

Verlässliche Echtzeitsysteme

Peter Ulbrich

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

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

Sommersemester 2017



Verlässliche Echtzeitsysteme

Lehrveranstaltungskonzept & Organisation

Peter Ulbrich

Lehrstuhl für Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität Erlangen-Nürnberg

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

Sommersemester 2017



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.



- **Systemprogrammierung**, Grundlagen der Informatik
- **C / C++, Java**
- Ein gewisses Maß an **Durchhaltevermögen**
- Freude an systemnaher und **praktischer Programmierung**

Wir arbeiten mit eingebetteten Systemen!



- **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 :-)



1 Vorwort

2 Die Veranstaltung

- Lernziele
- Einordnung

3 Organisatorisches

- Die Beteiligten
- Vorlesung und Übung
- Leistungsnachweise





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**
- ...





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?





Technik (von Echtzeit)

- Zusteller begrenzt
- Nicht-periodische
- Neue Verfahren und Algorithmen
- Mikrokerne schottet ab
- Verschlüsselungstechniken
- ...



Das ist jedoch nur

- Erfordert möglichst schnelle Reaktionen
- Implementierung ist aufwändig
- Verfahren und Architekturen



Wie lassen sich Ausnahmen vermeiden bzw. behandeln?



träge
pannend!
er ab
rheit

können



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)



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





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





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





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





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



- **Tafelübungen** \leadsto „*learning by exploring*“
 - Besprechung der Übungsaufgaben, Skizzierung von Lösungswegen
 - Vertiefung des Vorlesungsstoffes, Klärung offener Fragen

- **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!

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.

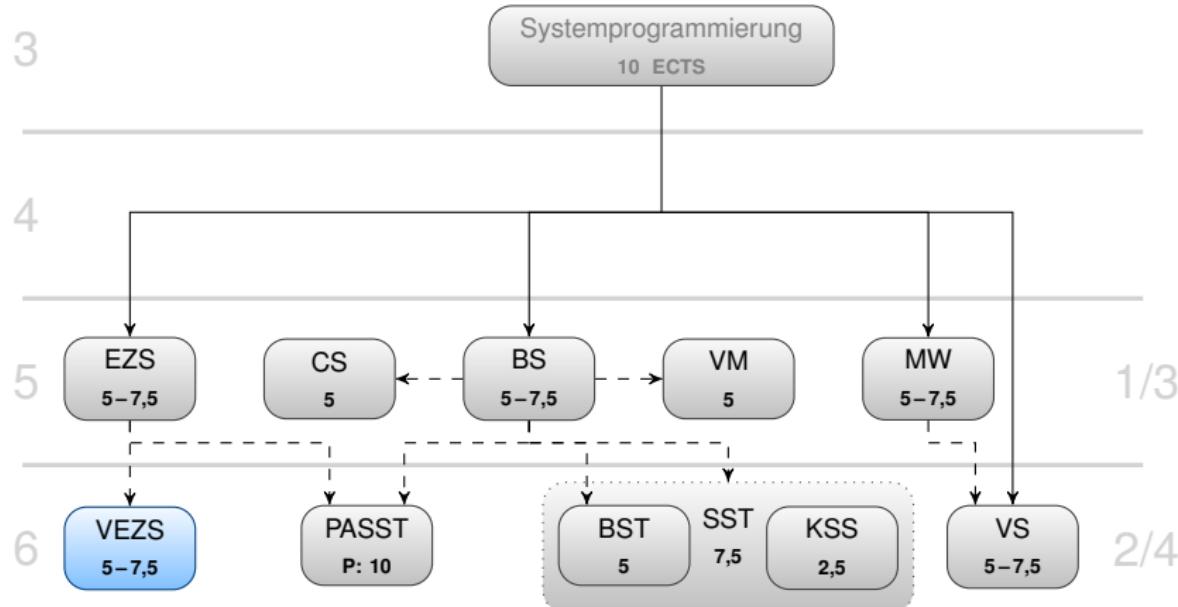
Wer, wie, was?

Wieso, weshalb, warum?

