Software Engineering

SS 2015

Hans-Georg Eßer
Dipl.-Math., Dipl.-Inform.

Foliensatz A: v1.0, 2015/04/07

- Einführung
- Grundlagen SE
- Seminararbeiten, Wissenschaftliches Arbeiten, LaTeX

Hans-Georg Eßer

- Dipl.-Math. (RWTH Aachen, 1991–1997)
  Dipl.-Inform. (RWTH Aachen, 2002–2005)
  Fachjournalist (FJS Berlin, 2005–2006)
  Promotionsstudium in Informatik
  (Univ. Erlangen-Nürnberg, 2010–2015)
- Chefredakteur Linux-Zeitschrift (seit 2000)
  und Autor diverser Computerbücher
- seit 2006 Dozent (u. a. FOM, HS München, TH Nürnberg, Univ. Erlangen-Nürnberg): Betriebssysteme, Rechnerarchitektur, IT-Infrastruktur, Informatik-Grundlagen, Systemprogrammierung, Betriebssystem-Entwicklung, Software Engineering, IT-Sicherheit
- Seit 2010 wiss. Mitarbeiter der Univ. Erlangen-Nürnberg

Zur Veranstaltung (1)

Service / Web-Seite: http://fom.hgesser.de
- Folien und Praktikumsaufgaben
- Vorlesungs-Videos („test, test “)
- Probeklausur gegen Semesterende
- Folien auch im Online-Campus

Prüfung und Benotung:
1. Lernfortschrittskontrolle (LFK)
2. Seminarausarbeitung + Vortrag
3. Klausur über 120 Minuten

Fragen: direkt in der Vorlesung oder danach
oder per E-Mail: h.g.esser@gmx.de

Zur Veranstaltung (2)

10 ECTS-Punkte – großes Modul

- Präsenzstudium: 48 Stunden *) (= 64 UE)
- Eigenstudium: 172 Stunden
- Lernfortschr.-Kontr.: 30 Stunden
- Workload: 250 Stunden

Klausur-relevant:

- Vorlesungen
- Übungsaufgaben
- Seminarvorträge (eingeschränkt)

*) Stunde
= Zeitstunde,
= 60 min.

Inhaltsübersicht

Inhalte

- Grundlagen des Software Engineering (SE)
  - Prozessmodelle (klassisch, agil)
  - Architekturmodelle (u. a.: UML)
- Secure SE
- Software-Dokumentation
- Versionsverwaltung, andere Werkzeuge

Literatur

- Ian Sommerville: Software Engineering, 9th ed. 2011, Pearson, ISBN 0137035152 (engl. Ausgabe)
- Helmut Balzert: Lehrbuch der Softwaretechnik (drei Bände)

weitere Literaturhinweise folgen ggf.

Dieser Foliensatz

Folien A

Vorlesungsübersicht
Einführung u. Motivation Software Engineering
Seminar / Wiss. Arbeiten / LaTeX
…
…
…

Einführung und Motivation

Einführung und Motivation

- Anfänge der Software-Entwicklung
  - Erstellung durch einzelne Programmierer
  - ganz früh: direkte Eingabe von Maschinen-Opcodes oder Code-Eingabe über Lochkarten-Leser
  - später: lokale Entwicklungsumgebung (Editor, Compiler/Assembler und Tools)
  - überschaubare Code-Größe (→ wenige 1000 LOC)
  - einfache Versionsverwaltung
  - Dokumentation: rudimentär oder gar nicht

Einführung und Motivation

- Zunehmende Komplexität
  - Code zu groß für einzelnen Entwickler → Teams
  - verteilte Code-Basis
  - bessere Versionierungstools, Repositorys (im Netz)
  - Coding-Guidelines
  - Dokumentation wird Pflicht (→ Qualität?)
  - „Software-Krise“

Eigenschaften von Software

- Original und Kopie sind gleich
- Herstellung vieler Kopien ist einfach
- Software verschleißt nicht durch Gebrauch
- keine Fehler durch Abnutzung
- Allerdings: Software „altert“
- Software ist immateriell

Quelle: Folien von Dipl. Ing. Felix Dutkowski, FOM

Kleine vs. große Projekte

Bild: © Fressnapf.de

Bild: © okal.de

- Hundehütte bauen
  - keine formale Projekt-organisation
  - kaum Planung nötig
  - 1 Person
  - einfache Werkzeuge
- SW-Entwicklung
  - Projekte ≤ 10.000 LOC bzw. ≤ 2 Personenjahre

- Einfamilienhaus bauen
  - Plan → Rohbau → Installation → Innenausbau etc.
  - genaue Planung nötig
  - großes Team
  - komplexe Werkzeuge
- SW-Entwicklung
  - Projekte mit Millionen LOC, hunderte Personenjahre

