

Vorlesung "Software-Engineering"

Prof. Ralf Möller, TUHH, Arbeitsbereich STS
Übung: Miguel Garcia

- Voraussetzungen:
 - Algorithmen und Datenstrukturen
 - Objektorientierte Programmierung

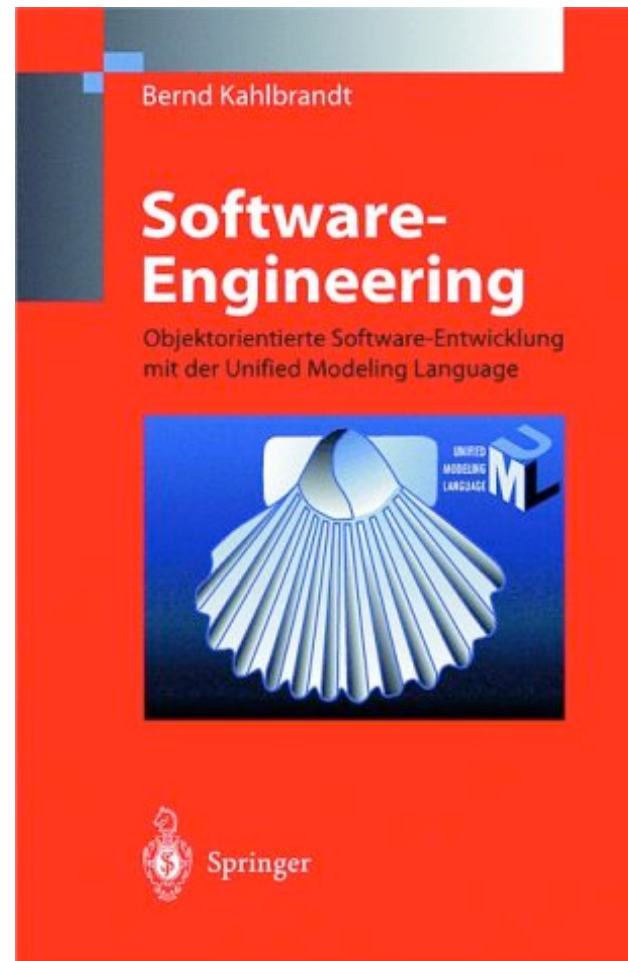
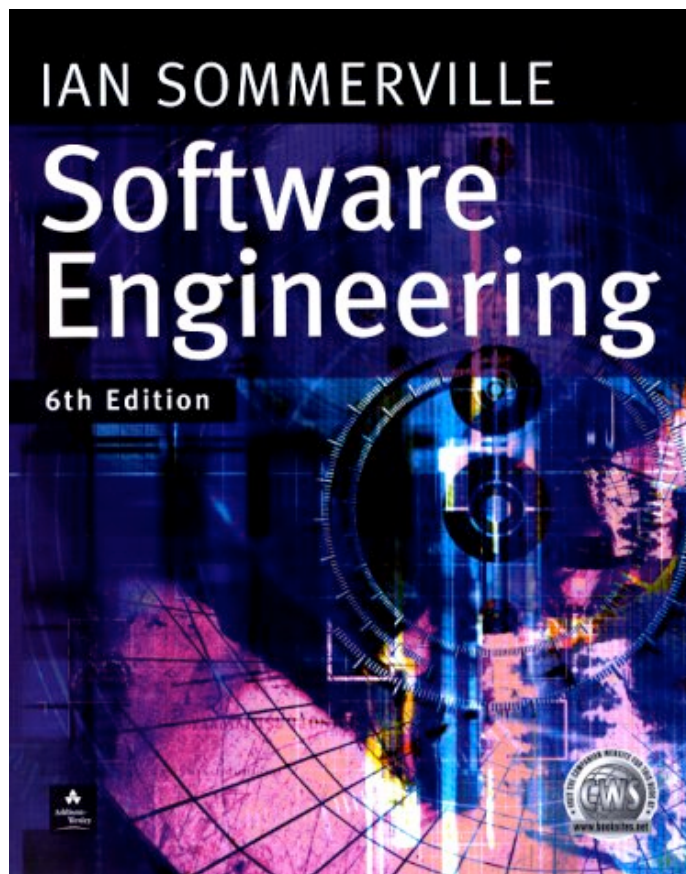
Organisatorisches

- Sprechstunde: n.V.
- Klausur
 - Wann? Am Ende des Semesters
 - Wie lernt man für dieses Fach?
 - Wiedergeben
 - Anwenden
 - Übertragen
- Web-Seite:
 - <http://www.sts.tu-harburg.de/~r.f.moeller/lectures/se-ss-05.html>

Literatur, Details und Zusatzinformationen



Literatur (2)



Weitere Literatur wird noch bekanntgegeben...

