Martin Glinz

Prof. Dr. rer. nat.

Software Engineering I: Grundlagen der Systementwicklung

(Grundstudium, drittes Semester)

Universität Zürich
Institut für Informatik

Im falschen Film?

Im falschen Film?

Im falschen Film?

Im falschen Film?

Im falschen Film?

Sind Sie sicher?

> "Software is arguably the world's most important industry. The presence of software has made possible many new businesses and is responsible for increased efficiencies in most traditional businesses."
>
> Grady Booch (Communications of the ACM, März 2001)

- Stellen Sie sich die Welt (und Ihr persönliches Leben!) ohne Software vor...
- ...und vergessen Sie nicht: Software hat es nicht nur in Ihrem PC
- Software ist überall.

Software ist überall

- Im Alltag: Telefon, Fernseher, Waschmaschinen, ...
- In Fahrzeugen:
  - Autos ($675 Stahl vs. $2500 Elektronik in neuen Autos von GM)
  - Lokomotiven (Re 4/4-460: 32 Prozessoren)
- In Flugzeugen (“fly-by-wire”)
- In der Industrie: Konstruktion, Produktionsplanung, Produktionssteuerung, Logistik, Bestellwesen, Buchhaltung,...
- In Handel und Dienstleistungen: Banken, Börse, Devisenmärkte, Reisebüros, elektronischer Handel mit Gütern und Information...
- In der Energieversorgung: Stromerzeugung, Stromverteilung
- In den Medien: Zeitungen, Fernsehen, Internet
- ...

Preisfrage

- Wie erhalten Sie Zugang zum größten Softwaresystem der Welt?
- Nehmen Sie einen Telefonhörer in die Hand.

Worum es geht

Professor, Software Engineering lehrend
(frei nach Wilhelm Busch)

Die traur'ge Wahrheit lautet schlicht
in der Praxis tun's die meisten nicht.

Man glaubt es kaum, es ist ein Graus,
sie werfen das Geld zum Fenster hinaus.

Vom Software Engineering sind sie weit entfernt –
der Grund ist der: sie haben's nie gelernt.

Drum, liebe Leute kommt herbei,
ihr braucht SE, ich bring's euch bei.

Ziele der Vorlesung

- Die Studierenden verfügen über ein Grundwissen in Software Engineering und kennen die Mittel zur wirtschaftlichen Herstellung guter Software.
- Auf dieser Grundlage können sie
  - den Stellenwert und Bedeutung von Software und Software Engineering in Wirtschaft und Alltag beurteilen,
  - ihr Wissen über Software-Entwicklung systematisch einordnen,
  - die Ursachen für Schwierigkeiten und Probleme in Software-Projekten erkennen,
  - eigene Software-Entwicklungsvorhaben systematisch und zielgerichtet angehen,
  - als Anwender bzw. als Manager erfolgreich mit Softwareentwicklern zusammenarbeiten.

Inhalt, Stoffplanung

1	Einführung
2	Zielsetzung, Messung
3/4	Der Software-Prozess
5	Software-Projektführung
6	Software-Aufwandschätzung
7/8	Konzipieren von Lösungen
9/10	Spezifikation von Anforderungen
11	Realisierung, Dokumentation
12/13	Qualitätsmanagement, Konfigurationsverwaltung
14	Produktivitätsfaktoren, Abschluss

Information

Informationen zur Vorlesung auf dem WWW:

http://www.ifi.unizh.ch/req/courses/se_I

Übungen

- Mini-Übungen in der Vorlesung mit kleinen Fallstudien, Gedanken-anstößen, etc.

- Zwei freiwillige Übungen als Hausarbeit mit Aufgaben im Stil von Vorprüfungsaufgaben

Unterlagen

- Ausführliches Skript
  - im Studentenladen käuflich erwerbbar (Fr. 20.–)
  - Im WWW herunterladbar
- Kommentiertes Verzeichnis von Lehrbüchern im Skript

Literatur

- Ausführliches kommentiertes Literaturverzeichnis im Skript
- Keine Empfehlung für ein bestimmtes Lehrbuch
- Für die Prüfungsvorbereitung genügt das Skript

Prüfung

- Software Engineering I ist Bestandteil der Vorprüfung Informatik Grundstufe

- Der Stoff der Prüfung ist durch einen Lernzielkatalog definiert (erhältlich auf den Webseiten zur Vorlesung)