Großes Projekt

Beispiel: Linux-Kernel

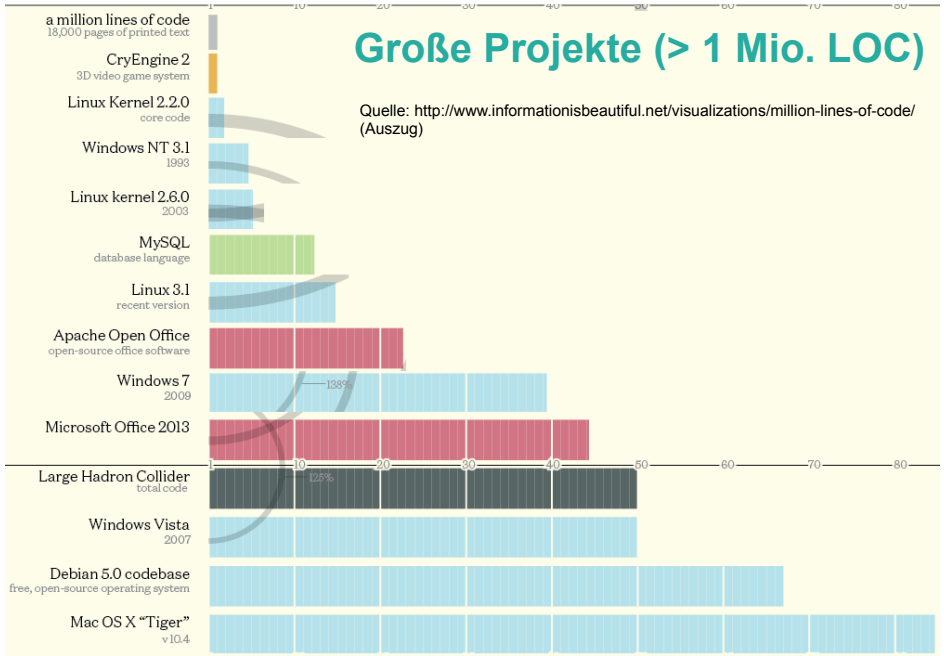
Kernel Version	Files	Lines
2.6.11 2005	17,090	6,624,076
2.6.12	17,360	6,777,860
2.6.13	18,090	6,988,800
2.6.14	18,434	7,143,233
2.6.15	18,811	7,290,070
2.6.16	19,251	7,480,062
2.6.17	19,553	7,588,014
2.6.18	20,208	7,752,846
2.6.19	20,936	7,976,221
2.6.20	21,280	8,102,533
2.6.21	21,614	8,246,517
2.6.22	22,411	8,499,410
2.6.23	22,530	8,566,606
2.6.24	23,062	8,859,683
2.6.25	23,813	9,232,592
2.6.26	24,273	9,411,841

Kernel Version	Files	Lines
2.6.27	24,356	9,630,074
2.6.28	25,276	10,118,757
2.6.29	26,702	10,934,554
2.6.30	27,911	11,560,971
2.6.31	29,143	11,970,124
2.6.32	30,504	12,532,677
2.6.33	31,584	12,912,684
2.6.34	32,316	13,243,582
2.6.35	33,335	13,468,253
2.6.36	34,317	13,422,037
2.6.37	36,189	13,919,579
2.6.38	36,868	14,211,814
2.6.39	36,713	14,537,764
3.0	36,788	14,651,135
3.1	37,095	14,776,002
3.2 2012	37,626	15,004,006

Quelle: http://storage.pardot.com/6342/48856/lf_kernel_development_2012.pdf

Große Projekte (> 1 Mio. LOC)

Quelle: http://www.informationisbeautiful.net/visualizations/million-lines-of-code/ (Auszug)



Software-Krise (1)

„The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly:

- as long as there were no machines, programming was no problem at all;
- when we had a few weak computers, programming became a mild problem,
- and now we have gigantic computers, programming has become an equally gigantic problem."

E. Dijkstra (1972)

Software-Krise (2)

Probleme:

- Budget-Überschreitung
- Zeit-Überschreitung
- ineffiziente Software
- Software von geringer Qualität
- Software erfüllt nicht die Anforderungen
- Projekt-Management u. Code-Pflege schwierig
- Entwicklung wird nie fertig

Software Engineering (SE)

- Neue Herangehensweise:
  Ingenieursperspektive (→ Engineering)
- SW-Entwicklung nicht als *Kunst*, sondern als Ingenieurswissenschaft begreifen
  - Einsatz von erforschten Methoden / Prozessen
  - Phasenmodelle, z. B. Wasserfall, Spirale
  - Life Cycle Management, z. B. Reifegradmodell (CMM), Test-driven Development (TDD)
  - Entwicklungsphilosophien, z. B. XP, Scrum, Pair Programming

Ingenieurswissenschaft

„Die Ingenieurwissenschaften sind angewandte Wissenschaften. Forschung wird meist in Bezug zu praktisch einsetzbarer Technik, Verfahren, Methoden, Anwendungen oder Geräten betrieben.“

(Quelle: http://de.wikipedia.org/wiki/Ingenieurwissenschaften)

SE-Definitionen

- „the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software“ [1]
- „an engineering discipline that is concerned with all aspects of software production“ [2]
- „the establishment and use of sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines“ [3]

[1] IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990, 1990
[2] Sommerville, Ian (2007), Kap. 1.1.2: „What is software engineering?“ in *Software Engineering*, S. 7.
[3] "Software Engineering". Information Processing (North-Holland Publishing Co.) 71: 530–538. 1972.

SE-Ziele

- Probleme (→ Software-Krise) in den Griff bekommen
  - im Budget bleiben, im Zeitplan bleiben (→ SW fertigstellen)
  - effiziente, qualitativ hochwertige, möglichst *fehlerfreie* SW entwickeln
  - die Anforderungen (Spezifikation) erfüllen
  - Projekt-Management und Code-Pflege ermöglichen
- Fehlerfreiheit: Testverfahren
- hier auch: *sichere* SW entwickeln

Fehlerfrei? – Y2K-Problem

- Y2K (Jahr-2000-Problem)
  - Jahr als zweistellige Ziffer (1985 → 85)
  - SW-Einsatz war nie über 1999 hinaus vorgesehen
  - SW war 2000 trotzdem noch im Einsatz
  - 2000 → 00 = 1900 …
  - noch besser: 19**99** ← **99**, 99 + 1 = **100** → 19**100**
- einer der größten Bugs der Informatik-Geschichte
- demnächst: Jahr-2038-Problem („Unixtime“)

Fehlerfrei? – typische Fehlerraten

- normale Software:
  1 bis 25 Fehler / 1000 LOC
- Clean Room Development:
  < 0,1 Fehler / 1000 LOC

Quellen:

[1] Steve McConnell: Code Complete, 2nd edition 2004, Microsoft Press, Kap. 22.4 („Typical Errors“), Abschnitt „How Many Errors Should You Expect to Find?“

[2] Cobb, R. H. and Mills, H. D. (1990). Engineering Software under Statistical Quality Control. IEEE Software,7(6), 44-54.

