TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik | Institut Systemarchitektur | Professur Rechnernetze

WS 2012
LV Informatik-I für Verkehrsingenieure

4. Softwaretechnologie

Dr. rer.nat. D. Gütter

Mail: Dietbert.Guetter@tu-dresden.de
WWW: wwwpub.zih.tu-dresden.de/~guetter/

Software - Probleme

ausgelöst durch Softwarefehler

- 1994 Flughafen Denver, Gepäcksystem bricht zusammen
  Schaden: 16 Monate Verzögerung bei Flughafeneröffnung
- 1995 Stellwerk Bahnhof Hamburg Altona
  falsch dimensionerter Kellerspeicher (Stack)
  1 Woche Verkehrs-Chaos im gesamten Bundesgebiet
- 1996 Rakete Ariane 5
  16-bit-Integer-Zahl; Überlauf (auch im Reservecomputer)
  Schaden: 500 Millionen $
- 2003 3 Jahre Probleme mit GPS-gestütztem Lkw-Maut-System
  Schaden: mehrere 100 Millionen Euro
- 6.5.2011 „Schwarzer Donnerstag 2.0"
  Computerhandel an Wall Street → Crash
  (zeitweiser) Verlust an Börsenwert: 1 Billion Euro
- 1.7.2012 „Schaltsekunde" verursacht Abstürze in Datenbanken
  → z.B. 50 Flugausfälle bei Fluggesellschaft Quantas
- …

Softwareentwicklung

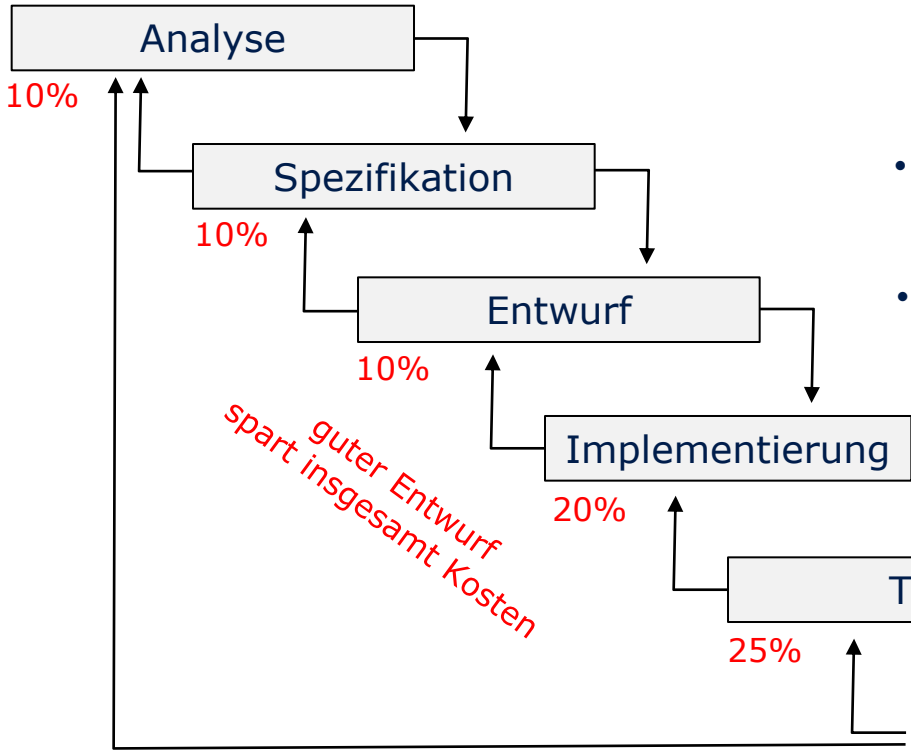
“im Kleinen”

- wenige Entwickler, kurze Entwicklungszeit (< 1 Mannjahr)
- Probleme überschaubar, Schwerpunkt: Programmierung, Testung

“im Großen”

- hoher Aufwand (u.U. > 1000 Mannjahre)
- nur ein Teil der Arbeiten: Programmierung, Testung
- im Vordergrund: Problemanalyse, Softwareentwurf, …
  - Intelligente Unterteilung des Projektes in Bausteine (relative Autonomie, gut durchdachte Schnittstellen, effiziente und sichere Testungsmöglichkeiten, mögliche Nachnutzbarkeit in anderen Projekten, …)
  - Absichern der Wartbarkeit und Weiterentwicklungsfähigkeit
  - hohe Produktivität der Projektbeteiligten (Einsatz effizienter Entwicklungsmethoden und –werkzeuge)

