Modellierung und Analyse eingebetteter und verteilter Systeme

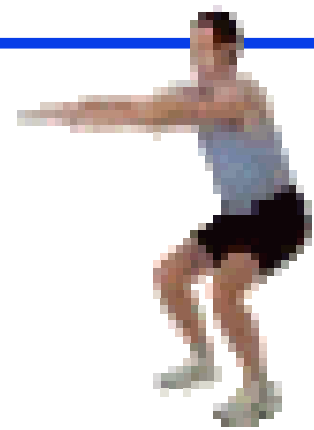
Prof. Dr. Peter Buchholz
Prof. Dr. Heiko Krumm

Basismodul in den Masterstudiengängen Informatik und Angewandte Informatik
für den Forschungsbereich: Eingebettete und verteilte Systeme

Vertiefungsvorlesung gemäß DPOs

Übungen

- Wer
- Wann
- Was



Literatur

- Folien werden in‘s WWW gestellt.
- Literaturverweise zu einzelnen Themen finden sich auf den jeweiligen Folienabschnitten
- Breitere Themenbereiche sind in folgenden Büchern und Buchteilen behandelt:
  - L. Lamport, N. Lynch; Distributed computing: Models and methods; in: Jan van Leeuwen (ed.), Formal Models and Semantics, volume B of Handbook of Theoretical Computer Science, chapt. 18, pgs. 1157-1199; Elsevier Science Publ. and MIT Press, 1990.
  - Z. Manna, A. Pnueli; The temporal logic of reactive and concurrent systems; Springer, Heidelberg, 1991.
  - Robin Milner; Communicating and Mobile Systems: the Pi-Calculus, Cambridge University Press 1999
  - M. Huth and M. Ryan. Logic in Computer Science. Modelling and reasoning about systems. Cambridge University Press, 2000.
  - C. G. Cassandras, S. Lafortune. Introduction to Discrete Event Systems. Springer 2008.
  - R. Jain. The Art of Computer Systems Performance Analysis. Wiley 1991.

Prüfungen

- Je nach Teilnehmeranzahl (Entscheidung nächste Woche):
  - Klausur
  - Mündliche Prüfung
- Leistungsnachweise (Diplomstudiengänge)
  - Übungsschein

Eingebettete Systeme

- ***Universalcomputer***
  Mit breiterem Anwendungsspektrum einsetzbares Computersystem (z.B. PC, Großrechner)

- ***Dediziertes System***
  Computersystem, das nur speziellem eng eingegrenztem Zweck dient (z.B. Router, Geldautomat-Steuerung)

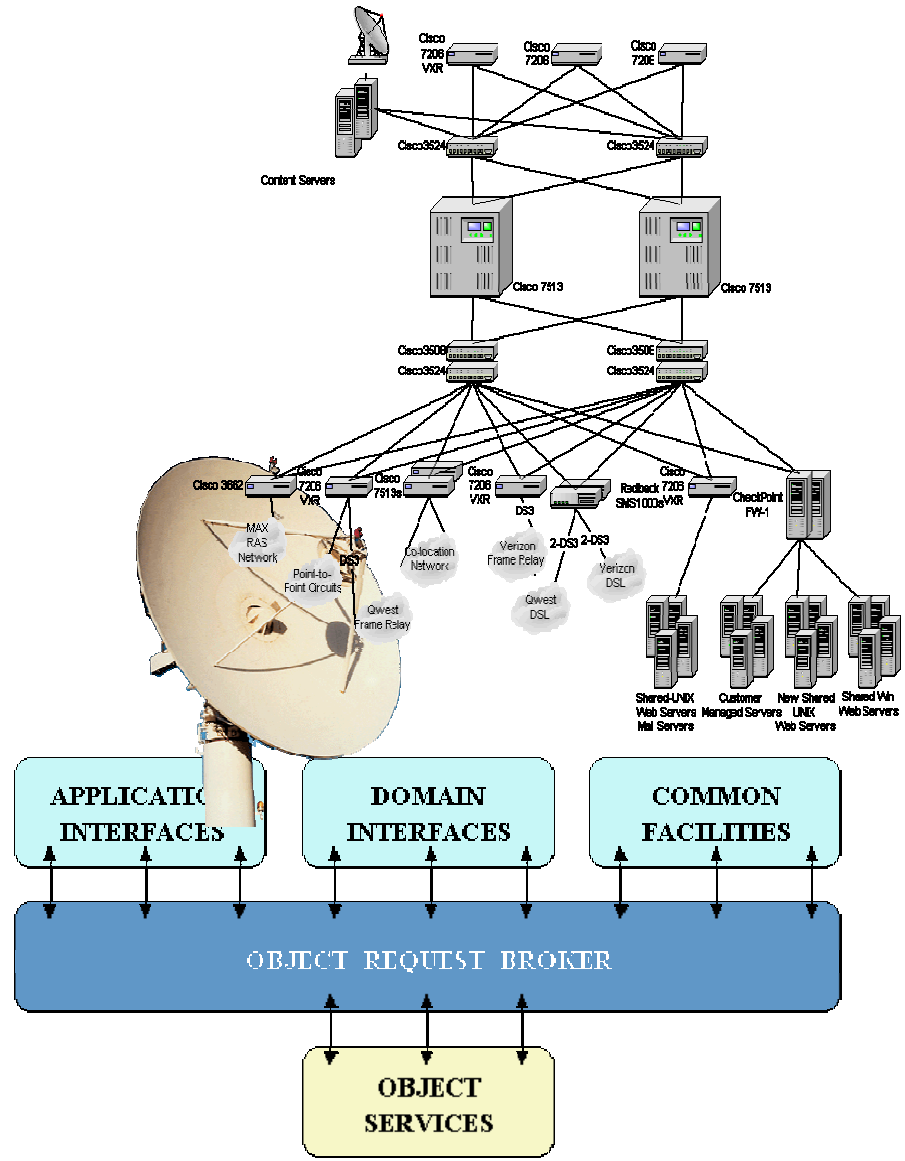
- ***Eingebettetes System***
  Computersystem, das integraler Teil eines Geräts ist (z.B. Steuerung einer Waschmaschine, Motorsteuerung im Auto)

Rechnernetze und verteilte Systeme

- ***Rechnernetz***
  Durch Telekommunikationssystem verbundene Rechnerknoten

- ***Telekommunikationssystem***
  System, das Teilnehmern Kommunikationsdienste anbietet (in der Regel selbst durch Rechnernetz implementiert).

- ***Verteiltes System***
  Anwendung, deren Komponenten sich an verschiedenen Orten befinden, Komponenten sind in Rechnernetz installiert, werden lokal von den Rechnerknoten ausgeführt und kommunizieren miteinander mit Hilfe eines Telekommunikationssystems.



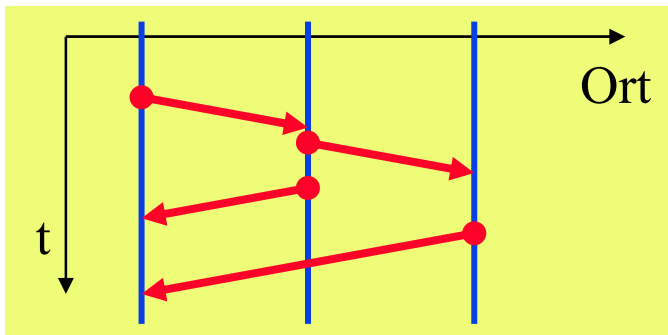
Systemarten

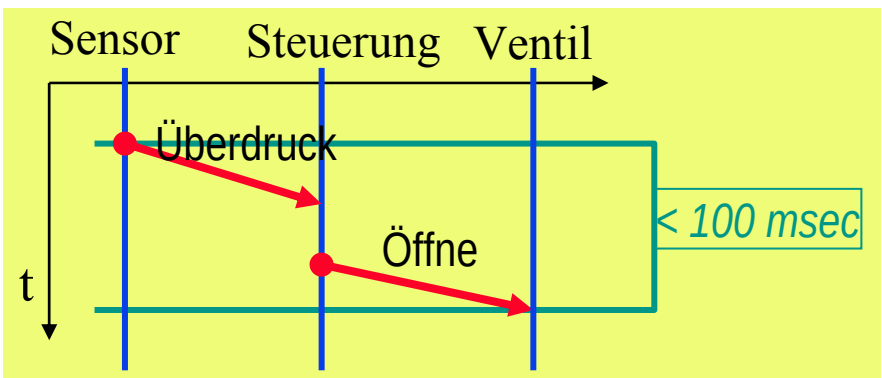
- ***Start/Stop-System***
  Das System wird mit einer Auftragsdefinition gestartet, führt die Aufgabe aus und terminiert mit einem Ergebnis (z.B. Compiler, Primzahlberechnung). Das System führt eine Berechnung aus.