Vorlesung "Software-Engineering"

- Lernziele der Vorlesung allgemein:
 - Fundamente: Überblicke, Grundprinzipien über die systematische, ingenieurmäßige Softwareentwicklung
 - Grundlagen der Informatik: Techniken und Methoden der SW-Entwicklung
- Lernziele heute:
 - Einführung in das Gebiet "Software-Engineering"
 - Grundlagen und Probleme
 - Software-Qualität

Anfänge der Software-Entwicklung

- Geringe Rechnerleistung
- Überschaubare Problemstellungen
- Bekannte Algorithmen (meist mathematisch/naturwissenschaftlich)
- Software-Entwicklung = Programmierung
- Wenige Benutzerinteraktion ("Batch-Verarbeitung")
- Wenige Programmierfehler
- Forderung nach Effizienz
- Programmierer = Benutzer = Spezialisten
- Seltener, isolierter Einsatz



Software-Engineering
nicht notwendig

Veränderung der Software-Entwicklung

- Stark wachsende Leistungsfähigkeit der Hardware
- Immer komplexere Aufgabenstellungen
- Erarbeitung neuer, nicht-numerischer Algorithmen
- Dialogbetrieb, Interaktivität der Programme
- Statt einzelner Programme große, verflochtene Programmsyst.
- Verteilte Anwendungen, Client-Server-Architekturen,
- Multi-Tier-Architekturen
- Zunehmende bewährte Altsysteme
- Zunehmende Abhängigkeit von DV-Systemen, sicherheitskritische Anwendungen

Veränderung der
Software-Entwicklung

Veränderung der Software-Entwicklung

- Nachträgliche Veränderung der Anforderungen und des Einsatzumfeldes
- Arbeitsteilige Systementwicklung
- Engpass Software-Entwickler
- Entwickler und Anwender (evtl. auch Auftraggeber) getrennte Personengruppen
- "DV-Laien" als Anwender
- Systementwicklung als kommerzielle Auftragsarbeit, Produktentwicklung
- Enorme wirtschaftliche Bedeutung



SW-Entwicklung wird zum wichtigen Problem!

Problembereiche der Software-Entwicklung

- Beherrschung der Komplexität der Aufgabenstellung
- Vollständige Erfassung und korrekte Spezifikation der Anforderungen
- Zerlegung des Systems in Teilsysteme und Spezifikation der Schnittstellen zwischen diesen
- Korrektheit und Zuverlässigkeit
 - Fehlerhäufigkeit und Aufwand der Fehlerlokalisierung und -beseitigung