Vorlesungsinhalte im Detail

- … stehen noch nicht fest
- Fokus auf modernen Verfahren
- spezielles Teilgebiet: Sicherheit (Secure Software Engineering)
- Ich werte den →Fragebogen aus und plane abhängig davon

Seminararbeiten

Seminararbeiten

- 28 Teilnehmer laut Campus-System
- Seminar-Themen:
  - SE-Praxis
  - Dokumentation und SW-Test
- bei Doppel-Themenbelegung Absprache
- Vorträge: während der Präsenztermine am **17.07. / 18.07.2015**
- Vortragsfolien **bis 10.07.2015** per Mail an mich

Anfertigen der Seminararbeit

Ausarbeitungen und Vorträge

Inhalte

- Wie lese ich Fachbücher/Skripte/Papers?
- Wie schreibe ich eine Seminararbeit?
- Wie halte ich einen Seminarvortrag?

Quellen lesen

- Was ist das Themengebiet der Quelle?
- Was ist die zentrale Fragestellung?
  - Welches Thema wird im Detail behandelt?
  - Welche Aspekte sind besonders wichtig oder vernachlässigbar?
  - Welche Aspekte bleiben in der Quelle unklar?
- Literaturangaben auswerten
  - Grundlagen
  - Weiterführende Literatur

Seminararbeit schreiben

- Gliederung erstellen
- Typisch:
  - Kap. 1: Einleitung
  - Kap. 2:
  - Kap. n–1:

    (Inhalt)
  - Kap. n: Zusammenfassung

Einleitung

- Einleitung besteht aus:
  - Motivation / Themenfeld
    - Welches Themengebiet behandelt die Arbeit
    - Warum ist es relevant?
  - Thema der (Seminar-) Arbeit
    - Fragestellung in dieser Ausarbeitung
    - Bezug auf das Paper / die Papers
    - Welche Lösung wird im Paper vorgeschlagen?
  - Struktur der Arbeit
    - Wie ist diese Arbeit aufgebaut? Roter Faden?

Einleitung

- Leser wollen i. d. R. nicht jedes Dokument vollständig lesen
- Darum: In der Einleitung vollständigen Überblick über die wichtigsten Inhalte geben (→ „Executive Summary“)

Inhaltliche Kapitel

- In den inhaltlichen Kapiteln kann sich die Grobstruktur der Ausarbeitung wiederholen, also
  - Kap. i.1: Einleitung (zu Thema i)
    - Worum geht es?
    - Roter Faden
    - Anknüpfen an vorangehendes Kapitel
  - Kap. i.2: …
  - Kap. $i.m$: Zusammenfassung
    - Worum ging es? Wieder: Roter Faden
    - ggf. Ausblick auf Folgekapitel

Redundanz ist OK, hilft bei Orientierung

Zusammenfassung

- Kurze Zusammenfassung der Ausarbeitung (Kenntnis der Arbeit kann hier vorausgesetzt werden)
- Falls sinnvoll: Hinweis auf offene Punkte, unbeantwortete Fragen

Inhaltsauswahl

- Häufiges Problem:
  Quelle viel zu lang / Quelle viel zu kurz
- Auswahl
- Ergänzung (weitere Literatur)
- Seminarausarbeitung und Vortrag müssen nicht deckungsgleich sein
- Ausarbeitung „in sich geschlossen“ (ohne weitere Literatur verständlich)
- Frage beantworten: Warum sollte man meine Ausarbeitung (statt der Originaltexte) lesen?

Quellen / Zitate (1)

- Beim Strukturieren bereits die relevanten Textstellen der Originalliteratur vermerken
  - Welcher Teil der Arbeit bezieht sich auf welchen Teil der Originalliteratur?
- Im Text eine Referenz einbauen falls ...
  - wesentliche Ideen, Zahlen, Bilder, Tabellen usw. übernommen werden
  - ganze Teile wortwörtlich übernommen werden (Zitate)
- Zitate als solche kenntlich machen („...“ [Referenz])
- Referenzen möglichst genau (am besten mit Seitenangabe, vor allem bei Büchern)

Kritischer Umgang

- Quellen kritisch bearbeiten
  - Nur weil etwas in einer Quelle steht, muss es nicht richtig sein → kein blindes Vertrauen
  - Technischen Beschreibungen und Erklärungen kann man auch widersprechen
  - Wenn unklar: Weitere Literatur zum selben Thema suchen, vergleichen
  - Kein direkter Zusammenhang zwischen „Zitierfähigkeit“ und Glaubwürdigkeit einer Quelle

Quellen / Zitate (2)

- Im Text stehen Referenzen (Verweise) auf das Literaturverzeichnis
  - Referenzen sind meist Kürzel in eckigen Klammern
  - Unterstützung durch LaTeX
- Arten des Zitierens:
  - Aktiv: In [1] wurde gezeigt … [1, S. 35]
  - Passiv: Maier und Müller [1] haben gezeigt …
  - Möglichst für eine Art entscheiden und konsistent verwenden

Quellen / Zitate (3)

- Informationen, die (unter Informatikern) allgemein bekannt sind, brauchen keinen Beleg durch Quellenangabe
  - „C ist eine häufig für systemnahe Programmierung verwendete Sprache.“ (ohne Beleg)
  - „Das Betriebssystem ZonkOS wurde nicht in C, sondern in Fortran implementiert [12].“ (mit Beleg)
- Zielgruppe: andere (Wirtsch.-) Informatik-Studenten, konkret: die übrigen Kursteilnehmer

Quellen / Zitate (4)

