Q1. Systemverhalten und nicht-funktionale Eigenschaften

Gliederung

1. Modellierung dynamischer Abläufe
2. Darstellung von Zeit in Zustandsmodellen
3. Systeme und Anforderungen an Systeme
4. Nicht-funktionale Maße
5. Festlegung und Vereinbarung von Systemanforderungen

Literatur (primär)

- C. G. Cassandras, S. Lafortune. Introduction to Discrete Event Systems. 2nd Ed. Springer 2008. Kap. 1
- N. J. Gunther. Analyzing Computer System Performance with Perl::PDQ. Springer 2005. Kap. 1
- R. Jain. The Art of Computer Systems Performance Analysis. Wiley 1991. Kap. 3
- A. Cockcroft, B. Walker. Capacity Planning for Internet Services. Sun Microsystems Press 2001. Kap. 2, 3
- D. A. Menasce, V. A. F. Almeida, L. W. Dowdy. Performance by Design. Prentice Hall 2004. Kap. 1, 3, 4.7.
- und einige Originalartikel

1.1 Modellierung der Anforderungen

System als Anzahl in Beziehung stehender Teile, die zu einem gemeinsamen Zweck interagieren!

In unserem Fall dynamische Systeme (d.h. Systeme, deren Zustand sich über die Zeit ändert)

Speziell Systeme der Informatik:

- Zusammenspiel von Software und Hardware, heute oft verteilt und/oder eingebettet
- Künstliches System, d.h. vom Menschen erschaffen
- Relativ genau definierte Anforderungen an das System

Ersatzsystem S‘ für ein System S bzgl. eines Analyseziels nennt man ein **Modell**

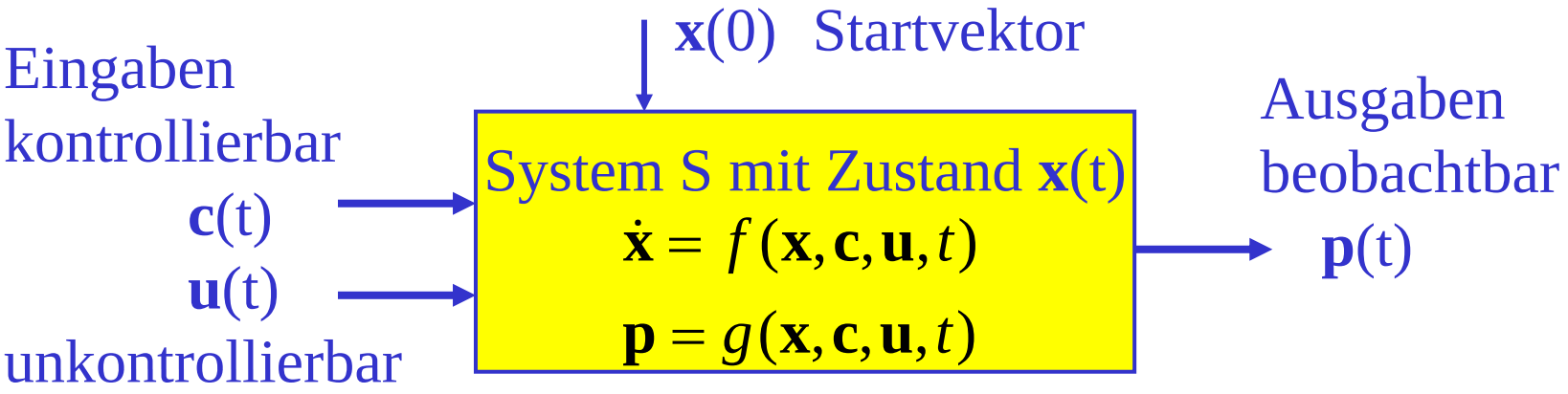
Es existieren mehrere Definitionen, hier nur zwei Beispiele:

Definition (nach Niemeyer)
Modelle sind materielle oder immaterielle Systeme, die andere Systeme so darstellen, dass eine experimentelle Manipulation der abgebildeten Strukturen und Zustände möglich ist.

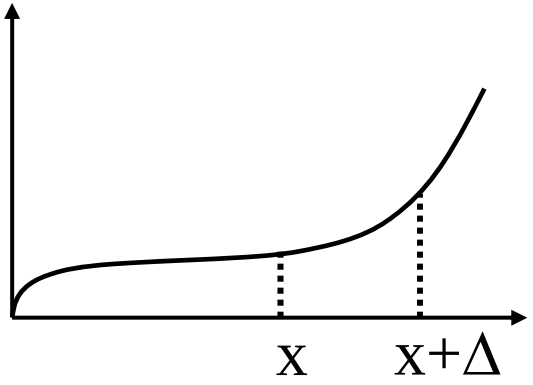
Definition (nach Cellier)
Ein Modell M für ein System S und ein Experiment E ist ein System S‘, auf das E angewendet werden kann und Aussagen über die Anwendung von E auf S erlaubt.

- Modelle dynamischer Systeme müssen den Ablauf über die Zeit beschreiben

Klassischer Ansatz aus den Naturwissenschaften: Differentialgleichungen



Kontinuierlichen Ablauf i.d.R. per Simulation abbilden



Kontinuierliche Modelle sind für Systeme der Informatik weniger geeignet:

- System läuft in Schritten ab
  - Anweisungen in einem Programm
  - Jobs auf einem Rechner
  - …
- Information ist digital und damit diskret

Kontinuierliche Beschreibung wäre nur eine Abstraktion

Besser:

Diskrete Modelle

- Systemzustand ändert sich zu diskreten Zeitpunkten
- Zustandsänderungen sind atomare Operationen

Man unterscheidet:

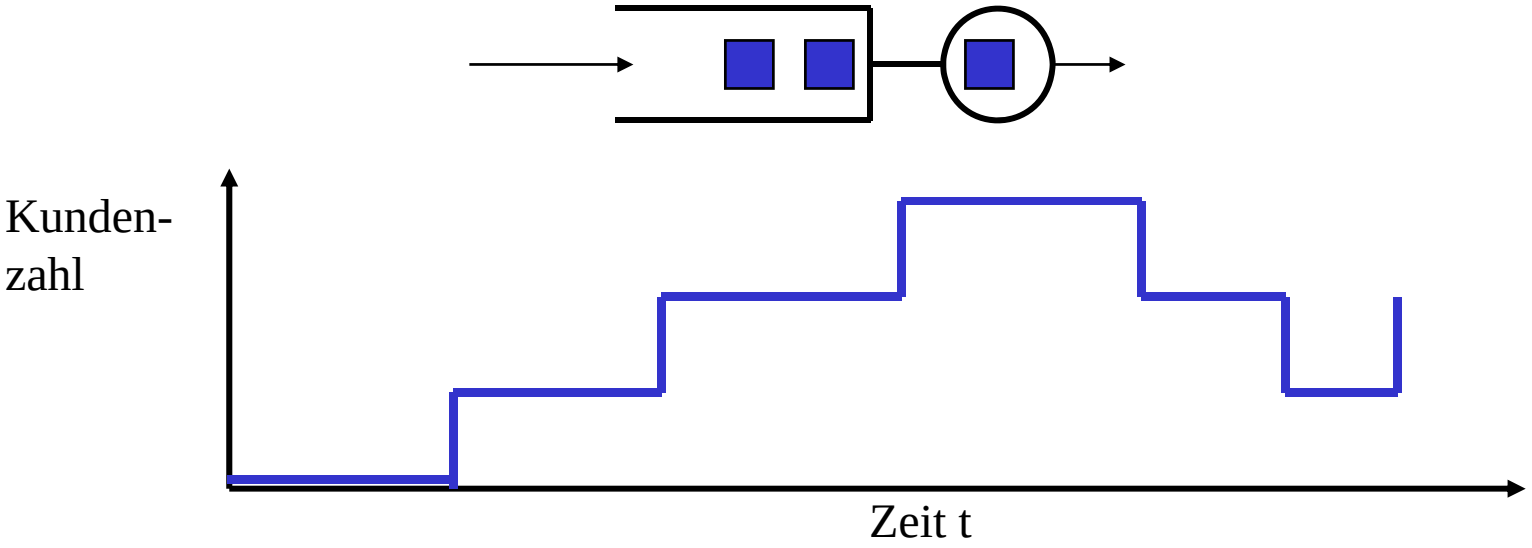
Zeitdiskrete Modelle:

- Zeitkonstante Δ bestimmt die Beobachtungszeitpunkte
- Zustand ändert sich zu Zeitpunkten $k \cdot \Delta$
- Kann auch bei der Beobachtung kontinuierlicher Abläufe eingesetzt werden (abtasten)

Zeitkontinuierliche Modelle:

- Zeitpunkt von Ereignissen bestimmt die Zustandsänderung
- Ereignisse können zu beliebigen Zeiten auftreten
- Ereigniszeitpunkte im Modell implizit oder explizit definiert

Beispiel ereignisdiskreter Ablauf



Zeitdiskrete Systeme werden eingesetzt bei

- Vorgegebenem Zeittakt (hardware-nah)
- Vorgegebenem Beobachtungstakt (Abtastung)

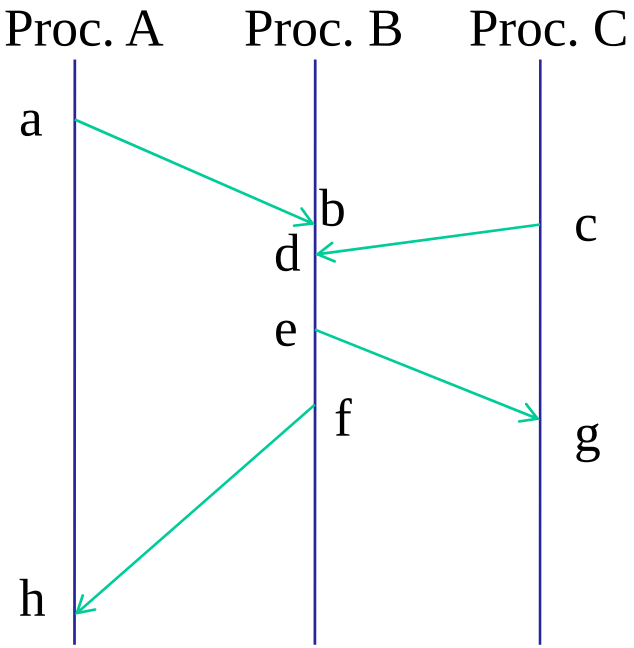
Ereigniskrete Systeme werden eingesetzt bei

- Modellen ohne Zeittakt (abstrakte Modellierung)
- Modellierung detaillierter Messungen

Zeit als Basis dynamischer Vorgänge

- Zeit als physikalische Größe
  seit der Relativitätstheorie ist bekannt, dass Zeit und Raum für große Dimensionen nicht unabhängig sind
  (für uns weniger relevant, siehe aber Gleichzeitigkeit in verteilten Systemen)
- Zeit in einem Rechner
  messbar über die lokale (Hardware-) Uhr mit vordefinierter Auflösung
  Ereignisse sind zeitlich geordnet
- Zeit in einem verteilten System
  verschiedene lokale Uhren, nicht vollständig synchronisierbar,
  Gleichzeitigkeit verteilter Ereignisse nicht definiert,
  damit auch Zustand zu einem Zeitpunkt nicht definiert

- Kausalität in verteilten Systemen



- Lokal gilt a<h, b<d<e<f, c<g
- Transitiv kann gefolgert werden a<g,c<h
- Keine Relation zwischen a und c sowie g und h

- Modelle sollten/müssten eigentlich nur Kausalitäten berücksichtigen (siehe z.B. zeitlose Petri-Netze)
- Viele Modelle zur quantitativen Analyse arbeiten mit absoluten Zeiten (d.h. globaler Zustand ist zu jeder Zeit vollständig definiert, siehe z.B. zeitbehaftete Petri-Netze)
  - Sequentielle Analysealgorithmen sind einfacher formulierbar
  - Verteilte Analyse wird erschwert (vgl. verteilte Simulation)

- Problem der unterschiedlichen Zeitskalen

Teilsystem	Zeit	Skala	Skalierte Zeit	Skalierte Skala
CPU-Zyklus	0.31	ns	0.31	Sekunde
L1-Cache	0.31	ns	0.31	Sekunde
L2-Cache	1.25	ns	1.25	Sekunde
Speicherbus	2.00	ns	2.00	Sekunde
DRAM	60.0	ns	1.00	Minute
Plattenzugriff	3.50	ms	1.35	Monat
NFS-Lesezugriff	32.0	ms	1.01	Jahr

Einbeziehung von Fehlern fügt Ereignisse mit einer Frequenz im Tages oder Wochenbereich ein (skaliert 1-10 Millionen Jahre)

⇒ Modelle können nicht alle Zeitskalen berücksichtigen

⇒ Dekomposition und separate Modellierung und Analyse

Modellierung von Zeiten:

- Deterministisch
- Durch Intervalle
- Stochastisch
  - Als Verteilung
  - Als stochastischer Prozess
  - Als empirische Verteilung

Basis der Modellierung:

- Schätzungen, Annahmen
- Messdaten

Vorgehen:

Suche nach einem adäquaten Modell, so dass

- Daten aus dem Modell stammen könnten
  (Methoden der Statistik siehe MAO)
- Modell im gewählten Analyseverfahren handhabbar ist

1.2 Zeit in Zustandsmodellen

Wecker beschreiben (bildlich) den Zeitablauf und werden mit Ereignissen verbunden:

- Ereignisse
  - starten Wecker
  - modifizieren Wecker
  - setzen Wecker zurück
- Ablauf eines Weckers initiiert ein Ereignis

(Absolute) Zeit ordnet Ereignisse (bis auf Gleichzeitigkeit)

Systemzustand: diskreter Zustand + Zustand aller Wecker

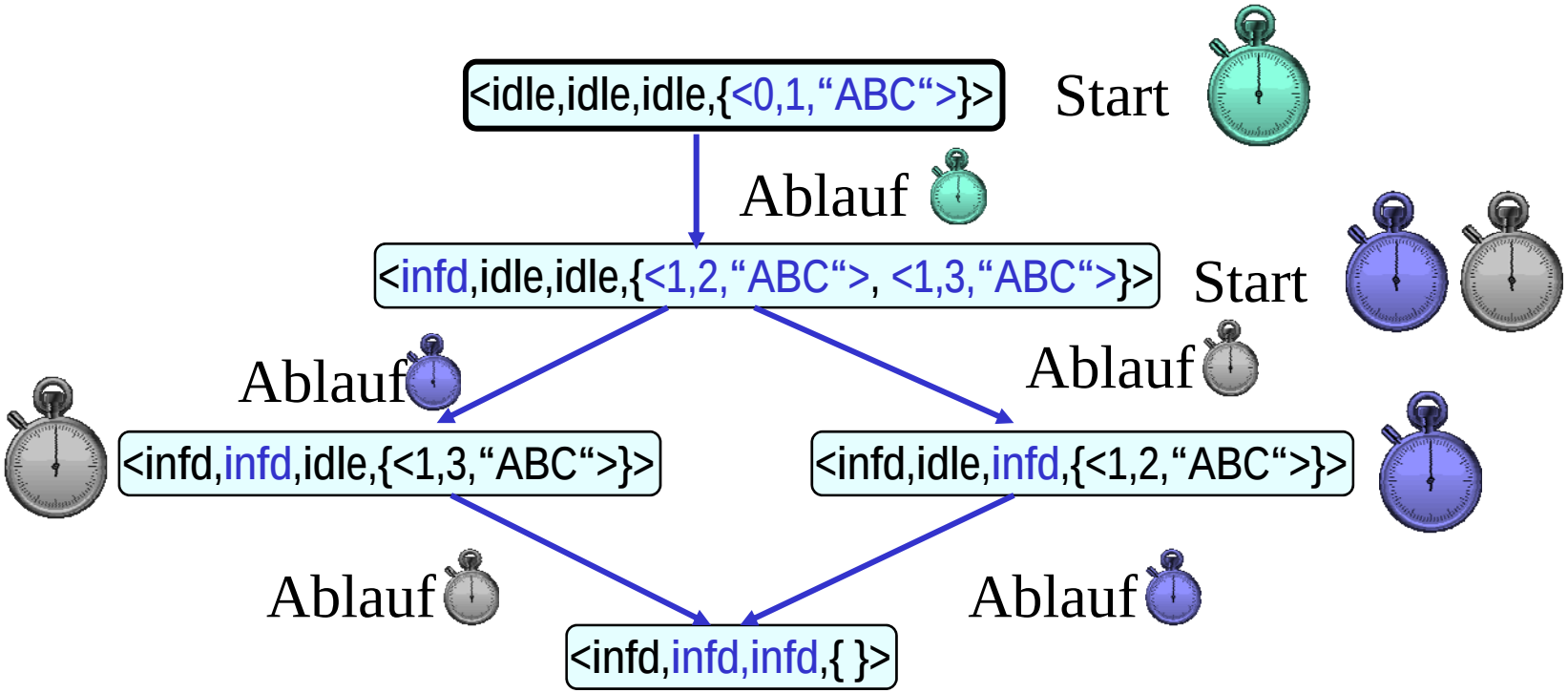
$\Rightarrow$ Komplexität nimmt zu!