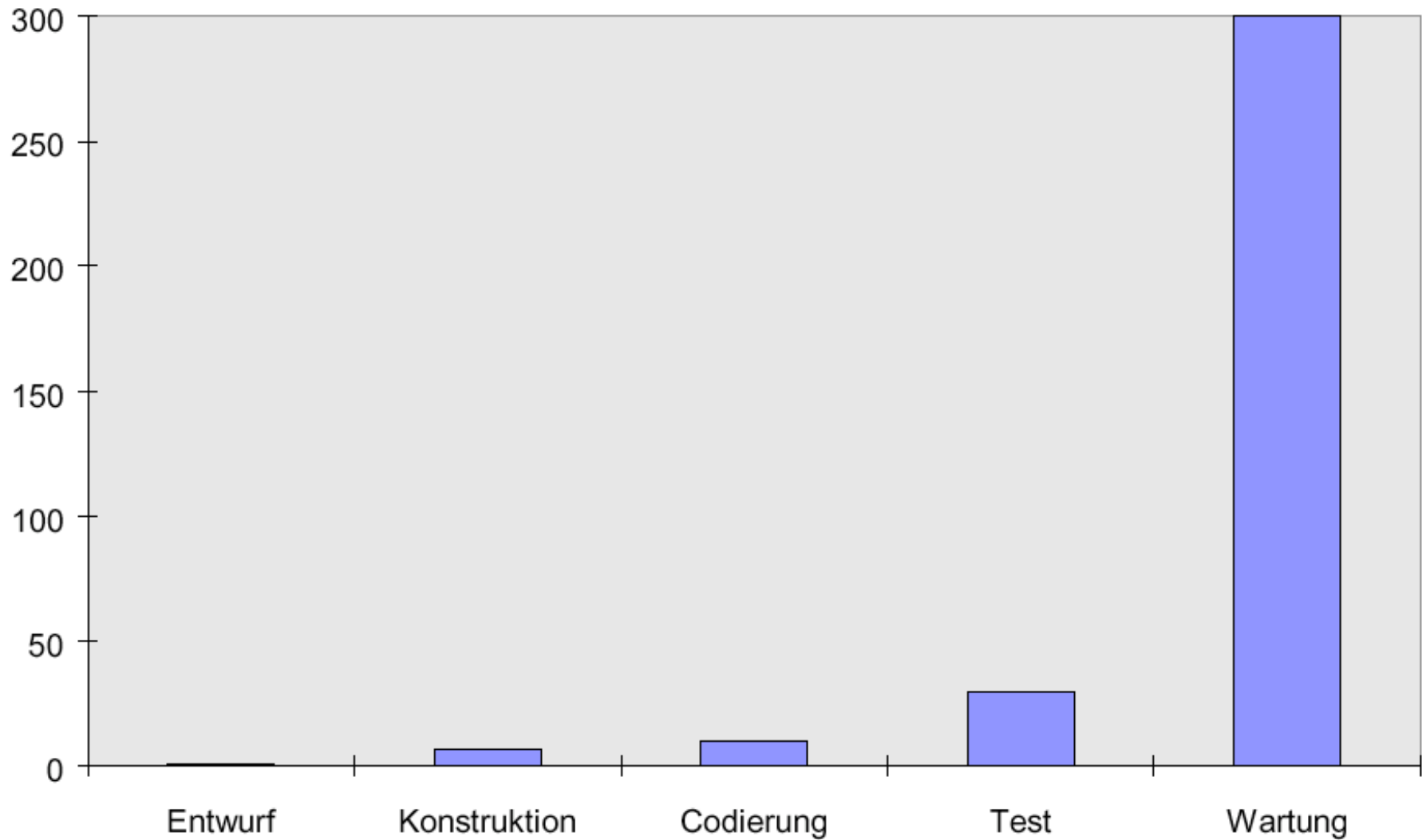
Problembereiche der Software-Entwicklung

- Effizienz der Programme
- Dokumentation und Wartbarkeit der Gesamtlösung
- Änderbarkeit und Erweiterbarkeit
- Übertragbarkeit auf verschiedene HW-Plattformen
- Planung und Durchführung von Projekten
- Kosten und Zeitbedarf der Software-Entwicklung
- Kommunikation zwischen den beteiligten Personen(-gruppen)

Mangelnde Zuverlässigkeit, Fehlerhäufigkeit

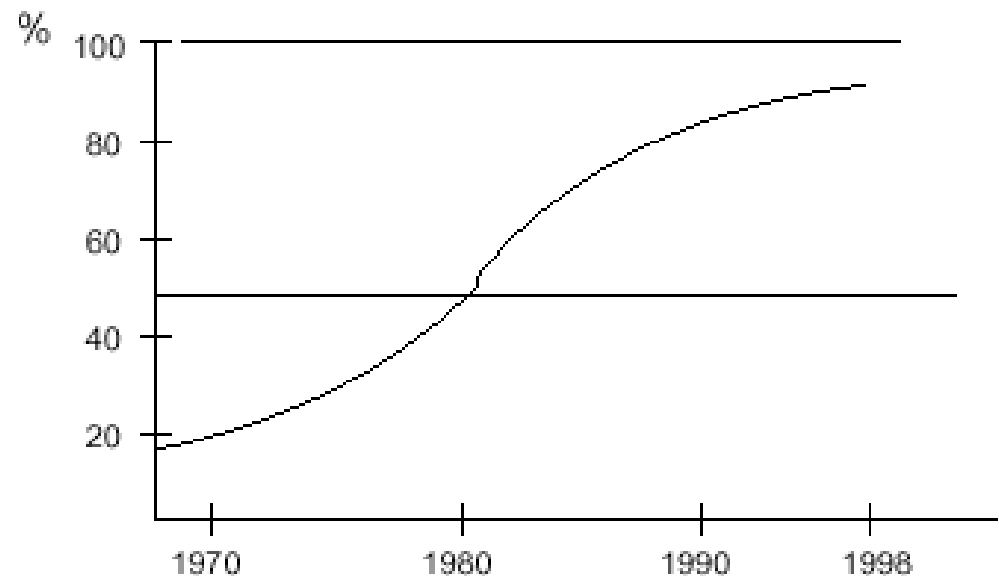
- Ein großes Software-Paket besteht aus mindestens 50.000 Zeilen Programmcode
- In 1000 Zeilen Programmcode werden während der Entwicklung durchschnittlich zwischen 50 und 60 Fehler entdeckt
- Die meisten Fehler entstehen bei Problemanalyse und Entwurf der Software
- Nach Auslieferung werden noch bis zu 4 Fehler pro 1000 Zeilen entdeckt
- Relativer Aufwand der Fehlerbeseitigung je nach Phase der Fehlerentdeckung hoch