- ***Reaktives / Interaktives System***
  Das System wird gestartet, hat u.U. eine Anlaufphase und ist ab dann bereit mit seiner Umgebung im Verlauf der Zeit zu interagieren. Das System besitzt ein Verhalten.

- ***Realzeitsystem***
  Reaktives System, an dessen Verhalten harte oder weiche Echtzeit-Bedingungen gestellt werden (minimale und maximale Reaktionszeiten, Reaktionen zu vorgegebenen absoluten Zeitpunkten).

{Vorbedingung}
Programm
{Nachbedingung}





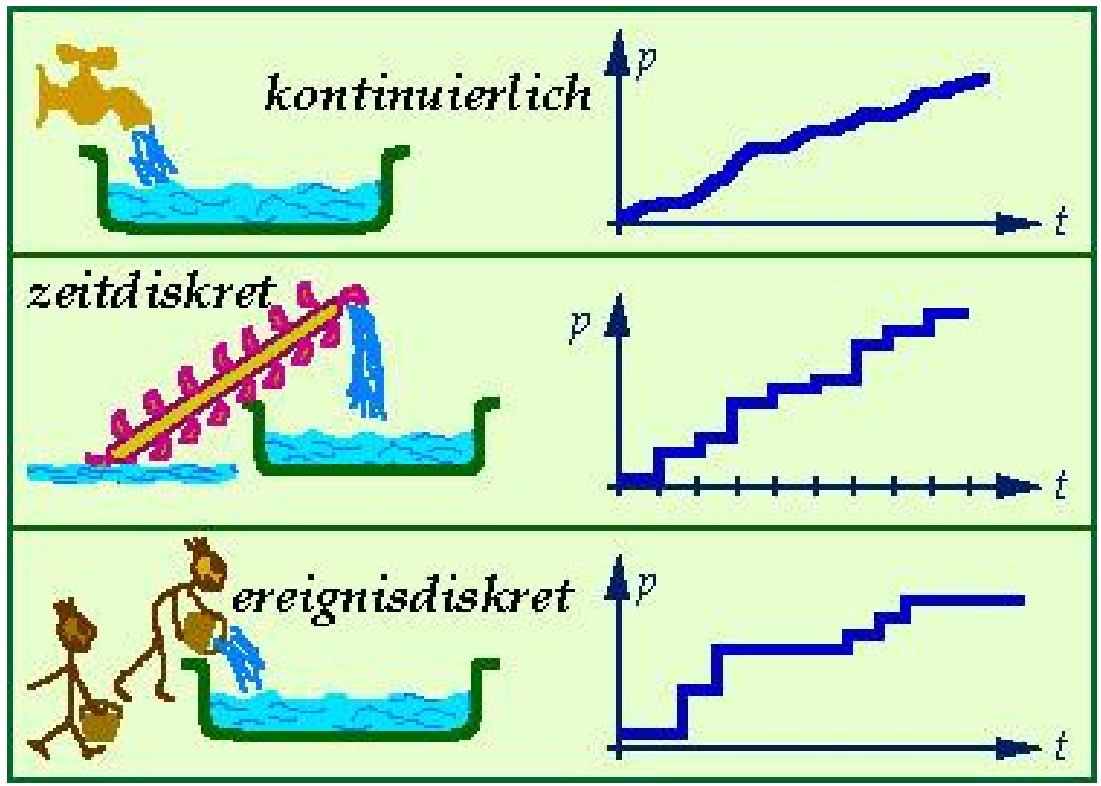
Dynamische Systeme

- ***Kontinuierliches System***
  Zustandsgrößen verändern sich kontinuierlich im Lauf der Zeit (z.B. Schwingkreis).

- ***Wertdiskretes System***
  Zustandsgrößen haben diskrete Wertebereiche.

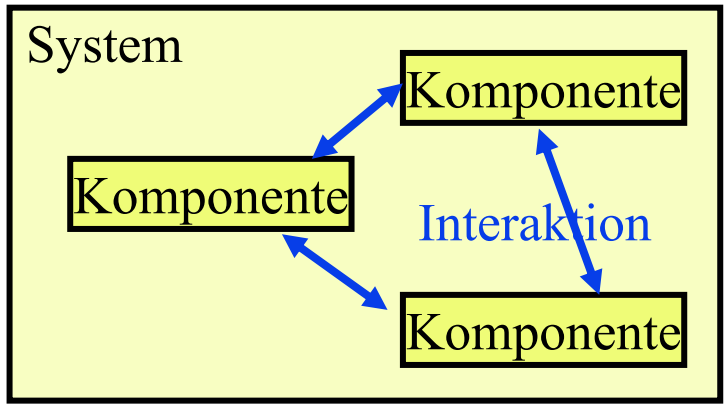
- ***Zeitdiskretes System***
  Zustandsgrößen ändern sich nur an einzelnen Zeitpunkten eines vorgegebenen Zeitrasters.

- ***Ereignisdiskretes System***
  Zustandsgrößen ändern sich nur, wenn Ereignisse auftreten.

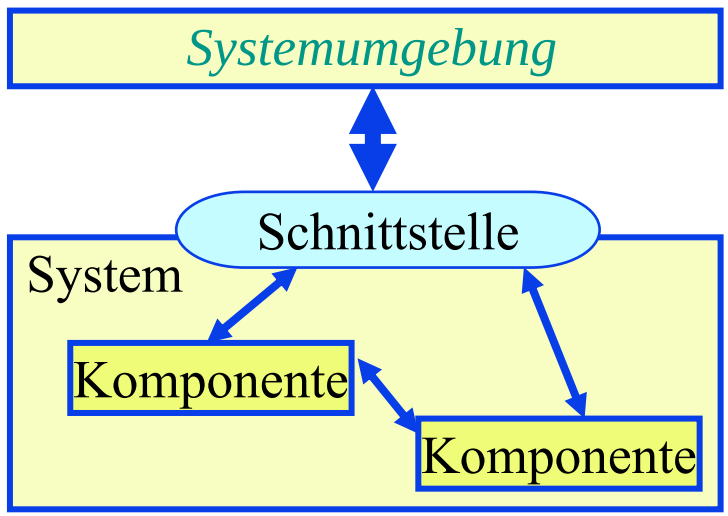


Systemstrukturen

- ***Geschlossenes System***
  Es treten keine relevanten Wechselwirkungen mit Komponenten auf, welche sich außerhalb des Systems befinden.

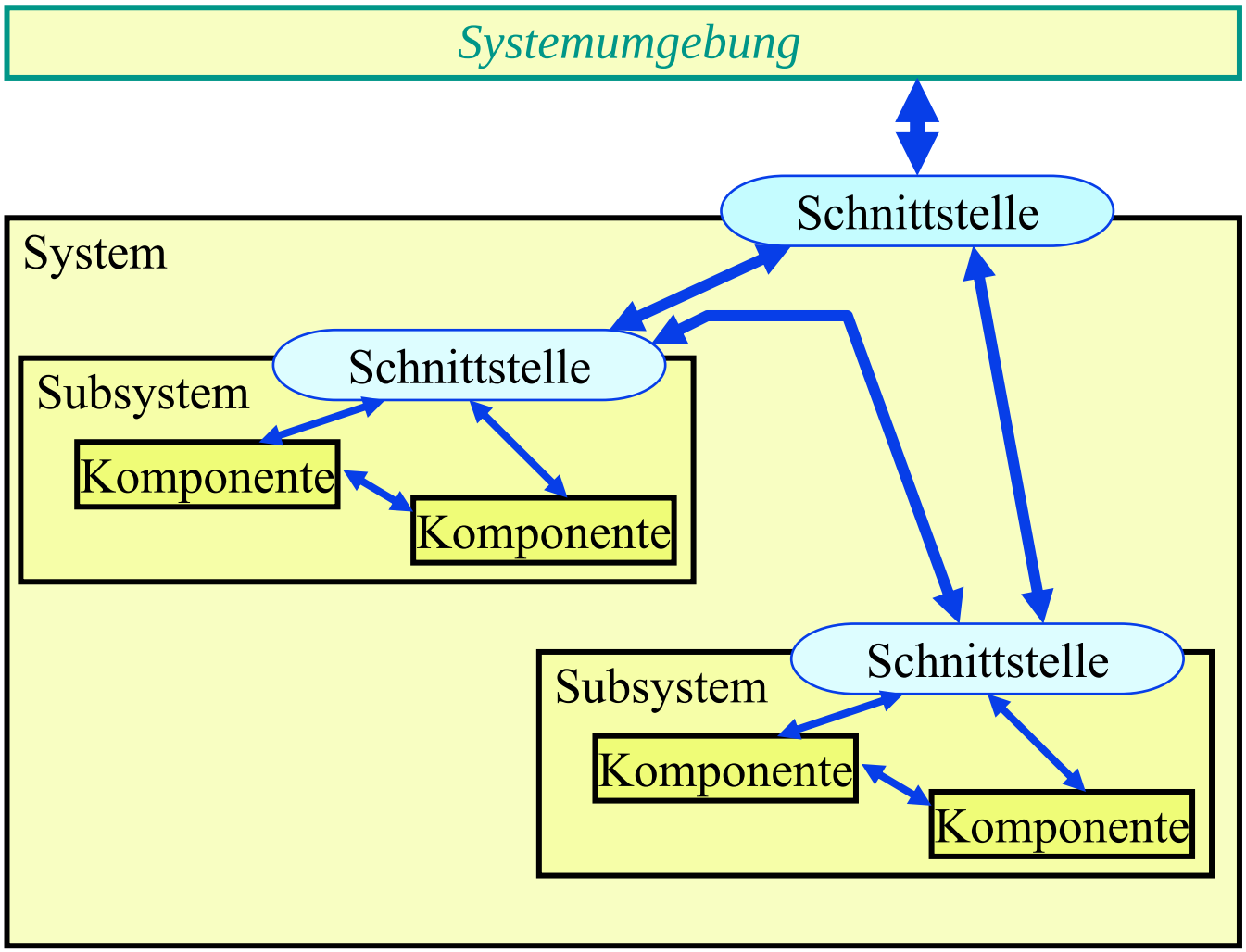


- ***Offenes System***
  Das System interagiert mit seiner Umgebung.



