Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Bachelor of Science (BSc) in Informatik

Modul Advanced Software Engineering 1 (ASE1)

LE 01 – Einführung und Überblick

Institut für Angewandte Informationstechnologie (InIT)

Walter Eich (eicw) / Matthias Bachmann (bacn)

http://www.zhaw.ch/de/engineering/institute-zentren/init/

Agenda

1. Motivation und Leitidee
2. Ziele, Inhalte und Ablauf des Moduls
3. Einführung ins Software Engineering
4. Wrap-up und Ausblick

Lernziele LE 01 – Überblick und Einführung

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering

- Sie sind in der Lage,
  - die Leitidee, Lernziele und den Ablauf des Moduls zu nennen.
  - zu erläutern, was für Zertifikate erlangt und was für Leistungsnachweise verlangt werden und wie sie sich darauf vorbereiten können.
  - die Disziplin Software Engineering zu definieren und abzugrenzen zu anderen Engineering Disziplinen.
  - die wichtigsten Themengebiete bzw. Subdisziplinen des Software Engineering zu charakterisieren und zu erläutern.
  - die begleitende Fallstudie und deren Domäne zu diskutieren.

Motivation und Leitidee für ASE1/2 (1/2)

- Gefestigtes und breites Wissen in den Kerndisziplinen des Software Engineerings ist essentiell für einen berufsbefähigenden Bachelor-Abschluss in Informatik.
- Die Pflichtmodule SWEN1/2 vermitteln die notwendigen theoretischen und praktischen Basis-Kenntnisse im Software Engineering.
- Um in verschiedenen Anwendungsbereichen, sowie in komplexen Aufgabebereichen als Softwareentwickler tätig zu sein, ist für die Praxis ein ergänztes, umfassendes und gefestigtes Software-Engineering-Wissen notwendig.
- Die Wahlmodule ASE1/2 reflektieren, vertiefen und festigen das bisherige Wissen im Software Engineering.
- Die behandelten Themen im Software Engineering werden mit einer begleitenden Fallstudien verzahnt mit der Theorie vermittelt, um verschiedene Methoden, Techniken und Lösungsansätze zu elaborieren (Software Engineering Lab).

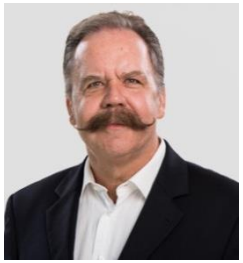
Motivation und Leitidee ASE1/2 (2/2)

- Das Erlernte Software Engineering Wissen kann durch mehrere internationale Zertifizierungen (Requirements Engineering IREB, Software Architektur iSAQB, Testing ISTQB) für den Arbeitsmarkt ausgewiesen werden.
- Die Theorie ist abgestimmt auf die Zertifizierungen und als Vorbereitung werden Übungen und Musterprüfungen gelöst.
- Die Absolventen erhalten damit eine solide und breite Grundausbildung im Software Engineering, sodass sie auch anspruchsvolle Situationen einwandfrei meistern und sich mit dieser Befähigung auf dem Arbeitsmarkt gut positionieren können.
- ASE1/2 decken die wesentlichen (technischen) Kerndisziplinen im Software Engineering ab und sind gemäss den im Projektverlauf (entlang der Zeitachse) auftretenden Fragen und Problemen gegliedert.

Behandelte Themengebiete ASE1/2

- ASE1 (HS):
  - Einführung ins Software Engineering: Grundbegriffe, Prozesse und Software-Qualität
  - Requirements Engineering (gemäss Curriculum IREB)
  - Softwarearchitektur (gemäss Curriculum iSAQB)
- ASE2 (FS):
  - Softwarekonstruktion
  - Testing (gemäss Curriculum ISTQB)
  - Software Maintenance und Operations (Einführung)
- Durchgängige Fallstudie
  - Vertiefung und Verankerung der Theorie
  - Ausgewählte Technikthemen zur Umsetzung (Spring Boot, Angular, …)

Die Dozenten stellen sich vor



Walter Eich

- Seit über 35 Jahren in der Software-Entwicklung unterwegs
- Seit 2013 Dozent und Studienleiter im MAS Informatik an der ZHAW School of Engineering
- Vorher 16 Jahre bei Zühlke als Systemarchitekt, Software Engineering Berater und Trainer
- Tätig am Institut für angewandte Informationstechnologie (InIT) im Forschungsschwerpunkt Software Systems (SWS)

Matthias Bachmann

- Seit über 35 Jahren in der Software-Entwicklung unterwegs
- Seit 2003 Dozent an der HSZ-T und an der ZHAW School of Engineering
- Seit 2005 Mitglied der Studiengangsleitung für den auslaufenden Studiengang
- Berater und Trainer für Softwarearchitektur und Java-Spring

Kontakt

Kontaktieren Sie mich oder Matthias Bachmann bei Interesse zur Entwicklung verteilter Software-Systeme, agiler Softwareentwicklung oder generell zu Themen im Software Engineering für Projekt- und Bachelorarbeiten.

Koordinaten:

Walter Eich

Institut für Angewandte Informationstechnologie (InIT)

Steinberggasse 13 , TG 203

8401 Winterthur

E-Mail: walter.eich@zhaw.ch

T: +41 58 934 49 82

Matthias Bachmann

Lagerstrasse 41

Büro ZL O2.07

8004 Zürich

matthias.bachmann@zhaw.ch

+41 58 934 82 52

Agenda

1. Motivation und Leitidee
2. **Ziele, Inhalte und Ablauf des Moduls**
3. Einführung ins Software Engineering
4. Wrap-up und Ausblick

Lernziele ASE1

- Sie sind in der Lage,
  - die Charakteristiken der gängigen Software Engineering Prozesse (plangetrieben, agil) und deren Anwendungsgebiete (Home ground) zu erläutern,
  - Kriterien für mehr oder weniger Zeremonie in einem Software-Prozess (Tailoring) zu nennen,
  - situationsgerecht Techniken zur Ermittlung, Kommunikation, Dokumentation und Konsolidierung von Anforderungen auszuwählen und anzuwenden.
  - die Bedeutung des Requirements Engineering für den Projekterfolg zu erklären,
  - gängige Techniken zum Entwurf, der Beschreibung und Kommunikation von Software-Architekturen anzuwenden,
  - Prinzipien guter Architektur und guten Designs in modernen, gängigen Frameworks erläutern,
  - Qualitätskriterien zu nennen und eine Software-Architektur zu bewerten.

