

Verlässliche Echtzeitsysteme

Peter Ulbrich

Lehrstuhl für Verteilte Systeme und Betriebssysteme
Friedrich-Alexander-Universität Erlangen-Nürnberg

<https://www4.cs.fau.de>

Sommersemester 2016



Verlässliche Echtzeitsysteme

Lehrveranstaltungs-konzept & Organisation

Peter Ulbrich

Lehrstuhl für Verteilte Systeme und Betriebssysteme
Friedrich-Alexander-Universität Erlangen-Nürnberg

<https://www4.cs.fau.de>

Sommersemester 2016



Vorwort

Die Lehrveranstaltung ist grundsätzlich für alle Studiengänge offen. Sie verlangt allerdings gewisse Vorkenntnisse. Diese müssen nicht durch Teilnahme an den Lehrveranstaltungen von I4 erworben worden sein.



Voraussetzungen

- **Systemprogrammierung**, Grundlagen der Informatik
- **C / 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 :-)



Gliederung

- 1 Vorwort
- 2 Die Veranstaltung
 - Lernziele
 - Einordnung
- 3 Organisatorisches
 - Die Beteiligten
 - Vorlesung und Übung
 - Leistungsnachweise



Verlässliche Echtzeitsysteme – Motivation



Technik (von Echtzeitsystemen) begeistert

- Zusteller begrenzen den zeitlichen Einfluss
 - Nicht-periodischer Aktivitäten auf periodische Arbeitsaufträge
- Neue Verfahren und Architekturen zu entwickeln, ist spannend!
- Mikrokerne schotten Programme räumlich voneinander ab
- Verschlüsselungsalgorithmen garantieren Datensicherheit
- ...



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 Ausnahmen 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 Fehlertoleranzmaßnahmen
 - 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 (aiT 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



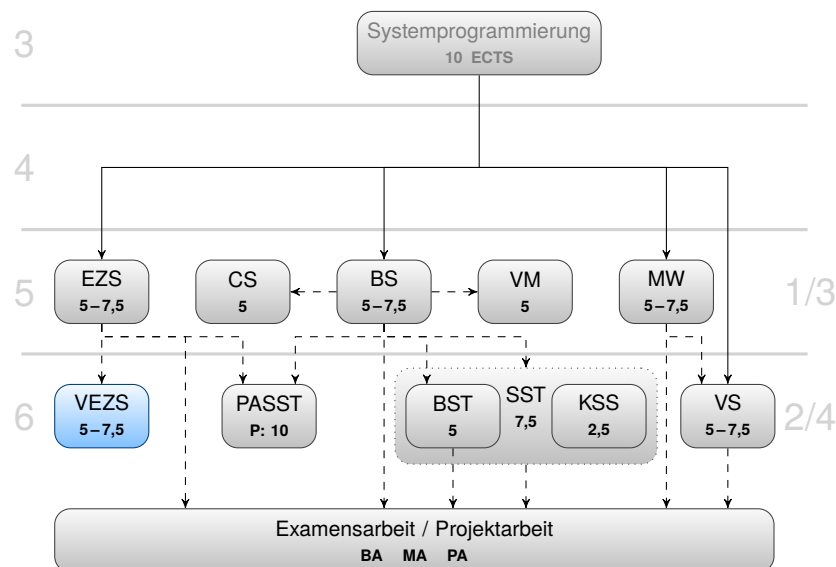
Bedeutung von Tafel- und Rechnerübungen

- **Tafelübungen** $\leadsto$ „*learning by exploring*“
 - Besprechung der Übungsaufgaben, Skizzierung von Lösungswegen
 - Vertiefung des Vorlesungsstoffes, Klärung offener Fragen
 - **Rechnerarbeit** $\leadsto$ „*learning by doing*“
 - Selbstständiges Bearbeiten der Übungsaufgaben am Rechner
 - Abgabe der bearbeiteten Übungsaufgaben
 - Klärung von Unklarheiten/Problemen bei/mit den Übungsaufgaben
 - Rechner ist allerdings **kein Tafelersatz**
- Bereitet euch vor! Wir erwarten konkrete Fragen!