- Mehrere Referenzen, Seitenangaben
  - [1, 2, 4]
  - [1–3, 5]
  - [4, S. 15, 6, S. 34] → besser zerlegen
  - LaTeX macht das automatisch
- Viele Autoren
  - Müller et al.: ab vier Autoren; in engl. Texten ab drei Autoren, vgl. Wikipedia [3]
  - im Literaturverzeichnis aber alle Autoren nennen

Quellen / Zitate (5)

- Zitierweise
  - Bitte keine Fußnoten für Literatur-Verweise,
  - sondern einfach z. B. „[6]“ im Text als Verweis auf Eintrag [6] im Literaturverzeichnis
  - bei längeren Quellen (wie dem TK-Skript) immer mit Seitenangabe, in der Form „[6, S. 119]“

Literaturverzeichnis

- alle verwendeten (und nur die!) Quellen aufführen
  - sortiert nach Nachname des Erstautors, dann nach Datum
- Eintrag enthält
  - Namen der Autoren, Titel der Publikation/des Beitrags
  - ggf. Name der Zeitschrift/Konferenz, Band, Nummer, Seitenzahl / Ort
  - Erscheinungsjahr
  - möglichst genau
- Quellen im Web: Autoren, Titel, URL und Zugriffsdatum (wenn es eine Originalquelle gibt, diese zitieren!)

Literaturverzeichnis

- Wikipedia: u. U. akzeptable Quelle. Ganz gut für technische Dinge
- Whitepapers von Unternehmen: da fehlen oft wichtige Angaben (z. B. Autoren), → Firma als Hrsg.
- „keinen Schrott zitieren“

Literaturverzeichnis

- Negativ-Beispiel

Bei einem Vergleich forensisch nutzbarer Informationen in Windows-Dateisystemen sieht man u. a., dass FAT [1] nur einen einzigen Zeitstempel pro Datei speichert, während NTFS [2] für jede Datei vier Zeitstempel verwaltet [3]. Ergänzend gibt ...

[1] http://de.wikipedia.org/wiki/File_Allocation_Table
[2] http://de.wikipedia.org/wiki/NTFS
[3] http://articles.forensicfocus.com/2013/04/06/interpretation-of-ntfs-timestamps/
[4] http://books.google.de/books?id=fummOICB9IgC&pg=PA148&lpg=PA148&dq=ntfs+%22delete+time%22+timestamp&source=bl&ots=FOFB7xD3MY&sig=awkTk3V-f1dR3E6ivx46-Gc7xs4&hl=en&sa=X&ei=loFXU6mOC4LNtQbLuwE&ved=0CDwQ6AEwAg#v=onepage&q=ntfs%20%22delete%20time%22%20timestamp&f=false

Literaturverzeichnis

Positiv-Beispiel

Bei einem Vergleich forensisch nutzbarer Informationen in Windows-Dateisystemen sieht man u. a., dass FAT [1] nur einen einzigen Zeitstempel pro Datei speichert, während NTFS [2] für jede Datei vier Zeitstempel (create, modify, access, change) verwaltet [3]. Der vierte Zeitstempel (change) wird für gelöschte Dateien auch als delete-Timestamp betrachtet [4, S. 148].

[1] Microsoft: Microsoft EFI FAT32 File System Specification, 2000, Whitepaper, http://download.microsoft.com/download/1/6/1/161ba512-40e2-4cc9-843a-923143f3456c/fatgen103.doc, Onlineabruf am 23.04.2014

[2] Microsoft: NTFS Technical Reference, 2003, http://technet.microsoft.com/en-us/library/cc758691%28WS.10%29.aspx, Onlineabruf am 23.04.2014

[3] wpathulin: „Interpretation of NTFS Timestamps“, Forensic Focus, 2013, http://articles.forensicfocus.com/2013/04/06/interpretation-of-ntfs-timestamps/, Onlineabruf am 23.04.2014

[4] Greg Gogolin: *Digital Forensics Explained*, Auerbach Publications, 2012, ISBN: 1-4398-7495-6

Text produzieren

- In eigenen Worten formulieren
- Keine Textblöcke aus der Quelle übernehmen, auch keine Wort-für-Wort-Übersetzung
- kein Copy & Paste
  - Plagiat, ist in Prüfungs- und Seminararbeiten ein Grund für Nichtbestehen; an einigen Hochschulen sogar für offizielle Rügen oder Exmatrikulation (Abschlussarbeiten)
  - fällt auf (Stilbruch etc.) und wird geprüft
  - Reines Umformulieren von Satz für Satz ist aber genauso Plagiat!

Text produzieren

- Eigene Beiträge kennzeichnen (wenn Sie z. B. eigene Praxistests etc. machen)
- Stil: Passivkonstruktionen, Substantivierungen meiden

Plagiat-Formen

- Copy & Paste (wortwörtliche Wiedergabe)
- Paraphrasierung (Wiedergabe mit eigenen Worten),
- Übersetzungsplagiat
- Strukturplagiat
- „Shake and Paste“ [1]

[1] http://plagiat.htw-berlin.de/ff/schule/3_2/wie

Aktuelles Plagiat (2013)

links: R. Cortiñas, F. C. Freiling, M. Ghajar-Azadanlou, A. Lafuente, M. Larrea, L. Draque Penso, I. Soraluze: "Secure Failure Detection and Consensus in „Trusted-Pals", IEEE TDSC (July-Aug. 2012, vol. 9 no. 4, pp. 608-623)

rechts: Khaderbasha, Shakeel, Babu: „Secure Failure Detection and Aggrements In Trusted Pals", Int. Journal of Innovative Technologies (Vol. 01, Issue 05, Oct 2013)

1 INTRODUCTION

CONSIDER a set of parties who wish to correctly compute some common function F of their local inputs, while keeping their local data as private as possible, but who do not trust each other, nor the channels by which they communicate. This is the problem of *Secure Multiparty Computation* (SMC) [1]. SMC is a very general security problem, i.e., it can be used to solve various real-life problems such as distributed voting, private bidding and online auctions, sharing of signature or decryption functions and so on. Unfortunately, solving SMC is—without extra assumptions—very expensive in terms of communication (number of messages), resilience (amount of redundancy), and time (number of synchronous rounds).