Inhalte und Ablauf des Moduls ASE1

SW#	KW#	Theorie Block / Vorlesung	Praktikum / Übungen	Selbststudium / Bemerkungen
		Do, 10:00 - 11:35 Uhr, ZL O6.12	Do, 08:00 - 09:35 Uhr, ZL O6.12 Do, 12:00 - 13.35 Uhr, ZL O6.12	
01	38	Überblick und Einführung	Aufgabe 0	SWEBOK Guide v3
02	39	Software Engineering Prozesse	Aufgabe 1	
03	40	Requirements Engineering 1 (Kap. 1, 2)	Aufgaben 2, 3	Syllabus CPRE-FL Pflichtlektüre RE Kap. 1, 2
04	41	Requirements Engineering 2 (Kap. 3, 4)	Aufgaben 4, 5	Pflichtlektüre RE Kap. 3, 4
05	42	Requirements Engineering 3 (Kap. 5, 6)	Aufgaben 6, 7	Pflichtlektüre RE Kap. 5, 6
06	43	Requirements Engineering 4 (Forts. Kap. 6, 7)	Aufgaben 8, 9	Pflichtlektüre RE Kap. 7
07	44	Requirements Engineering 5 (Kap. 8, 9)	Aufgaben 10, 11	Pflichtlektüre RE Kap. 8, 9 Prüfungsordnung, Probeprüfung CPRE FL
08	45	Zertifikatsprüfung CPRE FL, **12.11.2020, 10:00 Uhr, ZL O6.12** (75')	keine Vorlesung und Praktikum	
09	46	Softwarearchitektur 1 (Kap. 1, 2)	Aufgaben 12, 13	Syllabus CPSA-FL Pflichtlektüre SA Kap. 1, 2
10	47	Softwarearchitektur 2 (Kap. 3)	Aufgaben 14, 15	Pflichtlektüre SA Kap. 3
11	48	Softwarearchitektur 3 (Kap. 4)	Aufgaben 16, 17	Pflichtlektüre SA Kap. 4
12	49	Softwarearchitektur 4 (Kap. 5)	Aufgaben 18, 19	Pflichtlektüre SA Kap. 5
13	50	Softwarearchitektur 5 (Kap. 6)		Pflichtlektüre SA Kap. 6 Prüfungsregeln, Beispiel-Fragen CPSA FL
14	51	Zertifikatsprüfung CPSA FL, **17.12.2020, 10:00 Uhr, ZL O6.12** (75')	keine Vorlesung und Praktikum	
	52-02	**Semesterunterbruch: Prüfungsvorbereitung**		
15-16	03-04	**Semesterendprüfung** Keine Semesterendprüfung!		Kein Unterricht

Didaktisches Konzept

Vorlesung

- Vermittlung und Vertiefung der Grundlagen
- Diskussionen, Beispiele zu einzelnen Aspekten

Praktikum

- Durchgängige Fallstudie mit Lernaufgaben und Musterlösungen
- Ziele mit der Fallstudie:
  - Requirements Engineering: Wesentliche Artefakte einer Software Requirements Specification (SRS) erarbeiten
  - Software-Architektur: Wesentliche Artefakte einer Software Architecture Description (SAD) erarbeiten
  - Technologiegrundlagen für erfolgreiche Umsetzung mit gängigen Frameworks und Tools

Selbststudium

- Pflichtlektüre, Fachartikel, Review bzw. Aufgaben zur Vorlesung

Internationale Zertifizierungen

- Folgende Zertifizierungen können in ASE1 absolviert werden:
  - Certified Professional for Requirements Engineering (CPRE) Foundation Level, Webseite: https://www.ireb.org/
  - Certified Professional for Software Architecture (CPSA) Foundation Level, Webseite: http://www.isaqb.org/
- Multiple-Choice-Prüfungen (ca. 45 Fragen, 75 Minuten Zeit)
- Externe Organisation nimmt die Prüfungen ab (iSQI Organisation DACH, https://www.isqi.org/)
- Daten, Kosten:
  - CPRE FL, **12.11.2020, 10:00 Uhr**, EUR 210.- (inkl. MwSt.)
  - CPSA FL, **17.12.2020, 10:00 Uhr**, EUR 150.- (inkl. MwSt.)

Vor Ort auf Papier

Leistungsnachweise

- 2 Schriftliche Prüfungen (CPRE und CPSA Foundation Level Zertifizierungsprüfungen in %)
- Praktikumsaufgaben (in %)
- Die Schlussnote wird berechnet nach:

$$Note = \left(\frac{2 \times CPRE + 2 \times CPSA + Praktika}{5 * 100}\right) * 5 + 1$$

Anmerkungen

- Jede Zertifizierungsprüfung ergibt eine Prozentzahl von 0..100%. Die Details dazu sind in der jeweiligen Prüfungsordnung geregelt.
- Jede bewertete Praktikumsaufgabe ergibt eine Punktzahl (0..5). Die Summe der maximalen Punkte pro Praktikumsaufgabe ergibt 100%.
- Schriftliche Prüfungen werden mit 80% und Praktika mit 20% gewichtet.

