Objektorientierte Analyse
34b. 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 13-0.4, 14.06.13

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 für das Domänenmodell
   1. Strukturelle metamodellgetriebene Modellierung
   2. Modellierung von komplexen Objekten
      1. Modellierung von Hierarchien
      2. (Modellierung von komplexen Objekten und ihren Unterobjekten)
      3. Modellierung von Komponenten (Groß-Objekte)
   3. Strukturelle Modellierung für Kontextmodell und Top-Level-Architektur
3. Analyse von funktionalen Anforderungen (Verhaltensmodell)
   1. Funktionale Verfeinerung: Dynamische Modellierung von Lebenszyklen mit Aktionsdiagrammen (34)
   2. Funktionale querschneidende Verfeinerung: Szenarienanalyse mit Anwendungsfällen, Kollaborationen und Interaktionsdiagrammen
   3. (Funktionale querschneidende Verfeinerung für komplexe Objekte)
4. Beispiel Fallstudie EU-Rent

Punktweise und querschneidende dynamische Verfeinerung

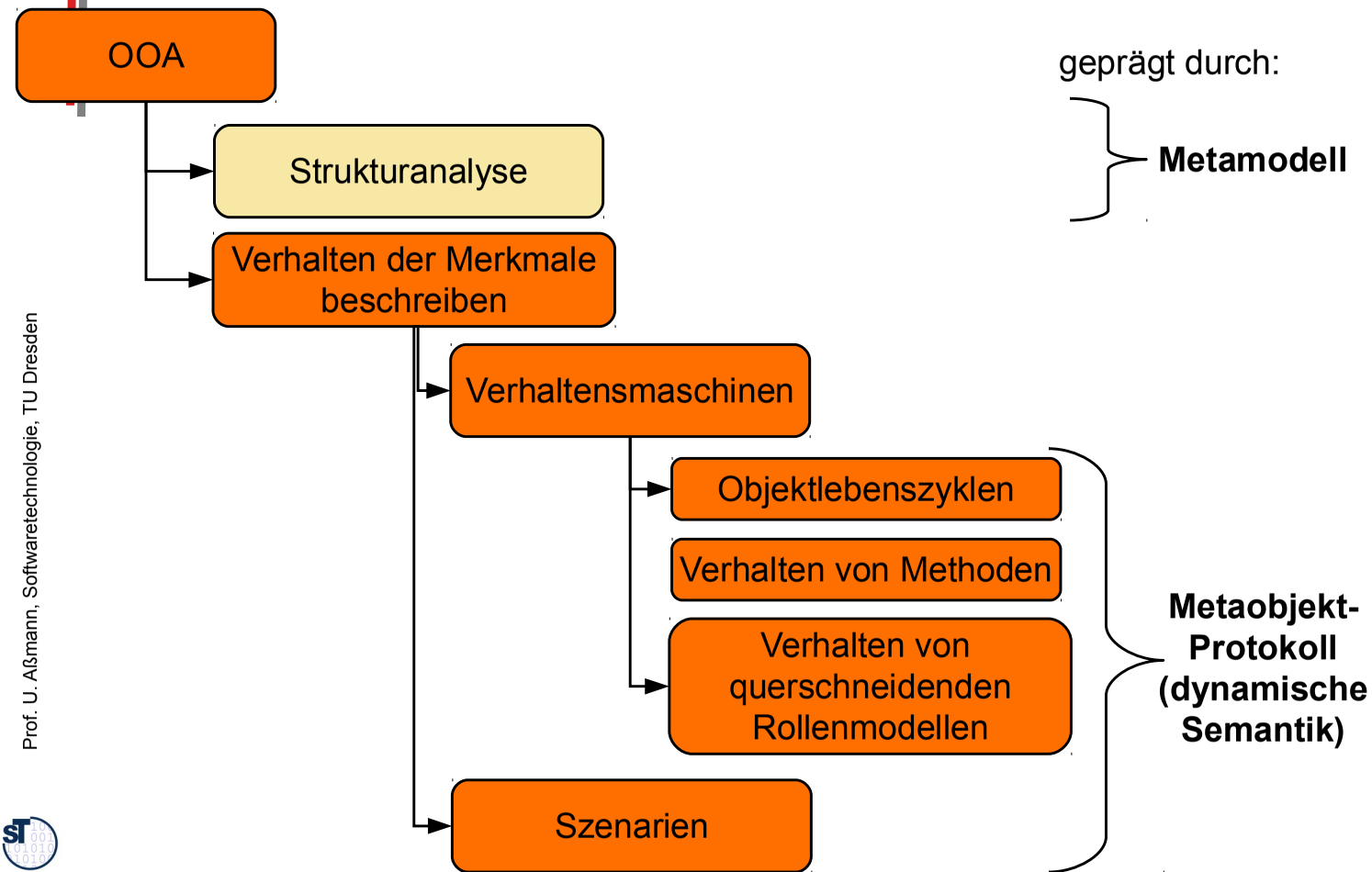
Punktweise funktionale Verfeinerung ist eine funktionale Verfeinerung eines Modellfragmentes (meist Objekt oder Methode), die *punktweise* geschieht, d.h. pro Modellfragment separat durchgeführt wird.

