Dependable Systems
Winter term 2015/2016

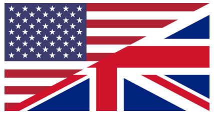
Dependable Systems

1. Chapter
Introduction

Prof. Matthias Werner
Operating Systems Group

1.1 Formalities

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

Lecture

- **Dependable systems** (565130)
- **Lecturer:** Prof. Matthias Werner
  - `matthias.werner@informatik.tu-chemnitz.de`
  - You will find supporting material as well as up-to-date information at `http://osg.informatik.tu-chemnitz.de/lehre/ds?lang=en`
- **Time & location:**
  - Mo, 13.45 – 15.15
  - Room 1/219

Tutorials

- **Tutor:** Jafar Akhundov
- **Time & location:**
  - Fr, 7.30 – 9.00
  - Room 1/205
- **Contents:**
  - Answering of open questions
  - Discussing of test problems' solutions
  - Consideration of example problems
- Tutorials are voluntary (lectures even so)
  - **Preparation and participation are required!**
  - Lack of preparation/participation may lead to cancelation

Please note

- Switch off your mobile phone!

Also please note

- The lecture starts at 13:45 – please be in time

Credits

- Course can be taken by students of following programs:
  - Master Computer Science
  - Master Applied Computer Science
  - Master Automotive Software Engineering
  - Master Biomedical Technology
  - Master Web Engineering

  deprecated:
  - Master High-Performance Computing
  - Diplom Computer Science
  - Diplom Applied Computer Science
- Students of other programs need approval
- Exam
  - Written exam
  - Registration with central examination office is required

Advice

- Learning is a necessary but not sufficient condition to pass this course
- You also should **understand** the topics

Note

Be able to answer no only to **What** questions, but also to **Why** questions!

Literature

- There is no single textbook for this course
- However, the following textbooks may be of use:

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

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

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

Literature (cont.)

- In addition, the following materials are provided:
  - Slides as handout (via homepage)
    - I'll try hard to provide it in advance to the related lecture
    - You may add your own notes
    - Handout is in 2x2 layout; if you need another layout, convert it by proper tools
  - Original articles (marked by)
    - Link at homepage
    - ➡ You need TUC trust center account

Handout

- Example script (bash) to convert the handout into 1x1 layout
  - Needs ghostscript and 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 Dependability

Term

- Past (appr. until 80s)
  Dependability is property of fault-tolerance systems
- Today:
  "Dependability" is umbrella term, that covers quite a number of concepts and measures.

General question:

"How to deal with unexpected/undesired events?"

- Unexpected event = impair
- E.g., attacks are **intended impairs**

Term (cont.)

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.

- **Attention:** following the definition, also unintended/undesired behavior is a service.
  - Example: damage for a third party

Term (cont.)

- Aspects
  - Reliability
  - Availability
  - Safety
  - Security
    - Privacy
    - Integrity
  - Maintainability
  - Correctness

➡ Dependability properties are **non-functional** properties

Non-functional Properties

- Usually, we are interested in the function or the result
  - Programming languages
  - Interface description
- Non-functional properties are “add-on” aspects
  - Execution time
  - Resource consumption
  - Reliability
  - Security

    ⋮

Issues...

... with non-functional properites

- Hard to define
- Hard to abstract
- **Divide et impera does not work in many cases**
- Interdependencies between different non-functional properties
- Frequently, probabilistic behavior

In-deep consideration often provide surprising results.

1.3 Case Studies

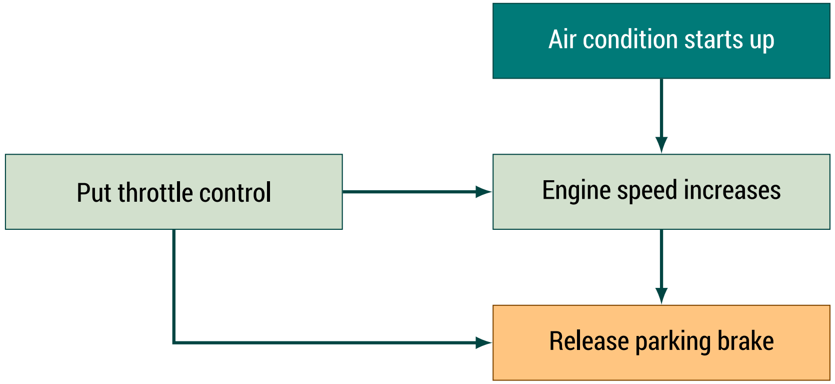
Example I: Specification vs. Implementation