Ablauf der Zeit kontinuierlich mit Rate 1!

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“?

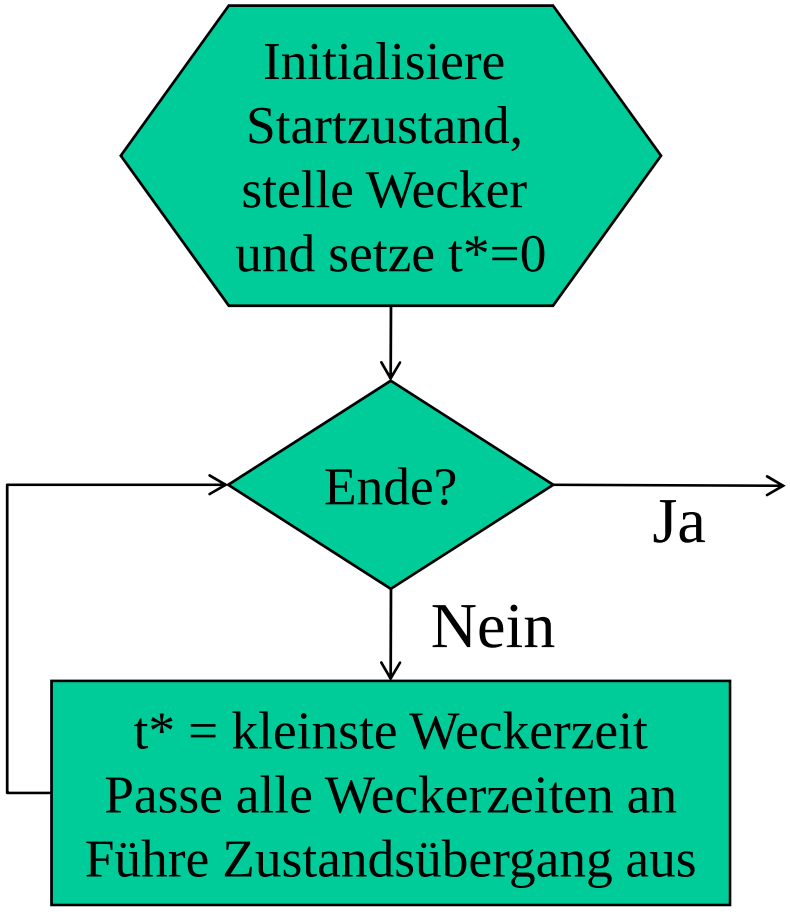
Sei U der Wert des Weckers und t* der Werte der Modellzeit
t* beginnt bei 0 und wächst mit konstanter Rate 1.

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 nicht-funktionaler 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)