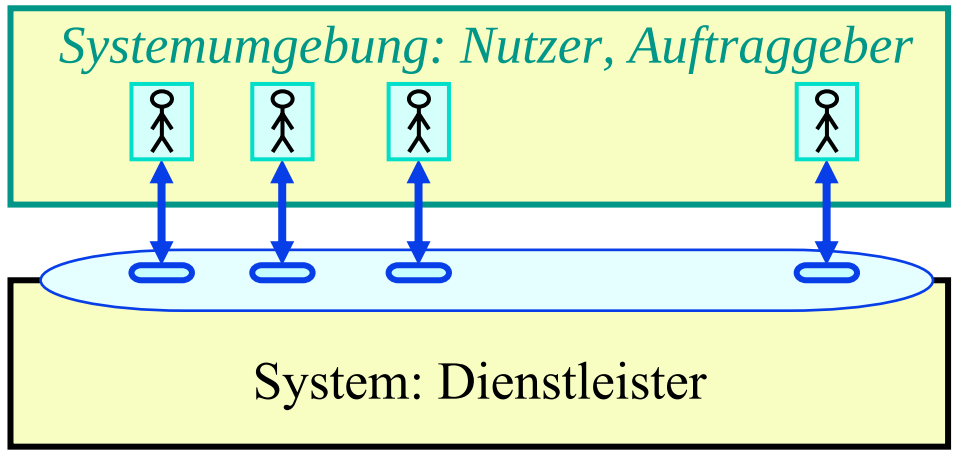
Systemstrukturen

- ***Hierarchisch strukturiertes System***
  Das System besteht aus Subsystemen.

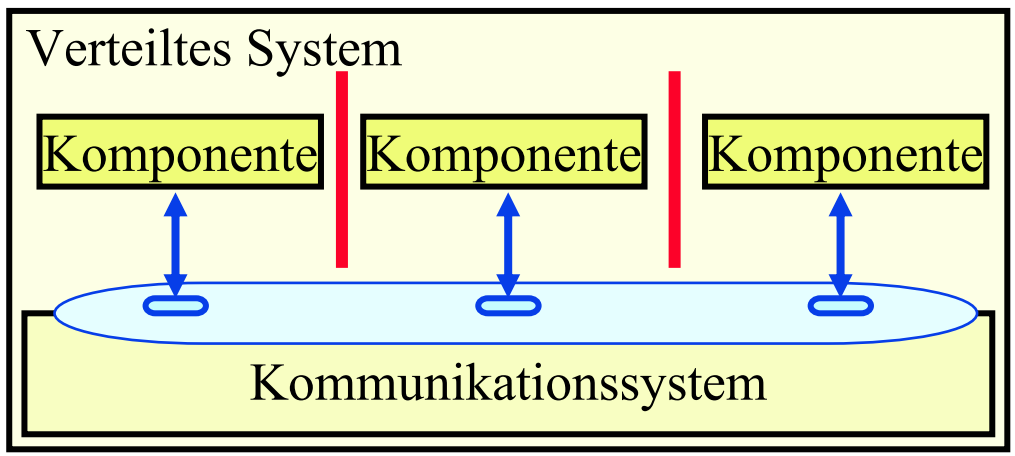


Wichtige Systemformen

- ***Dienstleistungssystem***
  Die Umgebung besteht aus Nutzern, welche dem System Aufträge geben.



- ***Verteiltes System***
  Die Komponenten des Systems interagieren – da voneinander entfernt – nicht direkt, sondern unter Nutzung eines Telekommunikations-Dienstleistungssystems.



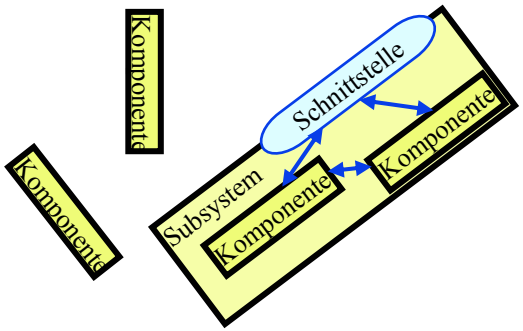
Systemaufbau

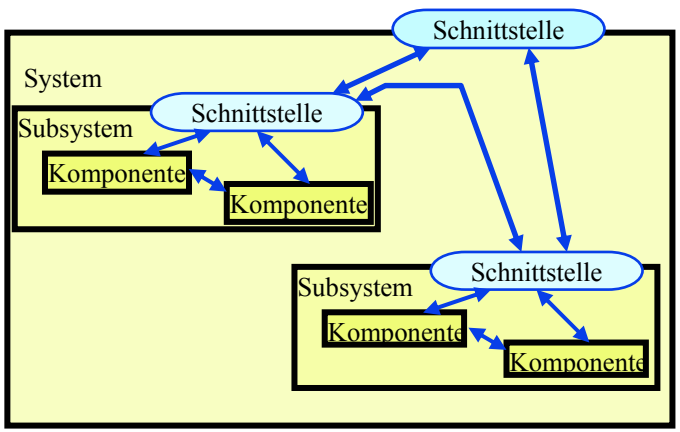
- Systeme bestehen aus Hardware, Systemsoftware und Software
  - Auf „natürliche“ Weise hierarchisch strukturiert
- Aufgaben für Informatiker/Informatikerinnen
  - Systementwurf, -planung, -realisierung
  - Betrieb und Wartung
  - Anpassung und Verbesserung
- Vorgehensweise
  - Üblicherweise Aufbau komplexer Systeme aus
    - » vorhandenen Komponenten
    - » angepassten Komponenten
    - » neu zu erstellenden Komponenten

    „Orchestrierung“ im SOA-Kontext

➢ Verhält mein System sich so, wie es soll?
Formaler: **Werden die geforderten Systemeigenschaften erreicht?**

