TECHNISCHE UNIVERSITÄT CHEMNITZ

Verlässliche Systeme
Wintersemester 2016/2017

Verlässliche Systeme

1. Kapitel
Einführung

Prof. Matthias Werner
Professur Betriebssysteme

1.1 Regularien

Language/Sprache

- Although English is the official course language, German is partly supported:
- The handouts are provided in English **and** German
- Literature is in English anyway.
- Language of a question determines the language of the answer

- Obwohl die Lehrveranstaltung als englischsprachig ausgewiesen ist, werden deutschsprachig Studierende teilweise unterstützt:
- Das Kursmaterial (Handouts) wird in Deutsch **und** Englisch bereit gestellt
- Literatur ist durchweg englisch
- Fragen werden in der bei der Fragestellung benutzten Sprache beantwortet

Vorlesung

- **Verlässliche Systeme** (565130)
  (für Fans der alten Rechtschreibung: „Verläßliche Systeme“)
- **Dozent:** Prof. Matthias Werner
  - matthias.werner@informatik.tu-chemnitz.de
  - Material und aktuelle informationen unter
    http://osg.informatik.tu-chemnitz.de/lehre/ds
- **Ort & Zeit:**
  - Mo, 15.30 – 17.00 Uhr
  - Raum 1/219

Übung

- **Übungsleiter:** Jafar Akhundov
- **Ort & Zeit:**
  - Mo, 11.30 – 13.00 Uhr
  - Raum 1/B006
- **Inhalt:**
  - Klärung von Fragen zur Vorlesung
  - Überprüfen von Lösungswegen von Hausaufgaben
  - Rechnen von Beispielaufgaben
- Übungen sind (wie die Vorlesung) ein **Angebot**
  - **Sie setzen Vorbereitung und Mitarbeit der Teilnehmer voraus**
  - Bei Mangel an Interesse/Vorbereitung spielt der Übungsleiter nicht Alleinunterhalter, sondern beendet die Übung

Zur Beachtung

- In allen Lehrveranstaltungen ist das Handy abzuschalten!

Anrechenbarkeit

- Anrechenbarkeit für:
  - Master Informatik
  - Master Angewandte Informatik
  - Master Automotive Software Engineering
  - Master Biomedizinische Technik
  - Master Web Engineering

  auslaufend:
  - Master High-Performance Computing
  - Diplom Informatik
  - Diplom Angewandte Informatik
- Studenten anderer Richtungen brauchen eine Anerkennung
- Prüfung
  - schriftliche Prüfung
  - Anmeldung über das Prüfungsamt

Hinweis

- Lernen des Stoffes ist eine notwendige aber nicht hinreichende Bedingung zum Bestehen des Moduls
- Sie sollten den Stoff auch **verstehen**.

Merke

Sie sollten nicht nur **Was**-Fragen beantworten können, sonder auch **Warum**-Fragen!

Literaturempfehlungen

- Die Vorlesung folgt keinem einzelnen Lehrbuch
- Jedoch sind einige Standardwerke empfehlenswert:

[AL82] Thomas Anderson und Pete A. Lee. *Fault Tolerance – Principles and Practice*. Prentice Hall, 1982

[Pra96] Dhiraj K. Pradhan, Hrsg. *Fault Tolerant Computer Systems*. Prentice Hall, 1996

[SS95] Daniel P. Siewiorek und Robert S. Swarz. *The Theory and Practice of Reliable Systems Design*. Digital Press, 1995

Literaturempfehlungen (Forts.)

- Zusätzlich wird folgendes Material zur Verfügung gestellt:
  - Handout der Folien im Netz
    - Ich bemühe mich, sie vorjeweiligen Behandlung in der Vorlesung ins Netz zu stellen
    - Können durch eigene Notizen ergänzt werden
    - Folien werden im Format 2x2 pro Seite zur Verfügung gestellt; wer es anders will, kann einschlägige Tools nutzen
  - Originalartikel (markiert durch )
    - Link auf der Homepage
    - ➡ Account beim TUC Web-Trust-Center nötig