Software Engineering I

Prof. Dr. Martin Glinz

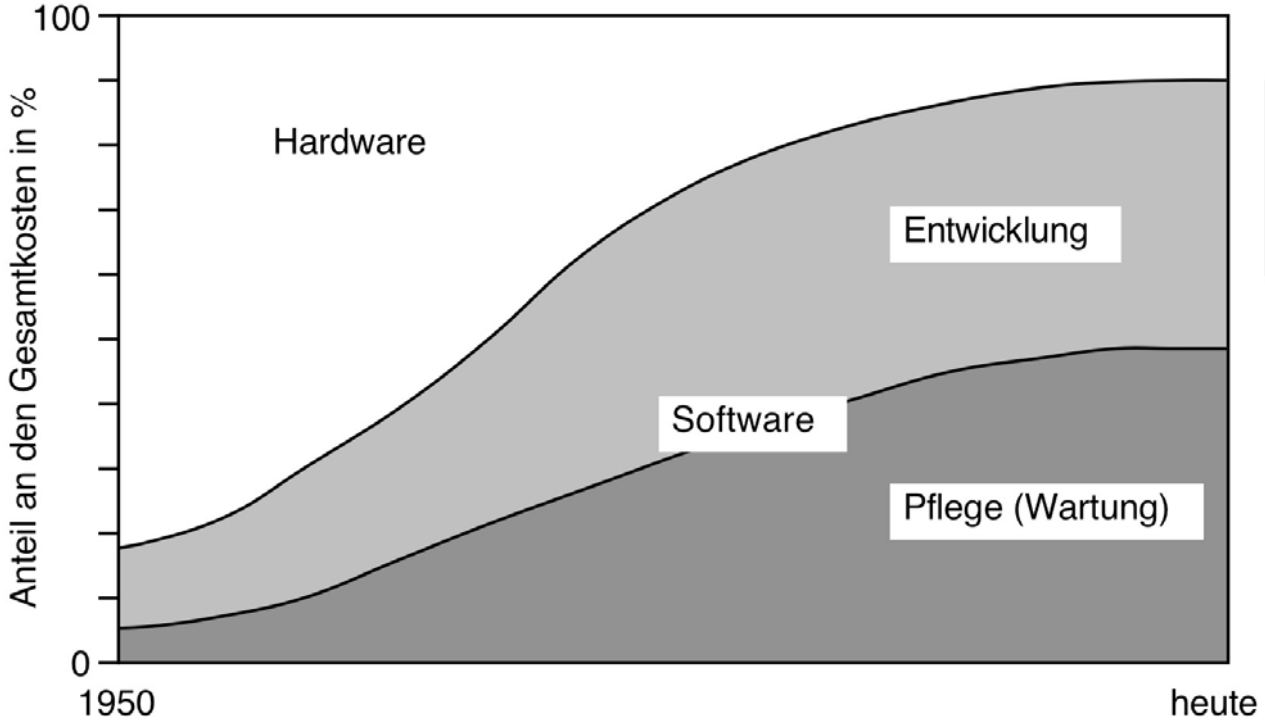
Kapitel 1

Einführung

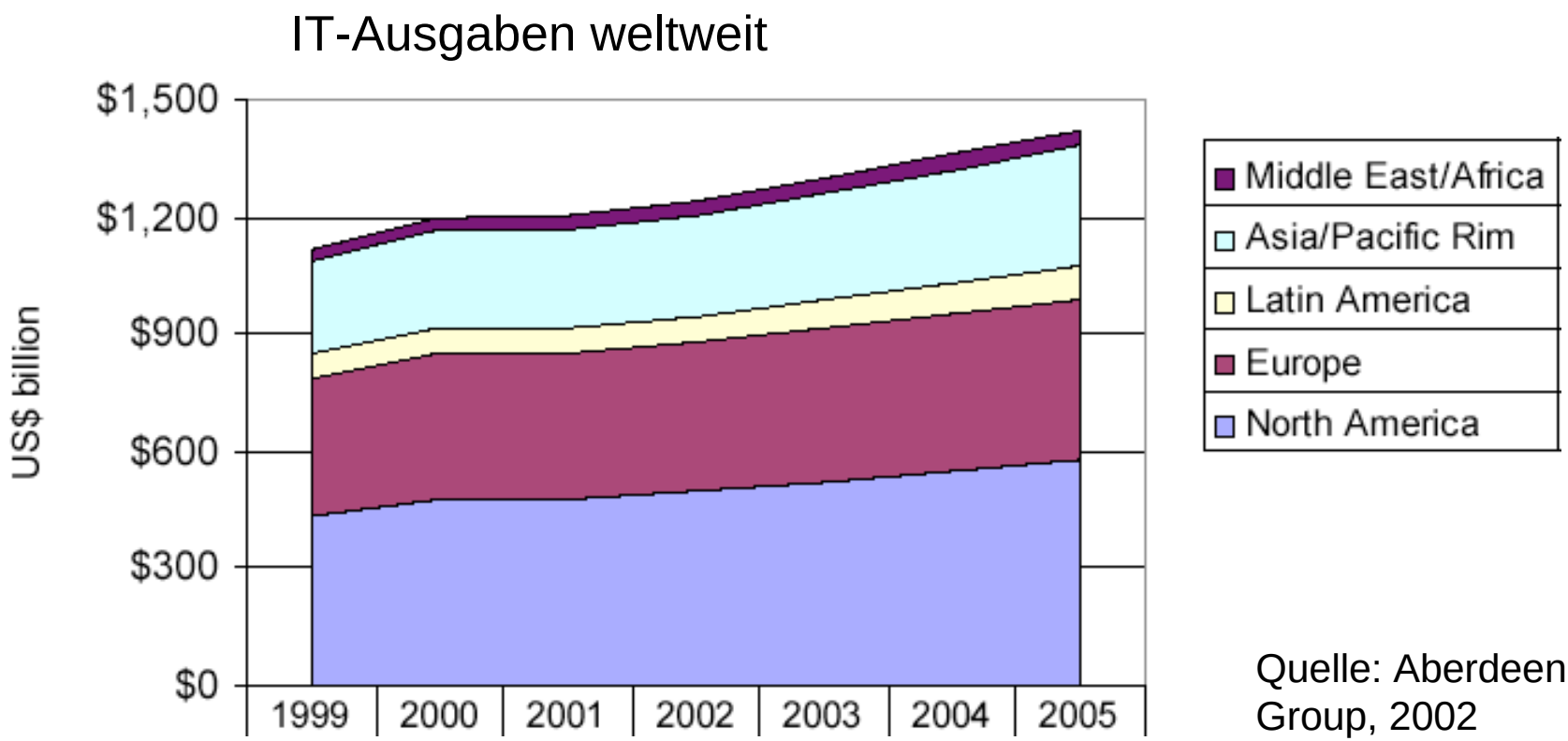
Universität Zürich
Institut für Informatik