Systemanalyse



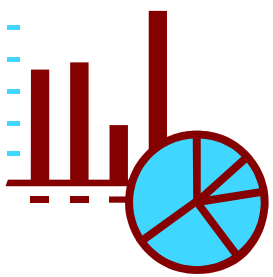


Systemanalyse möglichst entwurfs- und realisierungebegleitend

⇒ **modellgestützter Entwurf**
modellgestützte Realisierung

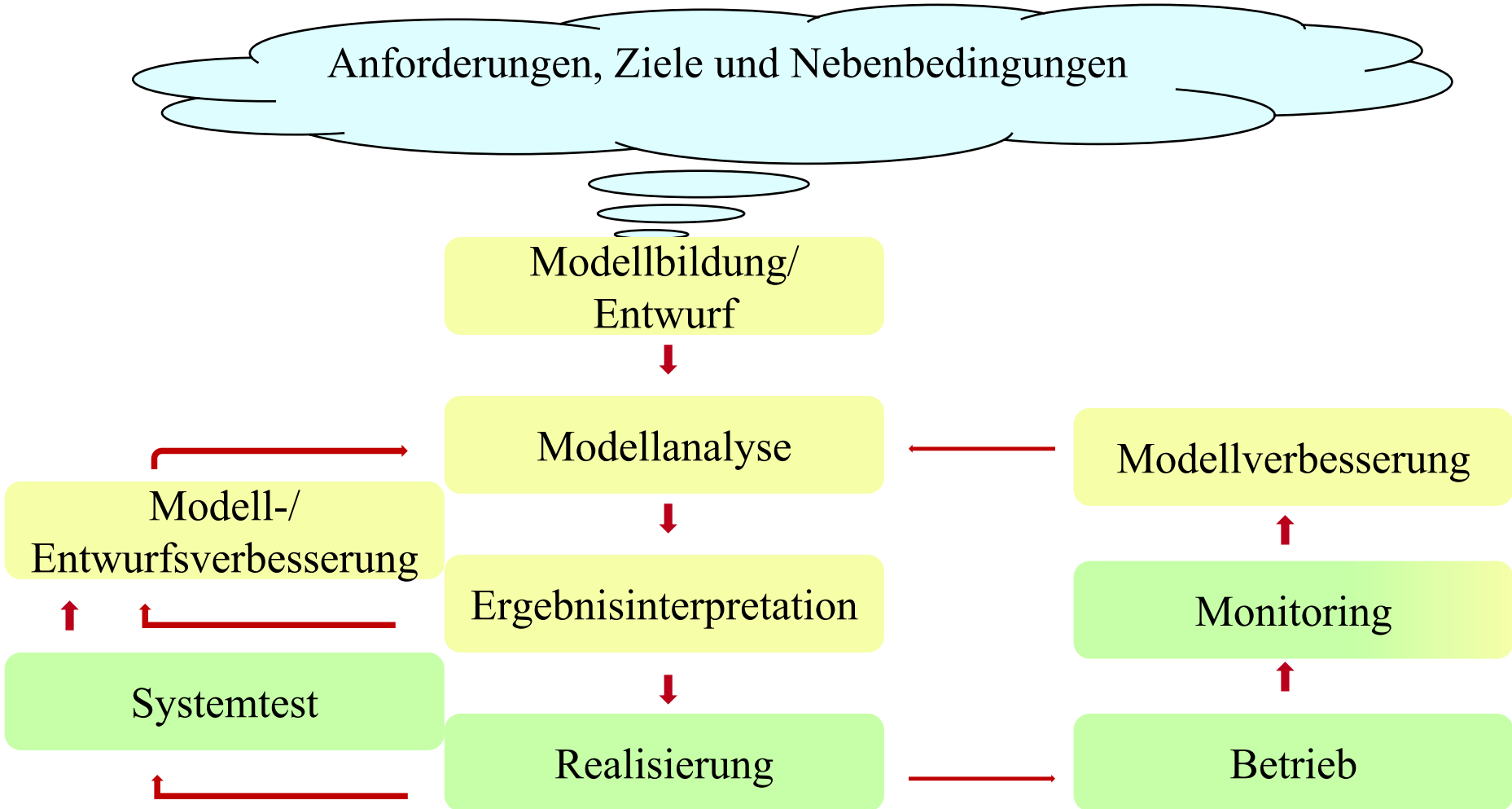
Kosten der Fehlerbeseitigung und Systemverbesserung steigen überproportional mit fortschreitender Systementwicklung

Analyse

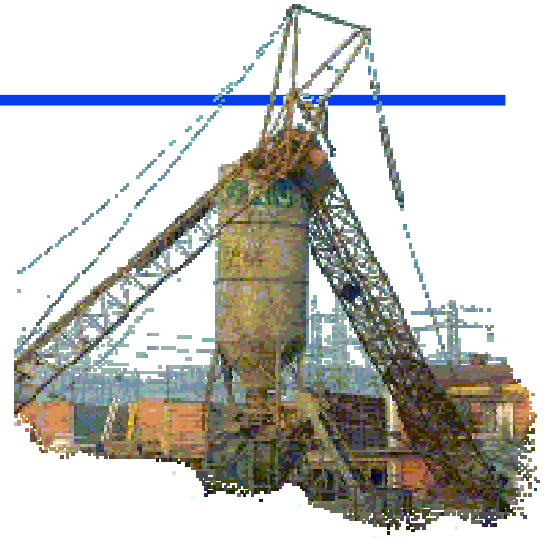


Testen,
Monitoring, ….

Modellgestützter Entwurf und Betrieb



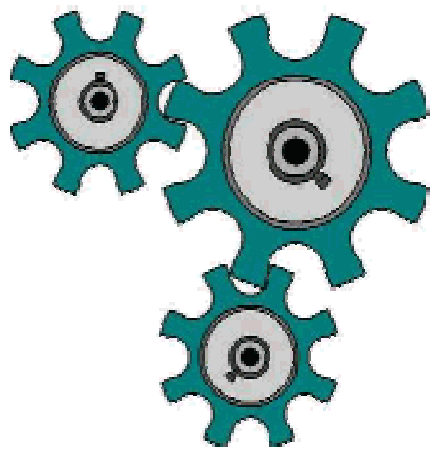
Systemeigenschaften



- Funktionalität und Funktionssicherheit
  - Safety-Eigenschaften
  - Liveness-Eigenschaften
- Datensicherheit
- Zuverlässigkeit
  - Fehler, Ausfälle und Häufigkeit
  - Fehlertoleranz
  - Kenngrößen (Verfügbarkeit, MBTF, MTTR)
- Realzeittreue
  - harte Echtzeitbedingungen
  - weiche Echtzeitbedingungen
- Leistung
  - Aufträge, Kunden und Bearbeitungszeiten
  - Kenngrößen (Durchsatz, Verweilzeit, Kundenzahl)
- Kosten

Die Threads der Vorlesung: Funktionalität

- Übersicht
  - Anwendungsdomänen und Systeme (TK-Systeme, Steuerungen, Verkehr, …)
  - Eigenschaften: Safety und Liveness
  - Komponenten, Ereignisse, Kopplung und Gesamtsystem
- Zustandstransitionssysteme
  - Einfaches STS
  - Mealy-Automat
  - Gefärbtes STS: LTS
  - Bisimulation
  - Safety, Liveness und Fairness im STS
  - Kopplung von Zustandstransitionssystemen
  - Systembildung und Erreichbarkeitsgraph
- Petrinetz und Partialordnungsmodelle
- Prozessalgebra: CCS
- Temporale Logik: LTL, CTL, CTL*
- Erreichbarkeitsanalyse und Model Checking
- Eigenschaftsbeweise im STS
  - Safety: Zustandsinvarianten und Induktionsbeweis
  - Liveness: Leads-to-Ketten, Fairness- und Lattice-Regeln



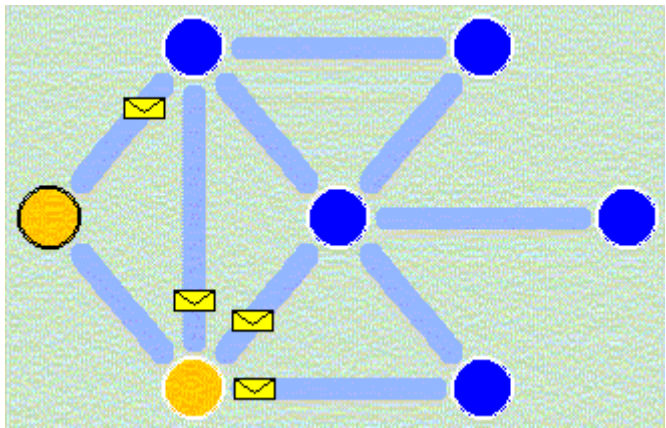
Die Threads der Vorlesung: Leistung und Realzeit

- Übersicht
  - Zeitliches Verhalten von Systemen (und Modellen)
  - Leistungsmaße und Realzeitmaße
  - Last und Maschine
  - Service Level Agreements (SLAs)
- Messung und Benchmarking
- Zeitbehaftete Modelle
  - Stochastische und zeitbehaftete Petri-Netze
  - Warteschlangennetze
  - Stochastische und zeitbehaftete Automaten
- Stochastische Prozesse (insbesondere Markov Prozesse)
- Leistungsanalyse
  - Mittelwertanalyse
- Analyse von Realzeiteigenschaften
  - Nachweis von Realzeiteigenschaften
  - Netzwerk- und Realzeitkalkül

Die Threads der Vorlesung: Zuverlässigkeit

- Übersicht
  - Fehler und Ausfälle
  - Zuverlässigkeit und Kenngrößen
  - Toleranzverfahren
- Systemmodelle
- Fehlermodelle
- Analyse
  - Hazard Analysis Verfahren
  - Fehlerbäume
  - Erreichbarkeitsanalyse und Model Checking