Handout

- Beispielsscript (bash), um Handout in 1x1-Format zu konvertieren
  - Benötigt ghostscript und pdf-Toolkit
  - *YMMV* 😊

```
#!/usr/bin/env bash
if [ -z "$1" ] || [ ! -f "$1" ] || \
   [ `file -Ib $1 | cut -f1 -d' '` != "application/pdf;" ];
then
  echo "No valid input file"
  exit 1
fi
x=( 0 -421 0 -421)
y=(-297 -297 0 0)
temp=`mktemp -u /tmp/pdf-cv-XXXXX`
for i in {0..3}; do
  gs -q -dNOPAUSE -dBATCH -P- -dSAFER -sDEVICE=pdfwrite \
   -g4210x2975 -sOutputFile=${temp}$i.pdf \
   -c "<</PageOffset [${x[$i]} ${y[$i]}]>> setpagedevice" \
   -f $1;
done
pdftk ${temp}?.pdf shuffle output ${1/.pdf/-1x1.pdf}
rm ${temp}?.pdf
```


1.2 Verlässlichkeit

Begriff der Verlässlichkeit

- Früher: (bis ca. 80er Jahre)
  Verlässlichkeit ist Eigenschaft fehlertoleranter Systeme
- Heute:
  Verlässlichkeit (*dependability*) ist ein Oberbegriff, der eine Vielzahl von Konzepten und Maßen abdeckt.

Allgemeine Frage:

„Wie umgehen mit unerwarteten/unerwünschten Ereignissen?“

- Unerwartete Ereignisse = Störungen
- z.B. sind Angriffe **beabsichtigte Störungen**

Begriff der Verlässlichkeit (Forts.)

Definition 1.1 (LAPRIE 1993)

Dependability is defined as the trustworthiness of a computer system such that reliance can justifiable be placed on the service it delivers.
The service delivered is it behaviour as it is perceptible to its user(s); a user is another system (human or physical) which interacts with the former.

- **Achtung:** nach der Definition fällt auch unerwünschtes Verhalten unter den Dienstbegriff.
  - Beispiel: Schaden gegenüber Dritten

Begriff der Verlässlichkeit (Forts.)

- Aspekte der Verlässlichkeit
  - Zuverlässigkeit
  - Verfügbarkeit
  - Sicherheit (*safety*)(nach außen)
  - Sicherheit (*security*)(nach innen)
    - Vertraulichkeit
    - Integrität
  - Wartbarkeit
  - Korrektheit (Wert und Zeit)

➡ Verlässlichkeitseigenschaften sind **nichtfunktionale** Eigenschaften

Nichtfunktionale Eigenschaften

- Üblicherweise interessiert die eigentliche Funktion oder ihr Ergebniswert
  - Programmiersprachen
  - Interfacebeschreibungen
- Nichtfunktionale Eigenschaften sind „Randaspekte“
  - Ausführungszeit
  - Ressourcenverbrauch
  - Zuverlässigkeit
  - Sicherheit

⋮

Probleme...

... mit nicht-funktionalen Eigenschaften

- Schwierig zu definieren
- Schwierig zu abstahieren
- **Divide et impera (teile und herrsche) funktioniert häufig nicht gut**
- Abhängigkeiten zwischen verschiedenen nicht-funktionalen Eigenschaften
- Vielfach probabilistische Abhängigkeiten

Genauere Betrachtung führt oft zu überraschenden Ergebnissen.

1.3 Fallbeispiele

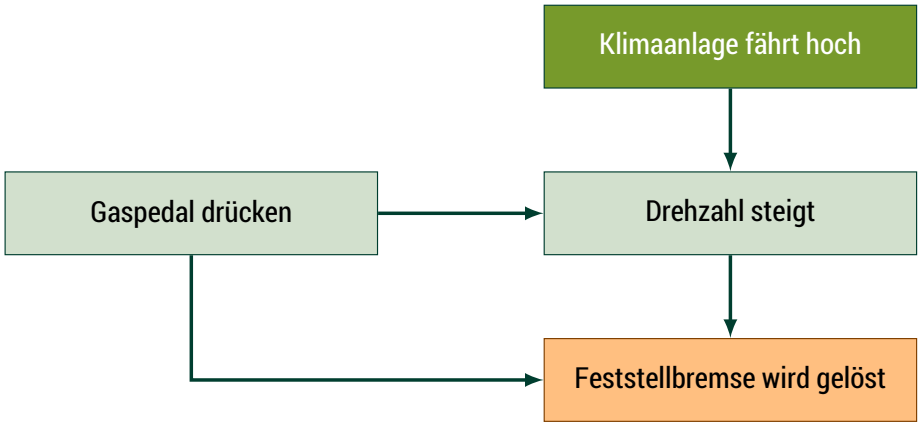
1.3.1 Beispiel I: Spezifikation vs. Implementation

