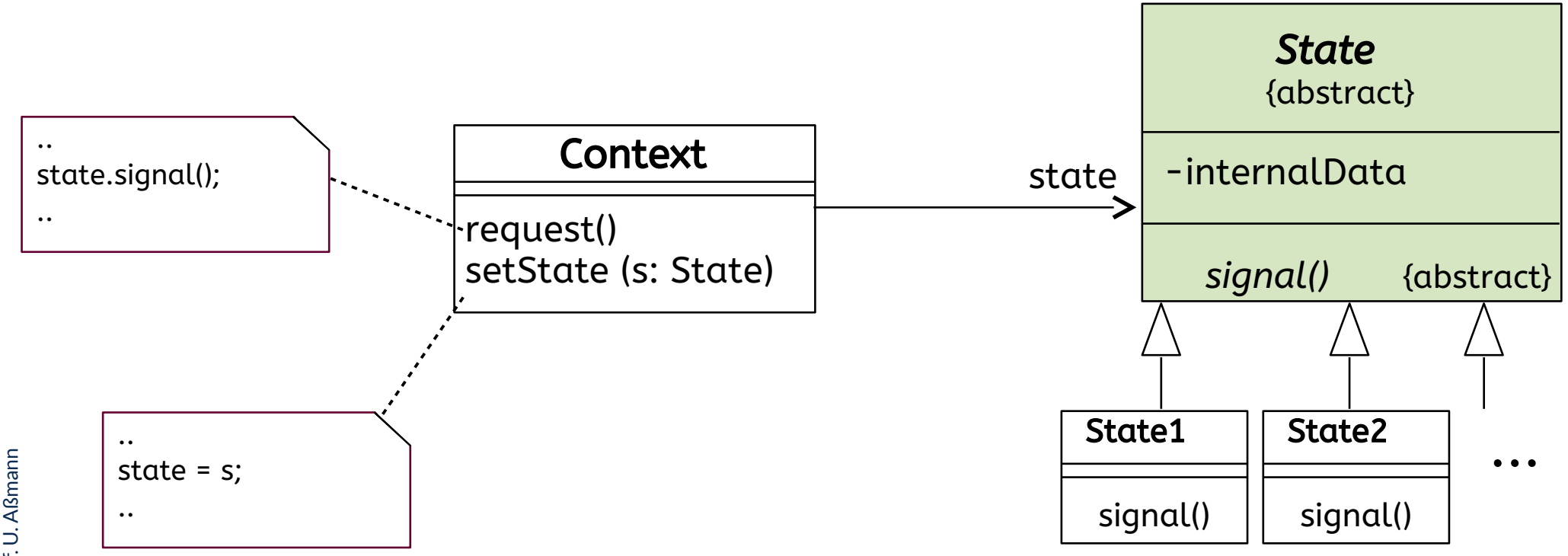
- Zum Überprüfen der *korrekten Aufrufsreihenfolgen*, die ein Benutzer an ein System absetzt
  - Ereignisse sind eingeschränkt auf Operationsaufrufe, d.h. es werden nur Aufruf-Ereignisse berücksichtigt
- Reaktion in gegebenem Zustand auf bestimmten Aufruf:
  - neuer Zustand
  - **Keine Aktionen** im Zustandsmodell!
- Zustandsmodelle definieren zulässige *Reihenfolgen* von Aufrufen (Schnittstelle)
- Protokollmaschinen sind Vertragsprüfer (“checker”), d.h. *Prüfer*, ob das System bzw. das Objekt einem Zustandsmodell folgt