Software-Entwicklung als Problem

- Ohne Software geht heute nichts mehr
- Die Entwicklung von Software wird nur ungenügend beherrscht
- Software-Kosten dominieren die Kosten von Informatiksystemen



Ausgaben für Software weltweit: horrende Summen



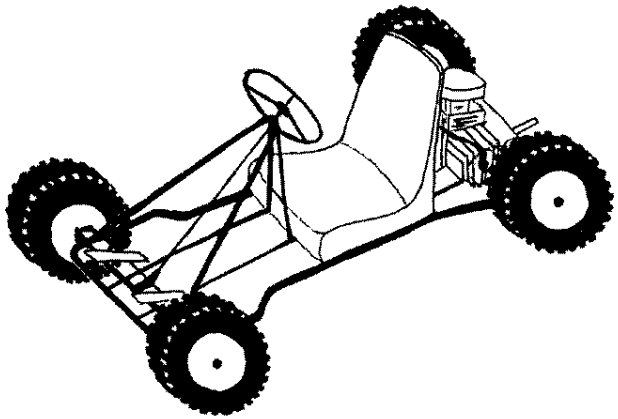
Quelle: Aberdeen Group, 2002

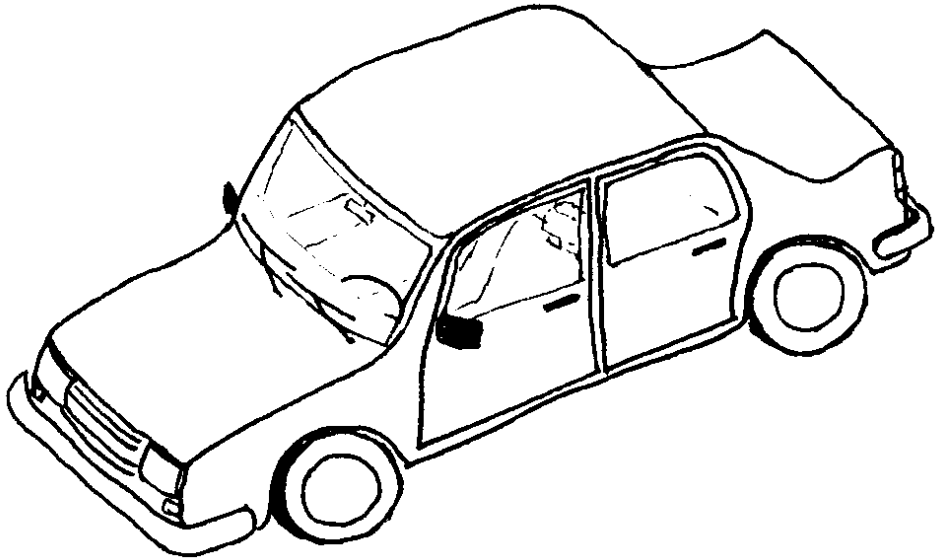
- Es gibt Verbesserungspotential
- ➪ Software Engineering

Was ist Software?