Softwarelebenszyklus



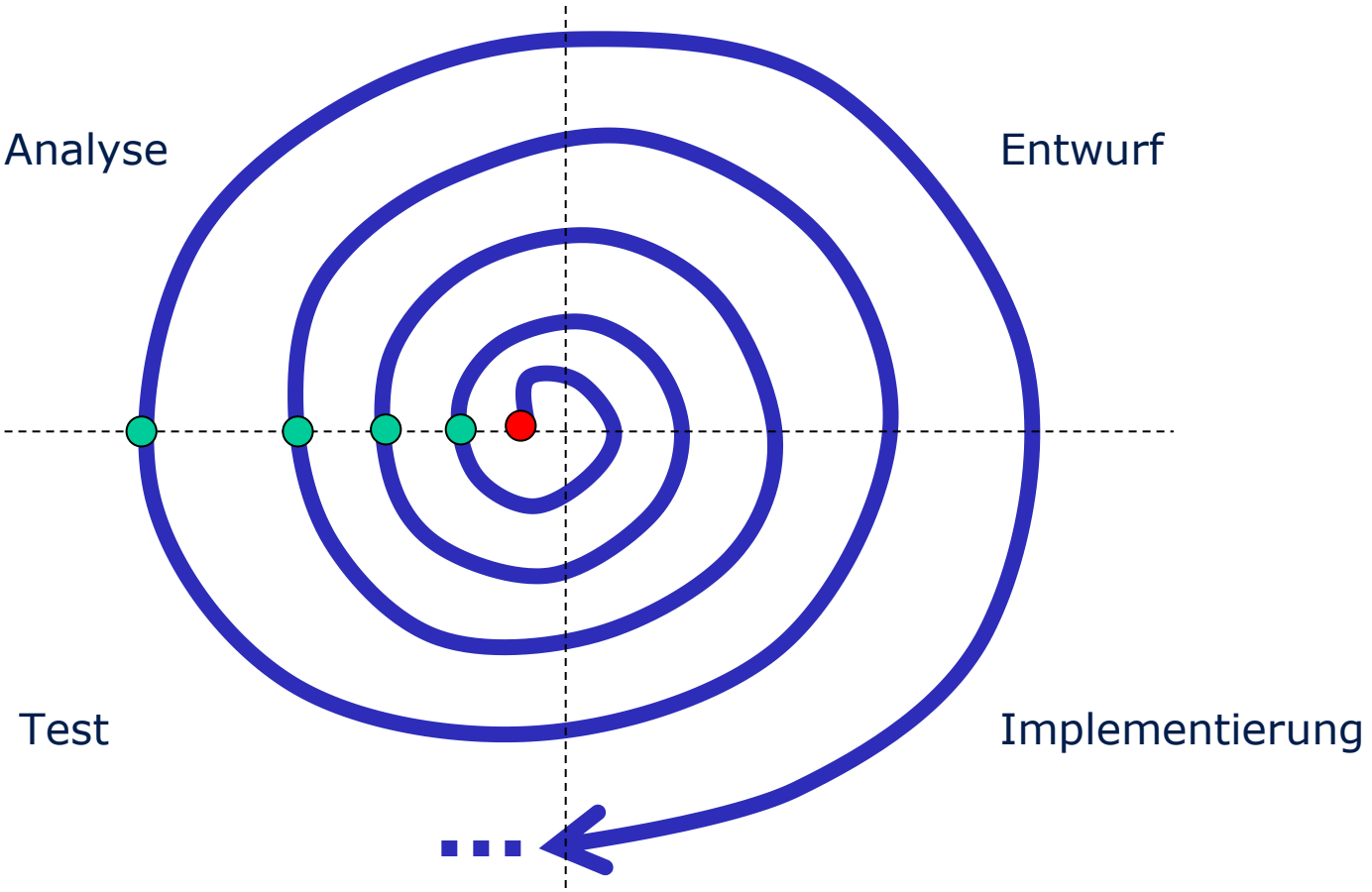
- Voruntersuchungen
  Wirtschftlichkeitsberechnungen
  Freigabe
- Festlegung der wesentlichen Anforderungen/Leistungsparameter
- Schrittweise Verfeinerung der Spezifikation bis in alle Details

- Modifikationen

- Daten sichern / Dokumentation

Spiralmodell

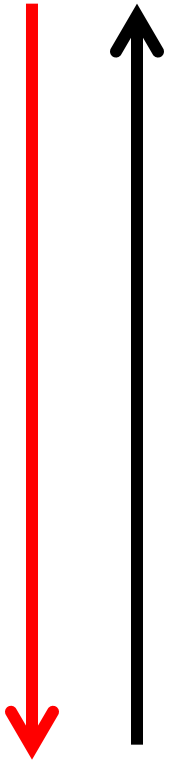
Projekt wird oft in mehreren **Generationen** überarbeitet.