Zeitaufwand je nach Entwicklungsphase



Kosten

- Dramatisch zunehmender Kostenanteil der Software-Entwicklung an den Gesamtkosten von DV-Projekten
- Geringe Produktivität: In großen Projekten pro Person im Durchschnitt weniger als 10 Zeilen ausführbares Programm am Tag
- Derzeit hoher Anteil der Wartungskosten an den Gesamtkosten (ca. 2 Drittel)
- Ziel: Minimale Gesamtkosten



Zeitbedarf



- Deutlich steigende Entwicklungsdauer für Software
- Nur 5% aller Projekte werden termingerecht fertig
- Mehr als 60% der Projekte sind $\geq 20\%$ in Verzug
- Zykluszeiten für Anwendungssoftware sind wesentlich länger als für Systemsoftware und Hardware

Entstehung des Fachgebietes "SW-Engineering"

- Der Begriff "Software-Engineering" wird Ende der sechziger Jahre geprägt, zunächst als Provokation
- Übertragen des erfolgreichen ingenieurmäßigen Vorgehens auf die Software-Entwicklung
- Weg von der "Kunst" des Programmierens hin zur Ingenieurwissenschaft des Programmierens
- Mittlerweile als Begriff und Fachgebiet etabliert
- Gehört zu den besonders nachgefragten Kompetenzen von Informatikern
- Forschung und Entwicklung nicht abgeschlossen, dynamisches und heterogenes Gebiet

Was ist Software-Engineering?



- Definition nach Pomberger/Blaschek
- "Software-Engineering ist die praktische Anwendung wissenschaftlicher Erkenntnisse für die wirtschaftliche Herstellung und den wirtschaftlichen Einsatz qualitativ hochwertiger Software"

Was ist Software-Engineering?



- Definition nach Balzert
- "Software-Technik: Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Software-Systemen."

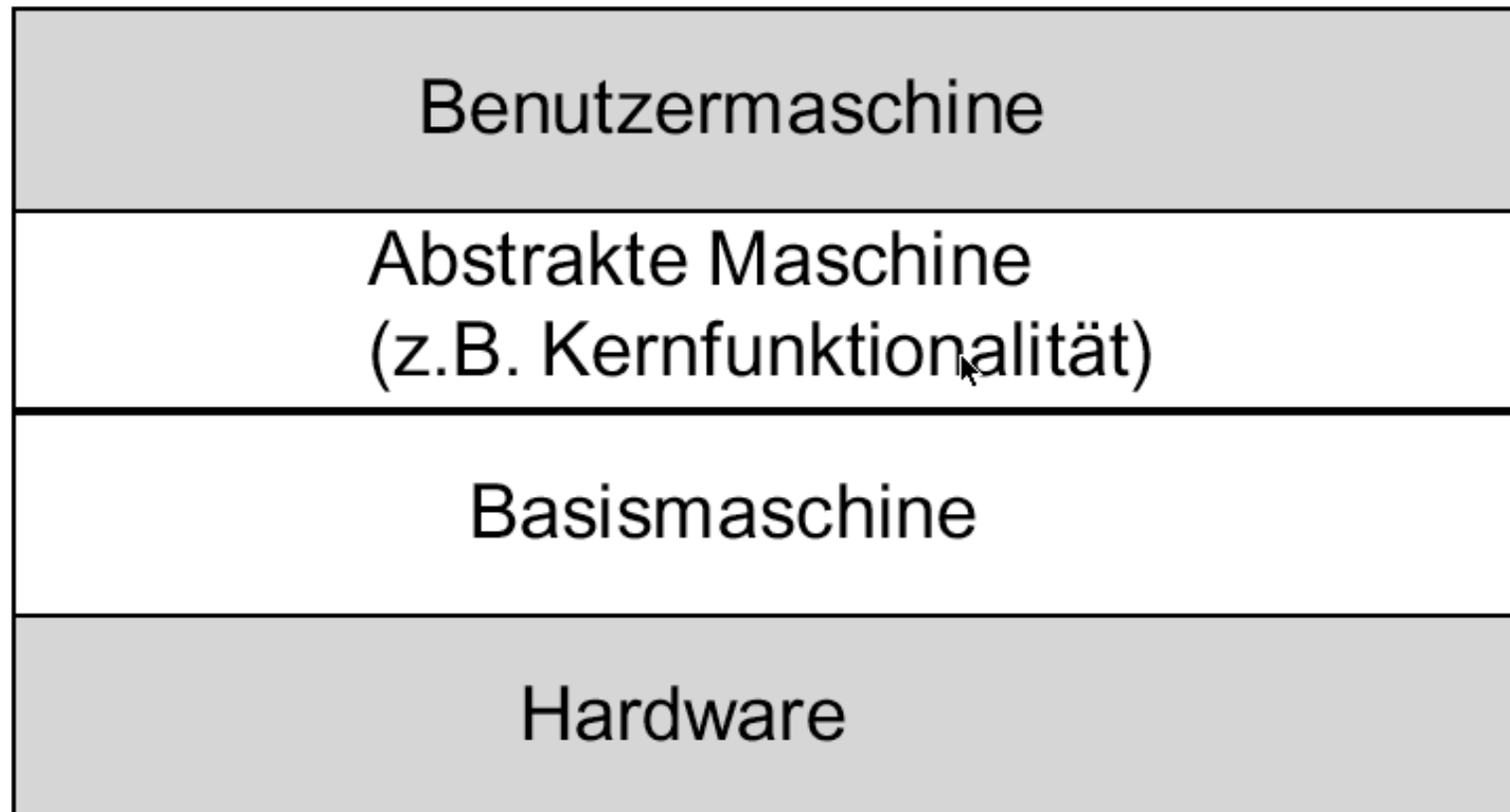
Was ist Software-Engineering?