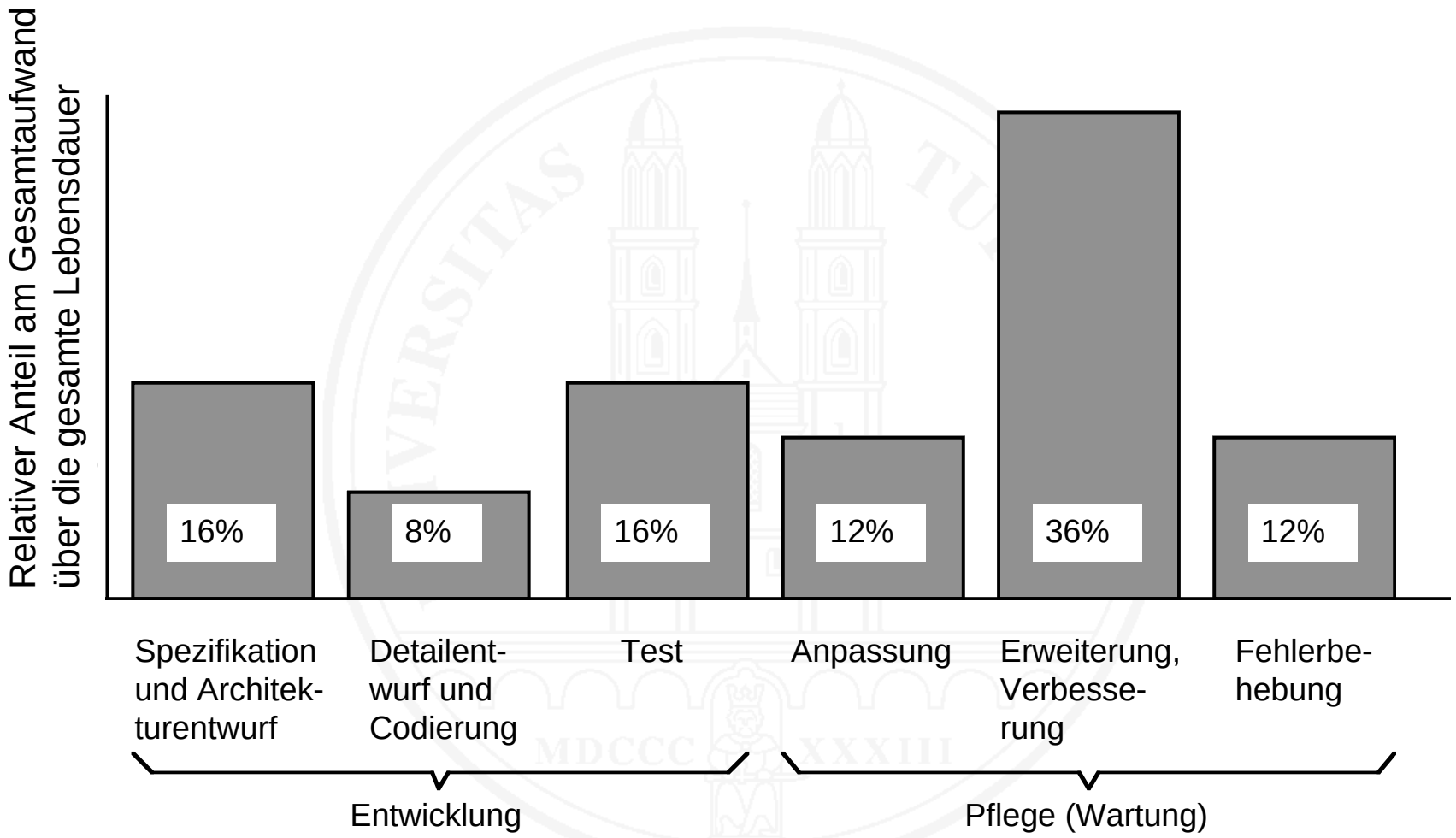
Software. Die Programme, Verfahren, zugehörige Dokumentation und Daten, die mit dem Betrieb eines Rechnersystems zu tun haben (IEEE 610.12).