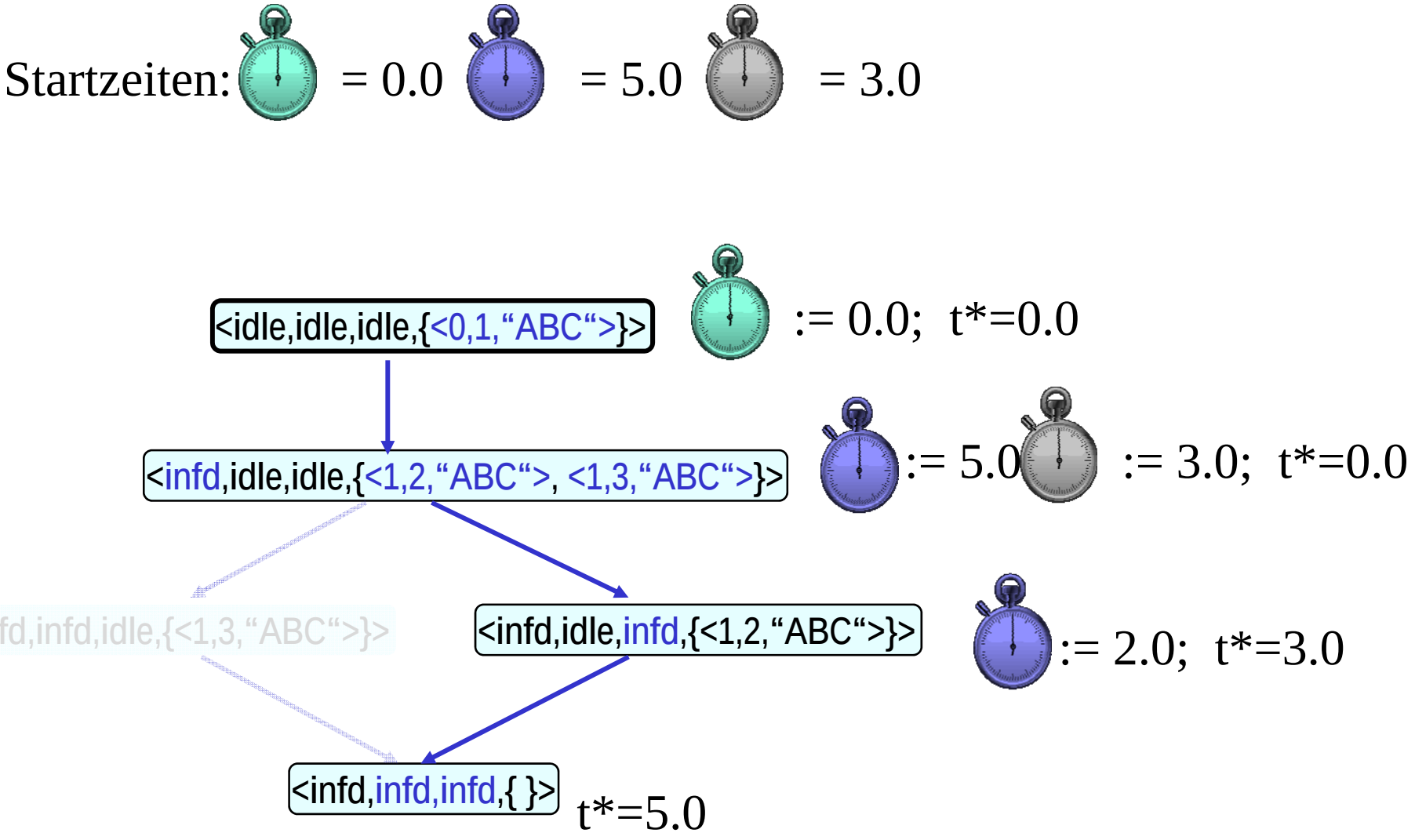
oder

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

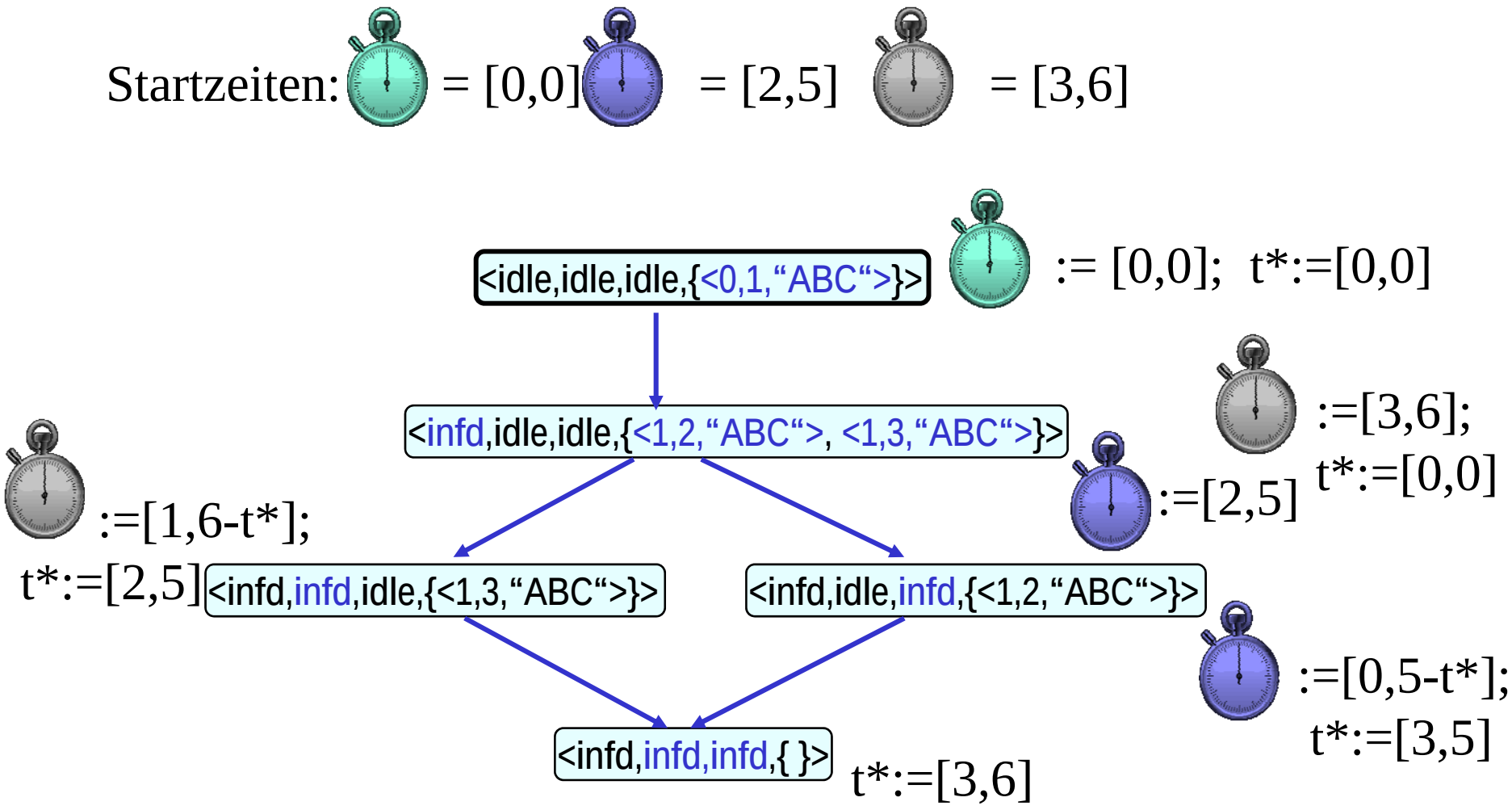
oder

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

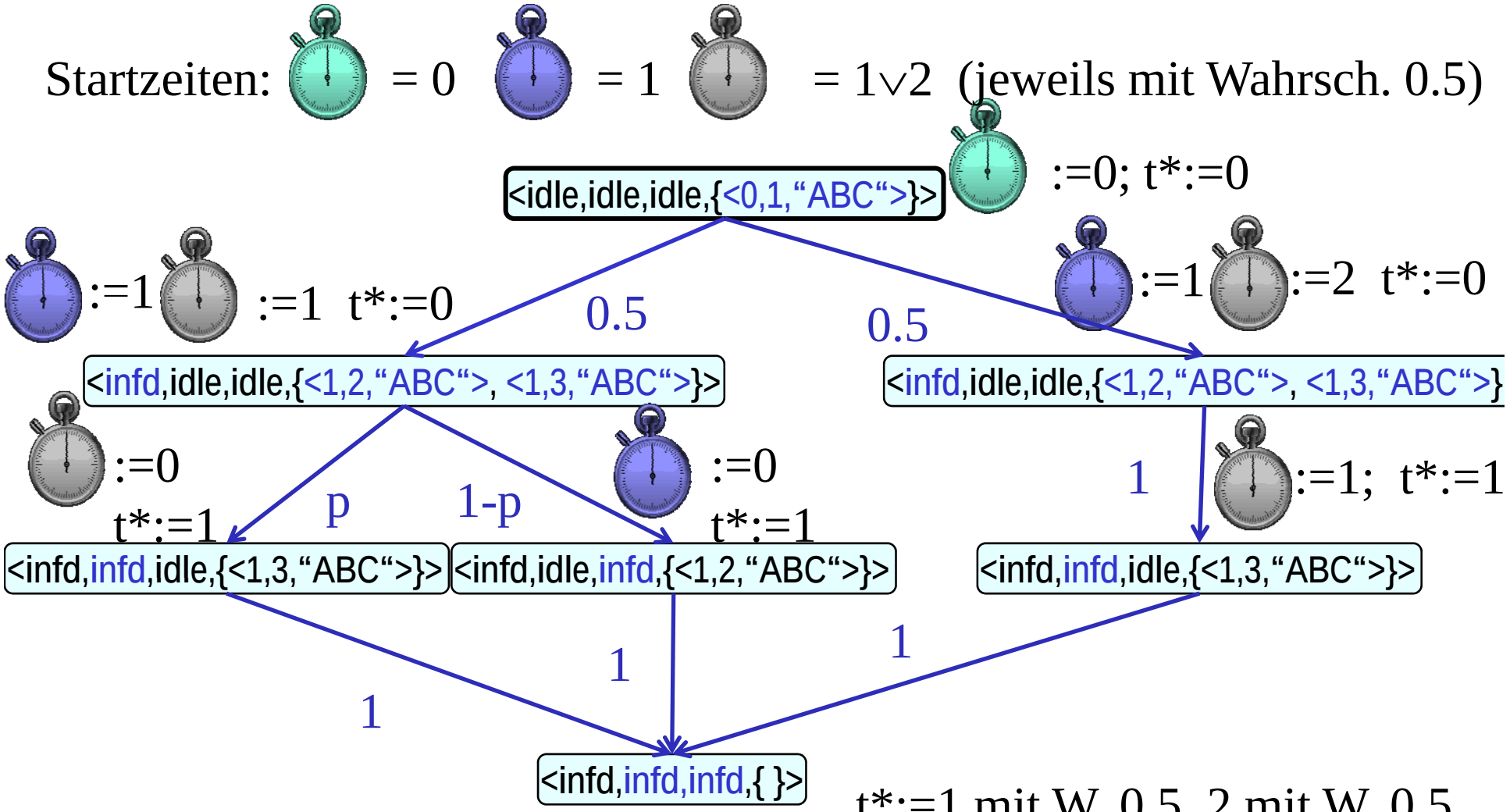
Deterministische Zeiten



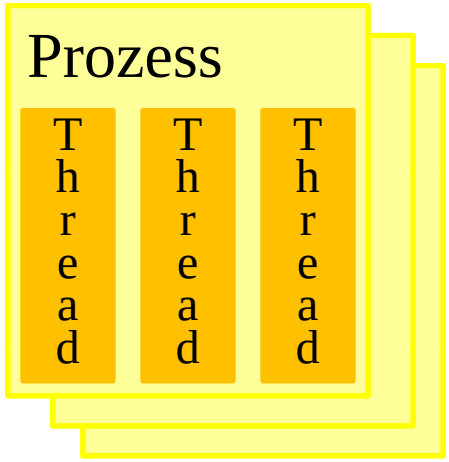
Intervallzeiten

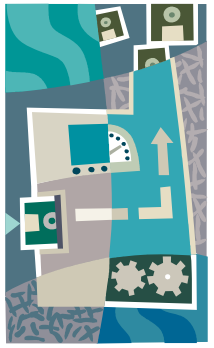


Stochastische Zeiten



1.3 Systeme und ihre Anforderungen





Systemsoftware

Systemhardware

Kommunikations-protokolle

Kommunikations-hardware

In mehreren Schichten, hierarchisch strukturiert

Allgemeine Anforderung:
System soll das tun, wozu es entwickelt wurde!

Oft wird unterschieden in

- Funktionale Eigenschaften, die die Funktion beschreiben
  (z.B. liefert das korrekte Ergebnis)
- Nicht-funktionale Eigenschaften, die die Systemverhalten quantifizieren
  (z.B. die mittlere Antwortzeit beträgt 3 Sekunden)

Diese Unterteilung ist nicht unumstritten:
„The fact that performance requirements are called non-functional properties in SE is a symptom of this problem. How can a software system function properly if some of ist requirements (e.g., performance requirements) are not met? In other words, all requirements should be functional. Otherwise, they should be no requirements“ (D. Menasce 2002).

