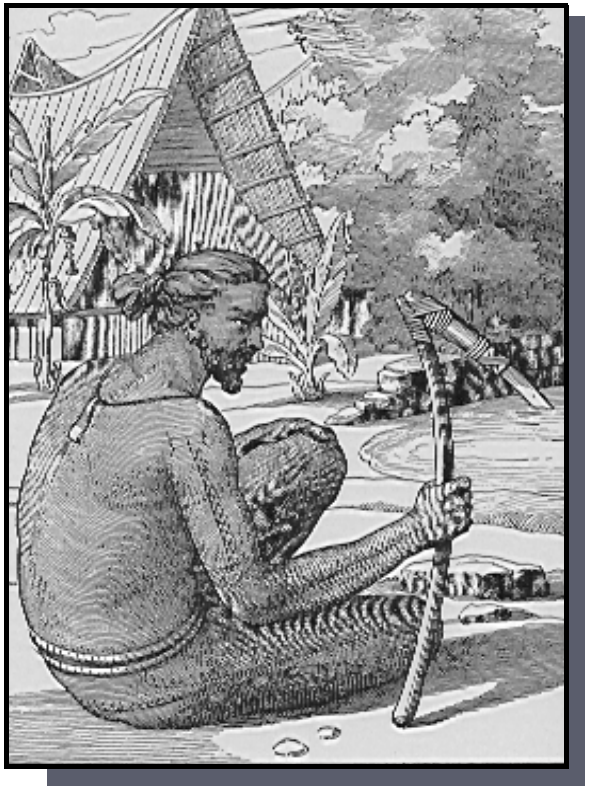
Software Engineering - Wartung und Qualitätssicherung



Not a Java Warrior with his favourite engineering tool!

Prof. Dr. Andy Schürr
Fachgebiet Echzeitsysteme
FB ETiT (Informatik)

Technische Universität Darmstadt,
Magdalenenstraße 4, 64289 Darmstadt

Andy.Schuerr@es.tu-darmstadt.de

Tel.: 06151 / 16-22350
Raum: S1|08|205

WWW-Seite der Vorlesung:

http://www.es.tu-darmstadt.de/lehre/vl-se-ii/

Bildquelle:

Jules Verne: The Great Explorers of the XIXth Century,
New York: Charles Scribner's Sons (1912)

WWW-Seite der Vorlesung:

http://www.es.tu-darmstadt.de/lehre/vl-se-ii/

Verankerung als Wahlmodul in Prüfungsplänen von Studiengängen:

- ETiT / DT, ...
- (Wirtschafts-)Informatik
- Informationssystemtechnik, Mechatronik, ...

Termine der Lehrveranstaltung:

- **Vorlesung**:
  Montag 10:00 (s.t.) bis 11:30 Uhr in S3|06|051 (ab 14.10.2019)
  Donnerstag 8:00 (s.t.) bis 8:45 in S3|06|051 (ab 17.10.2019)
- **Übung**: Donnerstag 8:45 bis 9:30 in S3|06|053 (ca. ab 07.11.2019)
- **Klausur** (laut TUCaN): Montag, 24.02.2018, 15:00 bis 17:00 (ohne Gewähr)

Wissensgebiete der Software-Technik:

Der IEEE Computer Society „Guide to the Software Engineering Body of Knowledge“ (SWEBOK V3.0, http://www.swebok.org) zählt folgende Wissensgebiete auf:

1. **Software Requirements**:

   es wird festgelegt, „**was**“ ein Software-System leisten soll (und warum)

2. **Software Design**:

   das „**Wie**“ steht nun im Vordergrund, der Bauplan (Architektur)

3. **Software Construction**:

   gemäß Bauplan wird das Software-System realisiert

4. **Software Testing**:

   Fehler werden **systematisch** gesucht und eliminiert

5. **Software Maintenance**:

   die Pflege und Weiterentwicklung der Software nach Auslieferung

6. **Software Configuration Management**:

   die Verwaltung von Software-Versionen und -Konfigurationen

7. **Software Engineering Management**:

   (Projekt-)Management von Personen, Organisationen, Zeitplänen, …

8. **Software Engineering Process**:

   Definition und Verbesserung von Software-Entwicklungsprozessen

9. **Software Engineering ~~Tools~~ Models and Methods**:

   Modelle und Methoden für die Software-Entwicklung

10. **Software Quality**:

    Messen und Verbessern der Software-Qualität

Aufteilung auf Lehrveranstaltungen

- **Software-Engineering - Einführung**
  - ⇨ Software Requirements
  - ⇨ Software Design
  - ⇨ Software Construction
  - ⇨ Software Engineering Models and Methods
- **Software-Engineering - Wartung und Qualitätssicherung**
  - ⇨ Software Testing
  - ⇨ Software Maintenance
  - ⇨ Software Configuration Management
  - ⇨ Software Engineering Management
  - ⇨ Software Engineering Process
  - ⇨ Software Engineering Models and Methods
  - ⇨ Software Quality

Zielsetzungen der Vorlesung:

- ☺ Zuhörer für die Realität der Software-Entwicklung fit machen
- ☺ Techniken und Werkzeuge zum Umgang mit bestehender Software vermitteln
- ☺ Überblick über systematische Software-Analyse- und Testverfahren geben

Zielsetzungen der Übung:

- ☺ praktische Vertiefung der Lehrinhalte an einem realen Beispiel
- ☺ Konfrontation mit kommerziellen und „Open Source“-Werkzeugen

Inhaltsverzeichnis der Vorlesung - 1

Inhaltsverzeichnis der Vorlesung - 2

Inhaltsverzeichnis der Vorlesung - 3

Übungen zur Vorlesung:

- es gibt meist klausurähnliche Hausaufgaben (**mit Korrekturservice**)
- Lösungen können/sollten gruppenweise abgegeben werden
- in Präsenzübungen werden Musterlösungen für Übungsaufgaben vorgestellt
- Einsatz verschiedener CASE-Tools für („Open Source“-)Softwareentwicklung
- kein Bonussystem (für Klausurnote)
- zusätzliche Sprechstunde bei Bedarf

Betreuer:

☞ **Sebastian Ruland** (sebastian.ruland@es.tu-darmstadt.de)

Zertifizierung zum Software-Tester:

- **ASQF** = **A**rbeitskreis **S**oftware-**Q**ualität **F**ranken bietet in Zusammenarbeit mit dem **iSQI** = international Software Quality Institute „standardisierte“ Ausbildung (Foundation/Advanced/Diploma Level) zum **Certified Tester** an (mit Unterstützung durch Gesellschaft für Informatik)
- in Deutschland waren 2003 fünf Kursanbieter **akkreditiert** und es gab etwa 550 **zertifizierte** Tester (auf Foundation Level); inzwischen gibt es mindestens 18 sogenannte „Premiumanbieter“ solcher Kurse
- wir werden auch dieses Jahr wieder Studierenden der TU Darmstadt die Möglichkeit bieten, am Ende der Vorlesungszeit gegen ermäßigte Prüfungsgebühr hier die Prüfung zum „Certified Tester“ abzulegen
- **Achtung**: Vorbereitung in Übung durch Schulungsleiter aus der Industrie!
- empfohlene Lehrmaterialien und weitere Links sind (am
  - ⇨ Lehrbuch hierzu [SL19] und http://www.dpunkt.de/certified-tester (am Fachgebiet ausleihbar und im Kittler-Student-Center einsehbar)
  - ⇨ weitere Informationen unter http://www.certified-tester.de/

Wichtige Literaturquellen:

[Ba00] H. Balzert: *Lehrbuch der Software-Technik (Band 1): Software-Entwicklung*, Spektrum Akademischer Verlag (2000), 2-te Auflage, 1136 Seiten

Sehr umfangreiches und gut lesbares Nachschlagewerk mit CD. Auf der CD findet man Werkzeuge, Videos, Übungsaufgaben mit Lösungen (aber nicht unsere), …

[Ba98] H. Balzert: *Lehrbuch der Software-Technik (Band 2): Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*, Spektrum Akademischer Verlag (1998), 769 Seiten

Hier findet man fast alles über das Gebiet Software-Technik, was nicht bereits in [Ba00] abgehandelt wurde. Wieder ist eine CD mit den entsprechenden Werkzeugen, ... beigelegt.

[Be00] R.V. Beizer: *Testing Object-Oriented Systems - Models, Patterns, Tools*, Addison-Wesley (2000), 1191 Seiten

Umfassende Quelle zum angesprochenen Thema. Gut geschrieben und umbedingt empfehlenswert, falls man die Thematik „Testen“ vertiefen will.

[CFP06] B. Collins-Sussman, B.W. Fitzpatrick, C.M. Pilato: *Versionskontrolle mit Subversion,* O’Reilly (2006)

Subversion (SVN) ist der moderne Nachfolger von CVS.

[Ka98] K. Fogel, M. Bar: ***Open Source-Projekte mit CVS***, mitp-Verlag (2002), 2-te überarbeitete Auflage, 428 Seiten

Dieses Buch liefert gut lesbar das notwendige Know-how für die Verwendung und Administration von CVS sowie eine Einführung in die Prinzipien des Managements von „Open Source“-Projekten

[He03] H. Herold: ***make - das Profitool zur automatischen Generierung von Programmen***, Addison Wesley (2003), 230 Seiten

Akzeptable Einführung in make, das „Open Source“-Werkzeug zur Automatisierung von Übersetzungs- und Programmgenerierungsprozessen.

[Li02] P. Liggesmeyer: ***Software-Qualität: Testen, Analysieren und Verifizieren von Software***, Spektrum Akademischer Verlag (2002), 523 Seiten

Ein Standardwerk zum Thema Software-Qualitätssicherung. Kapitel 3 und 4 der Vorlesung stützen sich vor allem darauf ab (es gibt eine neue Auflage von 2009).

[PBL05] K. Pohl, G. Böckle, F. van der Linden: *Software Product Line Engineering - Foundations, Principles, and Techniques*, Springer Verlag 2005, 467 Seiten

Ein (Lehr-)Buch zum „Software-Produkt-Familien“ (Management von Software-Varianten).

[So07] I. Sommerville: *Software Engineering*, Addison-Wesley - Pearson Studium, 8. Auflage (2007), 875 Seiten

Ins Deutsch übersetztes Lehrbuch, das sehr umfassend alle wichtigen Themen der Software-Technik knapp behandelt. Empfehlenswert (es gibt eine neue restrukturierte 10. Auflage von 2018)!

[SL19] A. Spillner, T. Linz: ***Basiswissen Softwaretest***, dpunkt.verlag (2019; 6. Auflage), 351 Seiten

Passend zum Lehrplan „Grundlagen des Testens“ für den ASQF Certified Tester, Foundation Level.

[Wh00] B.A. White: *Software Configuration Management Strategies and Rational ClearCase*, Addison Wesley (2000), 305 Seiten

Clearcase war lange Zeit „das“ kommerzielle Pendant zu CVS; es besitzt wesentlich mehr Features, die Administration erfordert allerdings auch mehr Aufwand.