- Ergebnis:
  - **Lebenszyklus** des Objekts
  - **Implementierung** einer Methode

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

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

Schritte der objektorientierten, metamodellgetriebenen Analyse



Prof. U. Aßmann, Softwaretechnologie, TU Dresden

34.6 Einsatzzwecke von Zustandsdiagrammen

5

Zustandsdiagramme in der Analyse

6

- 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) 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 Objekte beschreiben und zur Ableitung von Vertragsprüfern eingesetzt werden (*black-box object life cycle)*

Einsatzzwecke für Zustandsmodelle

	Anwendungsfall-Lebenszyklus	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 (Transduktor):

- 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

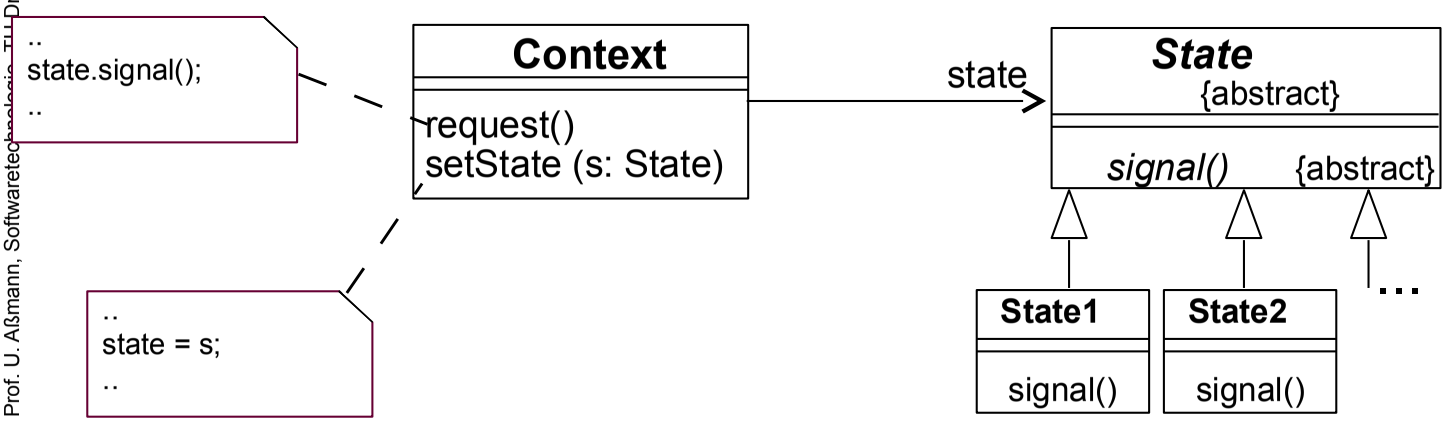
Protokoll-Maschinen (Akzeptoren, Prüfmaschinen):