Entwurfsstrategie

“top down” - Entwurf

1. Schrittweise Verfeinerung
2. Unterteilung in Subsysteme und Bausteine
3. Festlegen der Funktionalität
4. Festlegen von Schnittstellen



4. Testung Gesamtsystem
3. Testung von Subsystemen
2. Testung Zusammenwirken von Nachbarbausteinen
1. interne Testung der Bausteine

“buttom up” - Integration

Projektmanagement

Projektplanung

- Vorgaben zur Projektrealisierung (Ziel, Zeit, Budget)
- Risikoanalyse
- Grobplanung über Gesamtprojekt (Workflow)
- detaillierte Planung der nächsten Projektstufe

Projektorganisation

- Festlegung der Zuständigkeiten, Bildung von Arbeitsgruppen

Projektsteuerung/-kontrolle

- Kontrolle der Einhaltung der Projektziele, evtl. Korrekturen

Workflow

Aufbauorganisation

- Welche Mitarbeiter stehen zur Verfügung?
- Wer hat welche Qualifikationen?

Workflow-Management

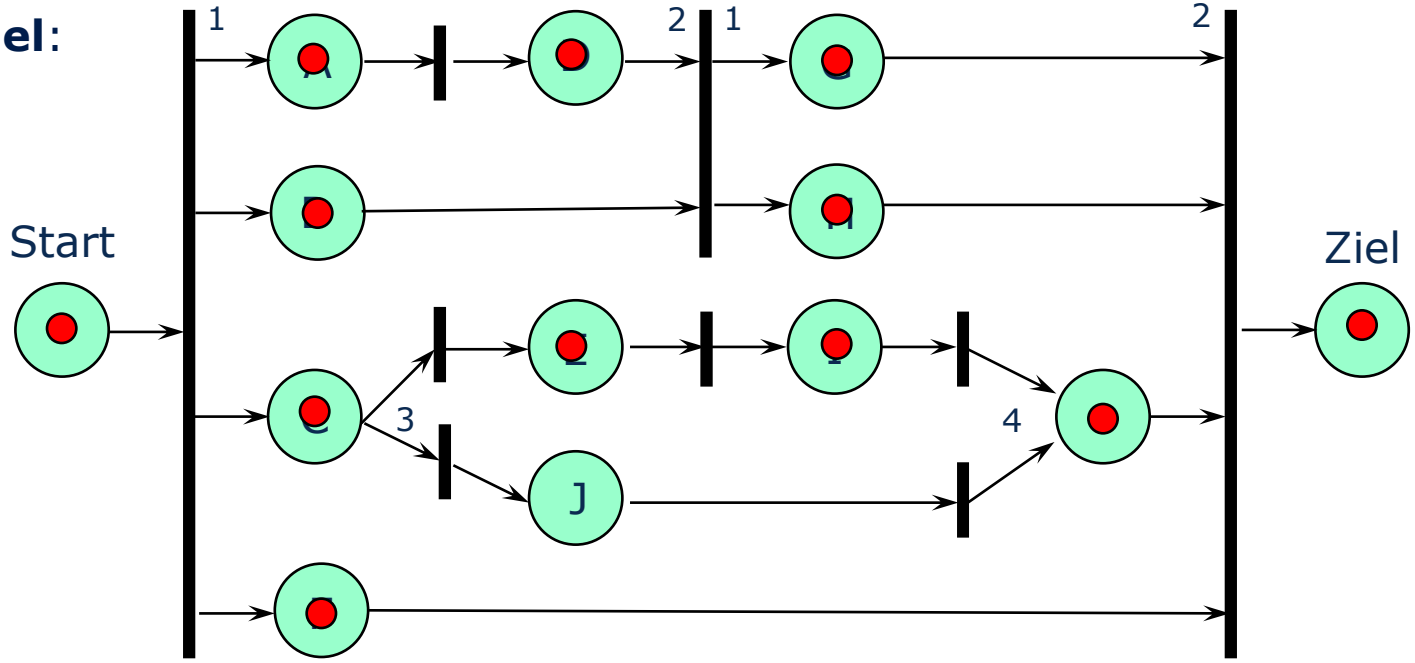
- Festlegung von Teilaufgaben
  und von Zeitschranken zur Bearbeitung der Aufgaben
- Zuordnung geeigneter Bearbeiter
  und weiterer Ressourcen (z.B. Rechentechnik)
- Einsatz geeigneter Workflow-Managementsysteme
- Darstellungsmethodik
  - Darstellung der logischen Abhängigkeiten der Teilaufgaben
    z.B. durch Petrinetze
  - Darstellung des Projektablaufes
    z.B. durch GANTT-Diagramme