- Automatische Handbremse
  - Beispiel: VW Passat
  - Wird beim Anfahren automatisch gelöst

Bildquelle: www.autozeitung.de

Automatisches Lösen der Handbremse

- Was gemeint war
- Was implementiert wurde
- Was passiert

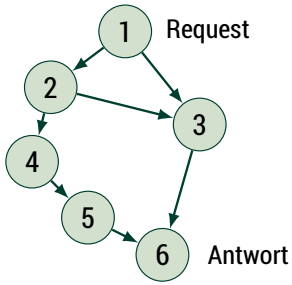


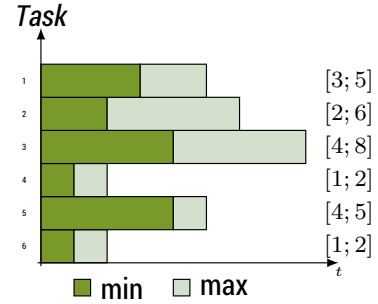
1.3.2 Beispiel II: Paradigma vs. Maß

- Dienst hängt von der Ausführung von 6 Tasks ab
- Beendigungswahrscheinlichkeit jeder Task ist innerhalb eines Intervalls gleichverteilt
- Ein oder zwei Prozessoren stehen zur Verfügung
- Deadline: 25 Zeiteinheiten
- Fehlerrate: $\lambda = 0.01$, $R(t) = e^{-\lambda \cdot t}$

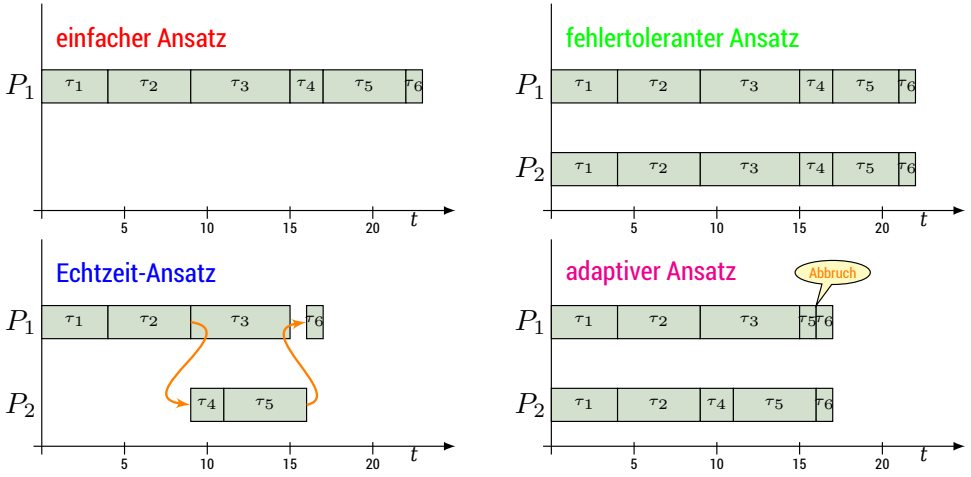
Fragestellungen

- Sollen ein oder zwei Prozessoren eingesetzt werden?
- Welche Schedulingstrategie soll benutzt werden?