- Das Fachgebiet Software-Engineering bietet dem Software-Entwickler einen "methodischen Werkzeugkasten"
- Software-Engineering bietet keine direkt anwendbaren "Kochrezepte" und "Bedienungsanleitungen" zur SW-Entwicklung
- Die Auswahl geeigneter Mittel des Software-Engineering muß weitgehend auf Wissen und Erfahrung der Systementwickler beruhen

Software...

- ist eine Sammelbezeichnung für Programme, die für den Betrieb von Rechensystemen zur Verfügung stehen (einschließlich der zugehörigen Dokumentation)
- ist im Zusammenspiel mit bestimmter Hardware ausführbar und ermöglicht deren Nutzung
- spezialisiert die zugrundeliegende universell programmierbare Hardware
- macht aus der konkreten Maschine eine neue Maschine (virtuelle Maschine)
- wird in Schichten unterteilt (Konzept der abstrakten Maschine)

Softwareschichten als abstrakte Maschinen



Ziel der Vorlesung ist ...

- ... Techniken zu erarbeiten, die es erlauben zu spezifizieren, was eine (virtuelle) Maschine tun soll.
- ... Schätzmethode für den Entwicklungsaufwand anwenden zu können
- ... Qualitätsmerkmale aufzeigen zu können
- ... Organisationsformen für die Entwicklung von SW zu verstehen

Besonderheiten von Software

- Software ist immateriell
- Software unterliegt keinem Verschleiß
- Software altert
- Software ist "weich", daher schnell änderbar
- Die Herstellung von Software beruht weniger auf allgemein akzeptierten Prinzipien
- Software ist (häufig) komplex
- Software ist schwer zu vermessen

Abgrenzung: Software - Programm

■ Programm

- Ausführbare Formulierung eines bestimmten Algorithmus (kleine Lösung)

■ Software(-System)

- Gesamtheit aller Software-Bausteine (Moduln), die in einem Zusammenhang stehen (gemeinsamer Zweck)
- Weist eine "Software-Architektur" auf

Software-Qualität



- "Qualität ist die Gesamtheit von Eigenschaften und Merkmalen eines Produkts oder einer Tätigkeit, die sich auf die Eignung zur Erfüllung gegebener Erfordernisse beziehen" (nach DIN 55 350)

- Software-Qualität...