35.2 Entwurfsmuster State

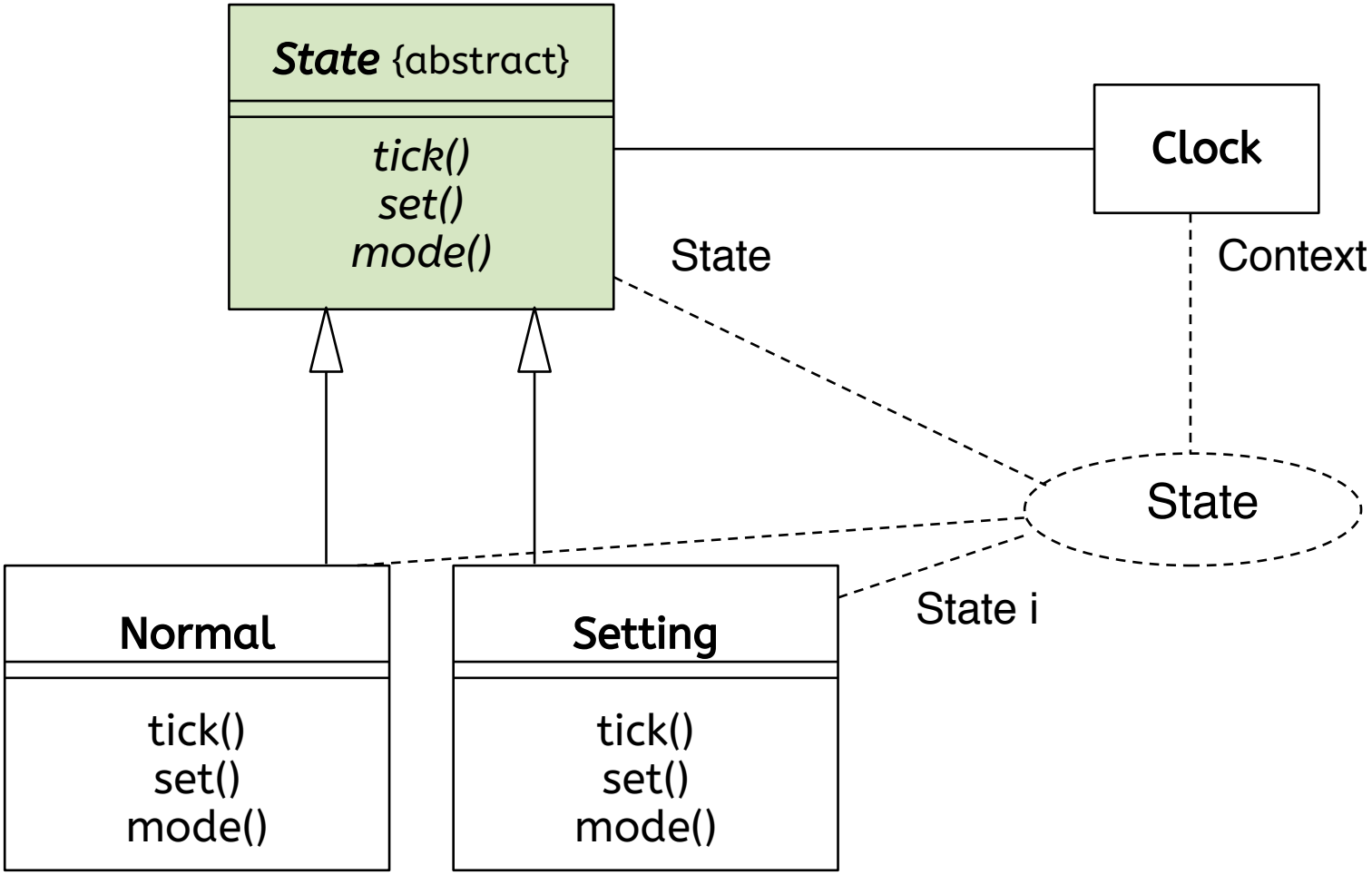
- Aus dem Gamma-Buch

Implementierungsmuster State

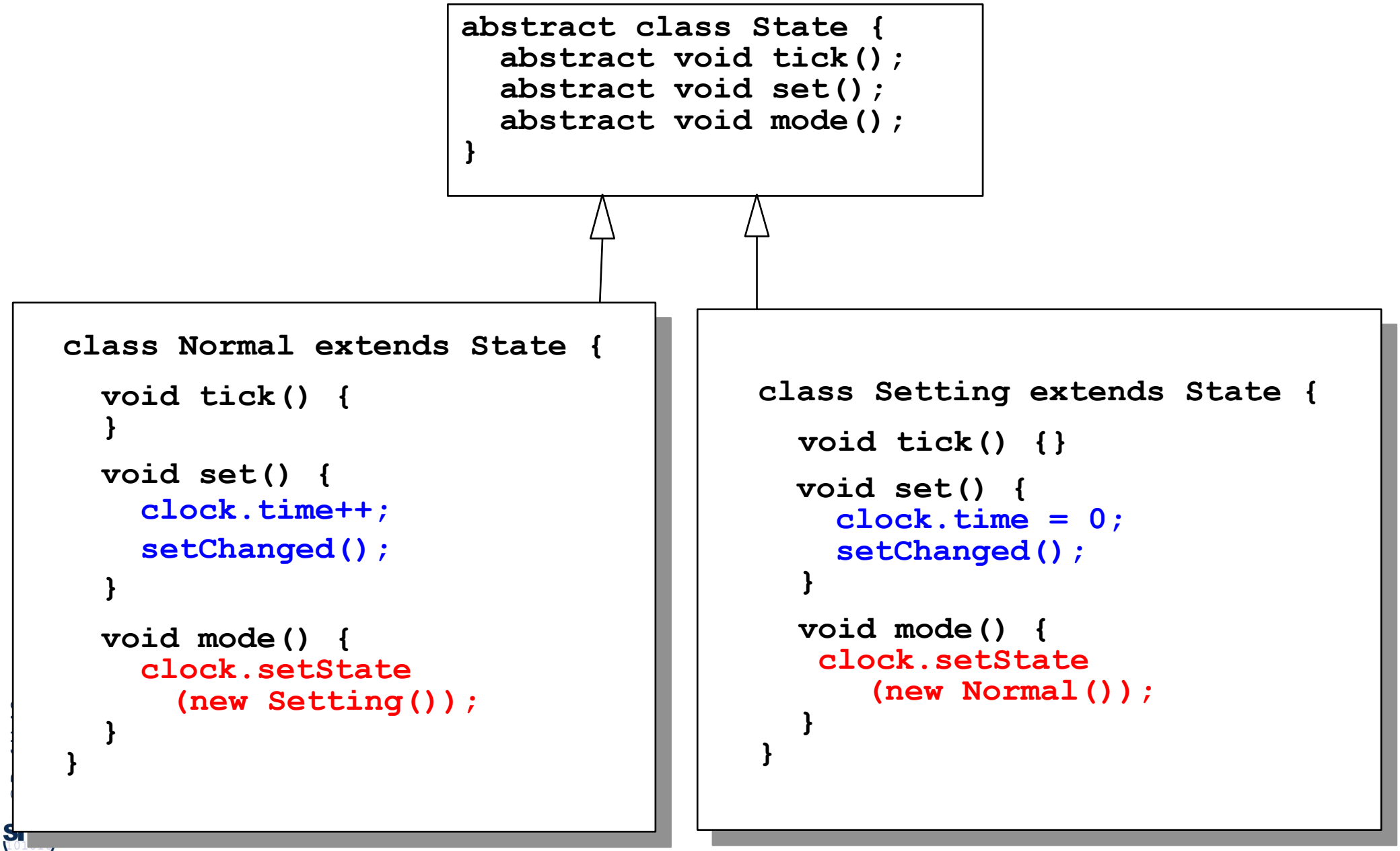
- Problem: Was, wenn der Zustand Informationen (Attribute) enthält?
- Lösung: Darstellung des Zustands durch *Zustandsobjekt*
  - Weiterschalten von Zuständen durch Auswechseln des Zustandsobjekts (Polymorphie)



State-Beispiel für Uhr (1)



State-Beispiel für Uhr in Java (2)



```
abstract class State {
  abstract void tick();
  abstract void set();
  abstract void mode();
}
```

```
class Normal extends State {
  void tick() {
  }
  void set() {
    clock.time++;
    setChanged();
  }
  void mode() {
    clock.setState
      (new Setting());
  }
}
```

```
class Setting extends State {
  void tick() {}
  void set() {
    clock.time = 0;
    setChanged();
  }
  void mode() {
   clock.setState
      (new Normal());
  }
}
```

State

Variante mit geschachtelten Klassen in Java *(inner classes)*


```
class Clock {

  private int time = 0;

  private State normal = new Normal();
  private State setting = new Setting();

  private State s = normal;

  abstract class State {...}

  class Normal extends State {
      void tick() {...}
      void set() {}
      void mode() { s = setting; }
  }

  class Setting extends State {
      ... analog
  }

  public void tick () {
      s.tick();
  }
  ... set(), mode() analog
}
```

Punktweise und querschneidende dynamische Verfeinerung

Punktweise funktionale Verfeinerung ist eine funktionale Verfeinerung eines Modellfragmentes (meist Objekt oder Methode), die *punktweise* geschieht, d.h. pro Modellfragment separat durchgeführt wird.