Literatur

- Pflichtlektüre
  - K. Pohl, C. Rupp: Basiswissen Requirements Engineering, 4. Auflage, dpunkt.verlag, 2015
  - M. Gharbi et al.: Basiswissen für Softwarearchitekten, 3. Auflage, punkt.verlag, 2017
- Weitergehende Literatur
  - C. Rupp et al.: Requirements-Engineering und – Management: Aus der Praxis von klassisch bis agil, 6. Auflage, Hanser-Verlag, 2014
  - G. Starke, Effektive Softwarearchitekturen: Ein praktischer Leitfaden, 8. Auflage, Carl Hanser Verlag, 2017
  - Software Architektur: Martin Fowler, Enterprise Patterns; Eric Evens Domain Driven Design (DDD)
- Weitere Literaturempfehlungen finden Sie auch auf:
  - Certified Professional for Requirements Engineering (CPRE) Foundation Level, Webseite: https://www.ireb.org/
  - Certified Professional for Software Architecture (CPSA) Foundation Level, Webseite: http://www.isaqb.org/

Unterlagen und Abgaben auf OLAT

- Alle Unterlagen zum Modul sind auf OLAT abgelegt.
- Der aktuelle und verbindliche Kursablauf ist auf OLAT ersichtlich.
- Unterlagen wie Handouts und andere im Kontaktunterricht erwähnten Dokumente sind in der jeweiligen Woche im Verzeichnis «Unterlagen» abgelegt.
- Die Praktika werden ebenfalls elektronisch aus-, abgeben und bewertet im Verzeichnis «Praktikum».
- Der verbindliche Abgabetermin für Praktika ist jeweils der Sonntag, 23:59 Uhr, in der nächsten Woche nach der Ausgabe.
- Das Abgabe-Format für Dokumente ist PDF!
- Bei den meisten Praktika ist eine Musterlösung verfügbar.

Agenda

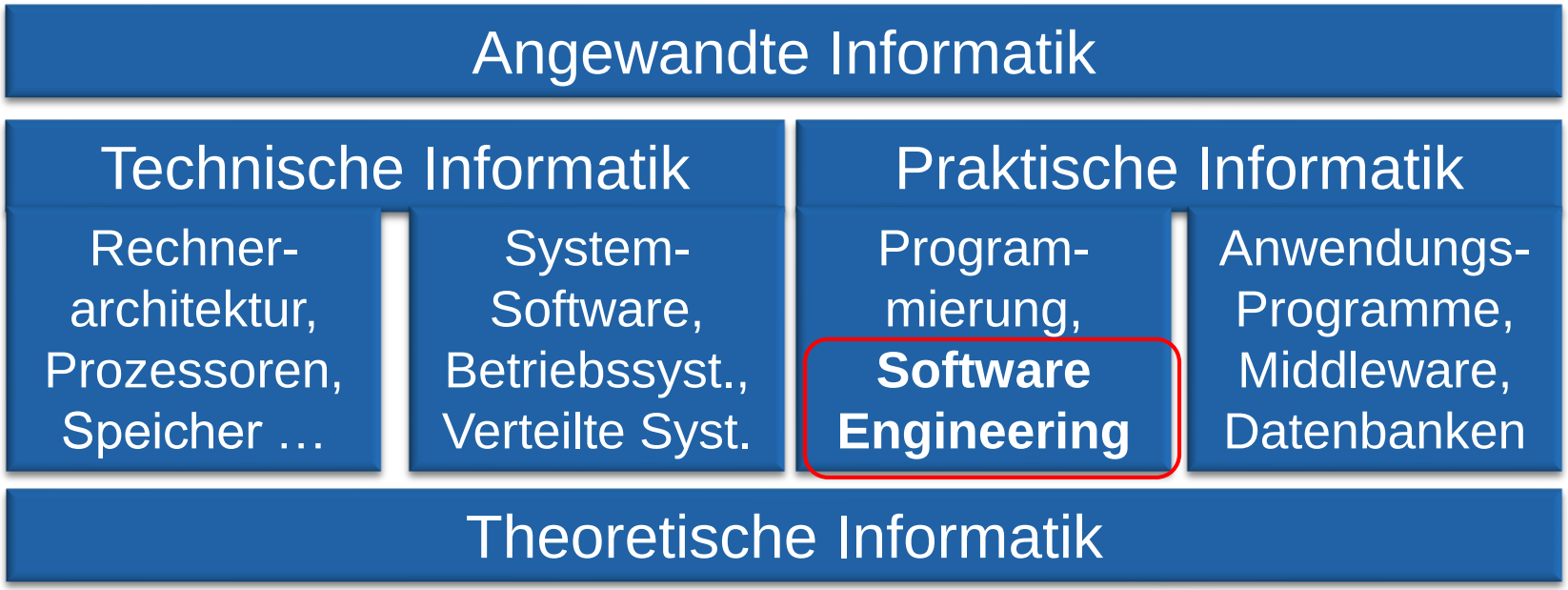
1. Motivation und Leitidee
2. Ziele, Inhalte und Ablauf des Moduls
3. **Einführung ins Software Engineering**
4. Wrap-up und Ausblick

Einführung ins Software Engineering
Agenda

- Was ist Software Engineering
  - Geschichte Software Engineering
  - Software Engineering vs. Programmieren
- Software Entwicklung als Prozess
  - Phasen bzw. wichtigste Phase
- Qualitätsaspekte
  - Cost to Fix
  - Spektakuläre Fehler
  - Qualitätsmodell
- Teilgebiete des Software Engineerings
  - IEEE SWEBOK
- Was sind die Herausforderungen in Software Engineering?

Gebiete der Informatik – Software Engineering

- Die Informatik unterteilt sich in die Teilgebiete der Theoretischen Informatik, der Praktischen Informatik und der Technischen Informatik.
- Die Bezeichnung Software Engineering besteht aus den beiden Worten **SOFTWARE** und **ENGINEERING**



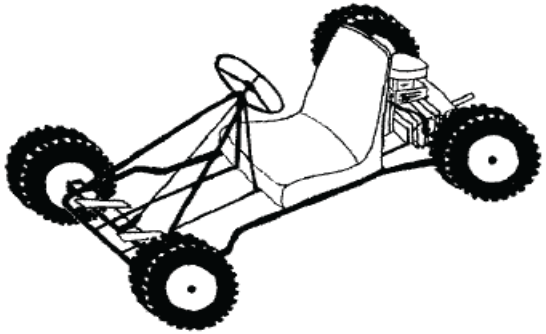
Was ist Software? (1/4)