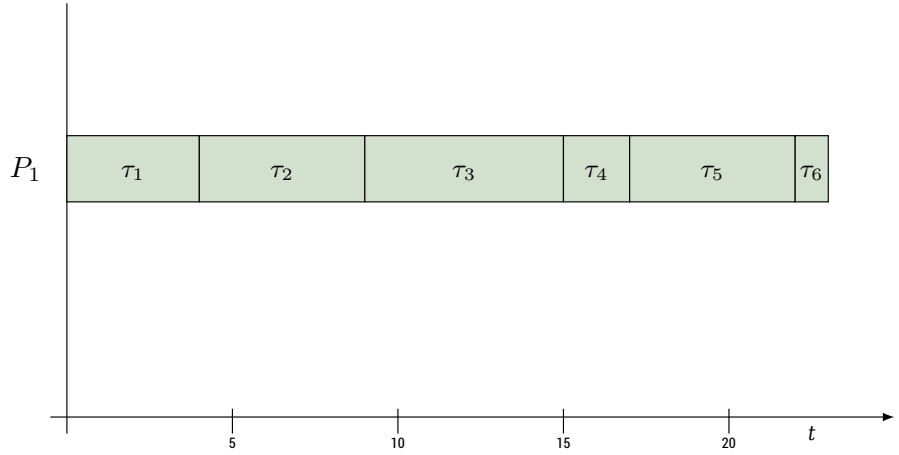




Alternative Designs



Alternatives Design I: Einfacher Ansatz



- Ein Prozessor, sequentielle Ausführung
- keine Fehlertoleranz, keine Echtzeit

... etwas Mathematik

Erfolgswahrscheinlichkeit

$$\mathcal{R}(t) = P[t_{ready} \leq t] = \int_0^t (e^{-\lambda\tau}\epsilon_1(\tau)) * \cdots * (e^{-\lambda\tau}\epsilon_6(\tau))d\tau$$

Dabei ist

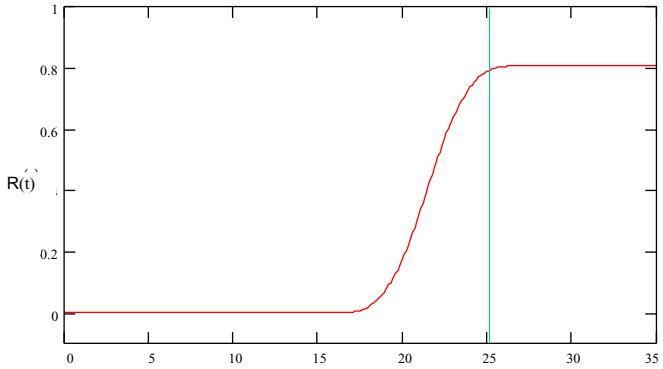
$e^{-\lambda t}$: Zuverlässigkeit
(Überlebenswahrscheinlichkeit im Intervall $[0, t]$)

$\epsilon(t)$: Beendigungswahrscheinlichkeit zum Zeitpunkt t bei Fehlerfreiheit
(Antwortzeitverhalten)

$f_x * f_y$: Faltung der Funktionen f_x und f_y

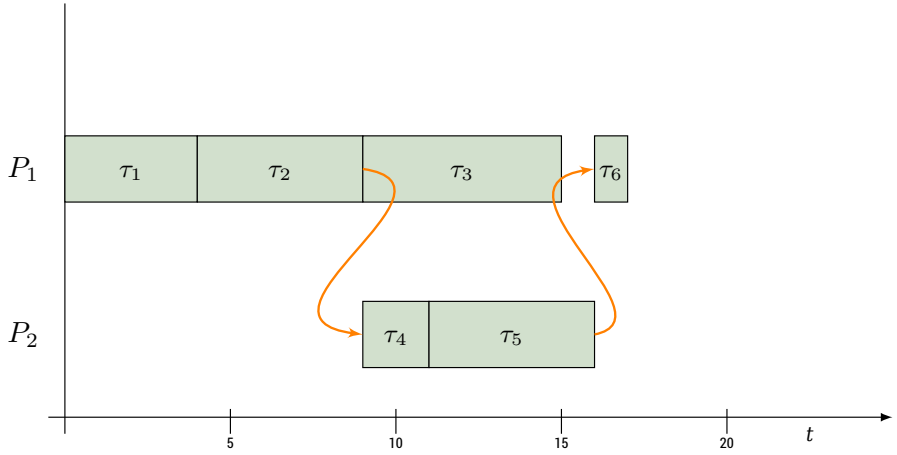
... etwas Mathematik (Forts.)

- Erfolgswahrscheinlichkeit



$\mathcal{R} = 78.85\%$

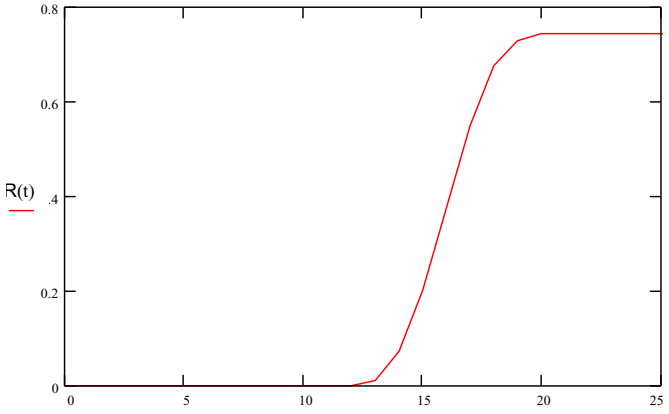
Alternatives Design II: Echtzeit



- Zwei Prozessoren, verteilte Ausführung
- Keine Fehlertoleranz, aber Echtzeitgarantie