- ist mehr als Korrektheit
- ist kein exakt definierter Begriff
- ist nicht exakt meßbar
- wird anhand von Qualitätsmerkmalen charakterisiert
- hängt von der Perspektive ab

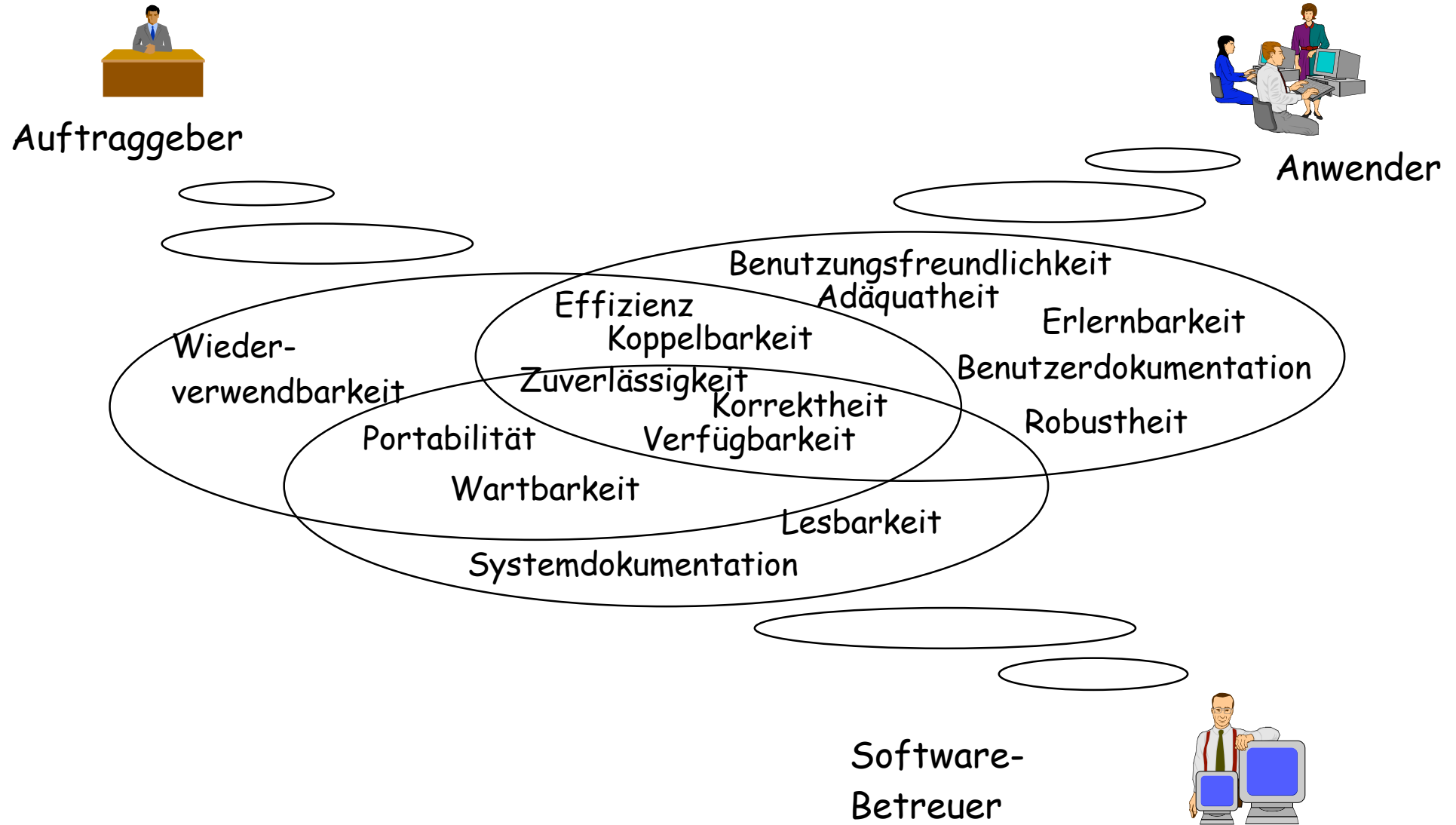


Software-Qualität: Merkmale

Benutzungsfreundlichkeit
Adäquatheit
Erlernbarkeit
Effizienz
Koppelbarkeit
Benutzerdokumentation
Wieder-
verwendbarkeit
Zuverlässigkeit
Korrektheit
Robustheit
Portabilität
Verfügbarkeit
Wartbarkeit
Lesbarkeit
Systemdokumentation

Bedeutung der Begriffe unklar, nicht eindeutig abgrenzbar

Software-Qualität: Perspektiven



Qualitätsmerkmale für die Anwendung (1)