Wir wollen trotzdem bei der Unterscheidung in funktionale und nicht-funktionale Eigenschaften bleiben. Ein System muss beide erfüllen, um korrekt zu sein!

Leistung
Wie lange dauert etwas?
Wie viel wird bearbeitet?
Im Mittel, in einem Zeitintervall, …

Zuverlässigkeit
Wie oft fällt das System aus?
Wie lange dauert ein Ausfall?
Im Mittel, in einem Zeitintervall, …

Nicht-funktionale Eigenschaften

Zeitverhalten
Wird eine Antwort innerhalb eines Zeitintervalls gegeben?
Wie oft wird eine Zeitschranke überschritten?

Kosten
Was kostet das System?
Was kostet die Abarbeitung eines Auftrags?

Leistung

Unterschiedliche Metriken für die verschiedenen Ebenen

- OLTP-System: Transaktionen pro Sekunde, mittlere Dauer einer Transaktion, Kosten pro Transaktion
- Web-Server: HTTP-Anfragen pro Sekunde, mittlere Zugriffszeit auf eine Web-Seite, Auslastung
- Kommunikationsnetz: Übertragungsrate in MB, RTT in sec (QoS)
- Router: Pakete pro Sekunde, übertragen Daten in MB pro Sekunde
- CPU: Millions of Instructions per Second (MIPS), Floating Point Operations per Second (FLOPS)
- Platte: mittlere Zugriffszeit, transferierte Daten pro Sekunde

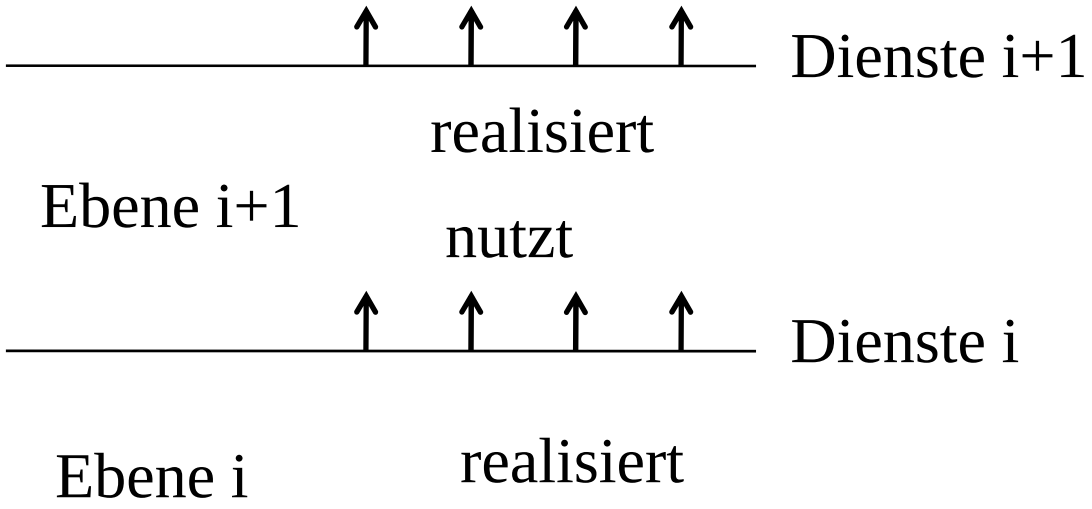
Angaben oft als Mittelwerte über einen (langen) Zeitraum!

Notwendige Länge des Zeitraums hängt vom System und Leistungsmaß ab!

Leistung

Systemleistung entsteht durch Zusammenspiel vieler (Hardware- und Software-) Komponenten

Hierarchische Sicht (virtuelle Maschinen):



- Jede Ebene durch Hardware und/oder Software realisiert
- Dauer einer Operation auf Ebene i resultiert aus Dauer der Berechnungen auf Ebene i und aller dadurch initiierten Berechnungen auf Ebenen j < i
- Üblicherweise Verkleinerung der Zeitskala von oben nach unten

⇒ Modelle beschränken sich auf wenige Ebenen

Leistung

Funktion von:

Nur sehr hardware-nahe Maße sind unabhängig von der Last
z.B. MIPS, FLOPS (deterministisch)
Auf höherer Ebene sind Spezifikationen der Last notwendig
z.B. Bearbeitungszeit einer Transaktion
Wie groß ist eine Transaktion?
Wie viele treffen pro Sekunde ein?

Deterministisches Verhalten ist zu beobachten

- Direkt auf der Hardware
- In speziell dafür entworfenen Systemen

Stochastisches Verhalten ist zu beobachten

- Auf Systemebene
- In Kommunikationsnetzen
- In verteilten Systemen

Last kann deterministisch oder stochastisch sein

Leistungsanalyse vernachlässigt durch Mittelwertbetrachtung oft die Schwankungen in den Leistungsgrößen

Zuverlässigkeit

Leistungsgrößen und auch Zeitschranken reichen alleine nicht aus, das System muss auch immer/meistens zur Verfügung stehen

Unterschiedliche Termini:

Zuverlässige / Fehlertolerante / Robuste Systeme

Im Englischen *dependable systems*

Je nach System führt ein Systemausfall

- Zum Verlust von Leben/Gesundheit (sicherheitskritisches System)
  z.B. Steuerungssysteme von chemischen Anlagen, Atomkraftwerken, Flugzeugen
- Zur Nichterreichung des eigentlichen Ziels, ohne Gefährdung von Menschen
  z.B. Computersystem eines Satelliten
- Zu finanziellen Verlusten durch entgangene Geschäfte
  z.B. Web-Server eines Online-Anbieters

Während im ersten Fall Ausfälle auf jeden Fall zu vermeiden sind, ist in den beiden anderen Fällen eine Kosten-/Nutzen-Abwägung möglich (Kosten zur Vermeidung eines Systemausfalls versus Kosten durch den Ausfall)

Zuverlässigkeit

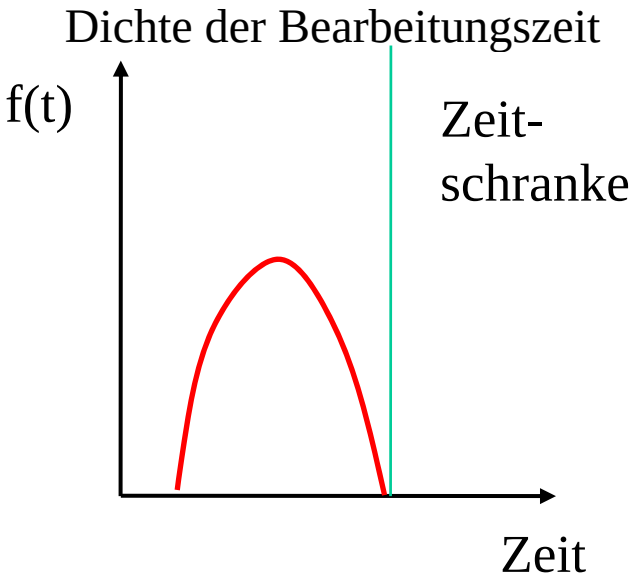
Zuverlässigkeitsmaße:

- Zuverlässigkeit (*Reliability)*
  Wahrscheinlichkeit, dass ein System eine vorgegebene Aufgabe erfüllt
- Verfügbarkeit (*Availability*)
  Wahrscheinlichkeit, dass ein System zu einem Zeitpunkt (fest oder beliebig) seine Aufgabe erfüllen kann
- *dependability*
  wird als Oberbegriff für Verfügbarkeit, Zuverlässigkeit, Sicherheit und Vertraulichkeit gesehen
- *performability* (*performance* + *availability*)
  Leistung unter Berücksichtigung von Ausfällen
- Überlebensfähigkeit (*survivability*)
  Wahrscheinlichkeit, dass ein System bei einer Menge von Ausfällen noch einen Mindestservice liefert
- Wartbarkeit (maintainability)
  Wahrscheinlichkeit, dass ein System nach einer Wartung innerhalb einer vorgegebenen Zeit seine Aufgabe erfüllen kann

Zeitschranken

Realzeit- oder Echtzeitsysteme:

Harte Realzeit:



- Bearbeitung muss erfolgt sein, wenn Zeitschranke erreicht wird
- Form der Dichtefkt. ansonsten beliebig
- Nachweis i.d.R. notwendig, um System einzusetzen