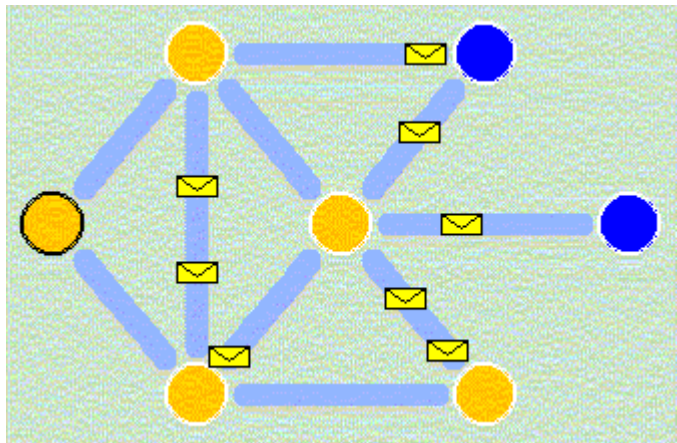
Erstes Beispiel: Ad-Hoc-Broadcast



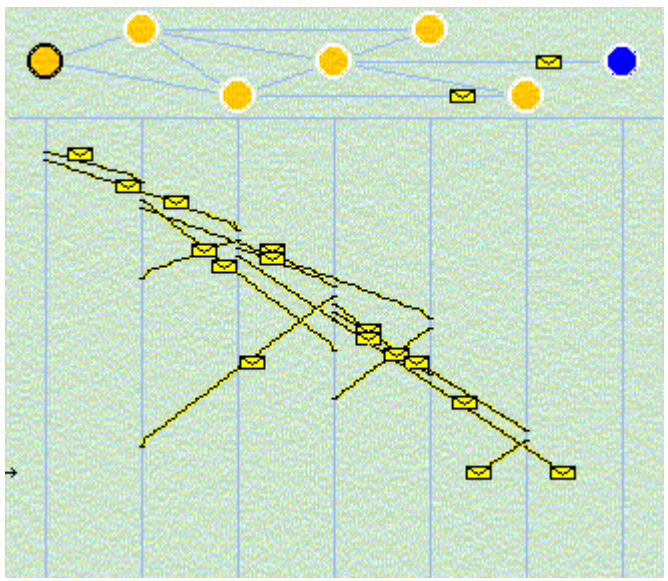
- Gegeben:
  - Netz aus Stationen und (bidirektionalen) Verbindungen
  - Netz ist zusammenhängend, jede Station kennt ihre Nachbarn bzw. die Verbindungen zu den Nachbarn
- Ziel:
  - unbestätigter Rundruf, ausgelöst durch eine Initiator-Station
  - alle Stationen sollen möglichst schnell informiert werden
  - Ad-Hoc-Durchführung, d.h. es soll keine Vorbereitungsphase (z.B. Spannbaum-Erkundung) vorausgesetzt werden
- Mechanismen:
  - ***Fluten***, d.h. eine Station leitet eine empfangene Nachricht an alle ihre Nachbarn weiter
  - ***Nachrichten-Auslöschung*** (Message Extinction), d.h. eine Nachricht, die für eine Station nicht neu ist, wird ignoriert

Ad-Hoc-Broadcast – Animation

- Animation in Topologie-Ansicht:
  http://ls4-www.informatik.uni-dortmund.de/RVS/MA/hk/OrdnerVertAlgo/ExtinctionAlgo.html

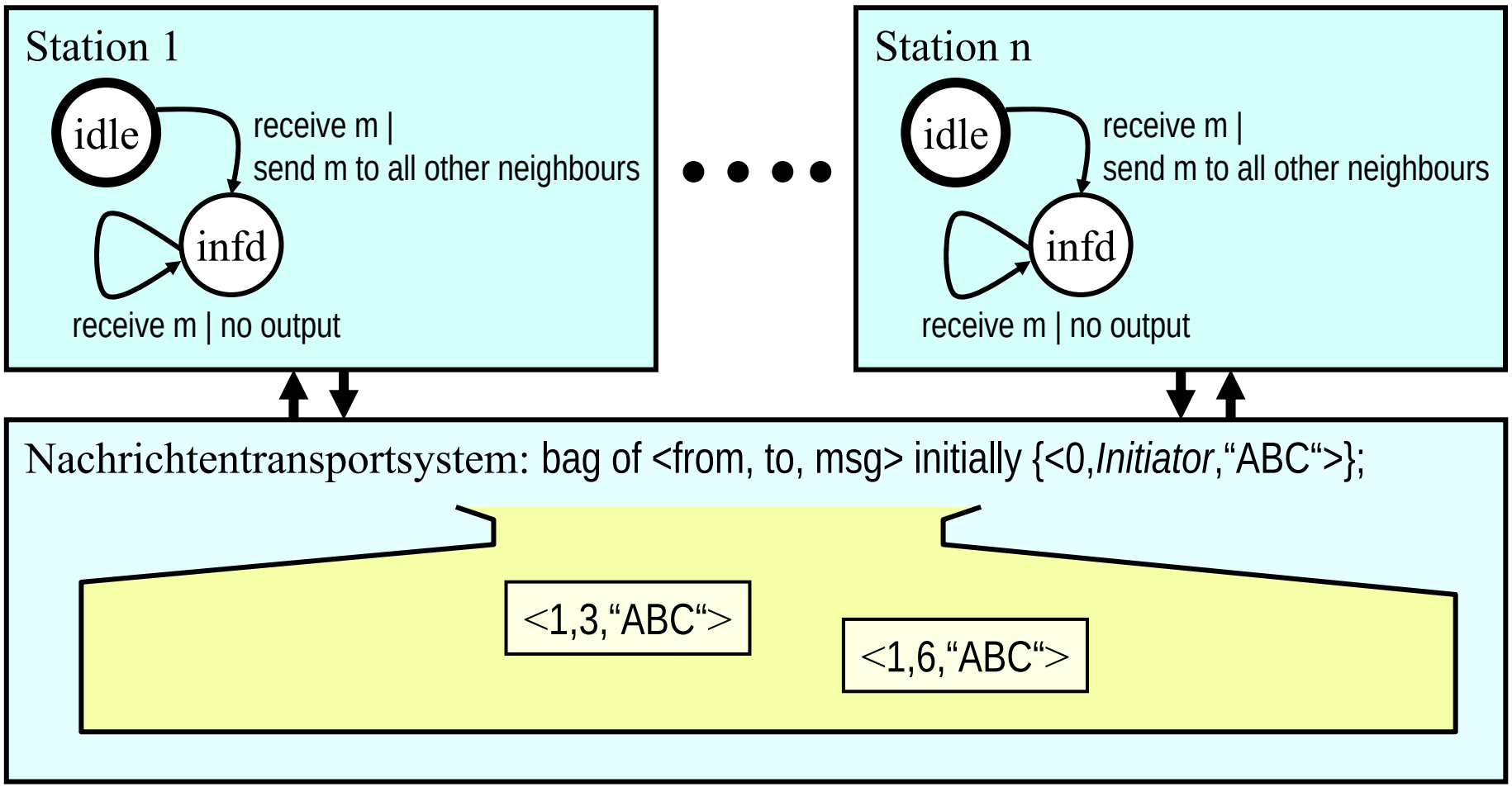


- Animation in Weg/Zeit-Ansicht:
  http://ls4-www.informatik.uni-dortmund.de/RVS/MA/hk/OrdnerVertAlgo/ExtinctionText1.html



Ad-Hoc-Broadcast: Funktionales Modell

- System aus gekoppelten Zustandsmaschinen (Mealy)
  (initial sei 1 Nachricht, adressiert an die Station Initiator, im Transportsystem)