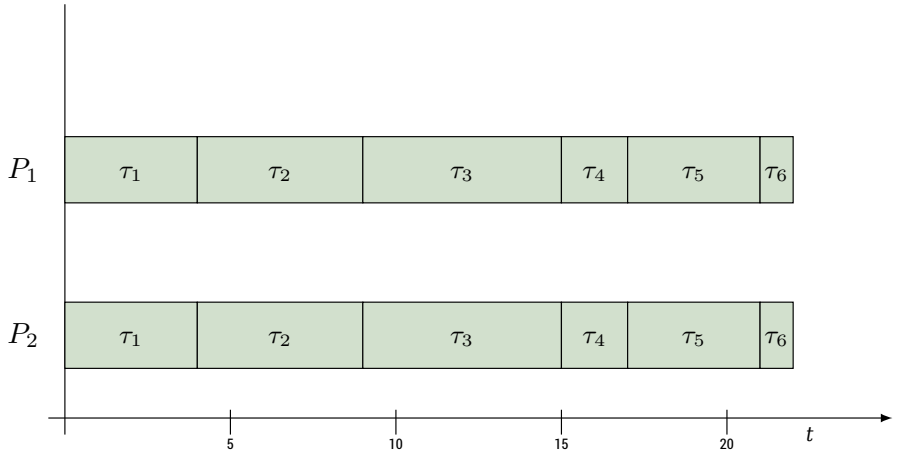
Alternatives Design II: Echtzeit (Forts.)

- Erfolgswahrscheinlichkeit



$\mathcal{R} = 74.48\%$

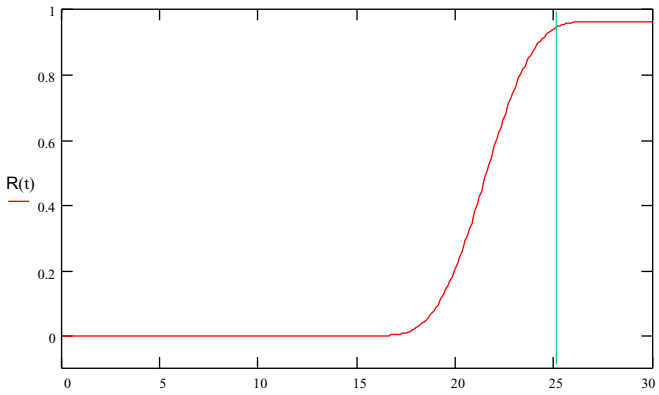
Alternatives Design III: Echtzeit



- Zwei Prozessoren, redundante Ausführung
- Fehlertoleranz, aber keine Echtzeit Garantie

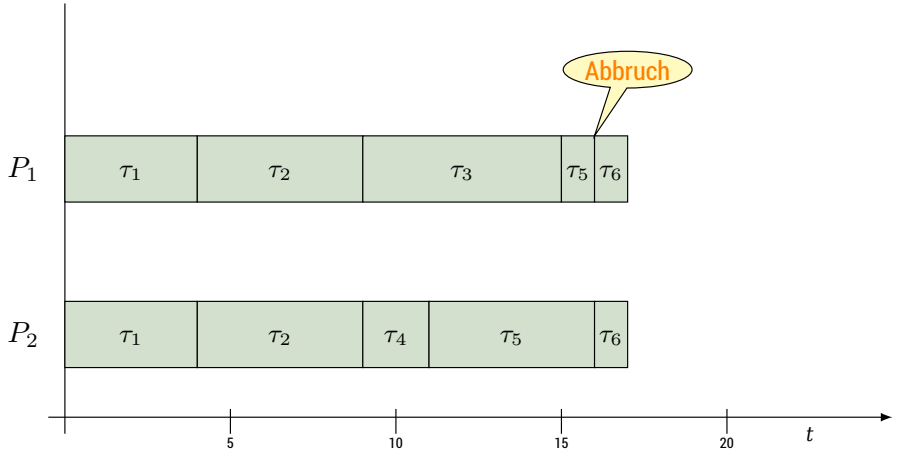
Alternatives Design III: Echtzeit (Forts.)