- 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

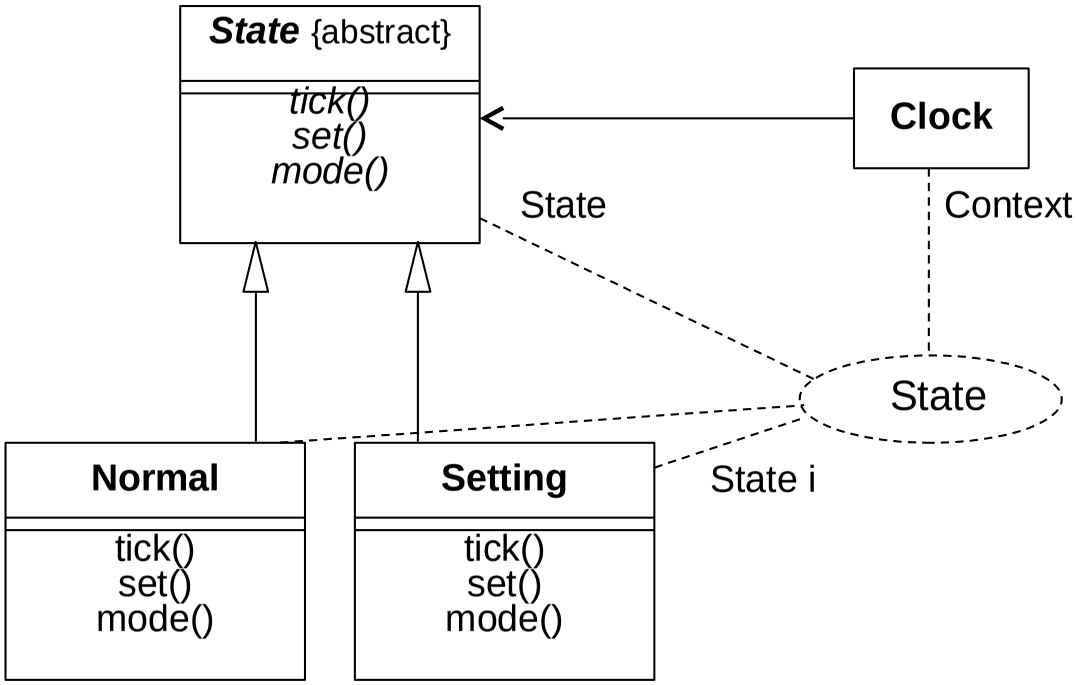
34.7 Entwurfsmuster State

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 *(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
}
```

Steuerungsmaschinen: Zusammenfassung

- Anwendungsgebiet:
  - white-box-Objektlebenszyklen
  - Gerätesteuerungen