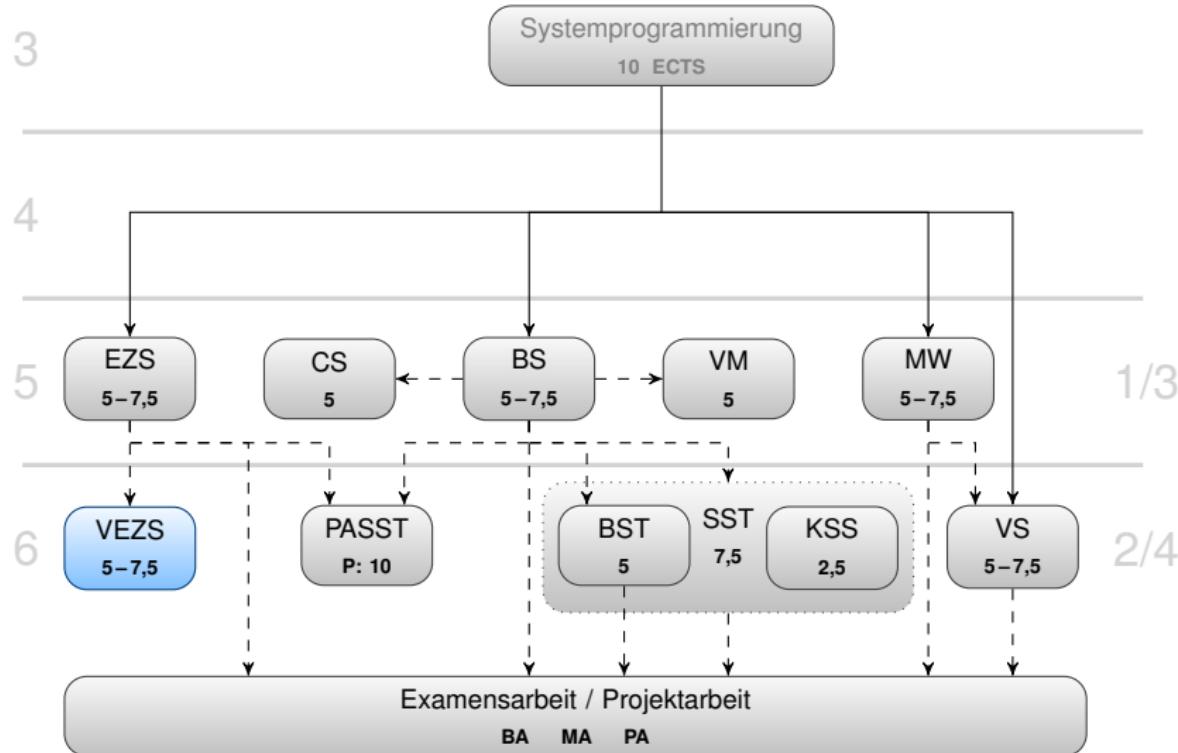
Wer nicht fragt, bleibt dumm!