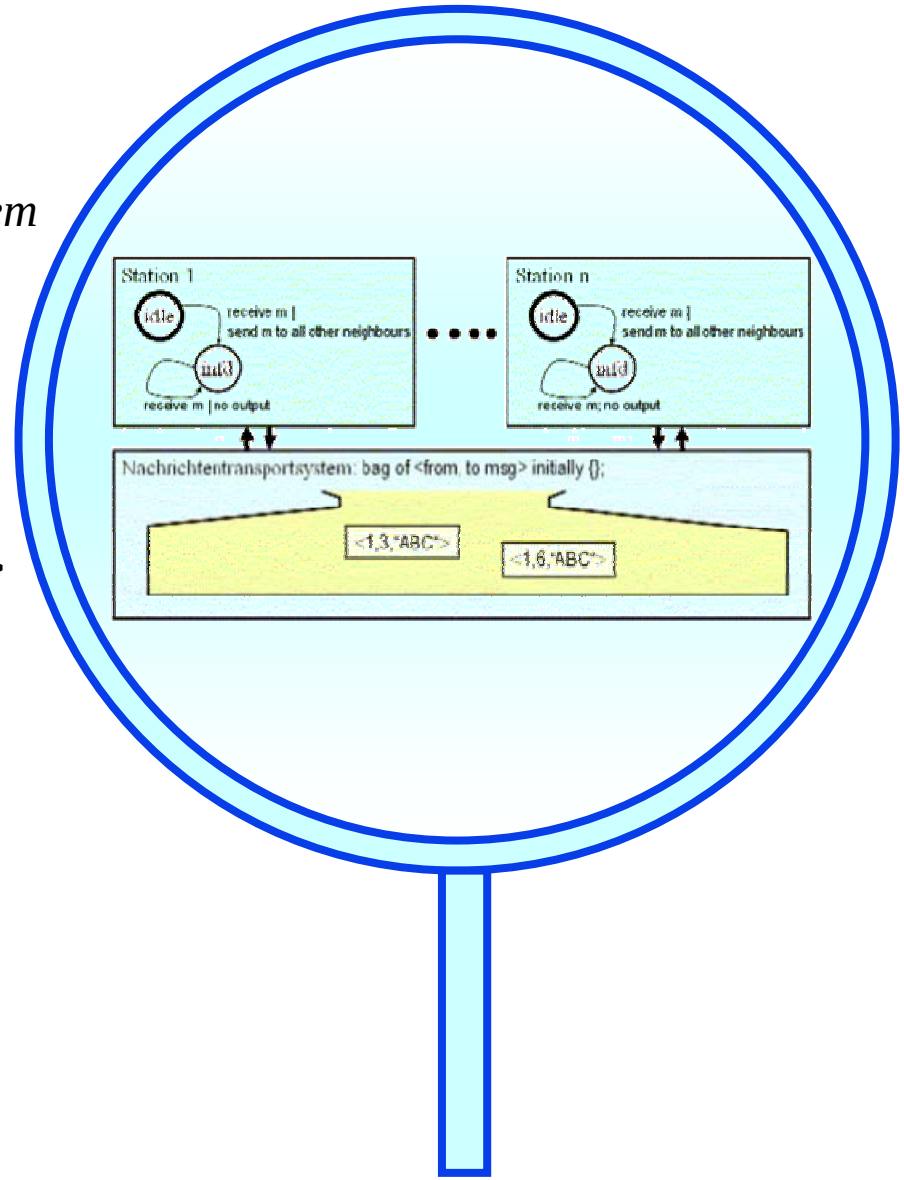
Ad-Hoc-Broadcast: Funktionales Modell

- Eine einzige (globale) Zustandsmaschine
  - Zustandsraum
    cs: array [1..n] of (idle, infd) ! *Stationen*
    nts: bag of < from, to, msg > ! *Transportsystem*

  - Zustand
    Vektor mit n+1 Stellen:
    z.B. (*es gelte n=3, d.h. es gibt 3 Stationen*)

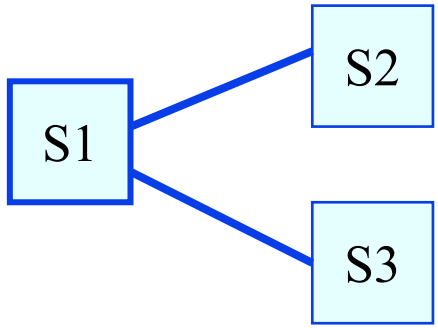
    <infd, idle, idle, {<1,2,“ABC“>, <1,3,“ABC“>} >

    In diesem Zustand hat Station 1 den Ablauf gestartet und 2 Nachrichten mit dem Inhalt „ABC“, je eine an 2 und an 3, gesendet. Die beiden Stationen 2 und 3 haben diese Nachrichten noch nicht empfangen. Sie sind noch idle und die Nachrichten sind noch im Transportsystem.

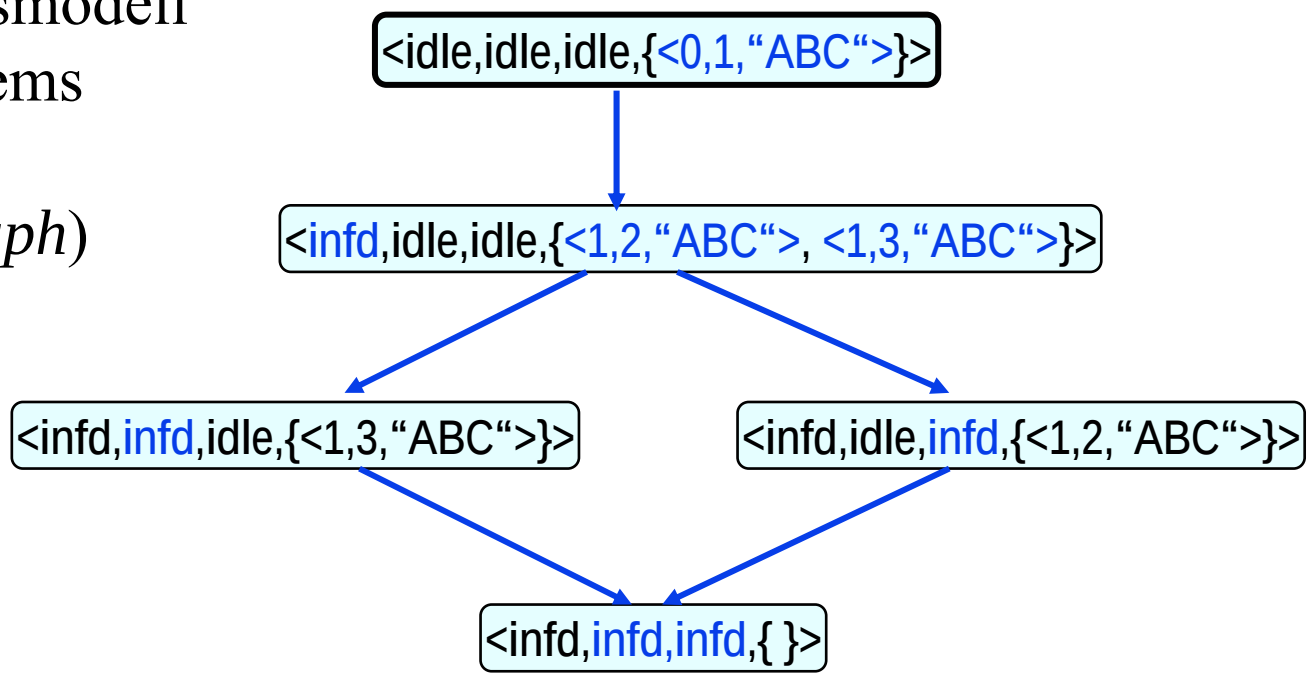


3-Stationen-Beispiel

- Netz aus 3 Stationen und 2 Verbindungen,
  S1 sei der Initiator,
  die zu verbreitende Nachricht sei „ABC“



- Zustandstransitionsmodell dieses Gesamtsystems (*so genannter Erreichbarkeitsgraph*)

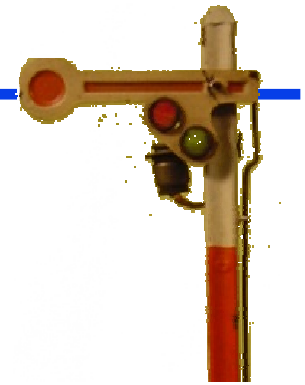


Ad-Hoc-Broadcast: Funktionale Eigenschaften

- Safety („*Was das System darf*“), z.B.
  - **Bei Terminierung wurden alle Stationen informiert.**
  - Es werden höchstens 2e Nachrichten ausgetauscht.
  - Wenn eine Station im Zustand infd ist, verlässt sie ihn nie wieder.

- Liveness („*Was das System soll*“), z.B.
  - **Nach dem Start tritt nach endlicher Zeit Terminierung ein.**
  - Jede Station wird schließlich informiert.
  - Jede gesendete Nachricht wird schließlich empfangen.

Ad-Hoc-Broadcast: Safety-Beweis



- Formalisierung der Eigenschaft als (zeit-)invariante Bedingung über den Systemzuständen (Zustandsinvariante)
  - Def.: Term ≡ cs[Initiator]=infd ∧ nts={}
  - Def.: Inv ≡ Term ⇒ ∀ i ∈(1..n): cs[i]=infd
  - geforderte Safety–Systemeigenschaft:
    Inv *gilt immer*

- Beweis per Induktion über Ablaufschritten, dass Invariante immer gilt (im Startzustand und allen erreichbaren Zuständen, benötigt wird dazu eine so genannte *induktive* Invariante)
  - „n=0“:
    Initialzustand ⇒ Inv !
  - „n→n+1“:
    Inv *gilt im Momentanzustand* ∧ *System führt Schritt aus*
    ⇒
    Inv *gilt im Folgezustand* !

Ad-Hoc-Broadcast: Liveness-Beweis

- Formalisierung der Eigenschaft als eine Bedingung über den Systemzuständen, welche nach endlich vielen Schritten gelten muss
  - Def.: Term ≡ cs[Initiator]=infd ∧ nts={}
  - geforderte Liveness–Systemeigenschaft:
    Term *gilt nach endlich vielen Schritten*
- Beweis durch Konstruktion einer Zustandsabbildung mit „*Lattice*“-Bildmenge, der Annahme, dass mögliche Systemschritte nicht unendlich lange verzögert werden, dem Nachweis, dass jeder Systemschritt dazu führt, dass das *Lattice*-Bild des aktuellen Zustands sich dem Minimum weiter nähert, und dem Nachweis, dass die Eigenschaft gilt, wenn das Minimum erreicht ist
  - Def.: lat ≡ <Anzahl Stationen im Zustand idle, Anzahl Nachrichten im nts>
    Lattice: Nat x Nat mit Ordnung $<$
    $\langle x,y\rangle < \langle x',y'\rangle \Leftrightarrow x < x' \vee (x = x' \wedge y < y')$
  - lat=<x,y> ∧ *Systemschritt erfolgt* ⇒ lat(*im Folgezustand*) $<$ <x,y> !
  - lat=<0,0> ⇒ Term !