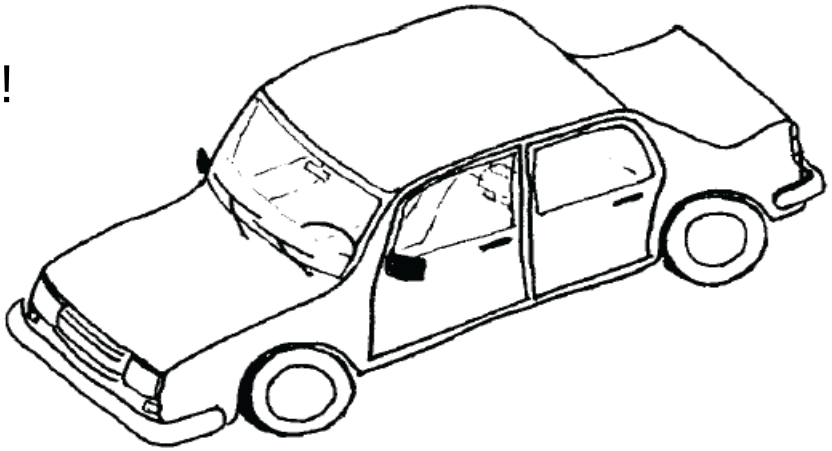
- Computerprogramme und dazugehörige Dokumentation
- Softwareprodukte sind:
  - Allgemein
    - entwickelt für den Verkauf an eine Gruppe verschiedener Anwender
  - Massgeschneidert (individuell)
    - entwickelt für einen einzelnen Kunden nach dessen Anforderungs-spezifikation
    - Das Softwareprodukt ist nur die Spitze des Eisbergs

Was ist Software? (2/4)

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

➪ *Mehr als nur Programme*!



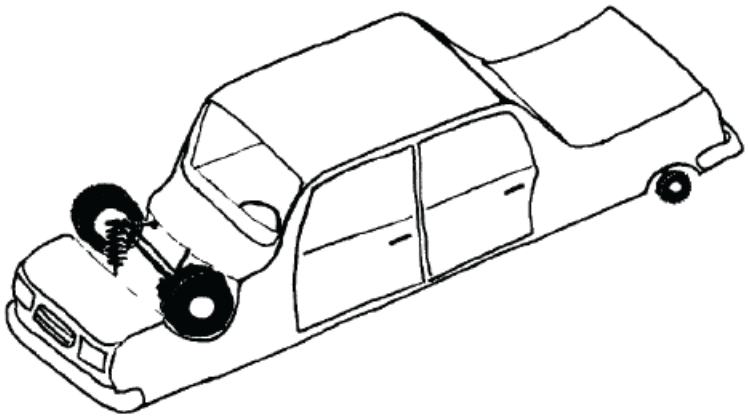


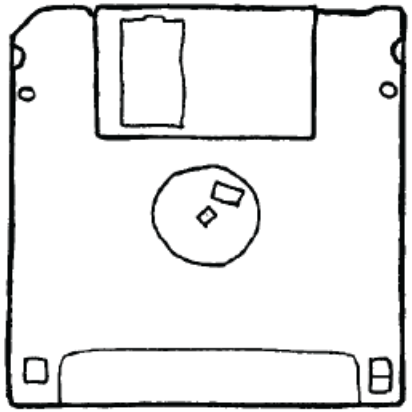
(Grafik: Glinz)

So wie ein Auto wesentlich mehr ist als nur ein fahrbarer Untersatz, so umfasst Software wesentlich mehr als nur Programme.

Was ist Software? (3/4)

- *Software kann man nicht anfassen.*





Diese mechanische Konstruktion ist offensichtlich falsch.

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

(Grafik: Glinz - UZH)

Was ist Software? (4/4)

- Fehler 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

Wozu dient Software?

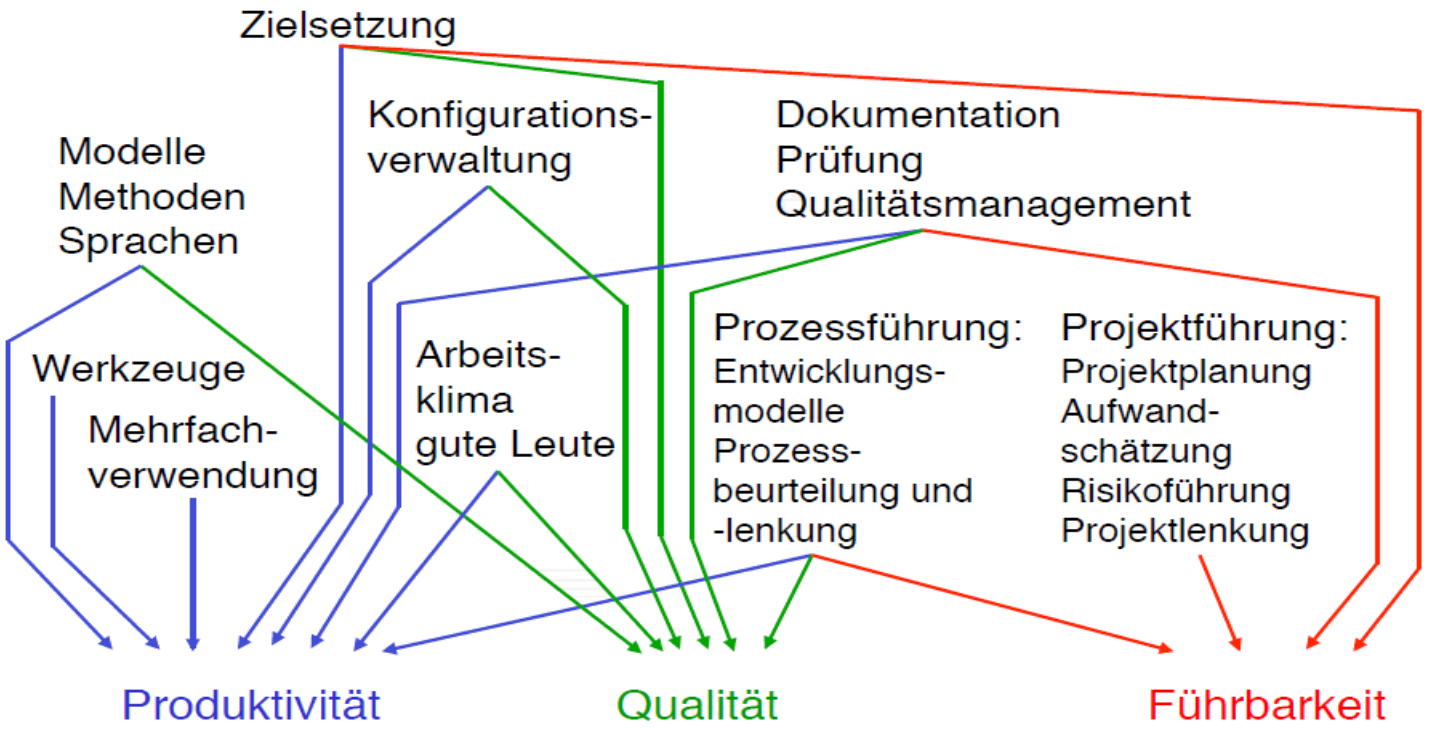
- Ein Problem zu lösen oder zu dessen Lösung beizutragen indem menschliche oder technische Arbeitsvorgänge automatisiert oder unterstützt werden.
- Wenn ein Problem von seiner Natur her komplex und schwierig zu lösen ist, so ist die Software zur Lösung dieses Problems in der Regel nicht weniger komplex und schwierig => Kenntnis der Fachdomäne.
  - Das Problem ist im Kontext seines Sachgebiets zu verstehen und befriedigend zu lösen.
  - Die Problemlösung muss auf adäquate Software-Strukturen abgebildet werden.
  - Sie *konstruiert und verändert* die Realität.
- Wir benötigen *Systeme*, nicht Programme
  - Software ist fast immer Bestandteil eines übergeordneten Systems
  - resultierende Software wird in der Regel in eine vorhandene Struktur von Software-Systemen eingebettet