➪ *Mehr als nur Programme.*



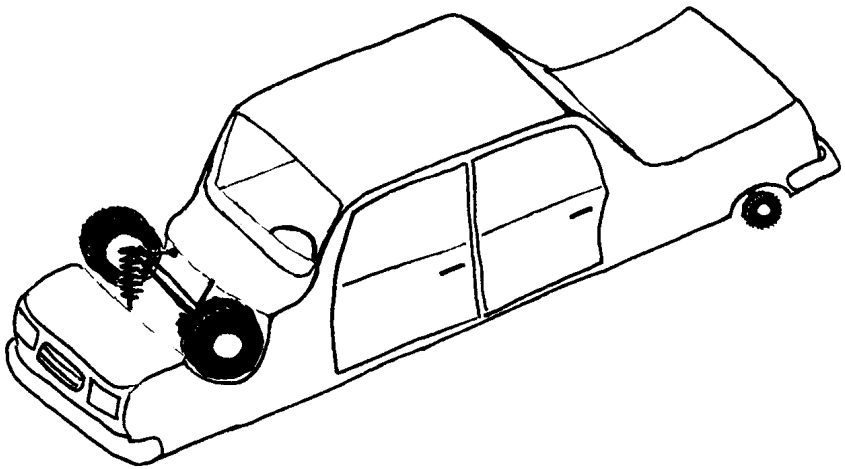


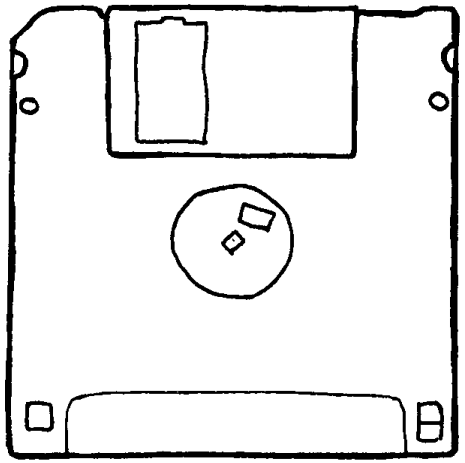
Verteilung des Aufwands über die Lebensdauer



Ein immaterielles technisches Produkt – 1

Software kann man nicht anfassen.





Diese mechanische Konstruktion ist offensichtlich falsch.

Wie erkennen wir aber Fehler in der hier gespeicherten Software-Konstruktion?

Ein immaterielles technisches Produkt – 2

- Beobachtbar nur
  - in den Wirkungen beim Ablauf auf Rechnern
  - indirekt über die Dokumentation der Software

- Kein Materialwert
- Keine physikalischen Grenzen
- Fehler sind schwieriger erkennbar
- Entwicklungsstand und Qualität schwer zu beurteilen
- Scheinbar leicht zu ändern

Verhält sich unstetig

- kleinste Änderungen in der Software
  - ➪ massive Änderungen im Verhalten
- Nachweis des wunschgemäßen Funktionierens schwierig

➪ schwieriger als andere technischen Produkte

Wozu Software?

- Probleme lösen
- Automatisierung oder Unterstützung von
  - menschlicher Arbeit
  - technischen Abläufen
- steht in ständiger Wechselwirkung mit
  - Arbeitsprozessen
  - Produktionsprozessen
  - Menschen

Konsequenzen

- Komplexes Problem ➪ komplexe Lösung
- Immer zwei Schwierigkeiten
  - Problem im Kontext verstehen
  - Problemlösung auf Software abbilden
- Problemlösungen schaffen neue Realitäten und Bedürfnisse

➪ Software konstruiert und verändert die Realität

Software-Entwicklung

Software-Entwicklung. Die Umsetzung der Bedürfnisse von Benutzern in Software. Umfasst Spezifikation der Anforderungen, Konzept der Lösung, Entwurf und Programmierung der Komponenten, Zusammensetzung der Komponenten und ihre Einbindung in vorhandene Software, Inbetriebnahme der Software sowie Überprüfung des Entwickelten nach jedem Schritt.

Von grundlegender Wichtigkeit für das Verständnis der Probleme:

- Klein ≠ Groß
- Aufwand steigt überproportional mit Produktgröße
- Software ist einer Evolution unterworfen
- Software wird von Menschen gemacht

Mini-Übung 1.1 (Aufgabe 1.4 im Skript)

Eine Kundenbetreuerin im Firmenkundengeschäft einer Bank hat auf der Grundlage eines Tabellenkalkulationsprogramms eine kleine persönliche Anwendung geschrieben, die sie bei der Überprüfung der Kredite der von ihr betreuten Firmen unterstützt. Die notwendigen Daten gibt sie jeweils von Hand ein.

Der Abteilungsleiter sieht diese Anwendung zufällig, ist davon angetan und beschließt, sie allen Kundenbetreuerinnen und -betreuern zur Verfügung zu stellen. Die notwendigen Daten sollen neu automatisch als den Datenbanken der Bank übernommen werden.

Mini-Übung 1.1 (Fortsetzung)

Die Kundenbetreuerin gibt an, für die Entwicklung ihrer Anwendung insgesamt etwa vier Arbeitstage aufgewendet zu haben. Der Abteilungsleiter veranschlagt daher für die Übernahme und die gewünschten Änderungen einen Aufwand von einer Arbeitswoche. Als die geänderte Anwendung endlich zur Zufriedenheit aller Beteiligten läuft, sind jedoch rund acht Arbeitswochen Aufwand investiert.

Der Abteilungsleiter erzählt die Geschichte einem befreundeten Berater als Beispiel, dass Informatik-Projekte nie ihre Termine einhalten. Darauf meint der Berater trocken, der investierte Aufwand sei völlig realistisch und normal.

Begründen Sie, warum.

