Verlässliche Systemsoftware

Lehrveranstaltungskonzept & Organisation

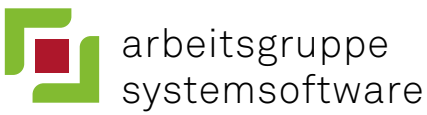
Peter Ulbrich

Lehrstuhl für Informatik 12 – Arbeitsgruppe Systemsoftware

Technische Universität Dortmund

`https://sys.cs.tu-dortmund.de`

06. April 2022



Vorwort

Die Lehrveranstaltung ist grundsätzlich für alle Studierende offen. Sie verlangt allerdings gewisse Vorkenntnisse.

Voraussetzungen

- **Betriebssysteme**,
  Allgemeine Grundlagen der technischen/praktischen Informatik
- **C**[1] / C++, (Java)
- Ein gewisses Maß an **Durchhaltevermögen**
- Freude an systemnaher und **praktischer Programmierung**

Wir arbeiten mit eingebetteten Systemen!

Die meisten sind überrascht, wie viel Spaß das macht :-)

[1] Selbsteinschätzung per Test im Moodle-Raum.

Gliederung

1 Vorwort

2 Die Veranstaltung
- Lernziele

3 Organisatorisches
- Die Beteiligten
- Vorlesung und Übung
- Vorlesung und Übung

Verlässliche Systemsoftware – Motivation

☞ **Systemsoftware** (von eingebetteten Systemen) **begeistert**

- Neue Verfahren und Architekturen zu entwickeln, ist spannend!
- Mikrokerne schotten Programme räumlich voneinander ab
- Verschlüsselungsalgorithmen garantieren Datensicherheit
- Echtzeitsysteme erlaubten ein zeitlich vorhersagbares Verhalten

  → Prioritätsorientierte Einplanung von Arbeitsaufträge
- . . .

⚠ **Das ist jedoch nur die halbe Miete**

- Erfordert möglichst fehlerfreie Implementierungen
- Implementierung muss mit Laufzeitfehlern umgehen können

→ Verfahren und Architekturen müssen korrekt arbeiten!

☞ **Wie lassen sich Fehler vermeiden bzw. behandeln?**

Verlässliche Echtzeitsysteme – Ausrichtung

Im Fokus dieser Veranstaltung: **Software**

1 Zuverlässige (robuste) Software entwickeln

- Robustheit gegenüber externen Fehlern (zur Laufzeit)
  - Wie erkenne und toleriere ich solche Fehler?
- Wie testet man, ob man korrekt mit solchen Fehlern umgeht?
- Hier „forschen“ wir (hoffentlich auch zusammen mit euch)

2 Software zuverlässig entwickeln

- Wie kommt man zu einer möglichst fehlerfreien Implementierung?
- Welche Werkzeuge helfen mir dabei?
  - Was tun diese Werkzeuge eigentlich?
  - Welche Grenzen haben diese Werkzeuge demzufolge?
- Hier „lernen“ wir zusammen mit euch

Lernziele – Robustheit

☞ **Zuverlässige (robuste) Software entwickeln**

- Maskieren von Fehlern durch Redundanz
  - → Replizierte Ausführung
  - – Homogene und heterogene Redundanz
- Härtung von Datenstrukturen und Kontrollfluss
  - → Informationsredundanz
  - – In Daten mithilfe von z.B. Prüfsummen
  - – In Berechnungen/Kontrollfluss mithilfe arithmetischer Codierung
- Evaluierung von Fehlertolanzmaßnahmen
  - – Fehlerinjektion und Testen

☞ Anknüpfungspunkte für den praktischen Einsatz aufzeigen

- Niemand braucht das 1001. Fehlertoleranzprotokoll!
  - Das den gegenwärtigen Stand der Kunst nicht reflektiert
  - Obendrein vielleicht fehlerhaft ist

Lernziele – Zuverlässigkeit

☞ Software zuverlässig entwickeln

- Typische Laufzeitfehler in C/C++-Programmen suchen und finden
  - Nullzeiger, Ganzzahlüberläufe, nicht initialisierte Speicherstellen, ...
  - → Durch Testen oder mittels statischer Analysewerkzeuge
- Testüberdeckung: Wie gut hat man getestet?
  - die Testüberdeckung für ein gegebenes Programm messen
  - Gibt es Zusammenhänge zwischen der Testüberdeckung, der Testfallanzahl und anderen Metriken?
- Design-by-contract: statische, werkzeug-gestützte Verifikation
  - Formulierung/Verifikation von Nachbedingungen für kleine C-Programme
  - Mithilfe von Werkzeugen (AbsInt Astrée) wie sie auch Airbus einsetzt

☞ Vorurteile gegenüber formalen Methoden abbauen

- Keine unverwendbaren Monster mehr
  - Vollbringen aber auch keine Wunder
  - Ihre Anwendung ist noch immer mühsam, aber sie lohnt sich

Bedeutung der Übungen

- Tafelübungen $\rightsquigarrow$ „*learning by exploring*“
  - Besprechung der Übungsaufgaben, Skizzierung von Lösungswegen
  - Vertiefung des Vorlesungsstoffes, Klärung offener Fragen
- Rechnerarbeit $\rightsquigarrow$ „*learning by doing*“
  - Selbstständiges Bearbeiten der Übungsaufgaben am Rechner
    - Abgabe der bearbeiteten Übungsaufgaben
    - Klärung von Unklarheiten/Problemen bei/mit den Übungsaufgaben

$\rightarrow$ Bereitet euch vor! Wir erwarten konkrete Fragen!