- Modelle müssen den schlechtesten Fall betrachten und nachweisen, dass Zeitschranke eingehalten wird
  Annahmen über den schlechtesten Fall sind notwendig
  - Schranken für Bearbeitungszeiten
  - Belastungsschranken

  oft konservative Annahmen
- Verteilte Systeme erfordert spezielle Kommunikation mit Zeitgarantien

Einsatzgebiet:

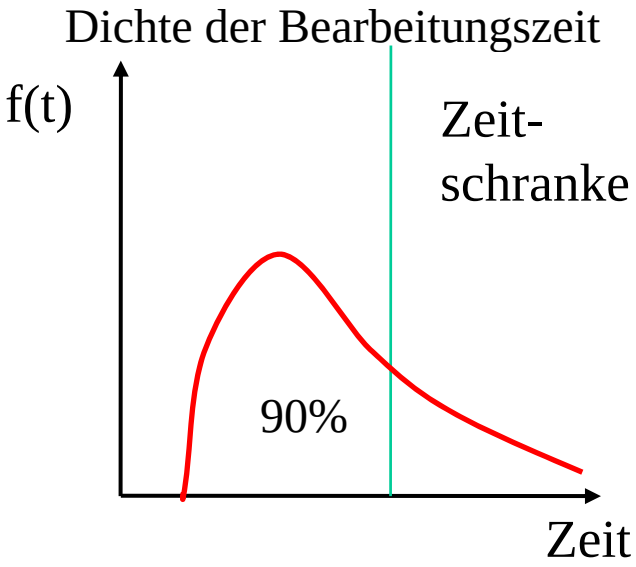
Sicherheitskritische Systeme

insbesondere Steuerungen

Zeitschranken

Realzeit- oder Echtzeitsysteme:

Weiche Realzeit:



- Bearbeitung sollte mit hoher Wahrscheinlichkeit erfolgt sein, wenn Zeitschranke erreicht wird
- Überschreitung der Zeitschranke führt meistens dazu, dass das Ergebnis unbrauchbar wird

- Modelle arbeiten mit Wahrscheinlichkeiten und müssen Wahrscheinlichkeiten bestimmen
  Mittelwerte sind nur in Ausnahmefällen akzeptabel (z.B. Mittelwert sehr viel kleiner als Zeitschranke)
  - Verteilungen von Bearbeitungszeiten und Belastung oder
  - Rechnen mit deterministischen aber weichen Schranken

Einsatzgebiet:

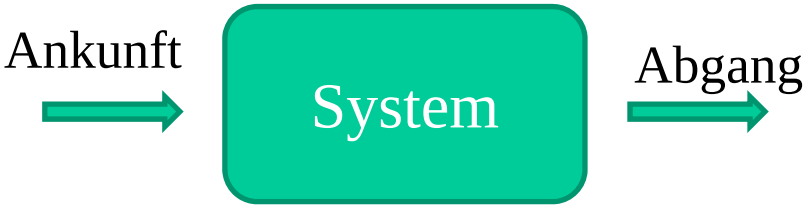
Multimediasysteme, Überwachung nicht sicherheitskritischer Anlagen

Kosten

- Wichtiges Maß in der Praxis, wenig beachtet in der Methodik
- Typischerweise eine betriebswirtschaftliche Fragestellung
  deshalb oft entkoppelt von der quantitativen Analyse
- Kostenmodelle
  - Statisch
    Kosten für Hardware, Software, Wartung und Betrieb
    sind unabhängig von der Systemdynamik
    dann entstehen feste Kosten für ein System/Modell
    beim Vergleich sind unterschiedliche Systeme bzgl.
    Leistung/Zuverlässigkeit/Zeitverhalten/Kosten zu vergleichen
    dies erfordert kostenmäßige Bewertung von
    Leistung/Zuverlässigkeit/Zeitverhalten
  - Dynamisch
    Kosten hängen vom dynamischen Ablauf ab
    z.B. Kosten für Datenübertragung pro Sendung, Arbeitskosten
    Methodik bisher praktisch nicht vorhanden

1.4 Nicht-funktionale Maße

Abstrakte Sicht auf ein System:



System bearbeitet Jobs

Eingangsparameter

- Ankunftsrate λ (Jobs pro Zeiteinheit)
- (mittlere) Bearbeitungszeit S

Maße für Jobs (traditionelle Leistungsmaße)

- (mittlere) Wartezeit eines Jobs, ohne dass er bedient wird W
- (mittlere) Verweilzeit eines Jobs im System R (:= W+S)
- (mittlerer) Durchsatz X, d.h. Anzahl bearbeiteter Jobs pro Zeiteinheit
- (mittlere) Auslastung U, d.h. prozentualer Anteil der Zeit zu dem das System arbeitet
- (mittlere) Population Q

Neben Mittelwerten/ Erwartungswerten können im stochastischen Fall auch weitere Maße untersucht werden

Maße für Realzeitsysteme:

- Für harte Realzeitbedingungen binäres Ergebnis
  Zeitschranke wird eingehalten ja/nein
  aber immer unter gegeben Voraussetzungen,
  z.B. maximales Lastaufkommen, funktionierendes System
  Parameter:
  - maximale Bearbeitungszeiten,
  - maximale Ankunftsraten
- Für weiche Realzeitbedingungen
  mit Zeitschranke T
  Prob[R < T]
  R ist meistens Zufallsvariable
  Parameter:
  - Verteilung Bearbeitungszeiten, Verteilung Ankunftsabstände

  oder
  - mittlere Bedienzeit und Ankunftsrate + maximale Abweichung in einem endlichen Intervall

Zuverlässigkeitsmaße:

Parameter:

- Fehlerrate λ(t) oft zeitabhängig mit Verteilungsfunktion F(t)
- Mittlere Zeit zwischen zwei Fehlern MTBF $(= \int_0^\infty t \cdot \lambda(t) dt)$
- Zeit bis zum nächsten Fehler MTTF
- Mittlere Zeit bis zur Reparatur MTTR

Resultate:

- Zuverlässigkeit zum Zeitpunkt t: Z(t) = 1 – F(t)
- Verfügbarkeit V = MTBF / (MTBF + MTTR) im Mittel
  Verfügbarkeit zum Zeitpunkt t V(t) muss mit komplexeren Modellen bestimmt werden (siehe Kap.Q9)

Analysetechniken

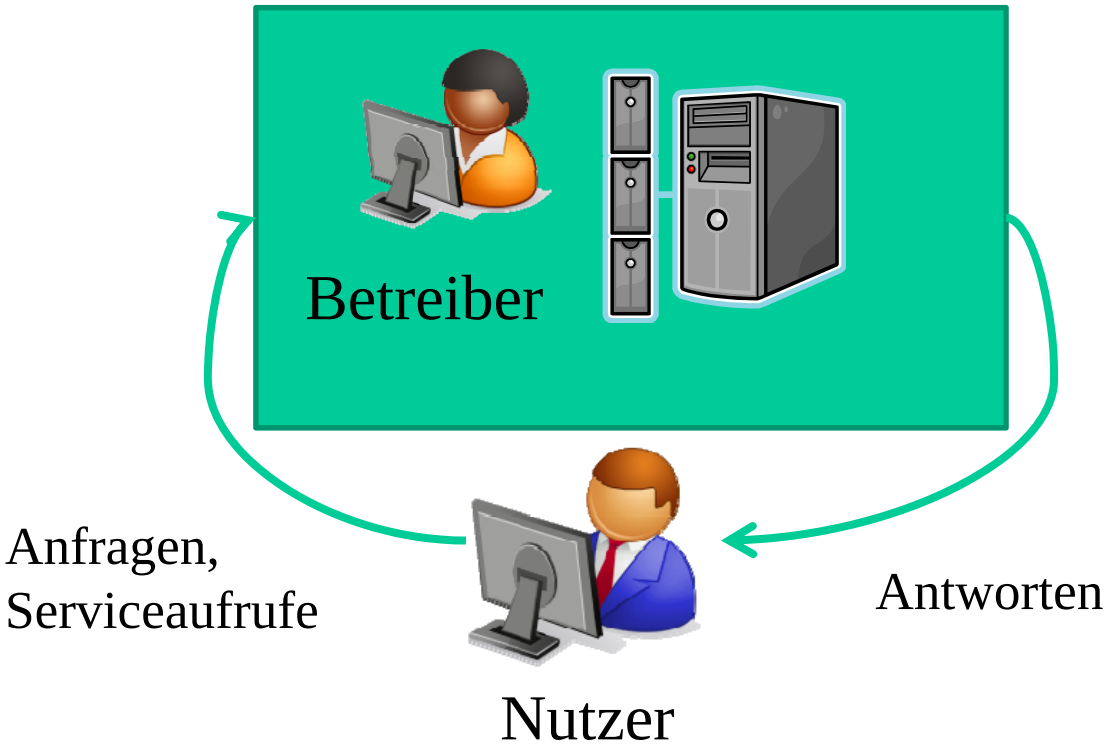
- Messung am realen System
  Ergebnisse deterministisch oder stochastisch
  Bei stochastischen Ergebnissen statistische Auswertung
  (siehe Q2 zur Messung)
- Simulation
  „Nachspielen“ der dynamischen Abläufe im Rechner mit Hilfe eines Modells
  Ergebnisse in der Regel stochastisch dadurch statistische Auswertung notwendig
  (siehe Vorlesung MAO im Bachelor/Wahlpflichtbereich)
- Numerische/Analytische Techniken
  Abstrakte Modelle zur Berechnung der notwendigen Kenngrößen
  Oft Berechnung von Erwartungswerten über potenziell unendliche Zeithorizonte (stationäre Analyse)
  aber auch detailliertere Resultate prinzipiell bestimmbar
  (ab Q3)