- Erfolgswahrscheinlichkeit



$\mathcal{R} = 94.02\%$

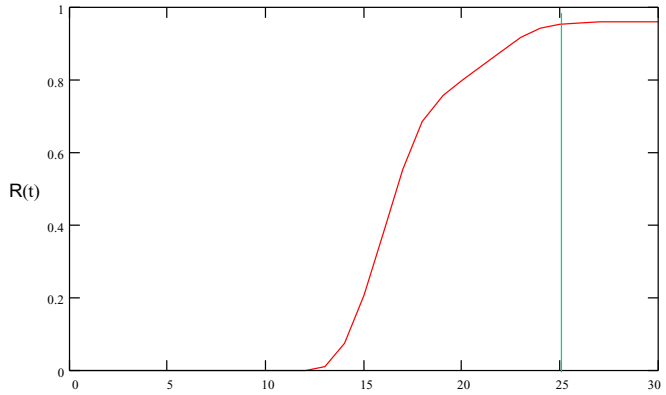
Alternatives Design IV: Adaptivität



- Zwei Prozessoren, Eager-Scheduling
- Fehlertoleranz und Echtzeit je nach Situation

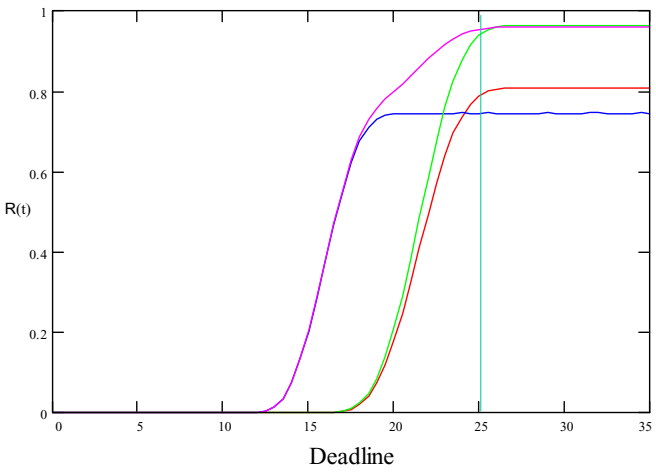
Alternatives Design IV: Adaptivität (Forts.)

- Erfolgswahrscheinlichkeit



$\mathcal{R} = 95.43\%$

Vergleich der Designs



- einfacher Ansatz
- Echtzeit
- Fehlertoleranz
- Adaptivität

Erkenntnisse aus der Fallstudie

- Schon einfache Systeme verhalten sich oft unerwartet
- Vorsicht bei „Verbesserungen“:
  Evtl. wird das Gegenteil erreicht
- Fragen beim Entwurf:
  - Was will ich erreichen? (Was ist für den Erfolg entscheidend?)
  - Was sind die Randbedingungen? (z.B. Ressourcen)

1.4 Inhalt der Vorlesung

Schwerpunkte der Vorlesung

- Fokus auf
  - Konzepten
  - Bewertung und Modellierung
  - (Klassische) Fehlertoleranz
- Nicht so sehr:
  - Echtzeit (➡ Empfehlung: Echtzeitsysteme, Sommersemester)
  - Sicherheit (➡ Empfehlung: Prof. Lefmann)
- können aber Einfluss haben (siehe Fallstudie)

Interessante Fragestellungen

- Wie kann man die Verlässlichkeit eines Systems bewerten?
- Warum reicht bei „bösartigen“ Fehler nicht die einfache Redundanz?
- Wie kann man fehlerhaftes Verhalten modellieren?
- Welche Ansätze gibt es für Tests?
- Was ist der Unterschied zwischen RAID 1+0 und 0+1?

Themen

- (Auffrischung der) Grundlagen der Stochastik
- Verlässlichkeitsmaße und Systembewertung
- Störungsmodelle
- Modellierung
- Tests und Fehlerdiagnose
- Konsens und Byzantinische Fehler
- Verifizieren und Testen von Software
- Fehlertoleranz in Software
- Fallstudien