Petrinetze

Graph: gerichtet, abwechselnd **Stellen** und **Transitionen**
Transitionen können nach festen Regeln "schalten"
dabei wandern **Marken** schrittweise von Start- zu Zielstelle.

In der Praxis gibt es viele Varianten von Petrinetzdarstellungen.

Beispiel:

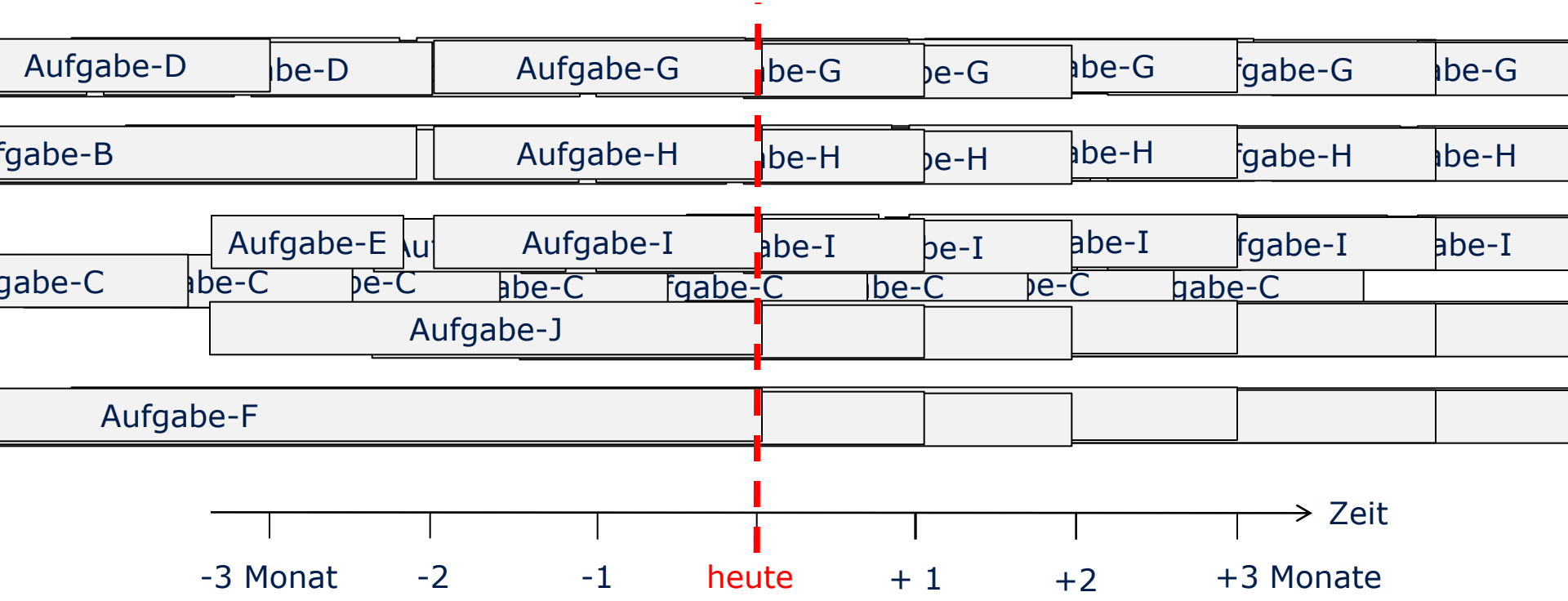


1/2- Parallelisierung/Synchronisation und 3/4- Verzweigung/Vereinigung

GANTT Diagramme

stellen anschaulich den Projektablauf in grafischer Form dar.

Zur Darstellung von Abhängigkeiten zwischen den einzelnen Teilschritten eignen sie sich weniger.