Klein vs. Groß in der Software-Entwicklung – 1

Klein	**Groß**
Programme von 1 bis ca. 300 Zeilen	Längere Programme
Für den *Eigengebrauch*	Für den Gebrauch durch *Dritte*
Vage Zielsetzung genügt	*Genaue Zielbestimmung* erforderlich
Ein Schritt vom Problem zur Lösung genügt: Lösung direkt programmiert	*Mehrere Schritte* erforderlich
Validierung am *Endprodukt*	*Auf jeden Entwicklungsschritt muss ein Prüfschritt* folgen
Eine Person entwickelt: Keine Koordination notwendig, keine Kommunikationsbedürfnisse	*Mehrere Personen* entwickeln gemeinsam: Koordination und Kommunikation notwendig

Klein vs. Groß in der Software-Entwicklung – 2

Klein	**Groß**
Komplexität des Problems in der Regel *klein*,	*Komplexität* des Problems *größer bis sehr groß*,
Behalten der Übersicht nicht schwierig	Maßnahmen zur Strukturierung und Modularisierung erforderlich
Software besteht aus *wenigen Komponenten*	Software besteht aus *vielen Komponenten*; Maßnahmen zur Komponentenverwaltung erforderlich
In der Regel *keine Dokumentation*	*Dokumentation* zwingend
Keine Planung und *Projektorganisation* erforderlich	*Planung* und *Projektorganisation zwingend* erforderlich

Mini-Übung 1.2 (Aufgabe 1.2 im Skript)

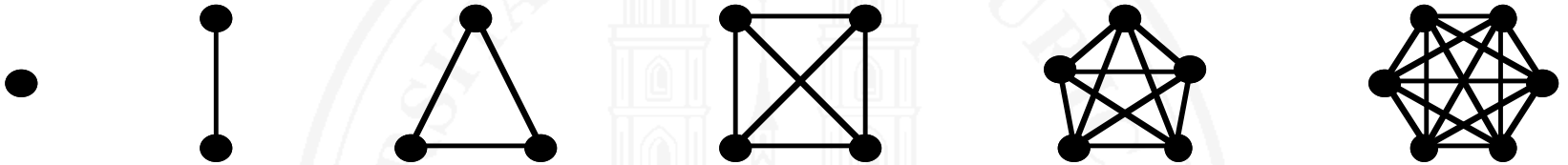
«Ein Mann braucht zum Bau einer 2 m langen Brücke 0,5 Tage. Wie lange brauchen 100 Leute für den Bau einer 2 km langen Brücke? Rechne.»

Begründen Sie, warum das eine Milchmädchenrechnung ist. Ziehen Sie Parallelen zur Entwicklung von Software.