    - 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

Prof. U. Aßmann, Softwaretechnologie, TU Dresden

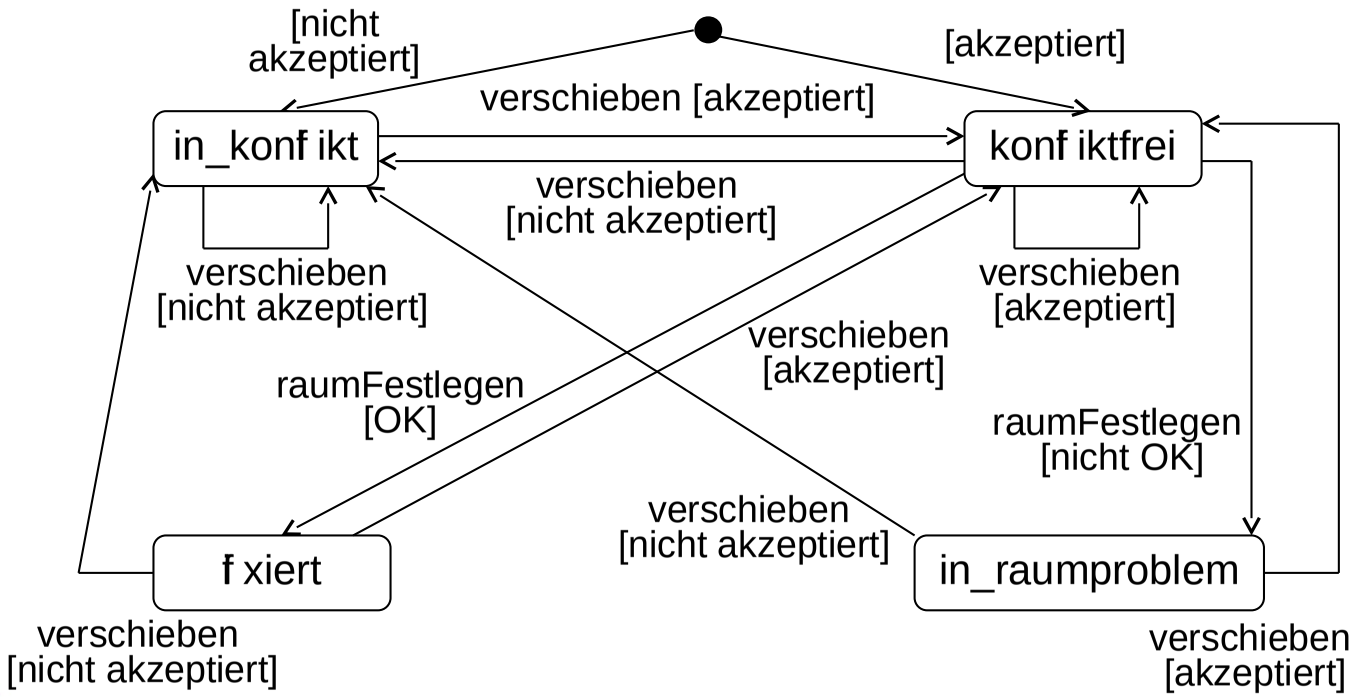
34.8 Vereinfachung von Zustandsdiagrammen durch Strukturierung

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!

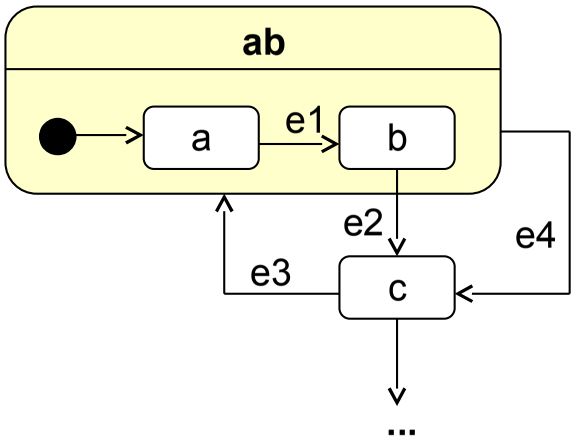
Black-Box Objektlebenszyklus (Protokollmaschine)