1.5 Festlegung und Vereinbarung von Systemanforderungen

Nicht-funktionale Eigenschaften müssen festgelegt werden

Formale Festlegung auf Basis der Sichtweise:

Client-Server, Nutzer-Betreiber, Käufer-Verkäufer, ….



- Festlegung der zu erbringenden Leistungen
  - Antwortzeiten
  - Durchsätze
  - Verfügbarkeit

  im **Service Level Agreement (SLA)**

  Vertrag zwischen Nutzer und Betreiber

SLAs sind Grundlage der Leistungserbringung

$\Rightarrow$ sie müssen validierbar/verifizierbar sein!

Schwierigkeiten bei der Formulierung von SLAs:

Beispiel für eine Anforderung an einen Web-Server:

Die Antwortzeit soll 2 Sekunden betragen!

Welche Antwortzeit, die maximale, die mittlere, ….

Die mittlere Antwortzeit soll 2 Sekunden betragen!

Über welchen Zeitraum wird der Mittelwert berechnet

Die mittlere Antwortzeit über 100 Aufrufe von Web-Seiten berechnet soll 2 Sekunden betragen!

Was ist eine Anfrage, unterschiedliche Web-Seiten sind unterschiedlich groß?

Die mittlere Antwortzeit über 100 Anfragen von Web-Seiten der durchschnittlichen Größe x berechnet soll 2 Sekunden betragen!

Größe x wäre noch genauer zu spezifizieren

Letzte Anforderung auf vorheriger Folie vom Betreiber grundsätzlich nicht erfüllbar, da

Leistung = Maschine × Last

Bisher wurden nur Anforderungen an die Maschine gestellt!

⇒ Beschränkung der Last ist ebenfalls notwendig

Die mittlere Antwortzeit über 100 Anfragen von Web-Seiten der durchschnittlichen Größe x berechnet soll 2 Sekunden betragen, wenn maximal 10 Anfragen pro Minute eintreffen und zwischen dem Eintreffen von zwei Anfragen mindestens 2 Sekunden liegen!

Probleme bleiben:

- Wie validiert man das SLA und wer validiert es?
- Zwischen Nutzer und Betreiber können weitere Komponenten liegen (z.B. Internet), die nicht in das SLA einbezogen sind
- …

Formulierung von SLAs ist komplex, ein allgemein akzeptierter Formalismus existiert bisher nicht!

Aspekte, die in SLAs angesprochen werden müssen:

- Beschreibung der Einbindung des angebotenen Dienstes
- Festlegung der Beteiligten
  (Nutzer, Betreiber + evtl. weitere Akteure)
- Definition des Services und der Anforderungen an den Dienst
- Spezifikation der Last
- Festlegung der Messmethode für Last und Leistung
- Festlegung von Problemberichten und Problemlösungen
- Festlegung von Konsequenzen bei nicht Erreichen der Dienstqualität
- Gültigkeitsdauer und Änderungsintervalle

Beispielsprache zur Spezifikation von SLAs für Web-Services

WSLA (IBM-Entwicklung) :

- Spezifikationssprache für SLAs + Infrastruktur
- Beschreibung der SLAs in XML-Schema
  - Direkte Verbindung mit der Dienstschnittstelle
  - Festlegung der beteiligten Dienste
- Automatische Generierung von Monitordiensten zur Überwachung der SLAs
  - (teilweise) Steuerung des Systems bei Überlast oder Nichteinhaltung von Zusicherungen