Kap. 34+35

- Ergebnis der Verfeinerung einer Klasse im Strukturmodell:
  - **Lebenszyklus** des komplexen Objektes
  - **Implementierung** einer Methode

Welchen Lebenszyklus durchläuft ein Objekt? Welche Zustände oder Aktivitäten hat eine Methode?

Querschneidende funktionale Verfeinerung ist eine funktionale Verfeinerung *mehrerer* Modellfragmente gleichzeitig, die *querschneidend* geschieht.

Kap. 36

- Damit kann man das Zusammenspiel mehrerer Objekte oder Methoden untersuchen, eine *Szenarienanalyse*, die quasi die Draufsicht auf ein Szenario ermittelt
  - Siehe Kapitel “Szenarienanalyse”

Welches Interaktionsprotokoll durchläuft eine Gruppe von Objekten?

Steuerungsmaschinen: Zusammenfassung

- Anwendungsgebiet: Punktweise Verfeinerung
  - white-box-Objektlebenszyklen
  - Gerätesteuerungen technischer Geräte (eingebettete Systeme)
    - Mikrowelle, Stoppuhr, Thermostat, ...
    - Große Bedeutung z.B. in Automobil- und Luftfahrtindustrie
    - Problem: Verhalten des gesteuerten Geräts muss *regulär* sein, d.h. die Zustandsmenge muss einer reguläre Sprache entsprechen
- Codegenerierung möglich mit State und IntegerState
  - bei genau definierter Aktionssprache (Aus den Aktionen muss Code generiert werden). Werkzeuge existieren
- Praktische Aspekte:
  - Kommunikation: Nachrichten empfangen/versenden
  - Nebenläufigkeit
  - Reaktivität (Akzeptieren von Nachrichten zu beliebigem Zeitpunkt)
  - Realzeitaspekte

35.3 Vereinfachung von Zustandsdiagrammen durch Strukturierung

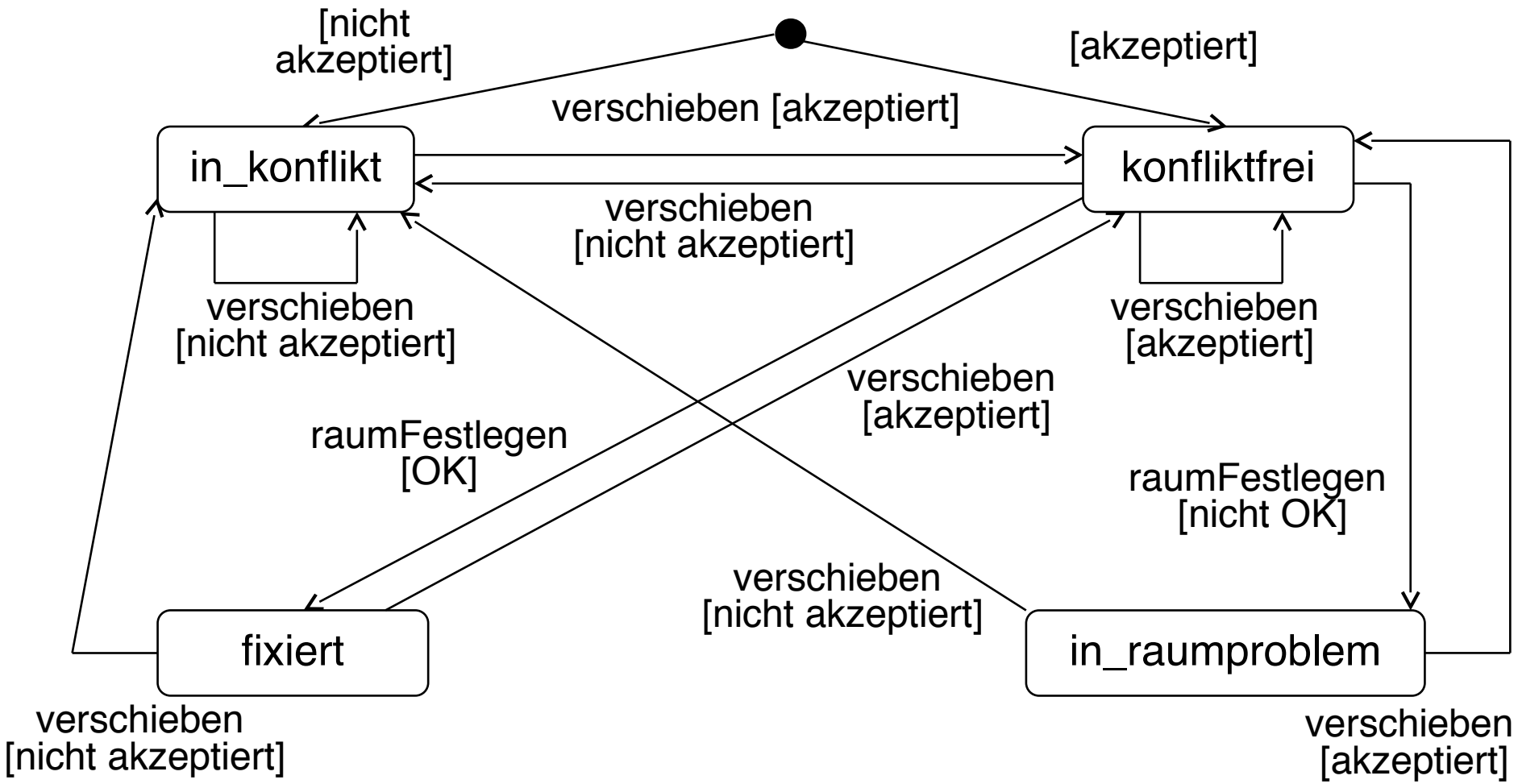
Unterspezifikation und Vervollständigung von Übergängen

- Was passiert, wenn ***kein*** Übergang im aktuellen Zustand für das aktuelle Ereignis angegeben ist?
- Möglichkeiten:
  - Unzulässig
    - Fehlermeldung (Fehlerzustand)
    - Ausnahmebehandlung
  - Zustand unverändert (impliziter "Schleifen"-Übergang)
  - Warteschlange für Ereignisse
  - Unterspezifikation ("wird später festgelegt")

- Achtung: Ein vollständiges Zustandsmodell (totale Übergangsfunktion) ist meist sehr umfangreich und unübersichtlich!