- Zulässige Zustände von Objekten der Klasse "Teambesprechung":
  - Merke: nur zur Generierung eines Vertragsprüfers einsetzbar, nicht zu einer vollständigen Implementierung

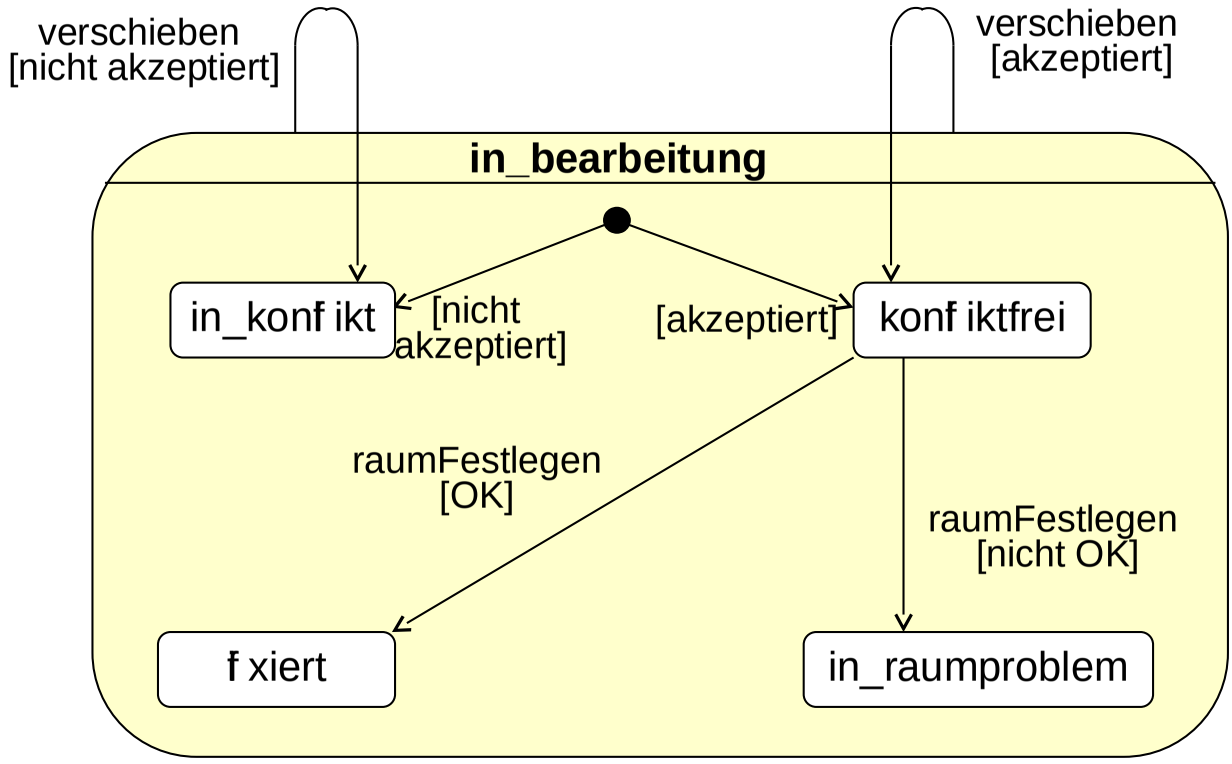


Ober- und Unterzustände

- 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).

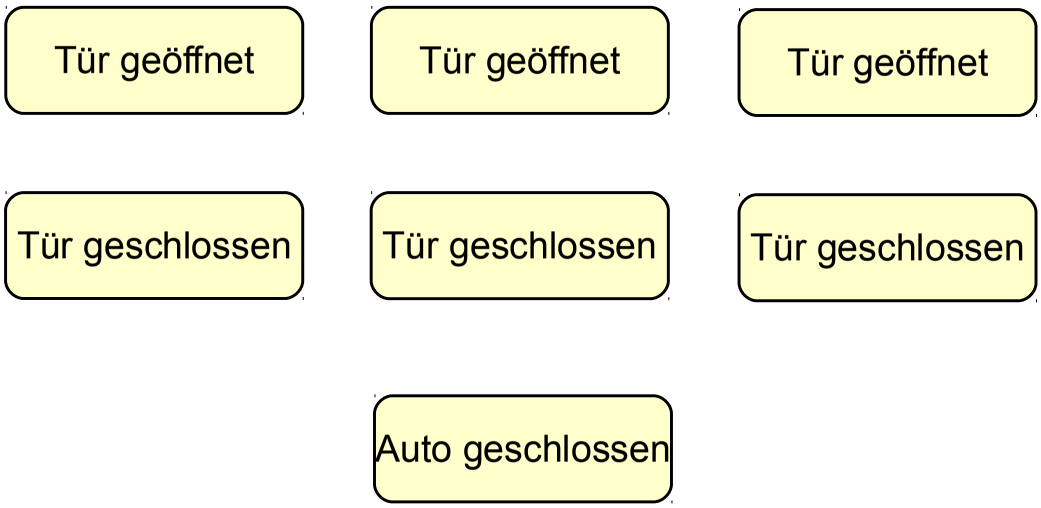