Paradigmen

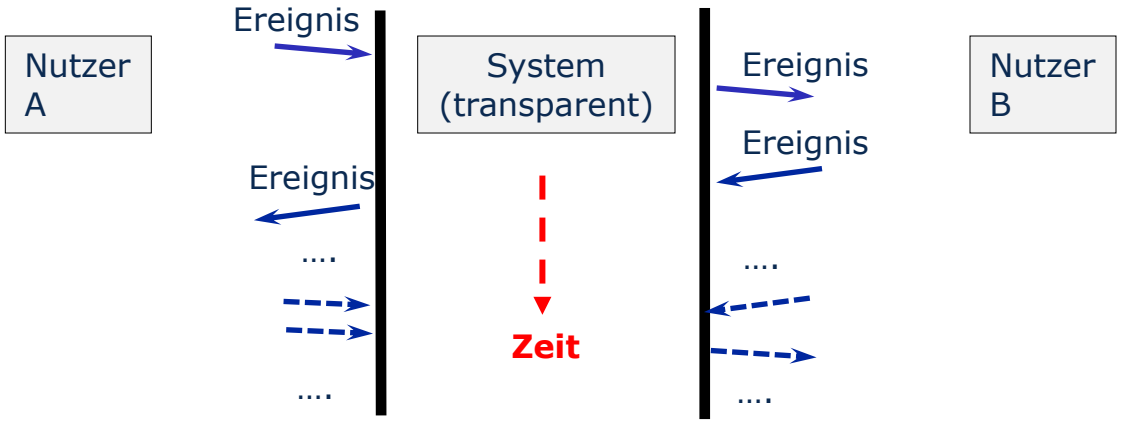
- Imperative / Funktionale / Logische Programmierung
  (s. Kap 3.1 "Programmierungstechnik")
- Datenmodellierung
  (s. Kap. 6.1 "Datenbanken")
- Ereignismodelle
- Zustandsmodelle
- Objektmodellierung — derzeit vorherrschende Methodik

  → **LV "Informatik-II"**

Ereignismodelle

beschreiben Systemverhalten aus Beobachtersicht.

Ablaufdiagramm
(Notation des zeitlichen Ablaufes von Interaktionen)



Vorteil: Anschauliche Darstellung des zeitlichen Systemverhaltens

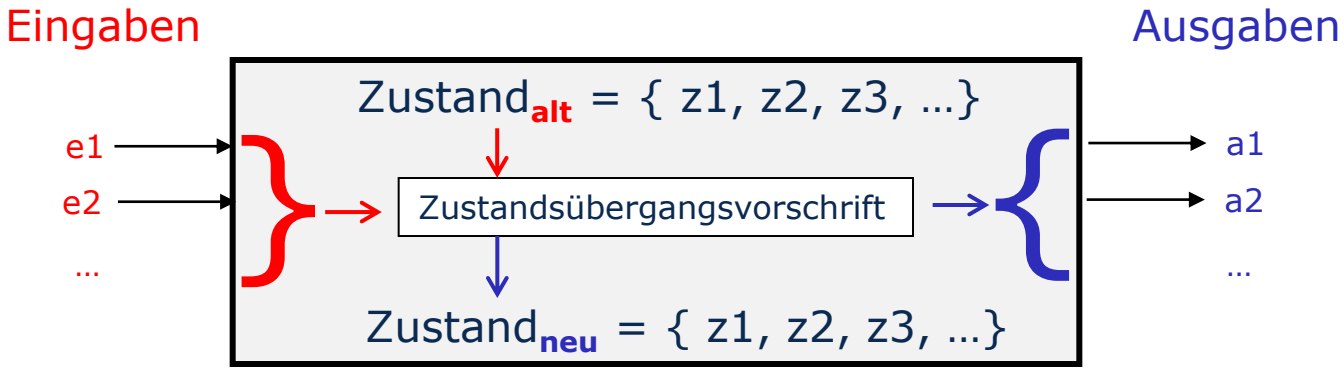
Nachteil: alternative Abläufe nicht in einem Diagramm darstellbar
schlecht geeignet als Programmiervorlage

Zustandsmodelle

beschreiben Systemverhalten auf Basis der Automatentheorie.