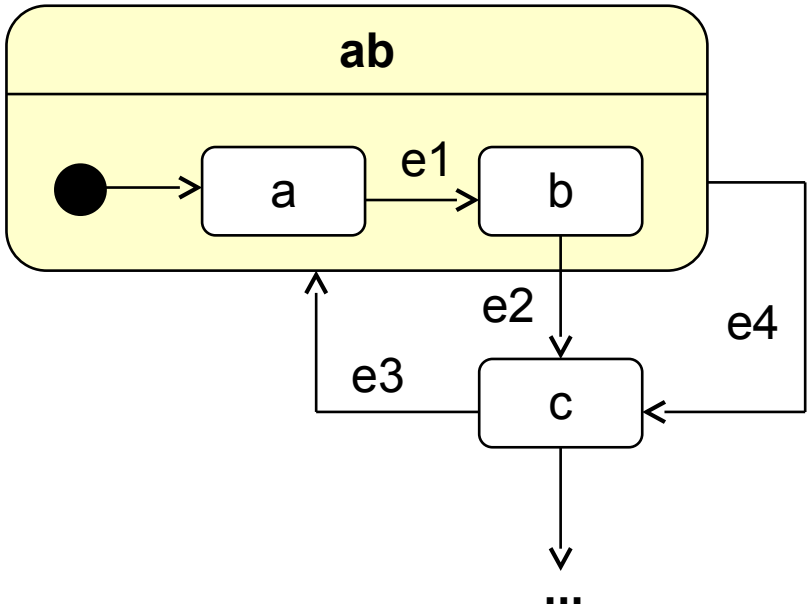
Black-Box Objektlebenszyklus (Protokollmaschine)

- Zulässige Zustände (und Aufruffolgen, Protokoll) von Objekten der Klasse "`Teambesprechung`":
  - Merke: nur zur Generierung eines Vertragsprüfers einsetzbar, nicht zu einer vollständigen Implementierung

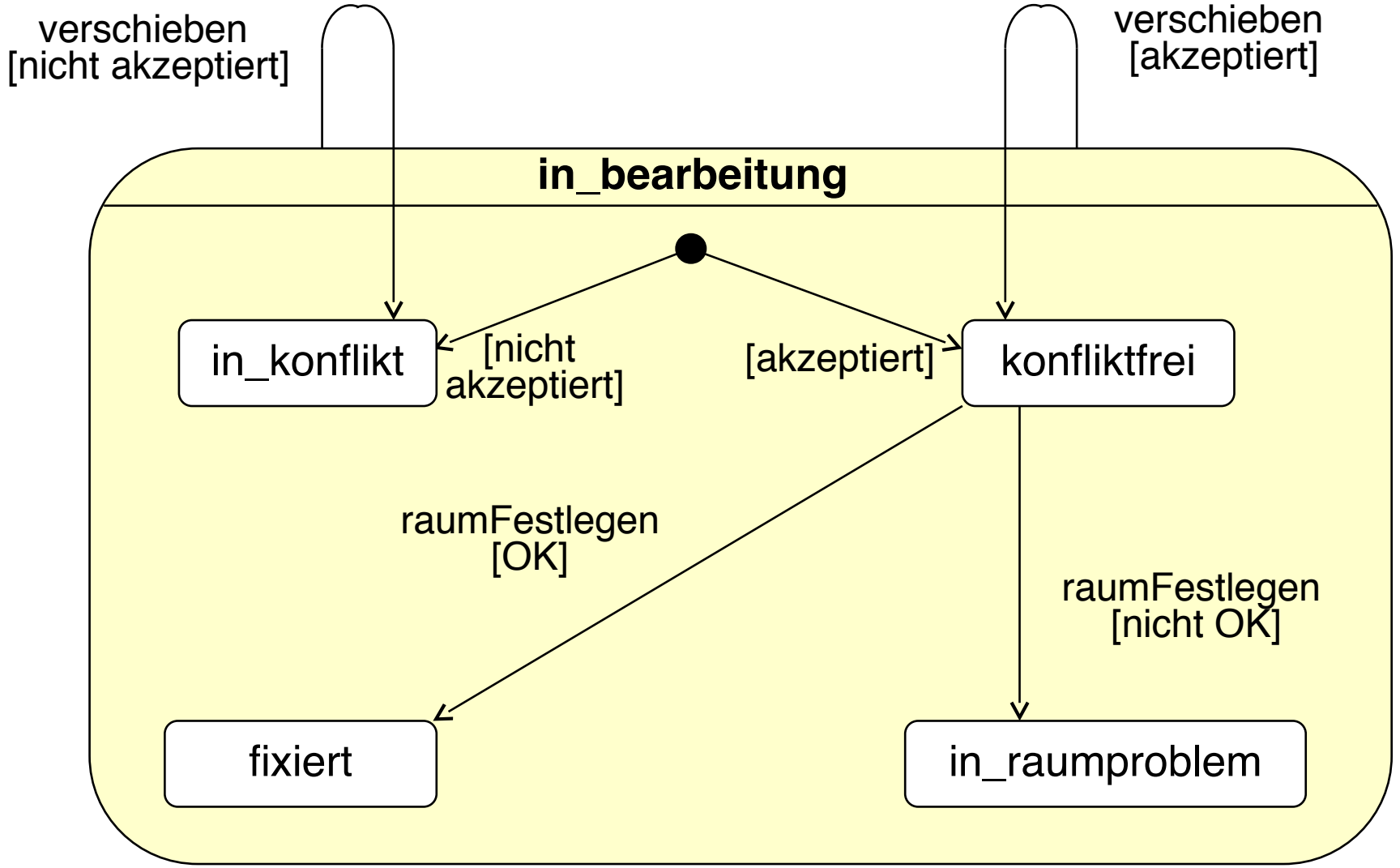


Ober- und Unterzustände

- Zur Vereinfachung, insbesondere, um eine ganze Gruppe von Zuständen einheitlich zu behandeln, können ***Oberzustände*** eingeführt werden.
  - Ein Zustand in den Oberzustand ist ein Übergang in den Startzustand des enthaltenen Zustandsdiagramms.
  - Ein Zustand aus dem Oberzustand gilt für alle Zustände des enthaltenen Zustandsdiagramms (*Vererbung* von Übergangsverhalten).