Ad-Hoc-Broadcast: Nicht-funktionale Eigenschaften

Einbeziehung der Zeit führt zu weiteren Fragestellungen

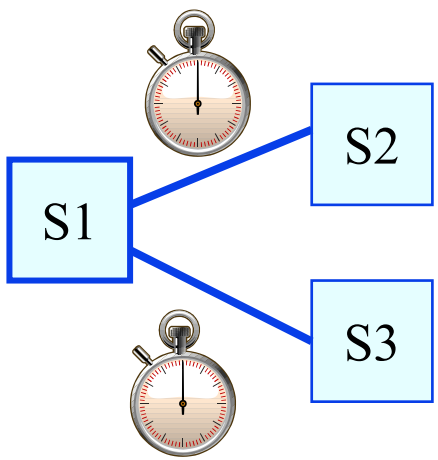
1. Wie lange dauert es im Durchschnitt, bis alle Stationen informiert sind?
2. Wie lange dauert es maximal, bis alle Stationen informiert sind?

Aber auch

- Wie groß ist die Auslastung des Kommunikationsmediums, das S1 und S2 bzw. S1 und S3 verbindet, wenn S1 10 Benachrichtigungen pro Sekunde verschickt?
- …

Wir betrachten 1. (Leistungsanalyse) und 2. (Realzeit)

Wo wird Zeit verbraucht?



Annahmen hier:

➢Nachrichteübertragung dauert Zeit

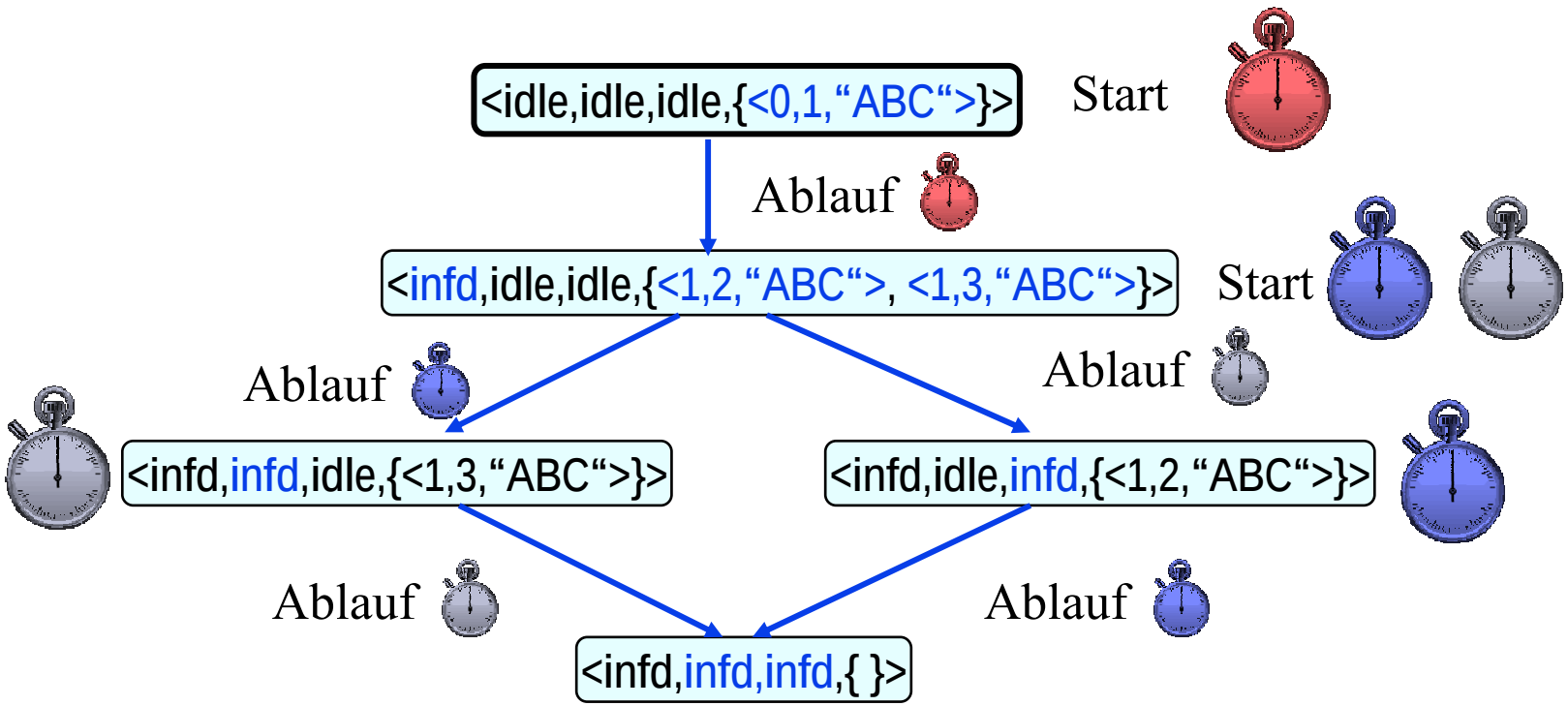
➢Zeit vergeht im Zustand

➢Versenden, Empfangen von Nachrichten und Zustandsänderungen in Nullzeit

Zeitbehaftete Zustandstransitionsmodelle

Allgemeine Form:

- Starte beim Senden einer Nachricht einen Wecker,
- Empfange die Nachricht und führe die interne Zustandsänderung beim Ablauf des Weckers durch



Zeitmodellierung

Was bedeutet „Starten eines Weckers“?

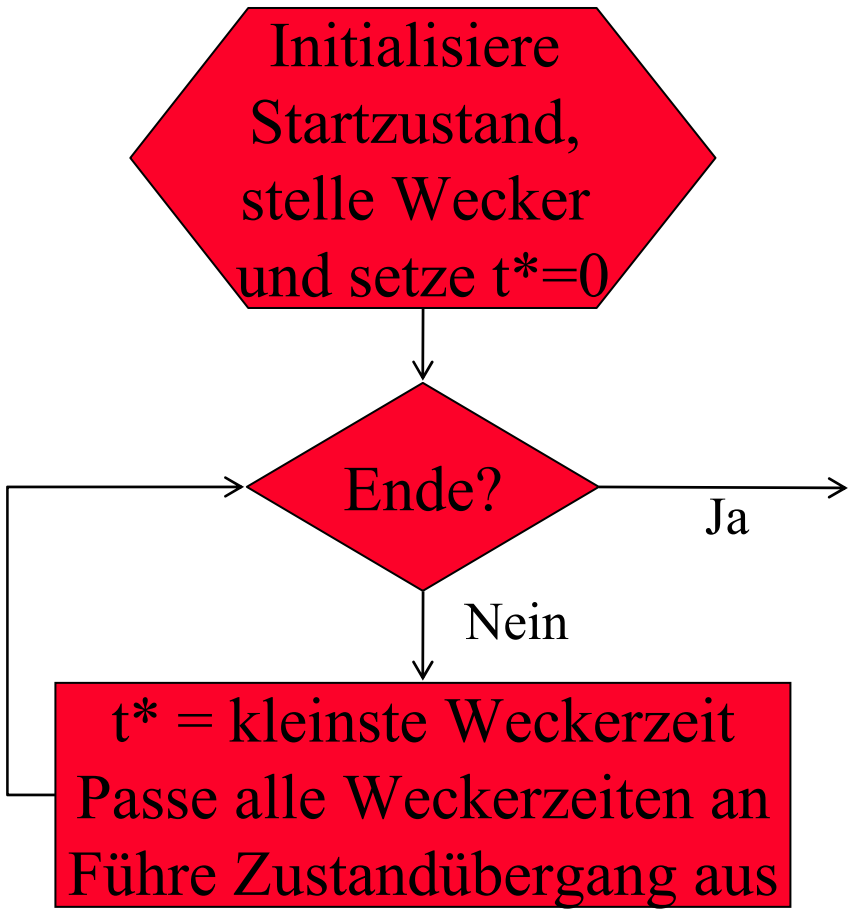
Sei U der Wert des Weckers und t* der Werte der Modellzeit
t* beginnt bei 0 und wächst mit konstanter Rate 1.