Was ist Engineering?

- **Engineering** is the science, skill, and profession of acquiring and applying scientific, economic, social, and practical knowledge, in order to design and also build structures, machines, devices, systems, materials and processes. [Wikipedia]
- Wird als **Ingenieurswissenschaften** übersetzt
- **Engineers** apply mathematics and sciences such as physics to find suitable solutions to problems or to make improvements to the status quo. More than ever, **engineers** are now required to have knowledge of relevant sciences for their design projects. As a result, they may keep on learning new material throughout their career.

Ziele und Mittel des Software Engineering

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



(Grafik: Glinz)

Die 5 P im Software Engineering

- Projekte
  - Klein, gross, sehr gross
  - Forschungsprojekt, Wartungsprojekt, Projekt in der Krise
  - Budget, Zeit, Risiko
- Personen
  - Rollen (Business Analyse, Software Architekt, Entwickler, Tester)
    Ausbildung, Erfahrung => Kompetenzen (Fa, Me, So, Se)
- Prozesse
  - Vorgehensmodelle: Phasen, Aktivitäten, Vorlagen, Richtlinien, Rollen
  - Wasserfall, Unified Process, V-Modell, Hermes, Agile (Scrum – Kanban)
- Produkte und Leistungen
  - Artefakte: Zwischenprodukte, Endprodukte
  - Qualität: Messung, Metriken
- Paradigmen
  - Strukturierte Analyse, OO, SOA, Patterns, ...

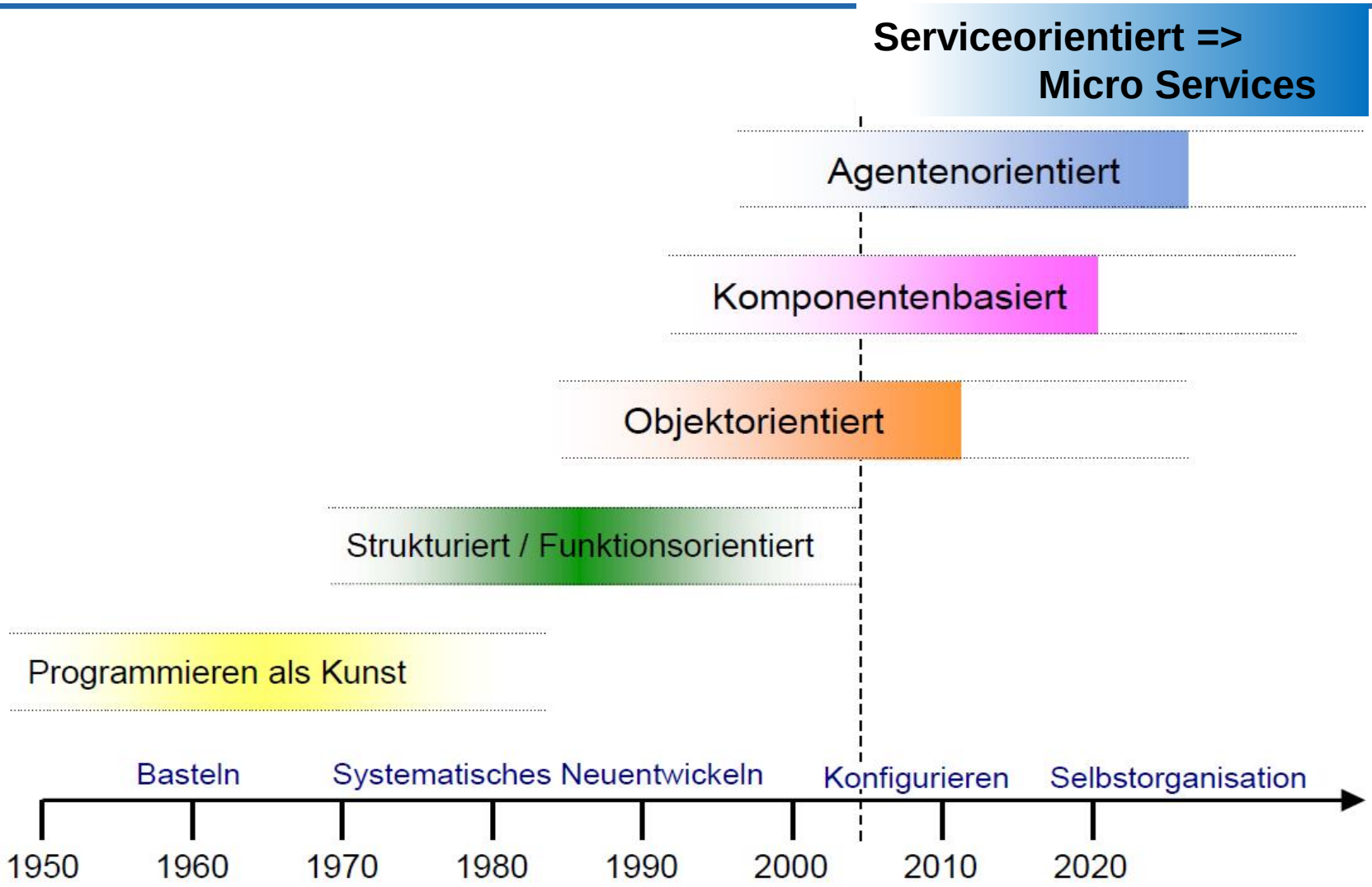
Geschichte Software Engineering (1/2)