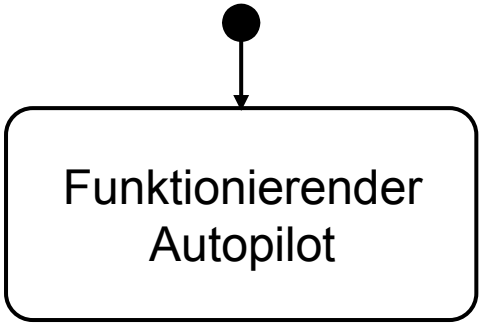


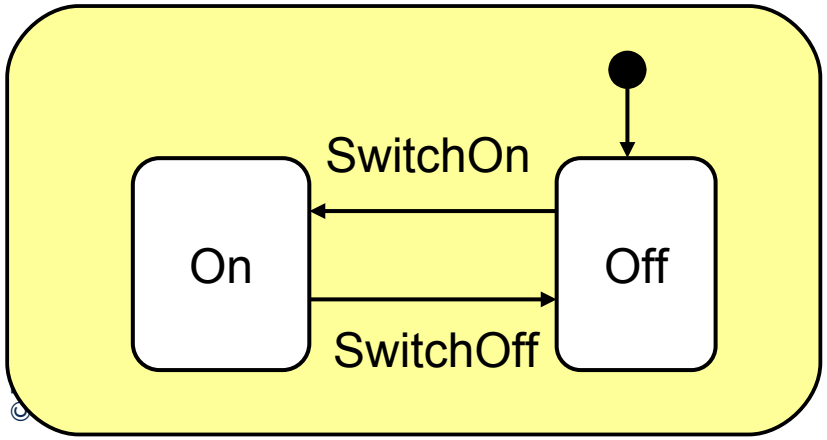
Zustandshierarchie Teambesprechung (jetzt einfacher notiert)

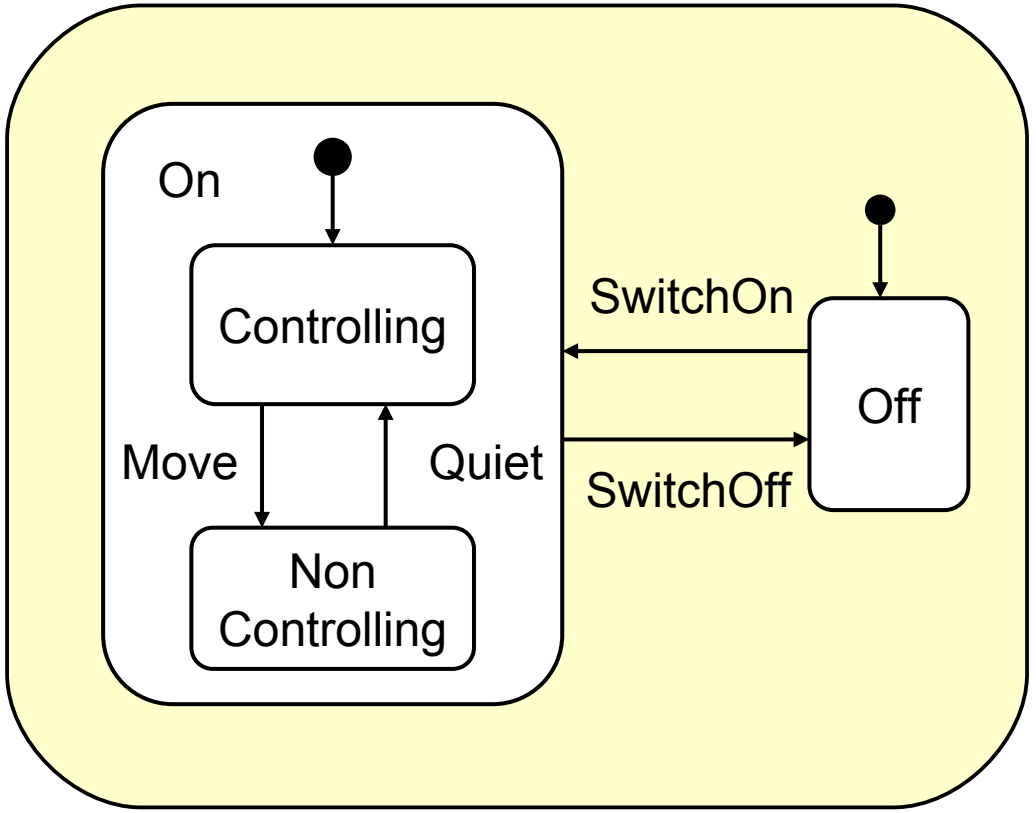


Warum kann man ein hierarchisches Zustandsdiagramm einfach verstehen?

- Es ist kein flacher Automat, sondern ein *hierarchisch gegliedeter*, der in einen einzigen Oberzustand gefaltet werden kann (*reduzibel)*