Fallunterscheidung bei der Initialisierung von Weckern:

1. Zeit genau bekannt und ist konstant
2. Zeit schwankt stochastisch und
   a. kann Werte aus einer abzählbaren Menge $\mathbb{T}$ annehmen
      p_t für $t \in \mathbb{T}$ ist die Wahrscheinlichkeit, dass Wert t angenommen wird
      (Zeit ist diskrete Zufallsvariable, zeitdiskretes Modell)
   b. kann Werte aus einer überabzählbaren Menge $\mathbb{T} \in \mathbb{R}_{\geq 0}$ annehmen
      initialer Wert wird gewählt gemäß Verteilungsfunktion $F_X(x)$
      (Zeit ist kontinuierliche Zufallsvariable, zeitkontinuierliches Modell)

Analyse von Zeitbehafteten Modellen

Simulieren und Beobachten:



Abläufe
beobachten (u.U. mehrfach)
und auswerten
d.h. oft Aussagen über
potenziell unendlich viele
Abläufe durch Beobachtung
endlicher vieler Abläufe
(⇒ Details siehe **MAO**)

In dieser Vorlesung:
Methoden zum Nachweis von
Eigenschaften!

Nachweis von nicht-funktionalen Eigenschaften

Notwendige Einschränkungen:

- Alle Wecker (bis auf evtl. einen Wecker) werden in jedem Zustand wieder neu gestartet
  (oder können wieder neu gestartet werden, ohne das Verhalten zu beeinflussen)

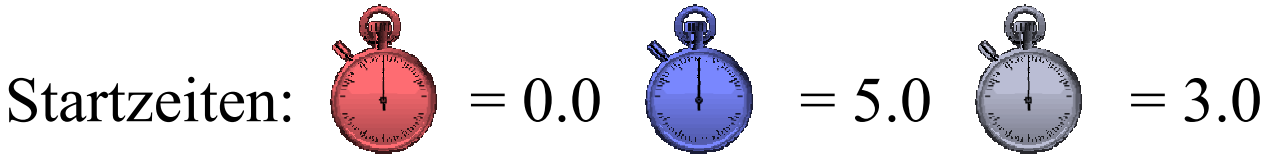
oder

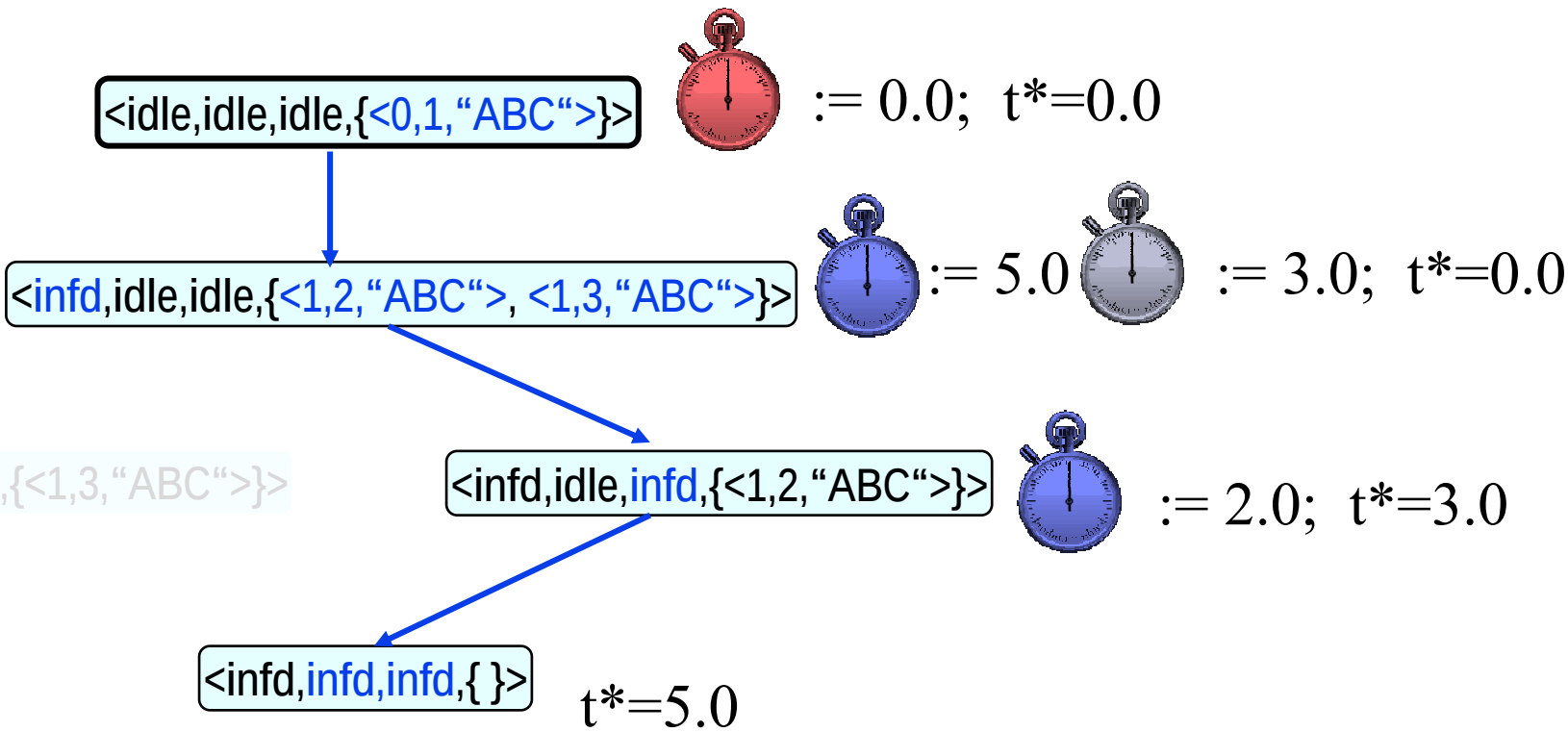
- alle Wecker (bis auf evtl. einen Wecker) können nur endlich viele initiale Werte annehmen

oder

- die initialen Werte aller Wecker sind durch endliche Intervalle gegeben und als Ergebnis werden Zeitintervalle berechnet

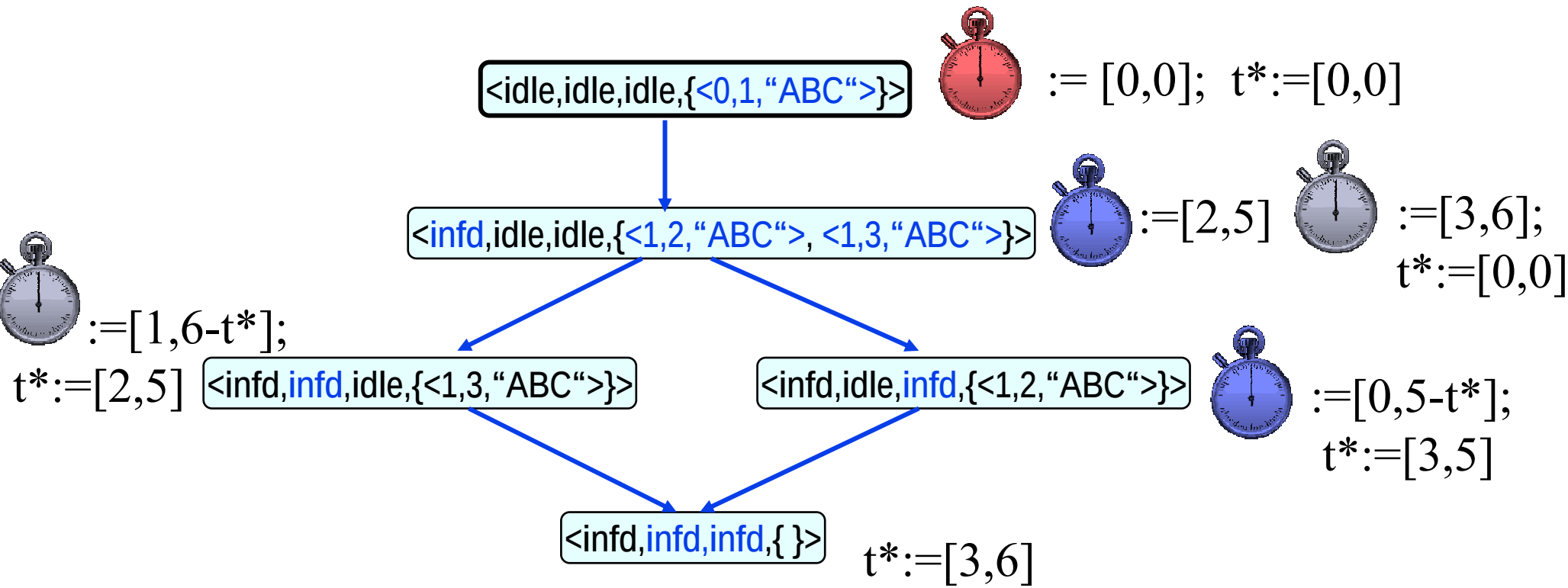
Deterministische Zeiten





Intervallzeiten

Startzeiten: = [0,0] = [2,5] = [3,6]



Stochastische Zeiten