TrustedPals [2] is a smart card-based security framework for solving SMC which allows much more efficient solutions to the problem. Conceptually, TrustedPals considers a distributed system in which processes are locally equipped with tamper-proof security modules. In practice, processes are implemented as a Java desktop application and security

1. INTRODUCTION

Consider a set of parties who wish to correctly compute some common function F of their local inputs, while keeping their local data as private as possible, but who do not trust each other, nor the channels by which they communicate. This is the problem of Secure Multiparty Computation (SMC) [1]. SMC is a very general security problem, i.e., it can be used to solve various real-life problems such as distributed voting, private bidding and online auctions, sharing of signature or decryption functions and so on. Unfortunately, solving SMC is without extra assumptions very expensive in terms of communication (number of messages), resilience (amount of redundancy), and time (number of synchronous rounds).

TrustedPals [2] is a smart card-based security framework for solving SMC which allows much more efficient solutions to the problem. Conceptually, TrustedPals considers a distributed system in which processes are

Bilder und Tabellen

- Jedes Bild hat eine Bildunterschrift
  - beschreibt das Bild – ohne den restlichen Text vorauszusetzen
  - ist keine Kopie eines Satzes aus dem Text
- Für Tabellen gilt dasselbe analog

Umfang, Layout

- Ziel: 10 bis 12 Seiten (echter Text)
  - zzgl. Anhang, Verzeichnisse
- Format / Layout:
  - DIN-A4, 12pt Times New Roman, 1,5-zeilig, Blocksatz, autom. Silbentrennung
  - Abstände: oben 3 cm, unten 2 cm, links 4 cm, rechts 2 cm
  - Seitenzahlen: oben (zwischen Textblock und Papierrand)

Formales im Detail

- FOM-Dokument: „Leitfaden zur formalen Gestaltung von Seminar- und Abschlussarbeiten"
- über Campus-System verfügbar (von Kursseite aus verlinkt)

Abschließende Tipps

- Konsistent sein
  - beim Zitieren
  - bei der Verwendung von Fachbegriffen
- Sich an die (deutsche) Sprache halten
  - englische Wörter übersetzen und (wenn nötig) den Originalbegriff beim ersten Mal nennen
  - steht das Wort im Duden? Dann ist es deutsch
  - Fachbegriffe wie IP-Stack, Bridge, Gateway sind ok (1x erklären)
  - teilweise Geschmacksache („I/O“ oder „E/A“?)
- Sparsame Typographie

Häufige Fehler (1)

- Beschwerden, dass der Stoff zu umfangreich ist. Aus jedem Thema lässt sich problemlos ein 10-, 20- oder 60-minütiger Vortrag gestalten. Das ist eben die Vortragskunst.
- Eine Motivation wird vernachlässigt.
- Sich nicht mit dem roten Faden oder der „Message“ auseinanderzusetzen.
- Es werden Dinge vorausgesetzt, die bei Teilen des Publikums nicht bekannt sind.
- Fachbegriffe werden nicht erklärt, Abkürzungen nicht ausgeführt.

Quelle: Weicker & Weicker, Lehrstuhl 4, Univ. Erl.-Nbg. [5]

Häufige Fehler (2)

- Gliederung wird anhand von Stichworten erstellt, die dann in jeweils ein bis zwei Sätzen ausgeführt werden. Meist geht hierbei der rote Faden verloren: die Arbeit wirkt abgehackt und unverständlich
- Gliederung der Folien für den Vortrag oder Gliederung des Vortrags für die Ausarbeitung verwenden
- Zu viel Zeit auf Folien verwenden, aber das Gesprochene improvisieren
- Den Folieninhalt nur vorlesen oder sich an den Folien entlang zu hangeln ohne den Vortrag lebendig werden zu lassen
- Mit Powerpoint-Effekten über inhaltliche Defizite hinwegtäuschen

Quelle: Weicker & Weicker, Lehrstuhl 4, Univ. Erl.-Nbg. [5]

Literatur

[1] Marcus Deininger, Horst Lichter, Jochen Ludewig, Kurt Schneider: Studienarbeiten. Ein Leitfaden zur Vorbereitung, Durchführung und Betreuung von Studien-, Diplom-, Abschluss- und Doktorarbeiten am Beispiel Informatik. vdf, 5. Auflage, 2005.

[2] Albers et al.: Gute wissenschaftliche Praxis für das Verfassen wissenschaftlicher Qualifikationsarbeiten, http://www.hochschulverband.de/cms1/uploads/media/Gute_wiss._Praxis_Fakultaetentage.pdf, Juli 2012.

[3] Wikipedia zu „et al.“: http://de.wikipedia.org/wiki/Et_al.

[4] Peter Rechenberg: Technisches Schreiben. Hanser, 2. Auflage, 2003.

[5] Karsten Weicker, Nicole Weicker: Seminarrichtlinien am Lehrstuhl 4, https://www4.cs.fau.de/Lehre/SS11/PS_KVBK/info/Weicker.pdf

[6] Markus Rath: Leitfaden zur Anfertigung von Seminar-, Studien- und Diplomarbeiten, http://www.wi-inf.uni-duisburg-essen.de/FGFrank/documents/Lehre/Leitfaden_WissArbeit.pdf

Fragebogen

bitte jetzt ausfüllen …

Arbeiten mit LaTeX

Motivation (1)

- besserer Textsatz

Das ist ein Beispiel für misslungenen Blocksatz. Vor allem die im Deutschen üblichen Langwortkombinationsmöglichkeiten führen dazu, dass in den Zeilen hässliche Lücken entstehen oder Buchstaben innerhalb der Wörter stark auseinander gezogen werden.