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



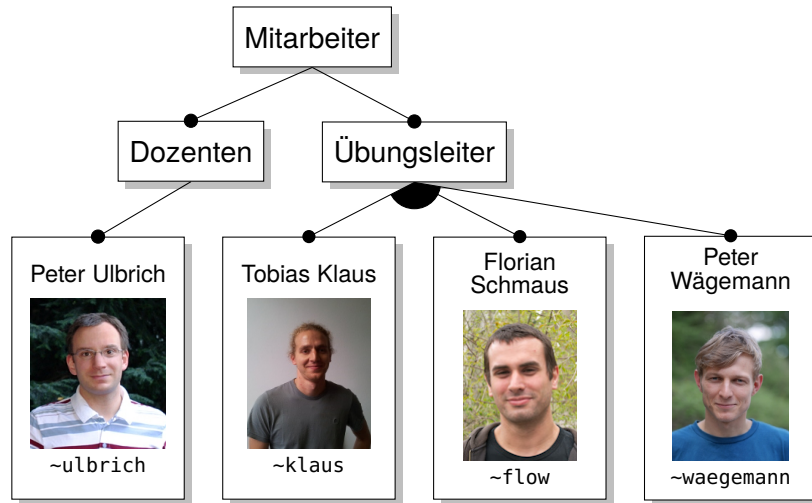
Einpassung in den Studienplan



Gliederung

- 1 Vorwort
- 2 Die Veranstaltung
 - Lernziele
 - Einordnung
- 3 Organisatorisches
 - Die Beteiligten
 - Vorlesung und Übung
 - Leistungsnachweise





Termine bis KW 28

- Donnerstag, 14:15 – 15:45, 0.031-113

Ausnahmen

- 05.05.: Himmelfahrt
- 26.05.: Fronleichnam

- **Handzettel** (engl. *handout*) sind verfügbar wie folgt:

- https://www4.cs.fau.de/Lehre/SS16/V_VEZS/Vorlesung
- Folienkopien werden vor der Vorlesung ausgegeben

- **Fachbegriffe** der Informatik (Deutsch ↔ Englisch)

- www.aktionlebendigesdeutsch.de

- **Änderungen und Hinweise:** siehe Webseite bzw. Mailingliste

Übungsbetrieb

Termine bis KW 28

Tafelübung

- Montag, 14:15 – 15:45, 01.253-128
- Donnerstag, 08:30 – 10:00, 01.253-128

Rechnerübung

- Montag, 12:15 – 13:45, 00.153-113
- Dienstag, 14:15 – 15:45, 00.153-113

Ausfälle

- siehe Webseite

Übung

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

Anmeldung

- Im Rahmen der Übungen (**Bettelverfahren**)
- Bitte tragt euch in die Mailingliste ein (siehe Webseite)

Studien- und Prüfungsleistungen (1)

VL – Vorlesung

2,5

Vorstellung und detaillierte Behandlung des Lehrstoffs

+

Ü – Übung

2,5

- Praktische Übungen
- Eine Aufgabe: 14 Tage
- Persönliche Abnahme

oder

EÜ – Erweiterte Übung

5

- Übung (Ü)
- + erweiterte Aufgaben
- + vertiefende Abfrage

+

RÜ – Rechnerübung

0

- **Betreutes** Arbeiten am Rechner
- Hilfe zu Werkzeugen und Techniken ...

Studien- und Prüfungsleistungen (2)

- **Wahlpflichtmodul** (Bachelor/Master) der Vertiefungsrichtung **Verteilte Systeme und Betriebssysteme**
 - eigenständig (nur VEZS) VL + Ü oder VL + EÜ
 - mit weiteren Veranstaltungen siehe Modulhandbuch
- Studien- und Prüfungsleistungen
 - Bachelor Prüfungsleistung
 - Master Prüfungsleistung
 - erworben durch
 - erfolgreiche Teilnahme an den Übungen
 - erfolgreiche Bearbeitung aller Übungsaufgaben
 - 30 min. mündliche Prüfung
- Berechnung der Modulnote
 - Note der mündlichen Prüfung + “Übungsbonus” in Zweifelsfällen



Forschung und Lehre funktioniert nur mit Studenten!

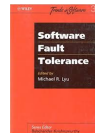


- Bachelor- und Masterarbeiten
- Bachelor-Praktikum und Master-Projekte
- studentische Hilfswissenschaftler (Hiwis)



Literaturempfehlungen

- [2] Fehlertoleranz in Software:
M. Lyu, editor. *Software Fault Tolerance, volume 3 of Trends in Software*.
John Wiley & Sons, Inc., 1995.
<http://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



Fragen...

42