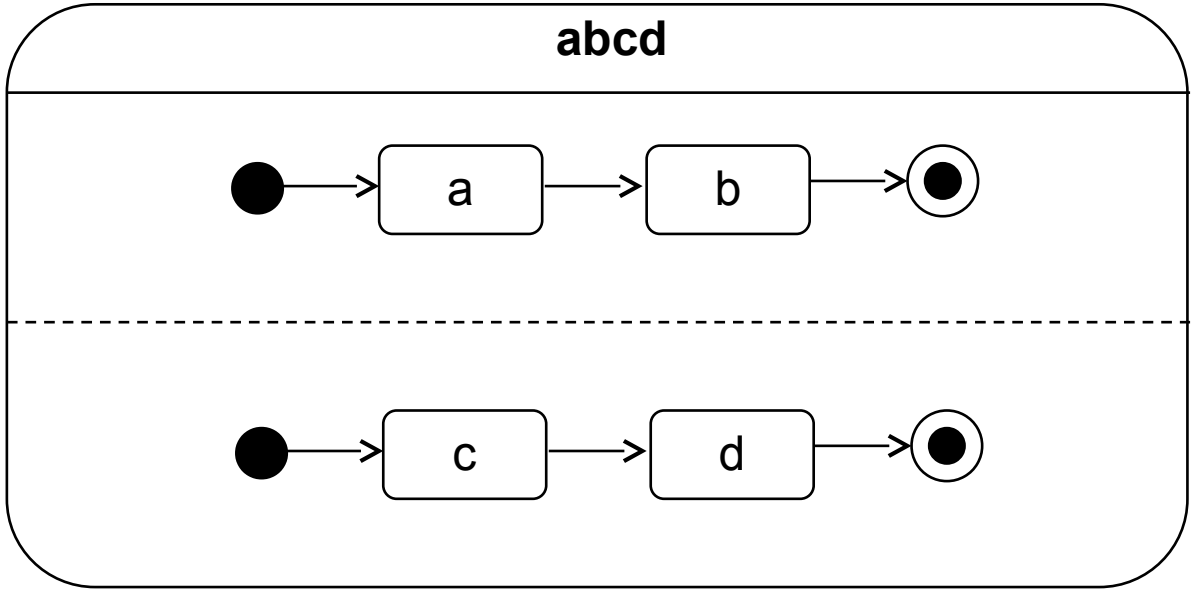






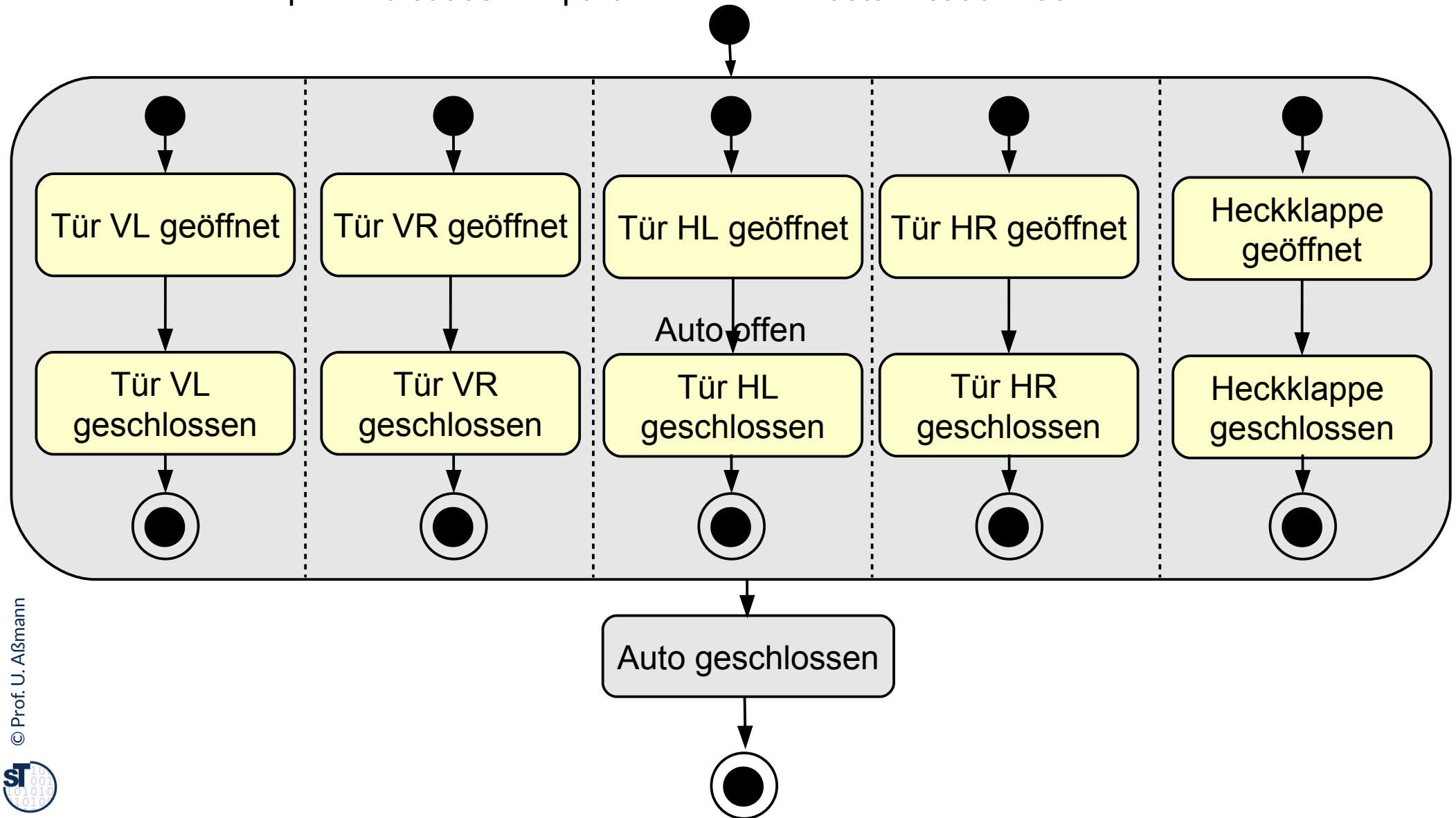
Nebenläufige Teilzustände

- Um voneinander zeitlich unabhängige Vorgänge einfach darzustellen, kann ein Zustand in nebenläufige Teilbereiche zerlegt werden (getrennte "Schwimmbahnen").
  - Ein Zustand des Oberzustands ist ein Tupel von Zuständen der Teilbereiche (Schwimmbahnen).



Türen eines Autos

- Die 4 Türen und die Heckklappe eines Autos bilden einen 5-tupeligen Zustandsraum, der komponiert ist aus den paralleleln Einzelzustandsautomaten