(Office, Silbentrennung aus)

Das ist ein Beispiel für misslungenen Blocksatz. Vor allem die im Deutschen üblichen Langwort-kombinationsmöglichkeiten führen dazu, dass in den Zeilen hässliche Lücken entstehen oder Buchstaben innerhalb der Wörter stark auseinander gezogen werden. *Silbentrennung hilft.*

(Office, Silbentrennung an)

Das ist ein Beispiel für ~~misslungenen~~ Blocksatz. Vor allem die im Deutschen üblichen Langwort-kombinationsmöglichkeiten führen dazu, dass in den Zeilen hässliche Lücken entstehen oder Buchstaben innerhalb der Wörter stark auseinander gezogen werden. *Silbentrennung in LaTeX: gelungen!*

(LaTeX)

Motivation (2)

- automatische Literaturverwaltung
  - Quelle verwendet → taucht in Lit.-Verzeichnis auf
  - Quelle *nicht* verwendet → taucht *nicht* auf
- Mathematische Formeln verwenden?

$$\int_0^\infty \frac{f(x)+g(x)}{f(x)-g(x)}\mathrm{d}x = \big(\varphi(z)\big)\big|_{z=0} + \lim_{n\to\infty}\frac{2^n}{n!}\Delta_n^{i_n}$$

- automatische Verzeichnisse (Inhalt, Abbildungen, Tabellen etc.)

Motivation (3)

- Kein Layout-Overkill

ZU VIEL DES GUTEN

In LibreOffice Writer kann man, wenn es denn sein muss, viele Schriftarten und schriftgrößen kombinieren und dazu noch *alle* möglichen *Textattribute einsetzen*. Zum Beispiel mit anderer Schrift für jedes Wort. Unbedingt dazu gehört dann auch die MICROSOFT-Schriftart Comic Sans MS.

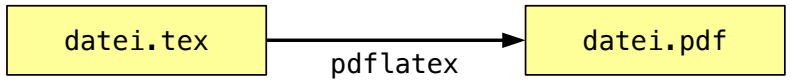
Ganz zu schweigen von Fontwork (das bei Word „Word Art“ heißt).

LaTeX (1)

- Seminararbeit evtl. mit LaTeX erstellen
- auch gute Vorbereitung für Bachelorarbeit
- hier eine Einführung in die grundlegenden Konzepte
  - LaTeX basiert auf TeX und ist ein Makropaket, das TeX leichter benutzbar macht
  - zentrale Idee: Trennung zwischen Layout und Inhalt
  - Autor konzentriert sich auf Inhalt, LaTeX übernimmt das Layout

LaTeX (2)

- LaTeX ist ein Dokumenten-*Compiler*.
- Text in einem Editor Ihrer Wahl erstellen
- Als Kodierung UTF8 einstellen
- speichern als `*.tex`-Datei
- übersetzen mit `pdflatex`, erzeugt `*.pdf`-Datei



LaTeX (3)

- Wenn LaTeX beim Setzen auf einen Fehler trifft, stoppt es den Vorgang und fragt den Benutzer
  - Wenn die Fehlermeldung unklar ist, einfach [Eingabe] drücken → LaTeX versucht, den Fehler selbst zu beheben
  - Aussehen der PDF-Datei deutet dann meist auf den Fehler hin
  - Wenn viele Fehler auftreten: „S“ eingeben (scrollmode)
  - Wie dauerhaftes Drücken von [Eingabe]

LaTeX (4)

- Sonderzeichen: \ $ # ^ _ muss man „escapen“, um sie benutzen zu können
  - `\$` erzeugt $
  - `\#` erzeugt #
  - `\textbackslash{}` erzeugt \
  - `\textasciitilde{}` erzeugt ~ (oder: `\verb#~#`)
  - `\^{}` erzeugt ^ (oder: `\verb#^#`)
  - `\_` erzeugt _
- Befehle i. d. R. mit `{}` abschließen: `\today{}`

LaTeX (5)

Beispiel für Weglassen der Klammern `{}`:

Heute ist der 35. Mai 2012. Oder: Heute ist der 35. Mai 2012. Falsch ist: Am 35. Mai 2012regnet es. Richtig: Am 35. Mai 2012 scheint die Sonne. Oder: Am 35. Mai 2012 schneit es.

```
Heute ist der \today.
Oder: Heute ist der \today .
Falsch ist:
 Am \today regnet es.
Richtig:
 Am \today{} scheint die Sonne.
 Oder: Am \today\ schneit es.
```

(Quelle: LaTeX2e-Kurzbeschreibung, Version 3.0 von Marco Daniel, Patrick Gundlach, Walter Schmidt, Jörg Knappen, Hubert Partl und Irene Hyna)

LaTeX (6)

- Leerzeichen / Leerzeilen (1)
  - Leerzeichen, Tabulatoren und Zeilenumbrüche werden alle als (ein) Leerzeichen interpretiert
  - auch mehrere davon (außer Umbrüche) werden zu einem Leerzeichen
  - Absatzumbruch durch Einfügen einer (oder mehrerer) Leerzeilen
  - keine Befehle wie `\newline` oder `\\` zum Erzwingen des Umbruchs verwenden
  - neue Seite: `\pagebreak`

LaTeX (7)

- Leerzeichen / Leerzeilen (2)
  - Folgende Texte erzeugen das gleiche Dokument:

```
Abc abc abc
Abc Abc Bcad

Abcda abc afb
```

```
Abc    abc    abc
Abc    Abc    Bcad

Abcda  abc  afb
```

LaTeX (8)

- Befehle und Gruppen
  - LaTeX-Kommandos sind entweder einzelne **Befehle** wie `\today{}` oder
  - Gruppen, die mit `\begin{gruppe}` begonnen und mit `\end{gruppe}` beendet werden

```
\begin{center}
Abgesetzter zentrierter Text
\end{center}
```