Wachstum des Kommunikationsbedarfs

Anzahl beteiligte Personen

1 2 3 4 5 6



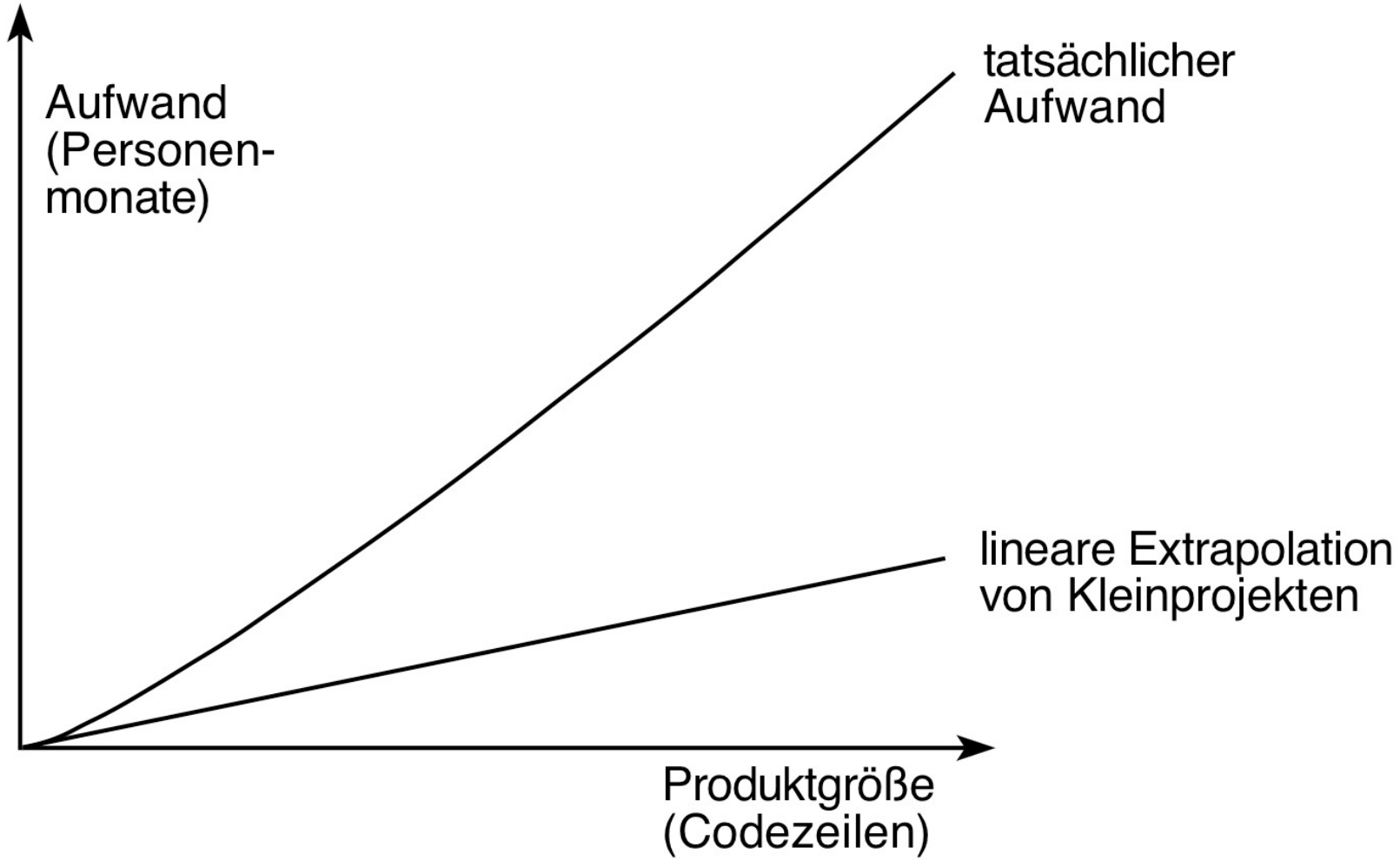
Anzahl Kommunikationspfade

0 1 3 6 10 15

Quantensprung: Kommunikation wird erforderlich

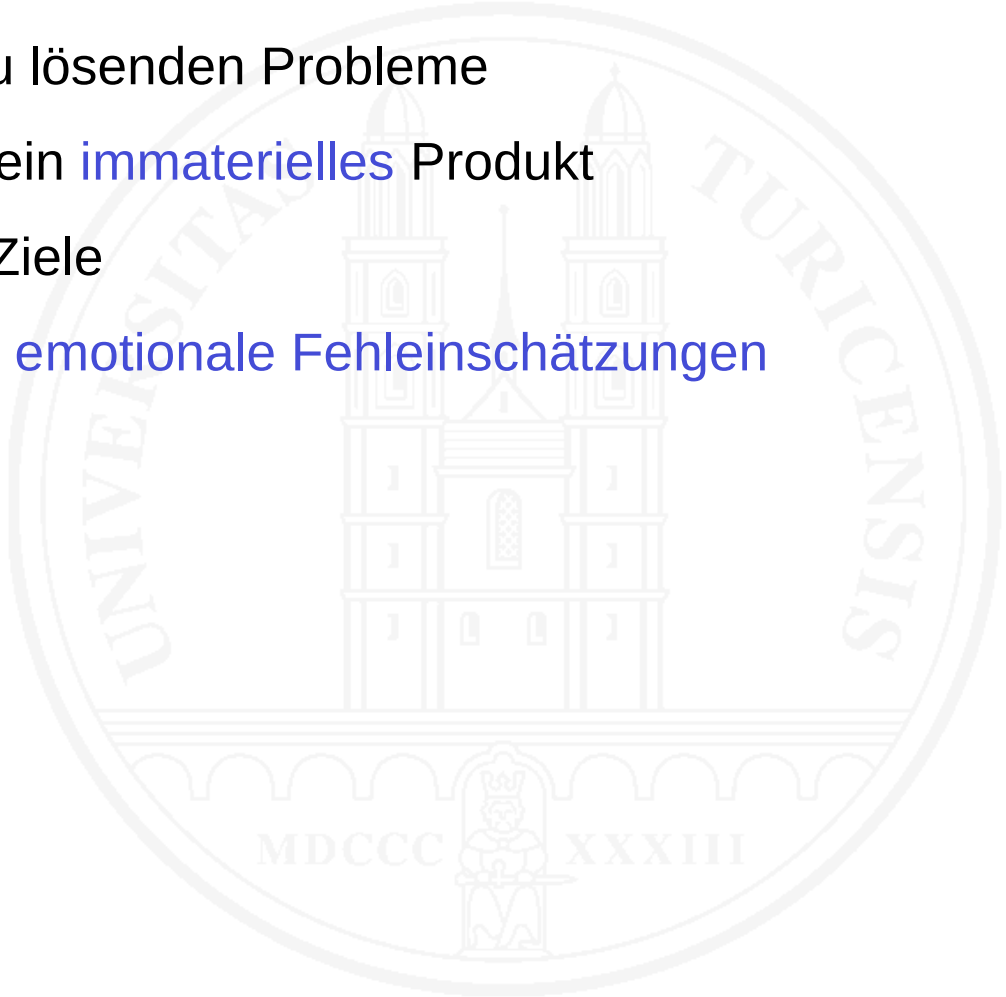
Quantensprung: Zahl der Kommunikationspfade übersteigt Zahl der Personen

Der Aufwand steigt überproportional



Warum ist Software-Entwicklung schwierig?