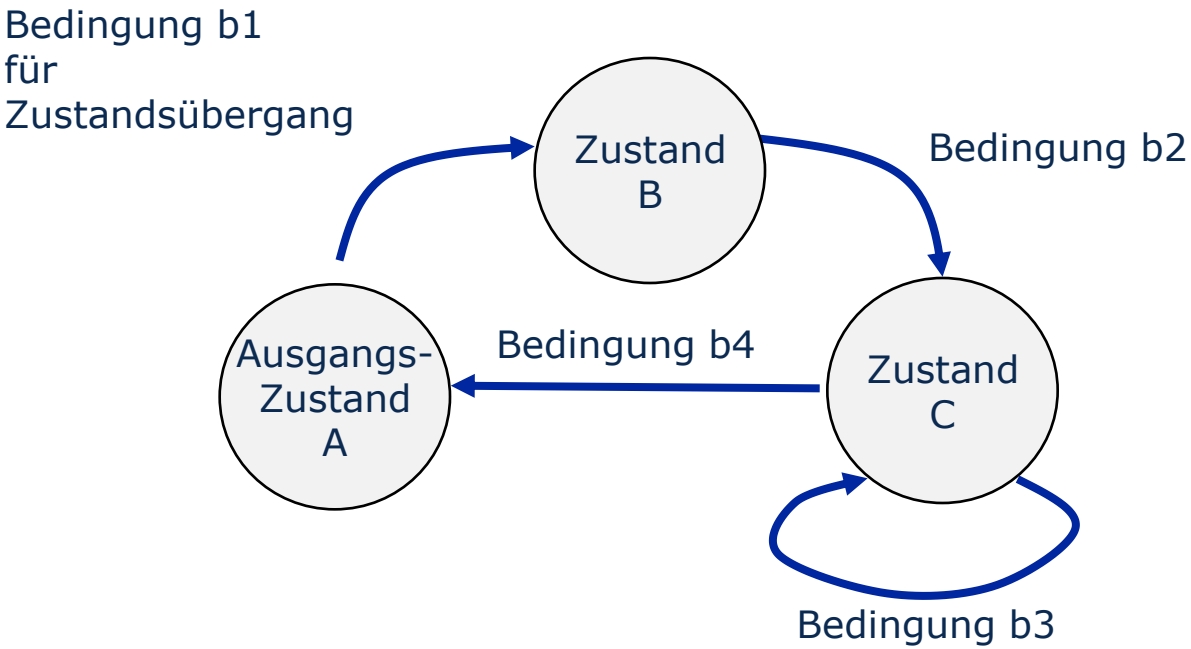
(endliche) **Automaten**

- arbeiten taktweise,
- besitzen Systemzustand (Repräsentation Systemvergangenheit),
- erhalten in jedem Arbeitstakt Eingabeinformationen
  und berechnen aus aktuellen Eingabe und Zustandsinformationen
  einen aktualisierten Systemzustand und Ausgabeinformationen.



Zustandsmodelle (2)

Zustandsdiagramm (Notation der möglichen Systemveränderungen)



Vorteil: vollständige Darstellung des Systemverhaltens in einem Diagramm
Implementierung kann aus Diagramm abgeleitet werden.

Nachteil: unübersichtlich, zeitliche Aufeinanderfolge kaum sichtbar

UML – (Unified Modeling Language)

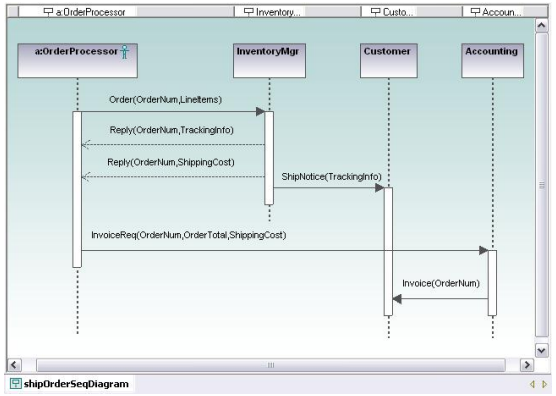
Grafische Modellierungssprache für Softwareentwurf

- sehr verbreitet
- 1997 standardisiert durch OMG (Object Mangement Group) und ISO (International Organization for Standardization)

UML unterstützt

- den objektorientierten Entwurf
- 7 Typen von Strukturdiagrammen
  7 Typen von Verhaltensdiagrammen

Sequenzdiagramm



vielfältige Toolunterstützung
(www.jeckle.de/umltools.htm)

Demo-Flash
(www.altova.com/videos.asp?type=0&video=introumodel)

UML – Diagramme

14 Typen