- Automatic parking brake
  - Example: VW Passat
  - Will be automatically released, when car accelerates

Picture from: www.autozeitung.de

Automatic Parking Break Release

- What was intended
- What has been implemented
- What did happen

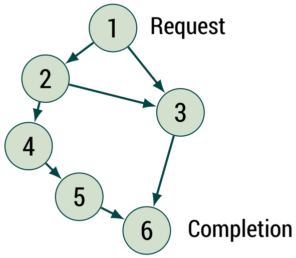


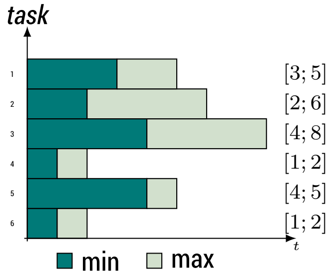
Example II: Measure vs. Measurement

- Service requires the execution of 6 task
- Probability of finishing is equally distributed within an interval
- One or two processors
- Deadline: 25 time units
- Fault rate: $\lambda = 0.01$, $R(t) = e^{-\lambda \cdot t}$

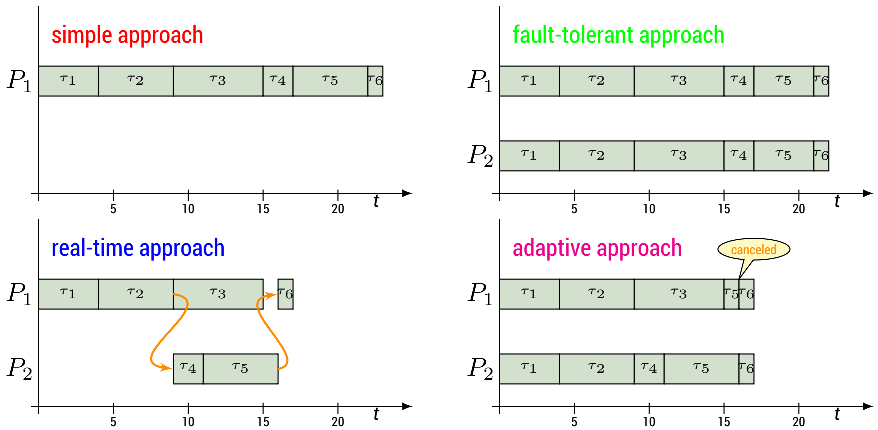
Questions

- How many processors should be used (one or two)?
- What scheduling policy should be used?

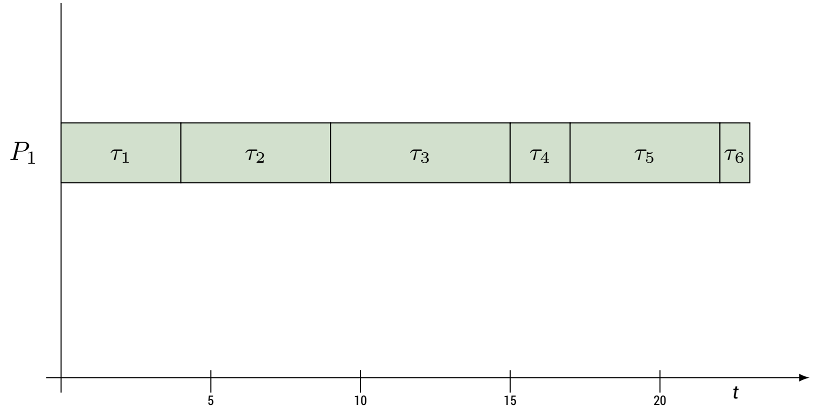




Design Alternatives



Design I: Simple Approach



- One processor; sequential execution
- Neither fault tolerance, nor real time

... Some Maths

Probability of success

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

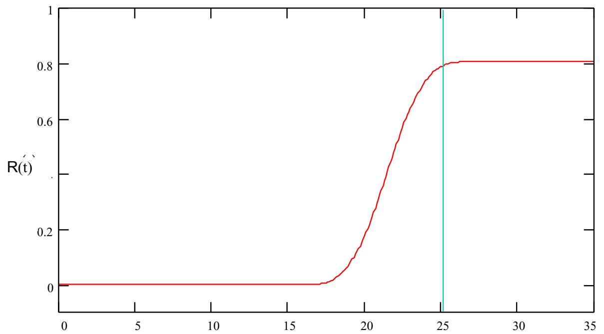
$e^{-\lambda t}$: reliability
(probability to survive during interval $[0, t]$)