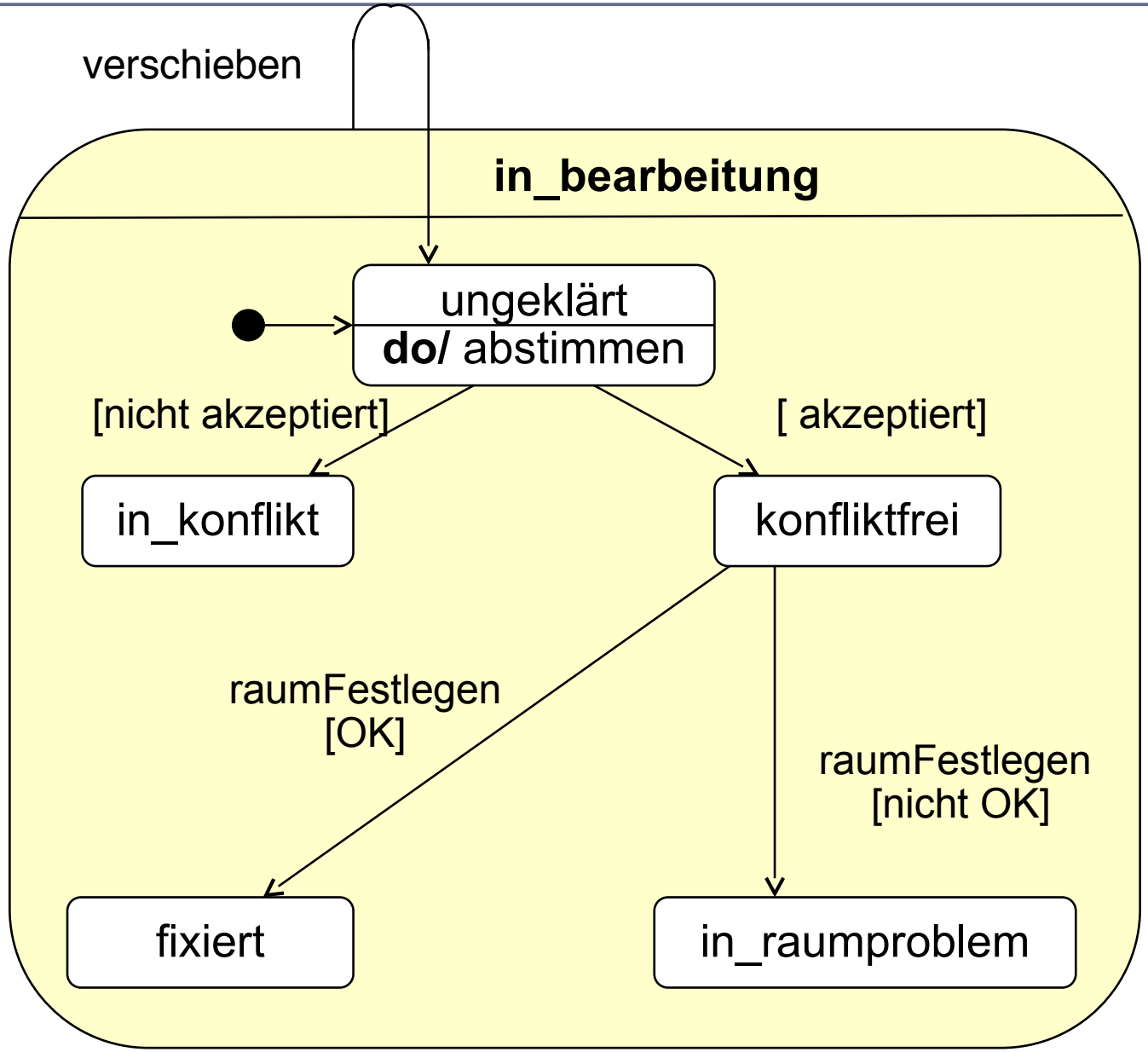


Interne Übergänge

- **Definition:** Ein ***interner Übergang*** eines Zustands *S* beschreibt einen Übergang, der stattfindet, während das Objekt im Zustand *S* ist.
- Es gibt folgende Fälle von internen Übergängen:
  - Eintrittsübergang *(entry transition)*
  - Austrittsübergang *(exit transition)*
  - Fortlaufende Aktivität *(do transition)*
  - Unterdiagrammaufruf *(include transition)*
  - Reaktion auf benanntes Ereignis
- **Notation:**

label = **entry**, **exit**, **do**, **include** oder *Ereignisname*

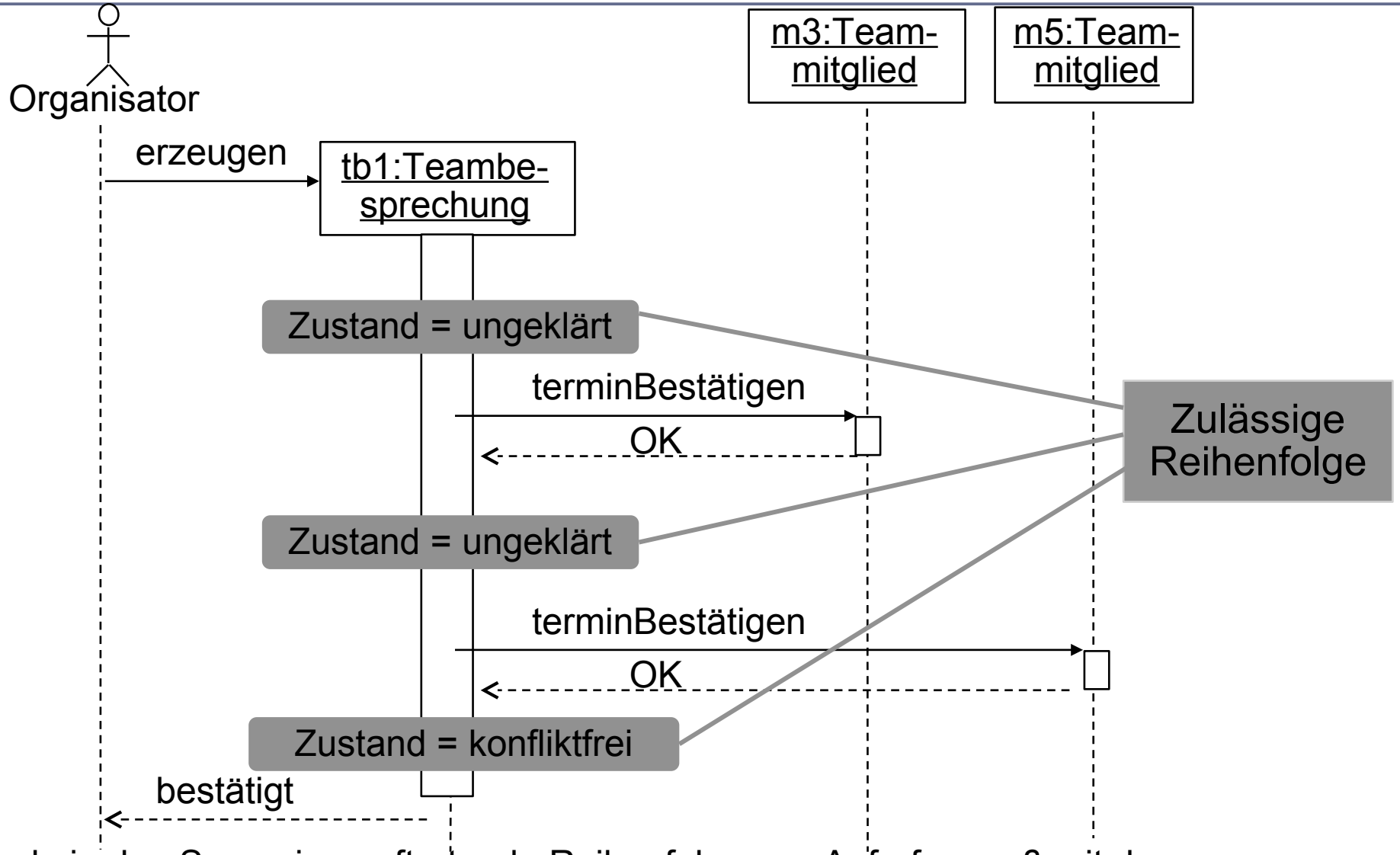
Verschiedene Aktivitäten



Aktivität
"abstimmen"
=
Abstimmung
mit den
Teammitgliedern
per Operation
"terminBestätigen"