LaTeX (9)

- Dokumentstruktur
  - einheitlicher Aufbau:

```
\documentclass{article}    % oder andere Klasse
% Präambel
% Befehle wie \usepackage{...} für LaTeX-Pakete
\begin{document}
% Dokument-Inhalt
Mein erstes Beispiel-Dokument.
\end{document}
```

  - **Klasse:** definiert globale Dokumenten-Eigenschaften (Layout)
  - **Pakete:** ergänzen Zusatzfunktionen

LaTeX (10)

- Aufzählungen (auch geschachtelt)
  - Bullet-Listen:

```
\begin{itemize}
\item Absatz 1
\item Absatz 2
\end{itemize}
```

→

- Absatz 1
- Absatz 2

  - Nummeriert:

```
\begin{enumerate}
\item Absatz 1
\item Absatz 2
\end{enumerate}
```

→

1. Absatz 1
2. Absatz 2

LaTeX (11)

- Einfache Auszeichnungen
  - `\textbf{Text}`: **fett**
  - `\emph{Text}`: hervorgehoben (*kursiv*)
  - `\underline{Text}`: <u>unterstrichen</u>
- Bilder und Tabellen: über `figure`-, `table`- und `tabular`-Umgebungen sowie den Befehl `\includegraphics{bilddatei}` (ohne Endung)
- Bild-/Tabellen-Unterschrift mit `\caption{Text.}`
- Labels: `\label{label}`, Verweise: `\ref{label}`, `\pageref{label}` (Seitenzahl)

BibTeX / Biber (1)

- Literaturverwaltung über separates Tool (`bibtex` oder `biber`)
- Quellen in `*.bib`-Dateien mit eigener Syntax verwalten
- Typische Sequenz, um Dokument `dokument.tex` zu bauen:

```
pdflatex dokument
bibtex dokument
pdflatex dokument
pdflatex dokument
```


BibTeX / Biber (2)

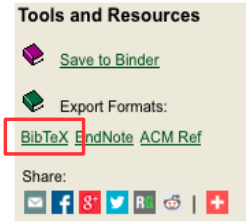
- Syntax in den `*.bib`-Dateien

```
@article{CS03,
 author = "Carrier, Brian and Spafford, Eugene H.",
 title = "Getting physical with the digital
          investigation process",
 journal = "International Journal of Digital Evidence",
 volume = 2,
 year = 2003
}
```

- `CS03` ist ein BibTeX-Label → `\cite{CS03}`
- Beispiele in `literatur/literatur.bib`

BibTeX / Biber (3)

- BibTeX-Tipps
  - für Bücher: Google-Suche nach „ISBN to Bibtex“ → http://manas.tungare.name/software/isbn-to-bibtex/ (oder andere Konverter-Seiten)
  - für Artikel: z. B. auf der ACM-DL-Seite (http://dl.acm.org), Beitrag suchen, Detailseite aufrufen, Bib-TeX-Export



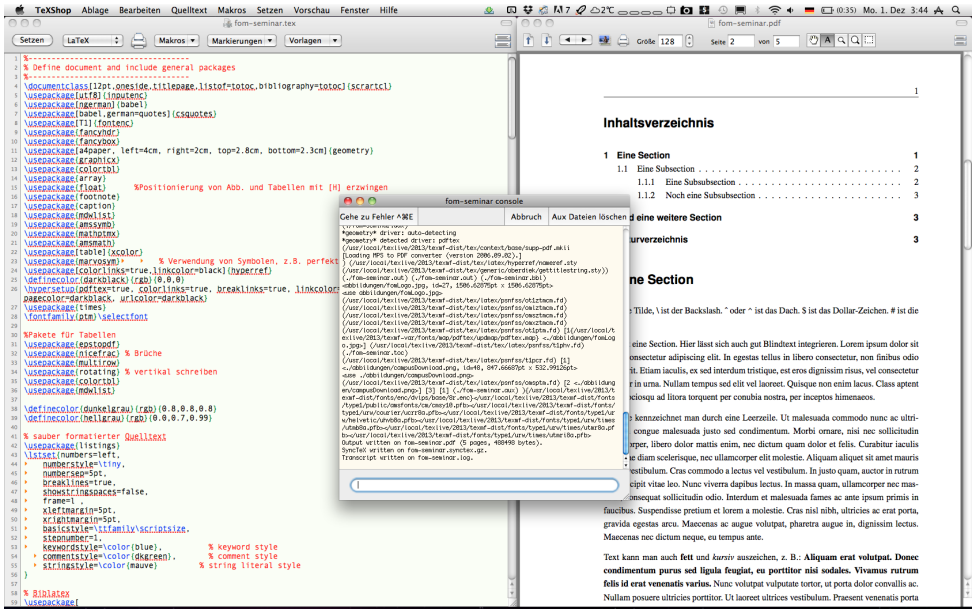
LaTeX-Tipps (1)

- Ausprobieren
- Mit funktionierenden Beispielen starten (z. B. mit der Vorlage; siehe auch folgende Folien)
- auf den Inhalt konzentrieren / nicht durch Layout-Fragen ablenken lassen
- ggf. nach speziellen Paketen suchen; LaTeX bringt umfangreiche Dokumentation mit:

  `texdoc paketname` (Mac, Linux)

LaTeX-Tipps (2)

- für mehr Komfort: LaTeX-IDE einsetzen
  - TeXShop (Mac), http://pages.uoregon.edu/koch/texshop/
  - TeXstudio (Windows, Mac, Linux u. a.), http://texstudio.sourceforge.net/
  - Features:
    - Aufrufe von pdflatex, bibtex/biber und anderen Tools über Tastenkombination
    - parallele Ansicht von Editor und Vorschau
    - SyncTeX
    - ggf. Debugging-Hilfe



(TeXShop, OS X)

LaTeX: Arbeiten mit der FOM-Vorlage (1)