Der, die, das.
Wer, wie, was?
Wieso, weshalb, warum?
Wer nicht fragt, bleibt dumm!

Gliederung

1 Vorwort

2 Die Veranstaltung
- Lernziele

3 Organisatorisches
- Die Beteiligten
- Vorlesung und Übung
- Vorlesung und Übung

sys.cs.tu-dortmund.de/People/*

Veranstaltungskonzept

- **Wissensvermittlung**
  - Stoff der Vorlesung $\rightsquigarrow$ Folien, Videoaufzeichnung und Screencasts
  - Stoff der Tafelübung $\rightsquigarrow$ Folien und Screencasts
- **Praktische Arbeit**
  - Übungs- und Programmieraufgaben
  - In Präsenz oder Distanz
- **Interaktion**
  - Vorlesung und Tafelübung $\mapsto$ Wöchentliche Präsenzveranstaltungen
  - Rechnerübung $\mapsto$ Betreute Laborarbeit, Heimarbeit (VC nach Absprache)
  - Matrix-Raum: `https://matrix.to/#/#vss-helpdesk:fachschaften.org`

Vorlesungsbetrieb und Lehrmaterialien

Termine bis KW 28

- Mittwoch, 14:15 – 15:45, 1.055 (OH12)

Ausnahmen

- ggf., siehe Moodle

- Handzettel (engl. *handout*) sind verfügbar wie folgt:
  - `https://moodle.tu-dortmund.de/course/view.php?id=34415`

☞ Fachbegriffe der Informatik (Deutsch ⟷ Englisch)
  - `www.aktionlebendigesdeutsch.de`

⚠ Änderungen und Hinweise: siehe Moodle (unbedingt einschreiben!)

Tafelübung

Termine bis KW 28

- Donnerstag, 08:15 – 9:45, 1.055 (OH12)
- Alternativ Rechnerübung zu diesem Termin

Ausfälle

- 26.05, 16.06.
- siehe Moodle-Raum

Übung

- Übungsaufgaben sind bevorzugt in 3er-Gruppen zu bearbeiten
- Rechnerarbeit: größtenteils in Eigenverantwortung

⚠ Anmeldung

- Bitte im Moodle-Raum eintragen. Gruppen melden sich per Mail.
- Es wird ein Pool-Account der IRB benötigt (ggf. bitte beantragen)

→ Freischaltung für AG-Server und git

Rechnerübung / Helpdesk

Termine bis KW 28

Tafelübung

- Donnerstag, 10:00 – 12:00, E07 (OH16)
- Alternativ auf Zuruf

Ausfälle

- 26.05, 16.06.
- siehe Moodle-Raum

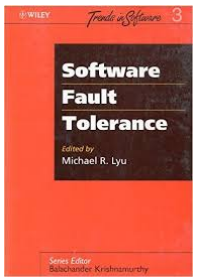
- **Inhalt & Ziele**
  - Laufende Übungsaufgabe
  - Lösen konkreter Probleme bei der Umsetzung der Aufgaben
- **Auf Zuruf**
  - In Präsenz im Labor der Arbeitsgruppe (E07)
  - Individuelle Videokonferenzen (Screensharing)

⚠ Termine und technische Umsetzung im Fluß (Moodle prüfen!)

Literaturempfehlungen

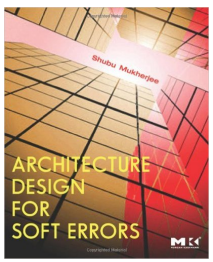
[2] Fehlertoleranz in Software:

M. Lyu, editor. *Software Fault Tolerance*, volume 3 of *Trends in Software*.
John Wiley & Sons, Inc., 1995.
https://www.cse.cuhk.edu.hk/~lyu/book/sft/



[3] Der „Klassiker" für transiente Hardwarefehler:

S. Mukherjee. *Architecture Design for Soft Errors*.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008



[1] Weiters Buch zu transienten Hardwarefehlern:

O. Goloubeva, M. Rebaudengo, M. S. Reorda, and M. Violante. *Software-Implemented Hardware Fault Tolerance*.
Springer-Verlag, New York, NY, USA, 1 edition, 2006