Quelle: http://www.ibm.com/developerworks/library/ws-slfram/

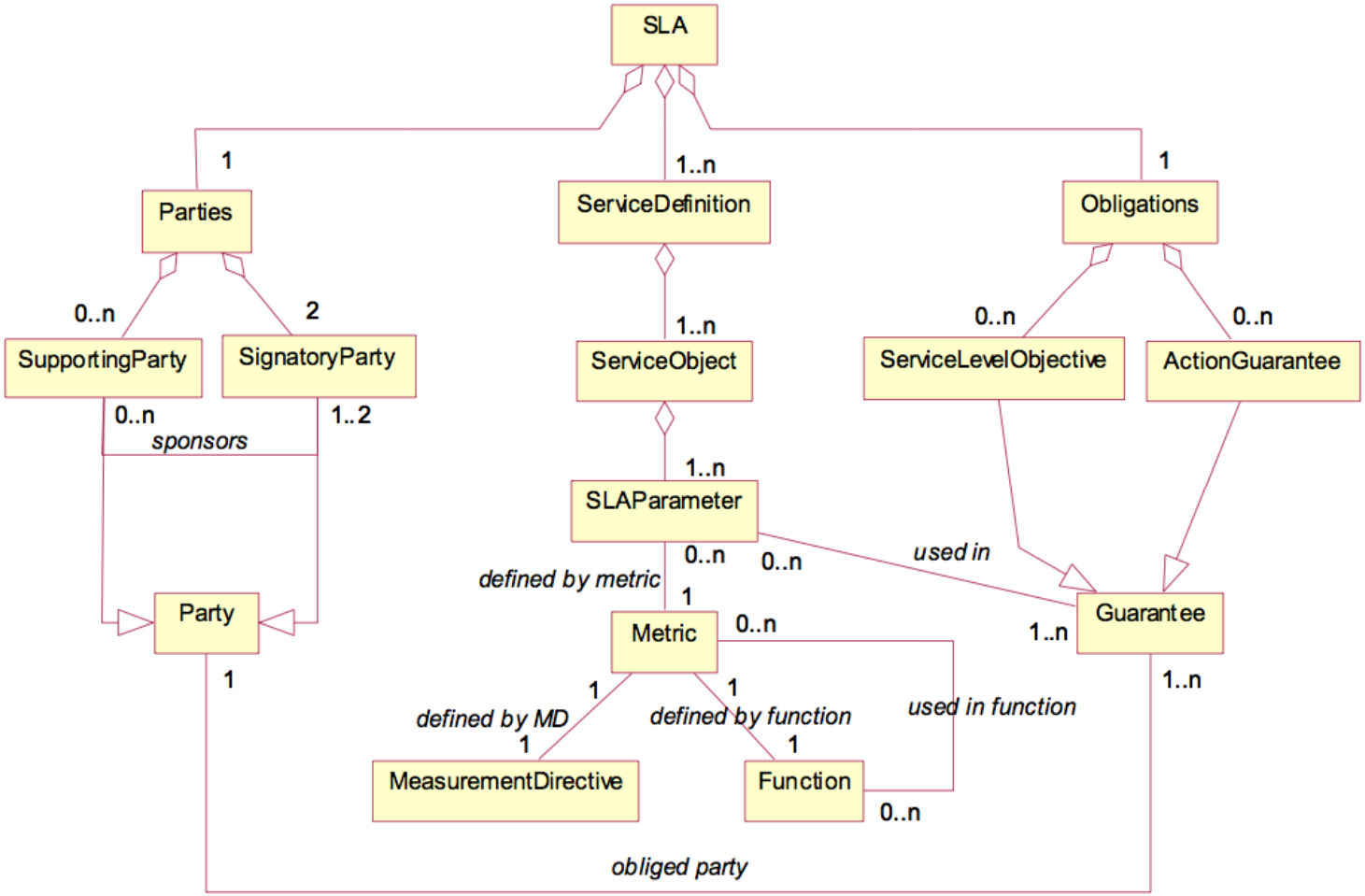
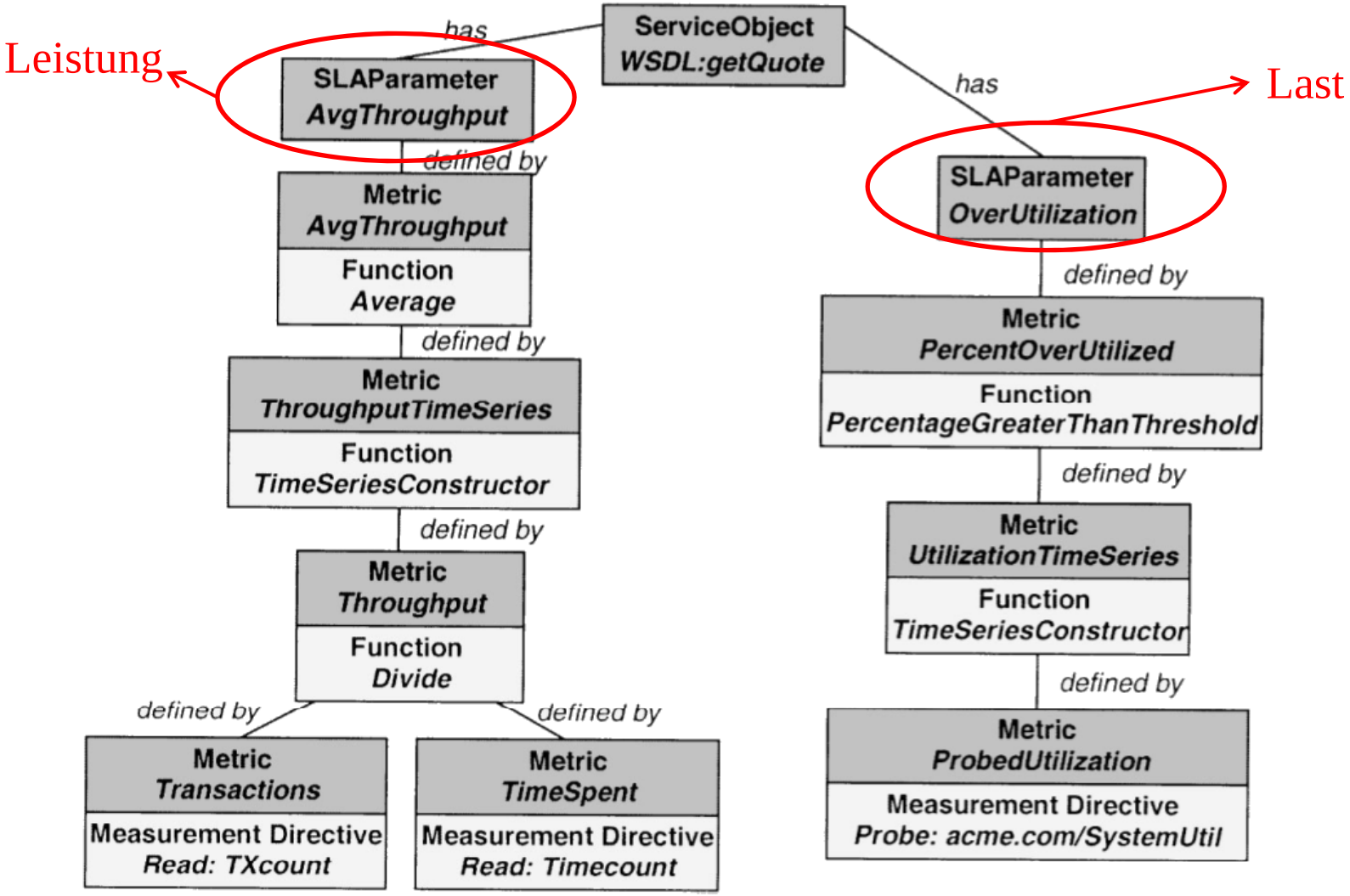


Figure 1: Overview of main WSLA concepts.

Quelle: H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. January 2003



Quelle: A. Keller, and H. Ludwig. The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. *Journal of Network and Systems Management*, vol. 11, no 1, March 2003.

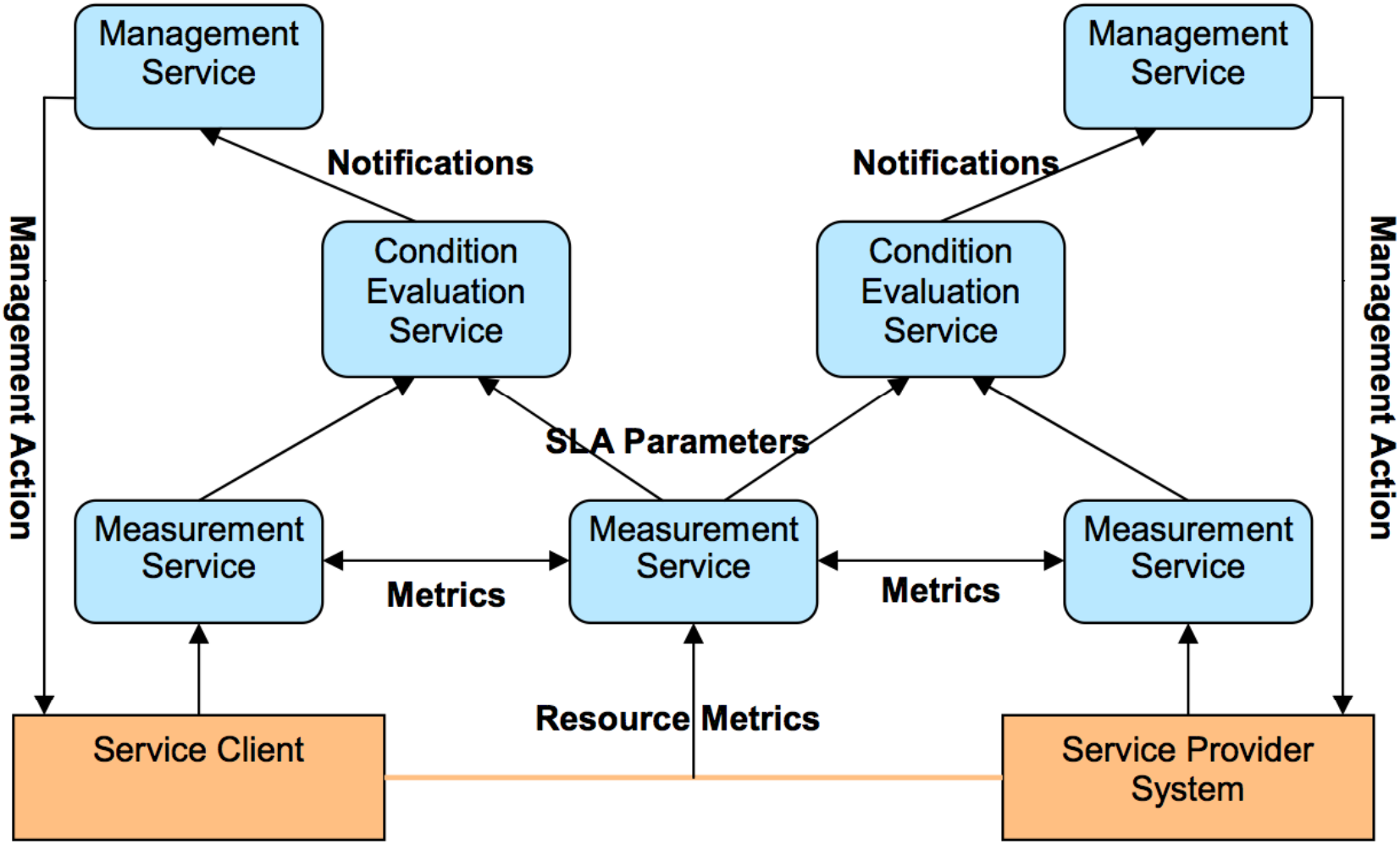


Figure 2: SLA Monitoring Model.

Quelle: A. Dan, H. Ludwig, G. Pacifici. Web Services Differentiation with Service Level Agreements. May 2003, http://www.ibm.com/developerworks/library/ws-slafram/.