- Der Begriff Software Engineering wurde Ende 1967 von einer Forschungsgruppe der NATO geformt.
- Auf den Software Engineering-Konferenzen der NATO 1968 in Garmisch und 1969 in Rom wurden erstmals in Anlehnung an andere Ingenieurdisziplinen Software-Programme als Industrieprodukt bezeichnet.
- Es wurde gefordert, Software Engineering nicht als Kunst zu sehen, sondern als ingenieurmässige Tätigkeit anzuerkennen.
- Ab Mitte der 60er und in den 70er Jahren kam der Begriff der „Software-Krise“ auf. Dieser bezog sich auf die sich nicht ändernde schlechte Qualität der erzeugten Software-Systeme. Als erste Reaktion wurden in der Praxis erprobte Vorgehensmodelle publiziert.

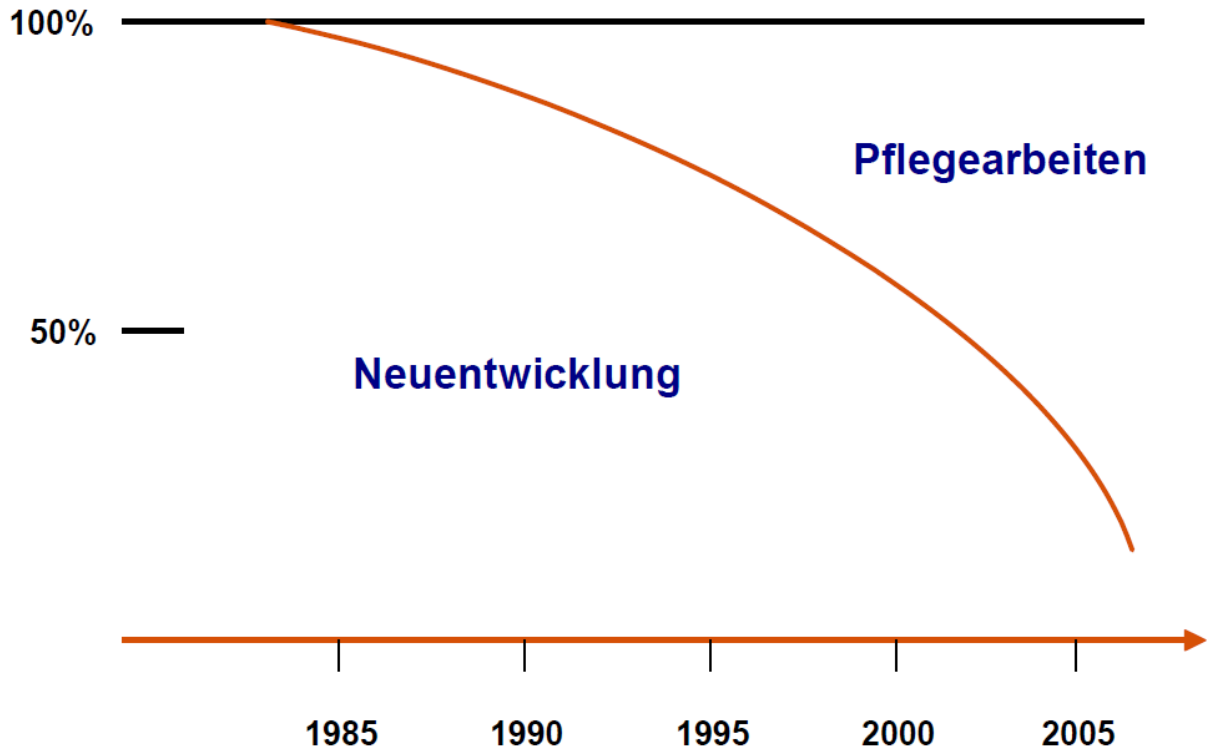
Geschichte Software Engineering (2/2)

- Ab den 80ern wurden einzelne Software Entwicklungsmethoden, welche sich in der Praxis bewährt hatten, formalisiert und einem breiteren Publikum zugänglich gemacht.
- Der Übergang von Wissenschaft und Wirtschaft zu einer professionellen Ingenieurdisziplin ist noch nicht vollzogen.
- Unter der Leitung der IEEE wird im SWEBOK – „Software Engineering Body of Knowledge“ versucht, eine inhaltliche Strukturierung vorzunehmen [www.swebok.org].