Zustandshierarchie Teambesprechung (jetzt einfacher notiert)



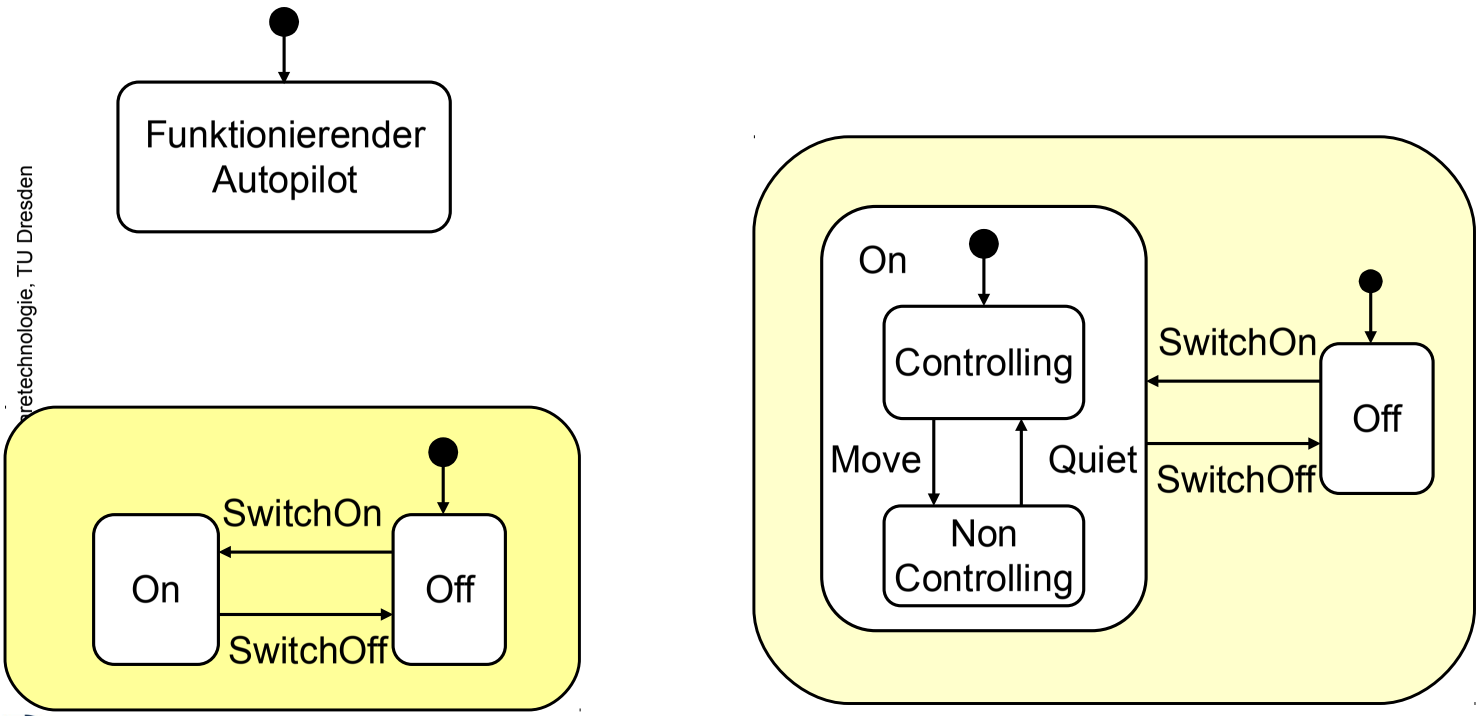
Türen eines Autos

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



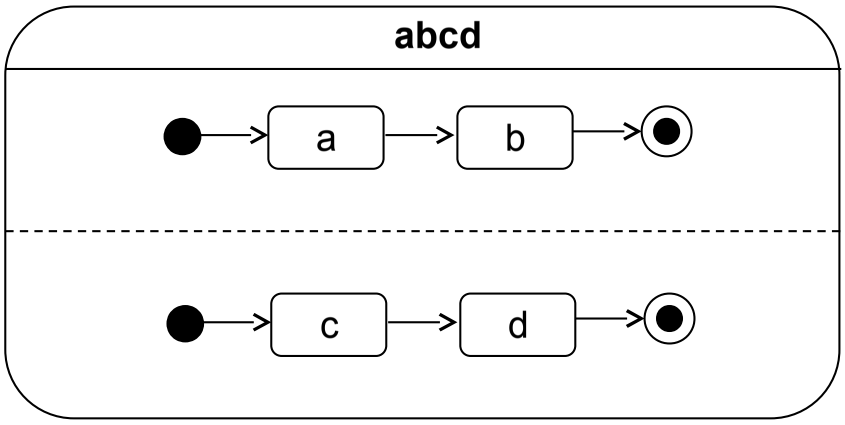
Warum kann man ein hierarchisches Zustandsdiagramm einfach verstehen?

- 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).



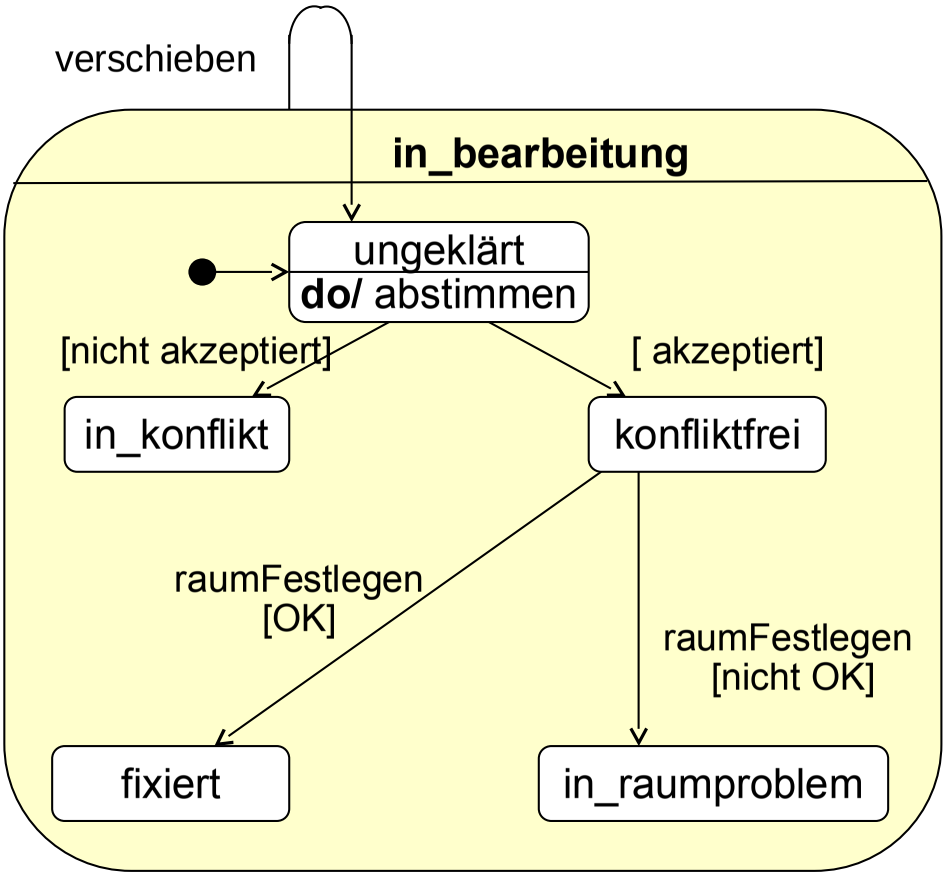