- Größe der zu lösenden Probleme
- Software ist ein immaterielles Produkt
- Bewegliche Ziele
- Fehler durch emotionale Fehleinschätzungen



Software Engineering

- Die Antwort auf die Software-«Krise»
- 1968 durch F.L. Bauer in Garmisch-Partenkirchen

Software Engineering ist das technische und planerische Vorgehen zur systematischen Herstellung und Pflege von Software, die zeitgerecht und unter Einhaltung der geschätzten Kosten entwickelt bzw. modifiziert wird (Fairley 1985).

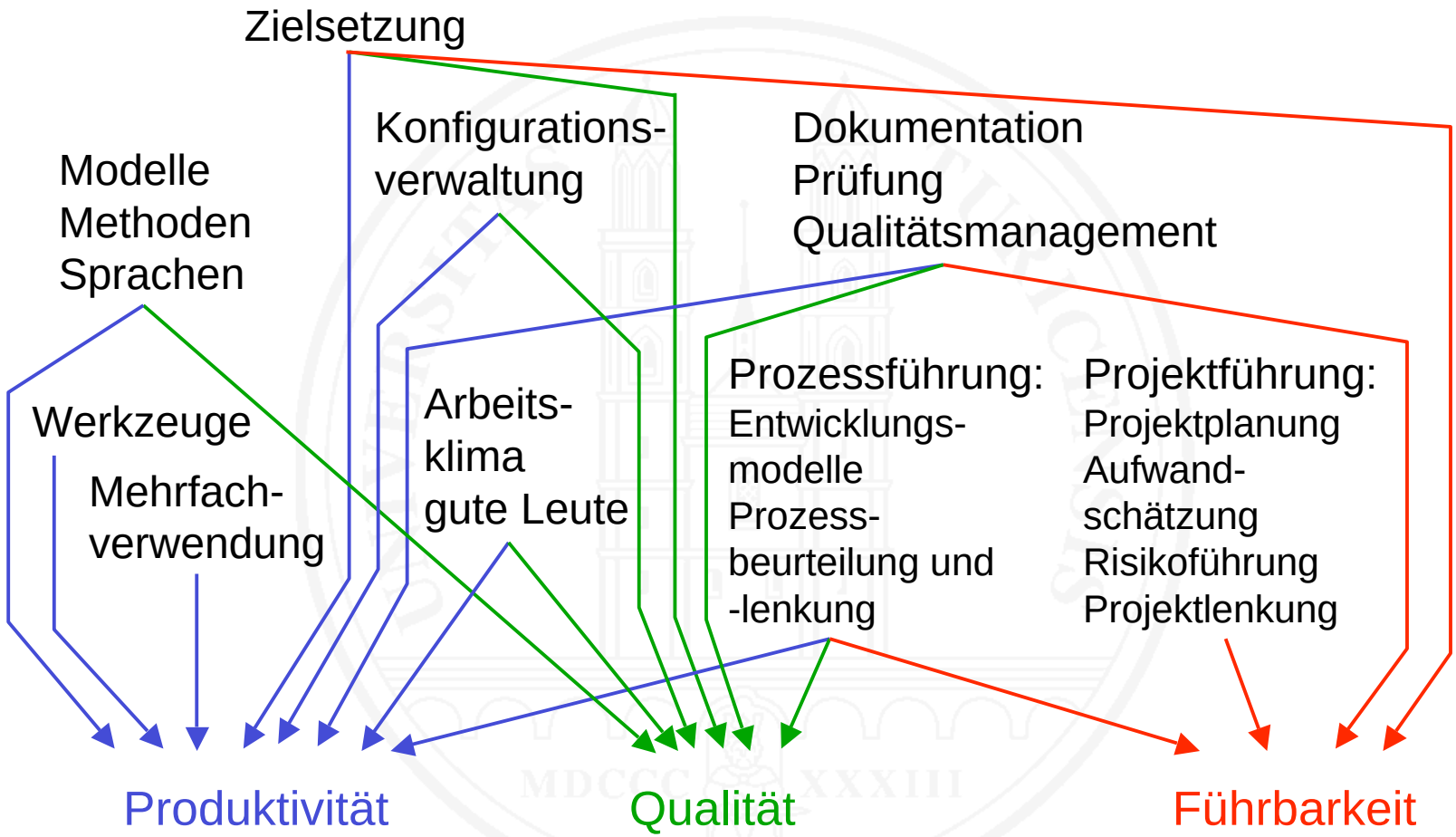
Software Engineering. Die Anwendung eines systematischen, disziplinierten und quantifizierbaren Ansatzes auf die Entwicklung, den Betrieb und die Wartung von Software, das heißt, die Anwendung der Prinzipien des Ingenieurwesens auf Software. (IEEE 610.12)

- Im deutschen Sprachraum auch: *Softwaretechnik*

Die Ziele des Software Engineerings

- Steigerung der **Produktivität**
- Verbesserung der **Qualität**
- Erleichterung der **Führbarkeit** von Projekten

Mittel des Software Engineerings und ihre Wirkung



In dieser Vorlesung: Die Mittel kennen lernen