Historie der Softwareentwicklungsmethoden

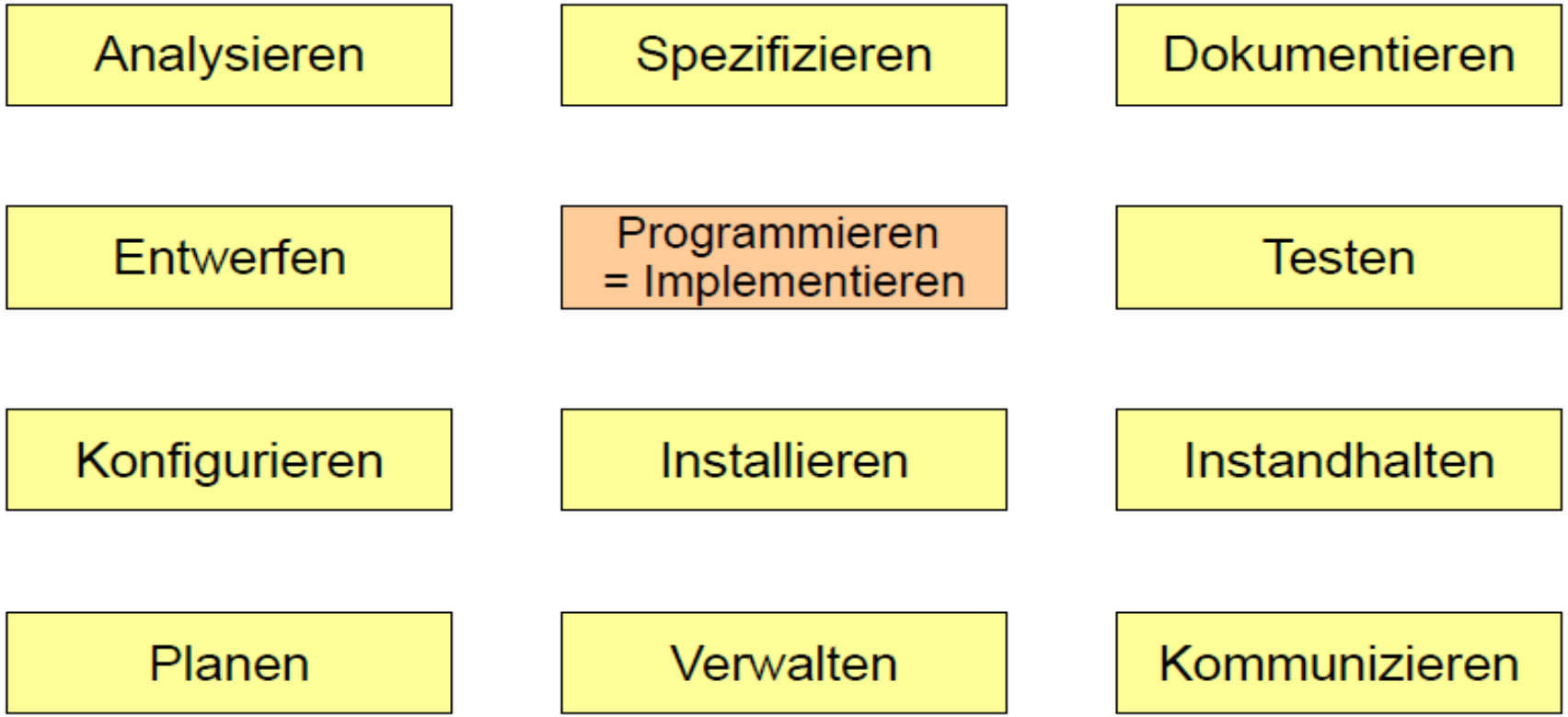


Historie Aufwendungen für Softwaresysteme



- 60-70 % aller Aufwendungen für Softwaresysteme sind Kosten für Pflegearbeiten.
- 60-70 % der Softwareentwickler sind mit Pflegearbeiten beschäftigt.

Software Engineering vs. Programmieren



- Software Engineering wenn
  - Auftraggeber ≠ Programmierer ≠ Benutzer
  - grosse, komplexe Software

Agenda

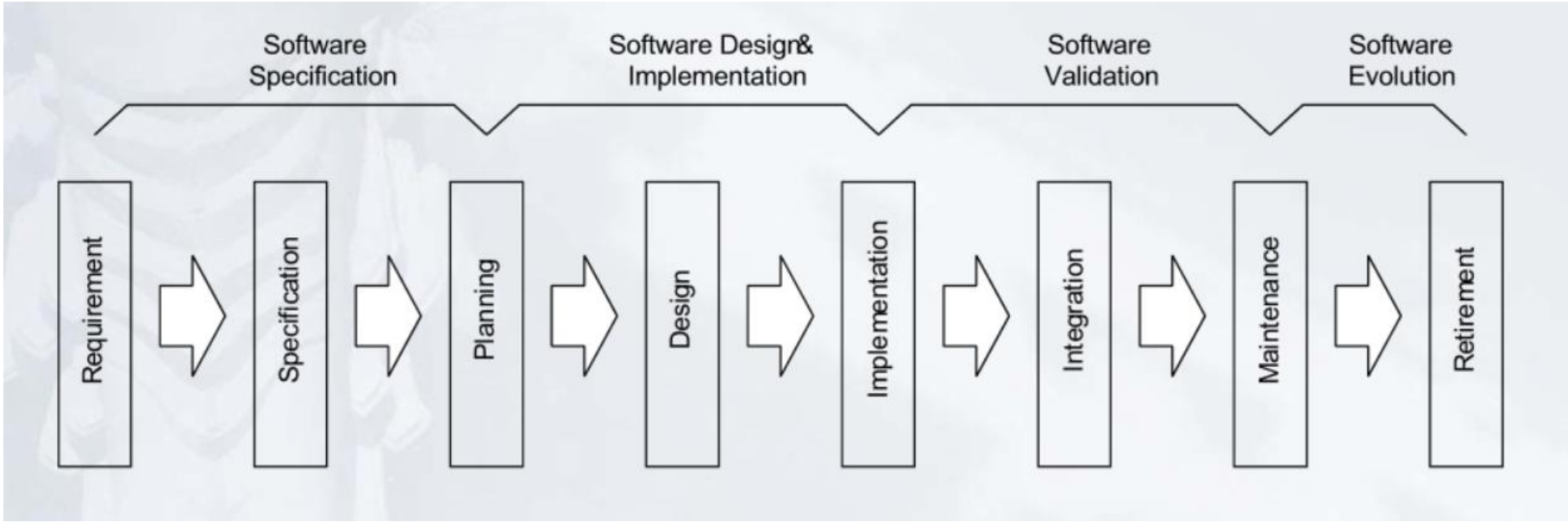
- Was ist Software Engineering
  - Geschichte Software Engineering
  - Software Engineering vs. Programmieren
- Software Entwicklung als Prozess
  - Phasen bzw. wichtigste Phase
- Qualitätsaspekte
  - Cost to Fix
  - Spektakuläre Fehler
  - Qualitätsmodell
- Teilgebiete des Software Engineerings
  - IEEE SWEBOK
- Was sind die Herausforderungen in Software Engineering?