Interne Übergänge

- **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:**

Zustand
label */ interner Übergang*

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

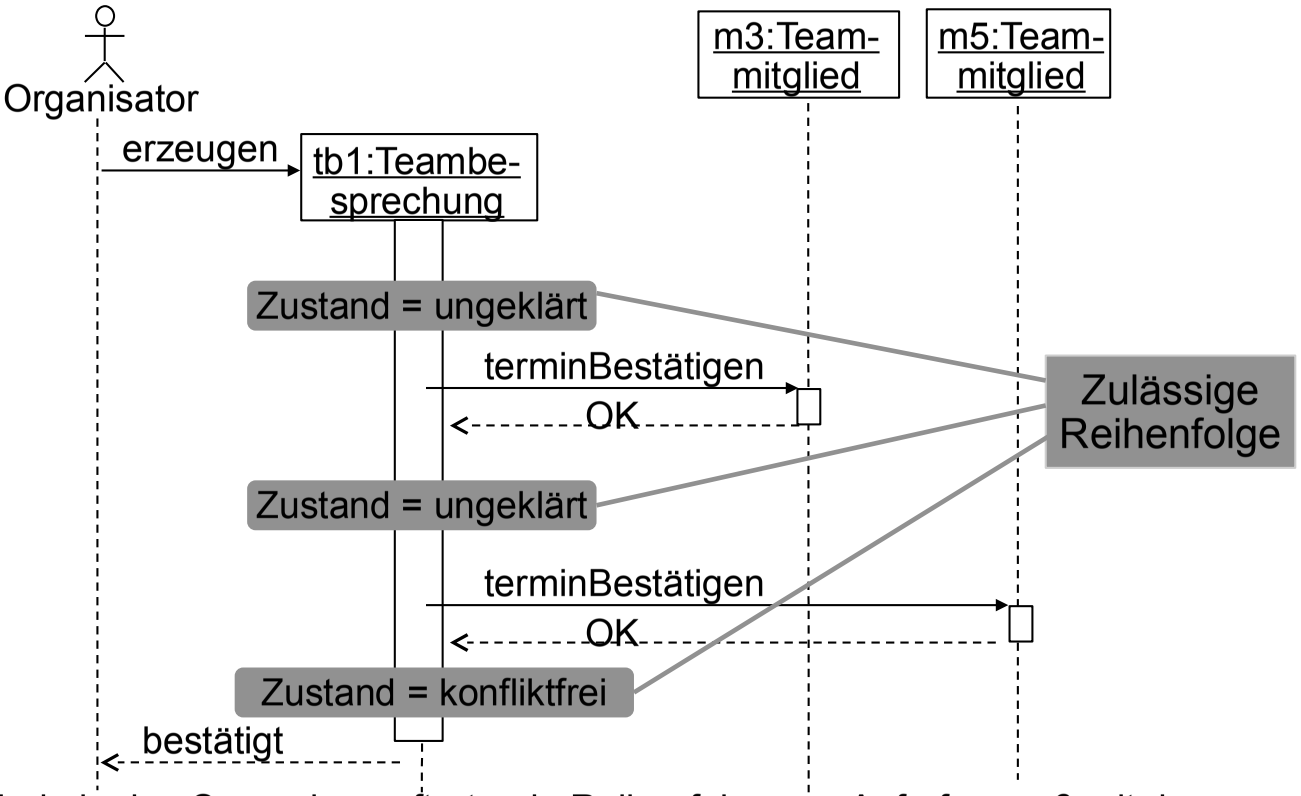
Verschiedene Aktivitäten



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

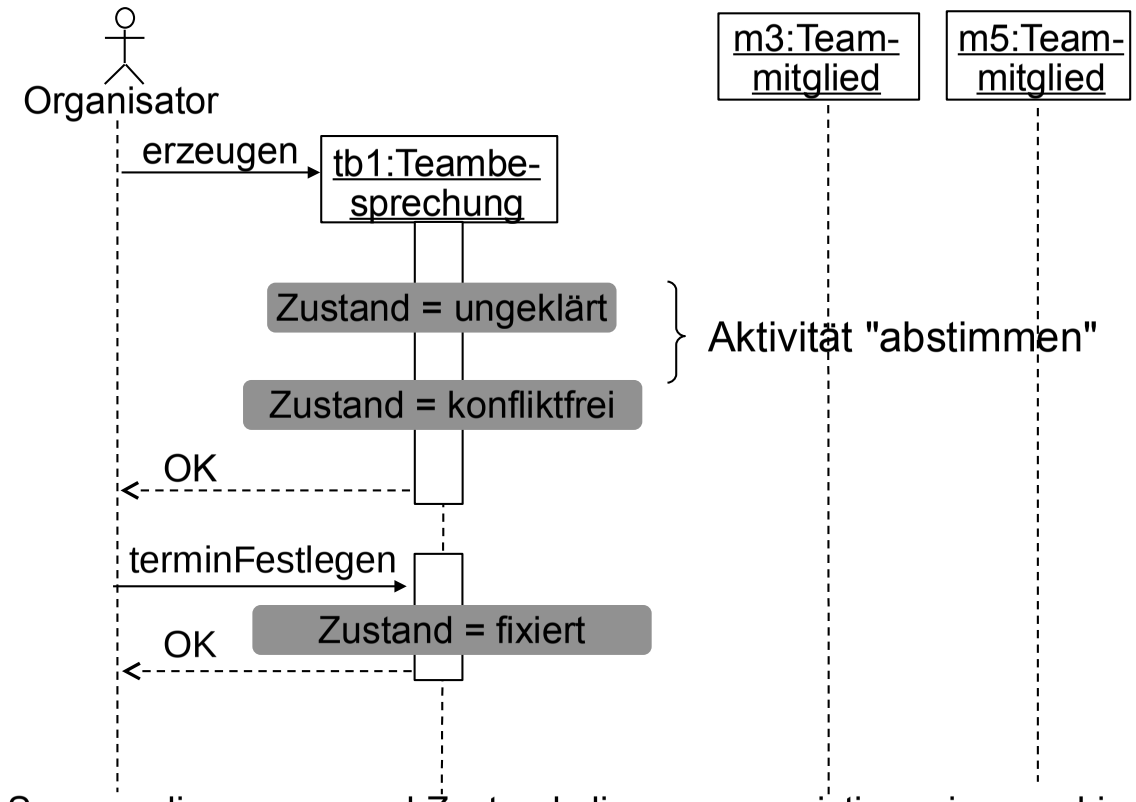
(Modellierbar als Unterdiagramm *UD*, d.h. **include** *UD* anstelle von **do**)

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

- Many slides courtesy to © Prof. Dr. Heinrich Hussmann, 2003. Used by permission.
- Typische Zustandsmaschinen:
  - Stellverhalten von Uhren
  - Autotüren und -heckklappen
  - Geldautomaten
  - Bahnkarten-Verkaufsautomat
  - Fahrstühle [Jazayeri]