$\epsilon(t)$: probability to finish till time t in case of no failure (response behavior)

$f_x * f_y$: convolution of f_x and f_y

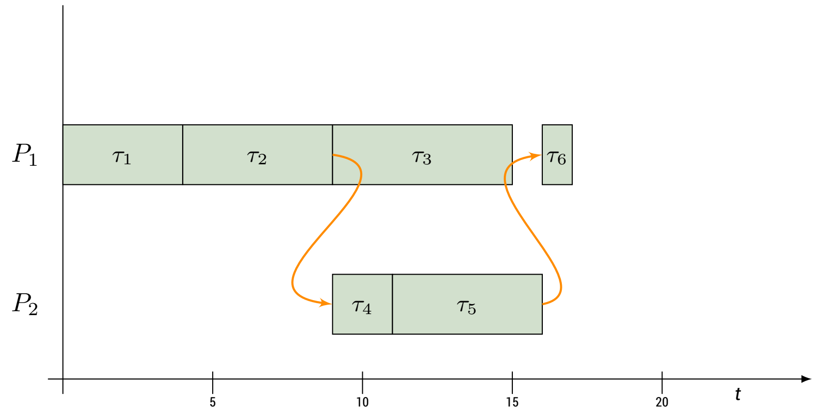
... Some Maths (cont.)

- Probability of success



$\mathcal{R} = 78.85\%$

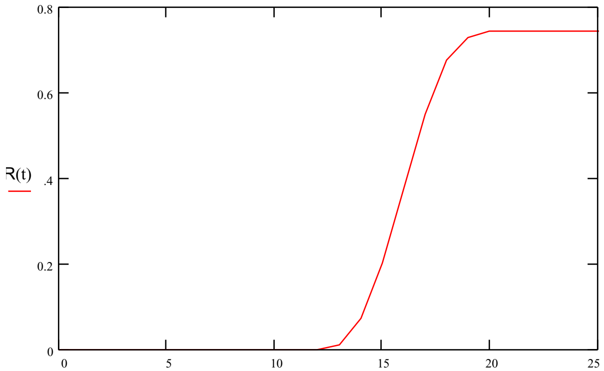
Design Alternative II: Real Time



- Two processors, distributed execution
- No fault tolerance, but real time

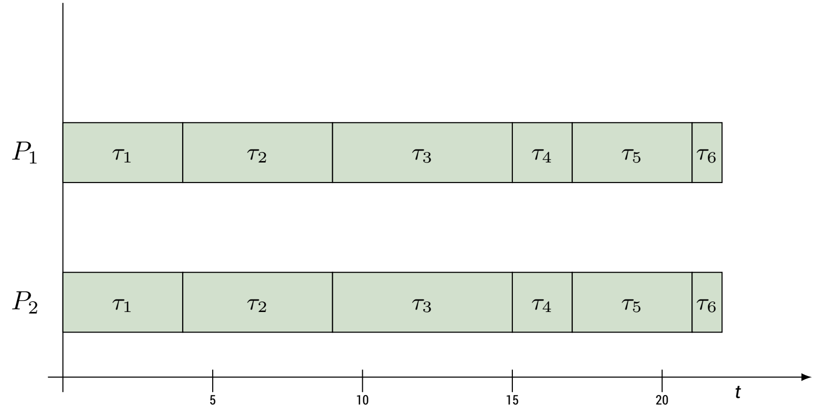
Design Alternative II: Real Time (cont.)

- Probability of success



$$\mathcal{R} = 74.48\%$$

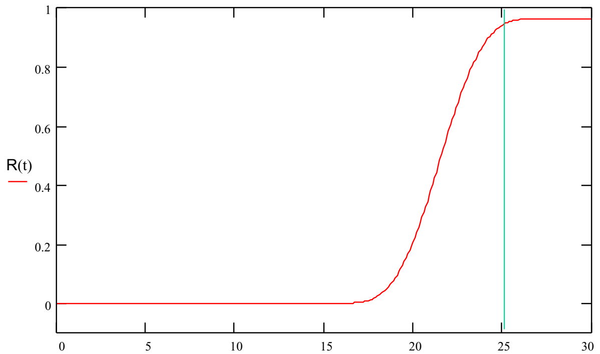
Design Alternative III: Fault Tolerance



- Two processors, redundant execution
- Fault tolerance, but no guaranteed feasability