■ Korrektheit

- Übereinstimmung zwischen funktioneller Spezifikation und Programmfunktionalität
- Korrektheit in der Praxis schwer nachweisbar
- Korrektheitsbeweise mit Programmverifikation nur für kleine Teilalgorithmen möglich
- Vollständige Tests aller Programmzustände zu aufwendig
- Korrektheit ist ein wichtiger aber vielfach theoretischer Anspruch
- Korrektheit in umfangreichen Programmsystemen besonders problematisch

Qualitätsmerkmale für die Anwendung (2)

■ Effizienz

- Bestimmt den Bedarf an Betriebsmitteln
- Effizientes Programm: kein unnötiger Verbrauch an Betriebsmitteln
- Unterscheidung von Speichereffizienz und Laufzeiteffizienz
- Konflikte zu Änderbarkeit, Testbarkeit, Portierbarkeit

Qualitätsmerkmale für die Anwendung (3)

■ Robustheit

- Definierte und sinnvolle Reaktion des Programms bei beliebiger externer Kommunikation
- Verhindern von undefinierten Systemzuständen und "Systemabstürzen"
- Besonders wichtig: Abfangen fehlerhafter Benutzereingaben
- Beseitigung der Fehlersymptome, nicht der Ursachen
- Spektrum von sinnvollen Reaktionsmöglichkeiten, abhängig von der Situation

Qualitätsmerkmale für die Anwendung (4)

■ Verfügbarkeit

- Wahrscheinlichkeit, daß ein System zu einem gegebenen Zeitpunkt funktionsfähig ist
- Kennwert in der Praxis:

$$V = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

MTBF: Mean Time Between Failures
MTTR: Mean Time To Repair

Qualitätsmerkmale für die Anwendung (5)

■ Zuverlässigkeit

- Zusammenspiel von Korrektheit, Robustheit und Verfügbarkeit
- Auftreten von Fehlern im Zeitablauf
- Berücksichtigung von Reparaturzeiten und Fehlerqualitäten
- Wahrscheinlichkeit, daß ein System seine Funktion während eines Zeitintervalls korrekt erfüllt
- Festlegung in den Spezifikationen

Qualitätsmerkmale für die Anwendung (6)

- Benutzungsfreundlichkeit
- Spezielles Forschungsgebiet: Software-Ergonomie
- Speziellere Merkmale der Benutzungsfreundlichkeit (nach DIN 66234 Teil 8 und DIN EN ISO 9241):
 - Aufgabenangemessenheit
 - Selbstbeschreibungsfähigkeit
 - Steuerbarkeit
 - Erwartungskonformität
 - Fehlerrobustheit

Qualitätsmerkmale für die Anwendung (7)

■ Datensicherheit / Datenschutz

- Schutz gegen unerwünschte bzw. unerlaubte Verfälschung/Zerstörung bzw. Preisgabe von Daten
- Problem durch Dezentralisierung/Vernetzung der DV verschärft
- Behandlung von Ausnahmesituationen (z.B. Stromausfall, Systemabsturz)
- Restartfähigkeit (Möglichkeit zum Wiederaufsetzen)
- Kombination von software-technischen und organisatorischen Maßnahmen

Qualitätsmerkmale für Entwicklung u. Wartung (1)

■ Verständlichkeit/Lesbarkeit

- Maß für den Aufwand, ein (fremdes) Software-Produkt zu verstehen
- Vielfältige Maßnahmen zur Erhöhung der Verständlichkeit möglich
- Voraussetzung für Änderbarkeit und Reparierbarkeit

Qualitätsmerkmale für Entwicklung u. Wartung (2)

■ Änderbarkeit

- Möglichkeiten zur Anpassung von (korrekter) Software an veränderte Einsatzbedingungen und Anforderungen
- Begrenzung des Aufwandes bei Änderungen
- Berücksichtigung bereits bei Software-Entwicklung
- Weitgehend abhängig von einer geeigneten Modularisierung der Software

Qualitätsmerkmale für Entwicklung u. Wartung (3)