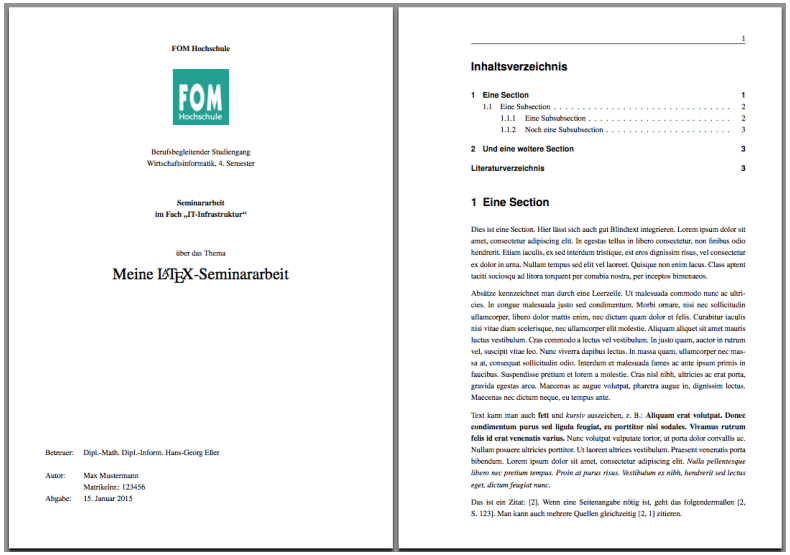
- Datei `fom-seminar.zip` (im Campus-System) herunterladen und entpacken; erzeugt Ordner `fom-seminar/`.
- Darin liegt eine Datei `fom-seminar.tex` – das ist das Hauptdokument
- Literaturverzeichnis: `literatur/literatur.bib`
- Übersetzen mit `./compile.sh` (Linux, OS X) bzw. `compile.bat` (Windows)

LaTeX: Arbeiten mit der FOM-Vorlage (2)

- Anpassungen:

```
% Autor
\newcommand{\myAutor}{Max Mustermann}
% Titel der Arbeit
\newcommand{\myTitel}{Meine \LaTeX{}-Seminararbeit}
% Betreuer
\newcommand{\myBetreuer}{Dipl.-Math. Dipl.-Inform. Hans-Georg Eßer}
% Matrikelnummer
\newcommand{\myMatrikelNr}{123456}
% Ort
\newcommand{\myOrt}{Nürnberg}
% Datum der Abgabe
\newcommand{\myAbgabeDatum}{15. Januar 2015}
% Semesterzahl
\newcommand{\mySemesterZahl}{4}
% Name der Hochschule
\newcommand{\myHochschulName}{FOM Hochschule}
% Studiengang
\newcommand{\myStudiengang}{Wirtschaftsinformatik}
% Art der Arbeit
\newcommand{\myThesisArt}{Seminararbeit}
```

LaTeX: Arbeiten mit der FOM-Vorlage (3)



FOM Hochschule

Berufsbegleitender Studiengang
Wirtschaftsinformatik, 4. Semester

Seminararbeit
im Fach „IT-Infrastruktur“

über das Thema

Meine LaTeX-Seminararbeit

Betreuer: Dipl.-Math. Dipl.-Inform. Hans-Georg Eßer

Autor: Max Mustermann
Matrikeln.: 123456
Abgabe: 15. Januar 2015

1

Inhaltsverzeichnis

1 Eine Section **1**
1.1 Eine Subsection 2
1.1.1 Eine Subsubsection 2
1.1.2 Noch eine Subsubsection 3
2 Und eine weitere Section **3**
Literaturverzeichnis **3**

1 Eine Section

Dies ist eine Section. Hier lässt sich auch gut Blindtext integrieren. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In egestas tellus in libero consectetur, non finibus odio hendrerit. Etiam iaculis, ex sed interdum tristique, est eros dignissim risus, vel consectetur ex dolor in urna. Nullam tempus sed elit vel laoreet. Quisque non enim lacus. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Absätze kennzeichnet man durch eine Leerzeile. Ut malesuada commodo nunc ac ultricies. In congue malesuada justo sed condimentum. Morbi ornare, nisi nec sollicitudin ullamcorper, libero dolor mattis enim, nec dictum quam dolor et felis. Curabitur iaculis nisi vitae diam scelerisque, nec ullamcorper elit molestie. Aliquam aliquet sit amet mauris luctus vestibulum. Cras commodo a lectus vel vestibulum. In justo quam, auctor in rutrum vel, suscipit vitae leo. Nunc viverra dapibus lectus. In massa quam, ullamcorper nec massa at, consequat sollicitudin odio. Interdum et malesuada fames ac ante ipsum primis in faucibus. Suspendisse pretium et lorem a molestie. Cras nisl nibh, ultricies ac erat porta, gravida egestas arcu. Maecenas ac augue volutpat, pharetra augue in, dignissim lectus. Maecenas nec dictum neque, eu tempus ante.

Text kann man auch **fett** und *kursiv* auszeichen, z. B.: **Aliquam erat volutpat. Donec condimentum purus sed ligula feugiat, eu porttitor nisi sodales. Vivamus rutrum felis id erat venenatis varius.** Nunc volutpat vulputate tortor, ut porta dolor convallis ac. Nullam posuere ultricies porttitor. Ut laoreet ultrices vestibulum. Praesent venenatis porta bibendum. Lorem ipsum dolor sit amet, consectetur adipiscing elit. *Nulla pellentesque libero nec pretium tempus. Proin at purus risus. Vestibulum ex nibh, hendrerit sed lectus eget, dictum feugiat nunc.*

Das ist ein Zitat: [2]. Wenn eine Seitenangabe nötig ist, geht das folgendermaßen [2, S. 123]. Man kann auch mehrere Quellen gleichzeitig [2, 1] zitieren.