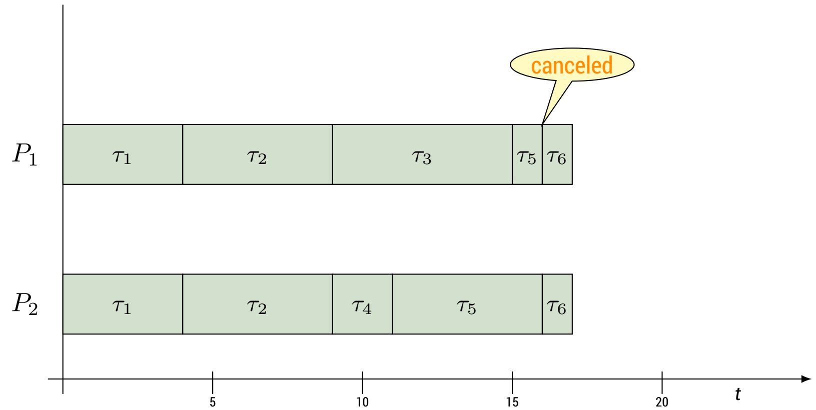
Design Alternative III: Fault Tolerance (cont.)

- Probability of success



$$\mathcal{R} = 94.02\%$$

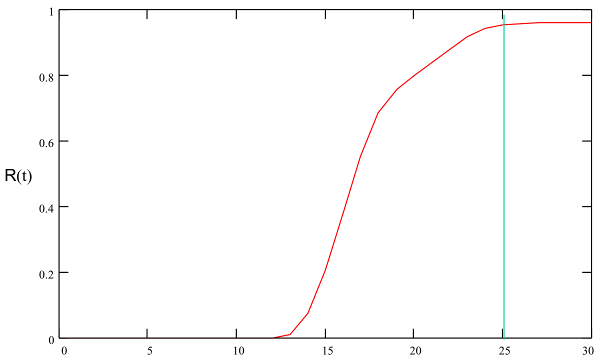
Design Alternative IV: Adaptivity



- Two processors; eager scheduling
- Fault tolerance and real time, dependent on situation

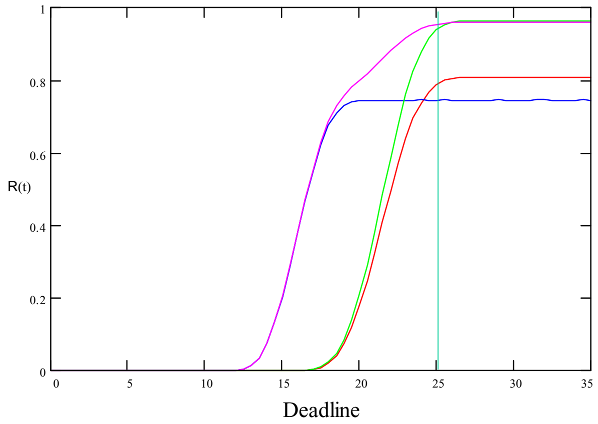
Design Alternative IV: Adaptivity (cont.)

- Probability of success



$\mathcal{R} = 95.43\%$

Comparison of Designs



- simple approach
- real time
- fault tolerance
- adaptivity

Lessons Learned

- Even simple systems may behave unexpected
- Beware of “improvements”:
  Possibility that they make things worse
- Design issues:
  - What is intended? (what is success?)
  - What are the side conditions? (e.g., resources)

1.4 Course Contents

Course’s Key Aspects

- Focus on
  - Concepts
  - Evaluation and modeling
  - (Classic) fault tolerance
- No focus on:
  - Real time (➡ real-time course (summer term))
  - Security (➡ Prof. Lefmann)
- but may interfere (c.f., case study)

Interesting Problems

- How to evaluate system's dependability
- Why is simple redundancy not sufficient in case of "malicious" faults?
- How to model faulty behavior?
- What test approaches do exist?
- What is the difference between RAID 1+0 and 0+1?

Topics

- (Recap of) stochastic basics
- Dependability measures and system evaluation
- Impairment models
- Modeling
- Tests and fault diagnosis
- Consensus and Byzantine Faults
- Verification and testing of software
- Fault tolerance in software
- Case studies