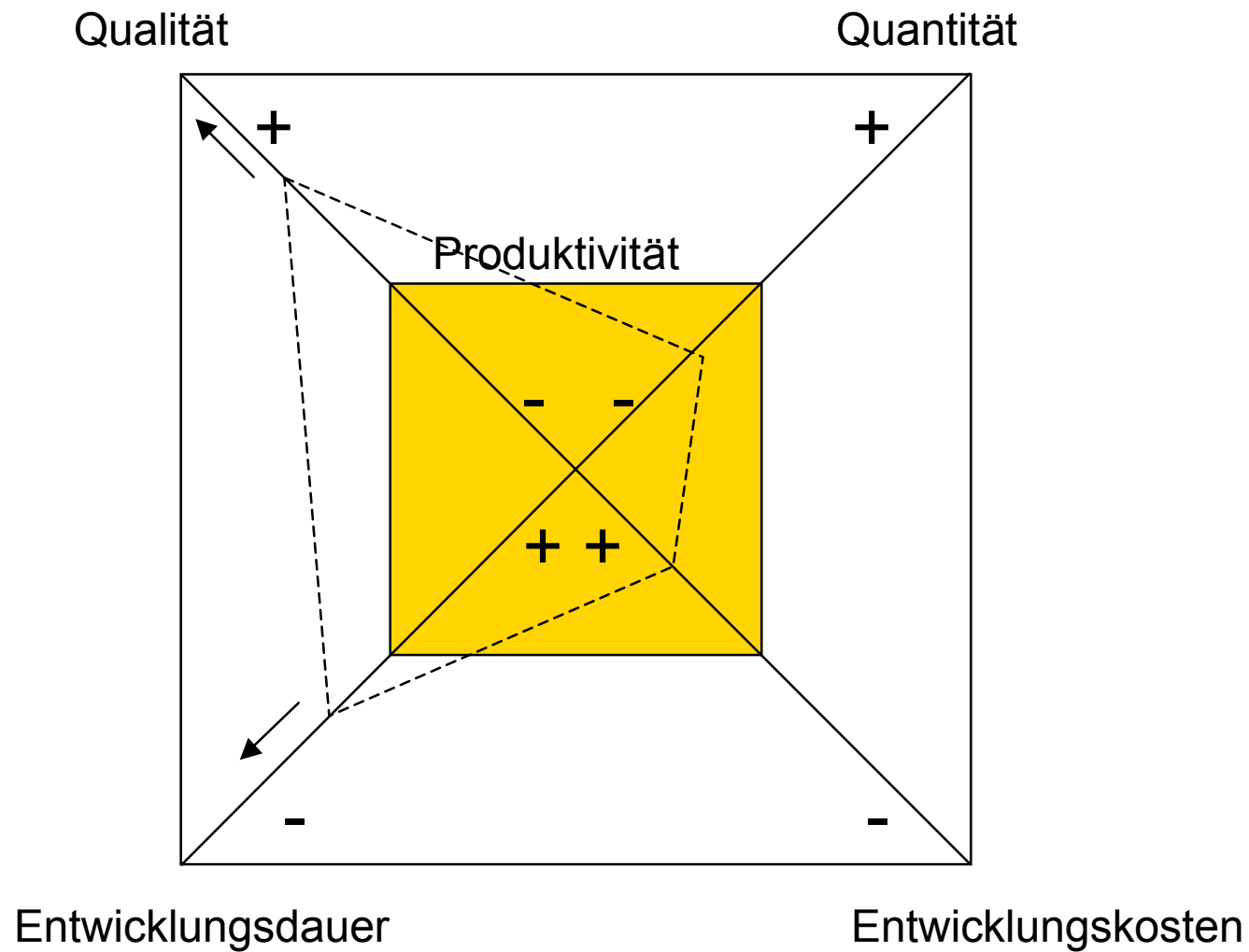
■ Prüfbarkeit/Testbarkeit

- Möglichkeiten zum Testen eines Programms hinsichtlich Korrektheit, Robustheit und Zuverlässigkeit
- Wesentlich abhängig von Modularität und Strukturierung
- Parallelentwicklung von Testumgebungen

Qualitätsmerkmale für Entwicklung u. Wartung (4)

- Wiederverwendbarkeit/Portabilität
 - Aspekt der Allgemeinheit der Software
 - Verlängerung der Lebensdauer von Software
 - Aufbau von Software-Bibliotheken und Nutzung objektorientierter Ansätze
 - Notwendigkeit zu ausführlicher Dokumentation
 - Ziel: Senkung von Entwicklungskosten

Qualität kann nicht isoliert betrachtet werden



Überblick über die Vorlesung



- Einführung
 - Begriffsbestimmung, Qualitätskriterien
- Phasen und Vorgehensmodelle
 - Projekttypen, Personen, Prozesse, Produkte und Leistungen
- Problemanalyse und Planung: Lastenheft
- Aufwandsabschätzung
- Spezifikation, Definition: Pflichtenheft
- Entwurf (Design)
 - Strukturen, Zustände, Prozesse (e.g., UML)
 - Model-driven Architecture
- Verifikation, Testen, Validierung
- Versionsverwaltung, Konfiguration
- Projektmanagement
- Qualitätssicherung
- Installation, Weiterentwicklung