Einpassung in den Studienplan



Einpassung in den Studienplan



1 Vorwort

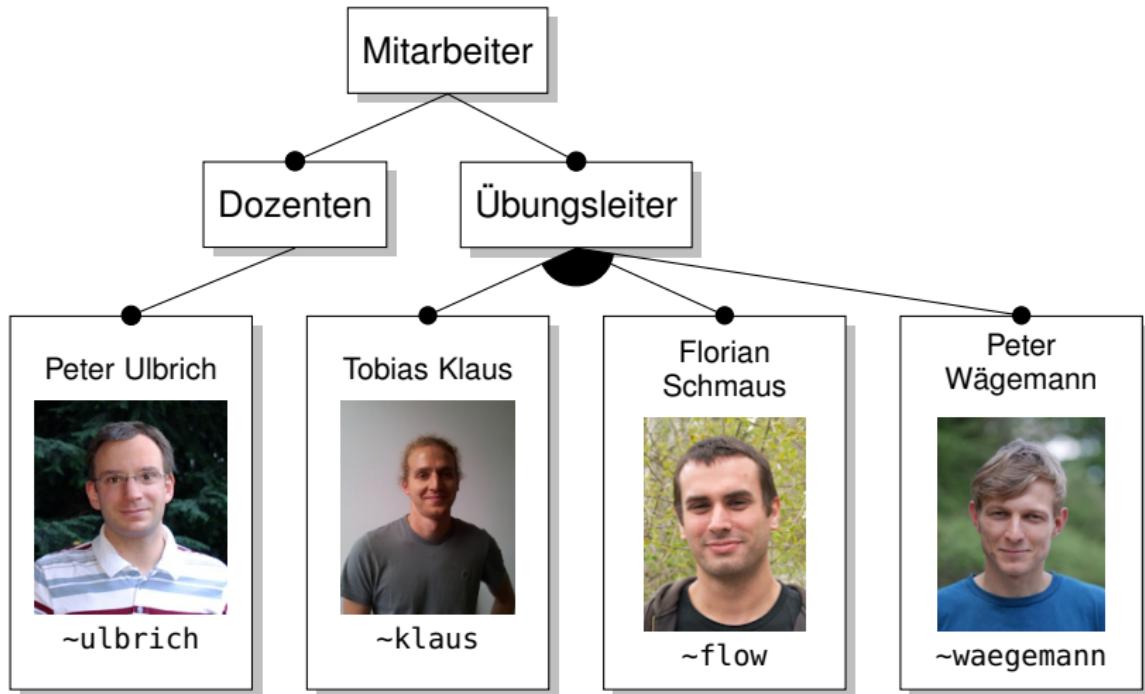
2 Die Veranstaltung

- Lernziele
- Einordnung

3 Organisatorisches

- Die Beteiligten
- Vorlesung und Übung
- Leistungsnachweise





Termine bis KW 30

- Donnerstag, 12:15 – 13:45, 0.031-113



Termine bis KW 30

- Donnerstag, 12:15–13:45, 0.031-113

Ausnahmen

- 25.05.: Himmelfahrt
- 15.06.: Fronleichnam

- **Handzettel** (engl. *handout*) sind verfügbar wie folgt:
 - https://www4.cs.fau.de/Lehre/SS17/V_VEZS/Vorlesung
 - Folienkopien werden vor der Vorlesung ausgegeben



Termine bis KW 30

- Donnerstag, 12:15–13:45, 0.031-113

Ausnahmen

- 25.05.: Himmelfahrt
- 15.06.: Fronleichnam

- **Handzettel** (engl. *handout*) sind verfügbar wie folgt:
 - https://www4.cs.fau.de/Lehre/SS17/V_VEZS/Vorlesung
 - Folienkopien werden vor der Vorlesung ausgegeben
-  **Fachbegriffe der Informatik** (Deutsch ↔ Englisch)
 - www.aktionlebendigesdeutsch.de



Termine bis KW 30

- Donnerstag, 12:15 – 13:45, 0.031-113

Ausnahmen

- 25.05.: Himmelfahrt
- 15.06.: Fronleichnam

- **Handzettel** (engl. *handout*) sind verfügbar wie folgt:
 - https://www4.cs.fau.de/Lehre/SS17/V_VEZS/Vorlesung
 - Folienkopien werden vor der Vorlesung ausgegeben
-  **Fachbegriffe der Informatik** (Deutsch ↔ Englisch)
 - www.aktionlebendigesdeutsch.de
-  **Änderungen und Hinweise:** siehe Webseite bzw. Mailingliste



Termine bis KW 30

Tafelübung

- Montag, 14:15 – 15:45, 0.031-113
- Mittwoch, 10:15 – 11:45, 0.031-113

Rechnerübung

- Montag, 12:15 – 13:45, 00.153-113
- Donnerstag, 10:15 – 12:45, 00.156-113



Termine bis KW 30

Tafelübung

- Montag, 14:15 – 15:45, 0.031-113
- Mittwoch, 10:15 – 11:45, 0.031-113

Rechnerübung

- Montag, 12:15 – 13:45, 00.153-113
- Donnerstag, 10:15 – 12:45, 00.156-113

Ausfälle

- siehe Webseite

Übung

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



Termine bis KW 30

Tafelübung

- Montag, 14:15 – 15:45, 0.031-113
- Mittwoch, 10:15 – 11:45, 0.031-113

Rechnerübung

- Montag, 12:15 – 13:45, 00.153-113
- Donnerstag, 10:15 – 12:45, 00.156-113

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 Mailingsliste ein (siehe Webseite)



VL – Vorlesung

2,5

Vorstellung und detaillierte Behandlung des Lehrstoffs



VL – Vorlesung

2,5

Vorstellung und detaillierte Behandlung des Lehrstoffs

+

Ü – Übung

2,5

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



VL – Vorlesung

2,5

Vorstellung und detaillierte Behandlung des Lehrstoffs

+

Ü – Übung

2,5

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

EÜ – Erweiterte Übung

5

oder

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



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 ...



- **Wahlpflichtmodul** (Bachelor/Master) der Vertiefungsrichtung
Verteilte Systeme und Betriebssysteme
 - eigenständig (nur VEZS)
 - mit weiteren Veranstaltungen

VL + Ü oder VL + EÜ
siehe Modulhandbuch
- Studien- und Prüfungsleistungen
 - Bachelor
 - Master

Prüfungsleistung
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





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



[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



42