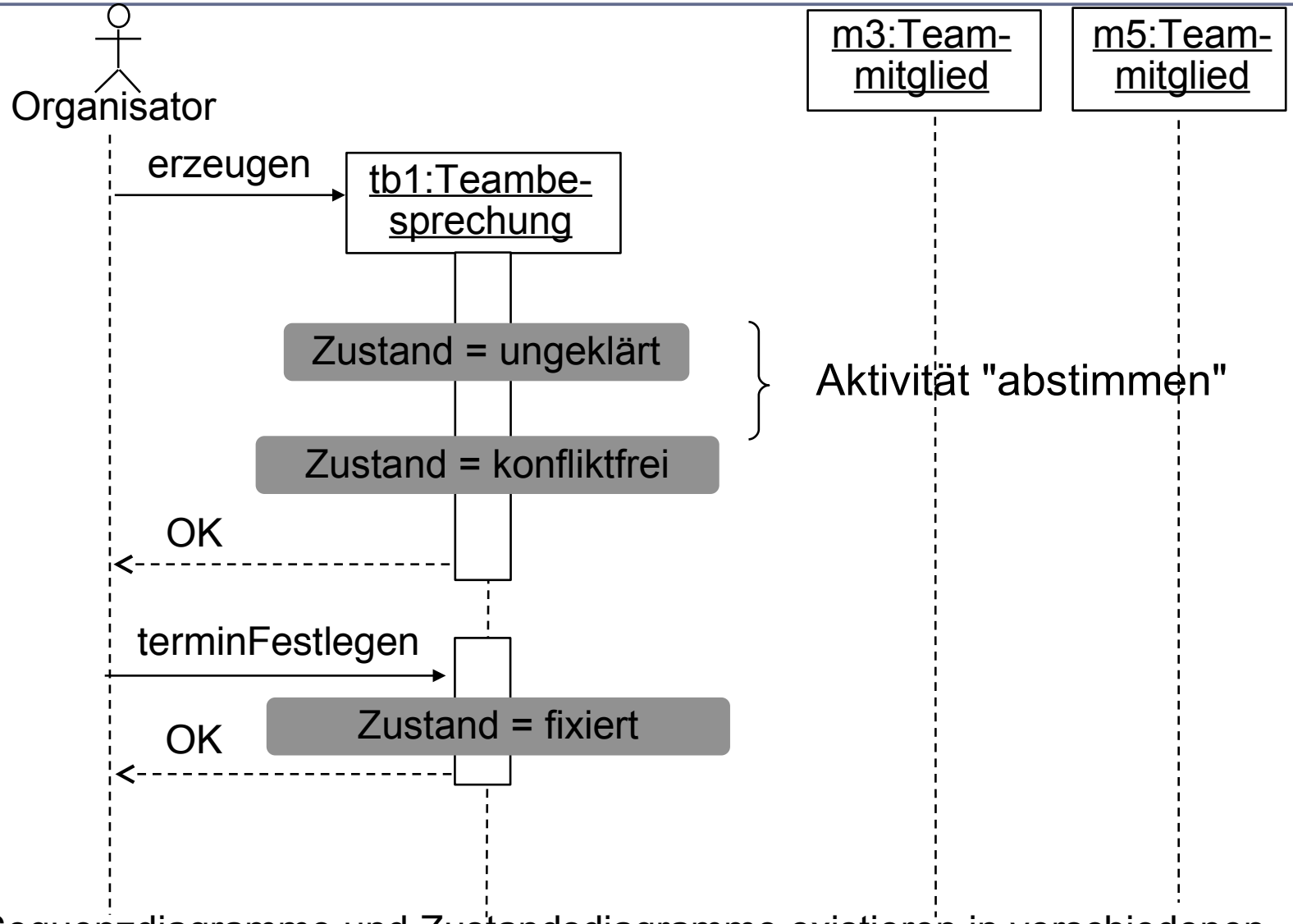
(Modellierbar als
Unterdiagramm *UD*,
d.h. **include** *UD*
anstelle von **do**)

Zusammenhang: Zustandsdiagramm - Sequenzdiagramm (1)



Jede in den Szenarien auftretende Reihenfolge von Aufrufen muß mit dem Zustandsmodell verträglich sein.

Zusammenhang: Zustandsdiagramm – Sequenzdiagramm (2)



Sequenzdiagramme und Zustandsdiagramme existieren in verschiedenen Abstraktionsstufen.

Zustandsmodellierung: Zusammenfassung

Typische Anwendung:	Analysephase	Entwurfsphase
Zeitbezogene Anwendungen (Echtzeit, Embedded, safety-critical)	**Verhaltens- und Steuerungsmaschinen** Skizzen der Steuerung für Teilsysteme und Gesamtsystem	Detaillierte Angaben zur Implem., automatische Codegenerierung, Verifikation mit Model checking
Datenbezogene Anwendungen (Informations-systeme, DB-Anwendungen)	**Protokollmaschinen** Lebenszyklen für zentrale Geschäftsobjekte, Geschäftsprozesse	Genaue Spezifikation von Aufrufreihenfolge (Vertragsprüfung)

The End

- Welche Arten von Aktionsdiagrammen gibt es?
- Wie verhalten sich Ober- und Unterzustände in einem Statechart zueinander?
- Was ist der Unterschied zwischen Steuerungs- und Protokollmaschine?
- Gibt es eine Möglichkeit, ein Statechart in ein Aktivitätsdiagramm zu wandeln?

- Many slides courtesy to © Prof. Dr. Heinrich Hussmann. Used by permission.