Motivation und Zielsetzung

- Der Produktlebenszyklus startet bei der ersten Idee und endet bei der Ausmusterung der Softwarelösung.
- Warum ist eine strukturierte Softwareentwicklung notwendig?
  - Fehlerentstehung und Fehlerkosten während der Softwareentwicklung.
  - Strukturierung von Software Entwicklungsprozessen.
  - Anforderungen des Kunden sind die Basis für die Softwareentwicklung; durch sie wird definiert, was das Produkt leisten soll.
- Je nach Projektkriterien (Größe, Art, Typ, usw.) stehen zahlreiche Prozessmodelle zur Verfügung, die den wesentlichen Phasen des Life-Cycles folgen (Vorgehensmodelle).
- V-Modell und Inkrementelle Entwicklung als Beispiele für Vorgehensmodelle.

Übersicht über den Software Life-Cycle Prozess

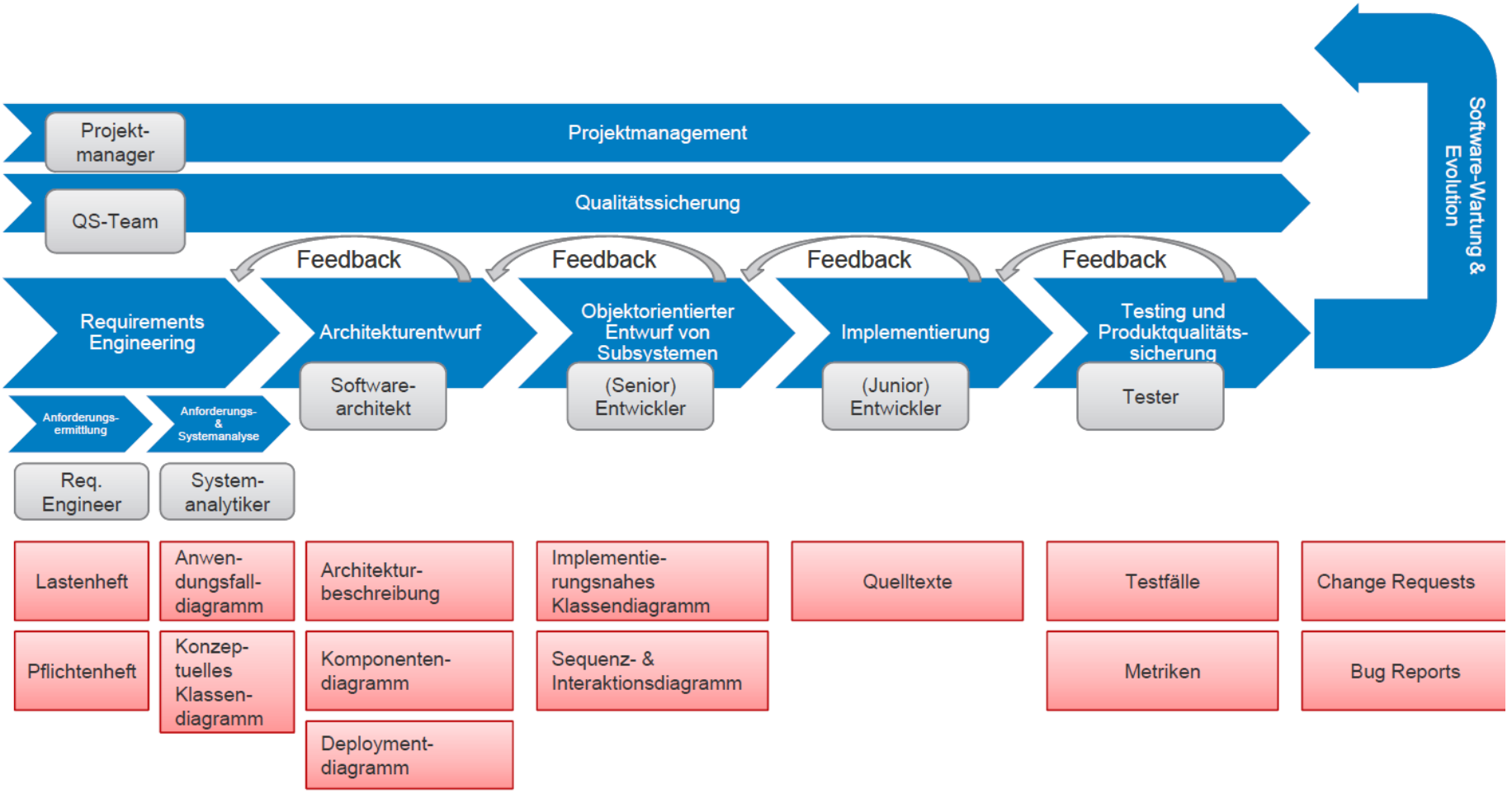
- Ein Software Prozess ist eine Abfolge von Schritten (Phasen) mit all seinen Aktivitäten, Beziehungen und Ressourcen.
- Der Software Life-Cycle beschreibt ein Basiskonzept für Software Engineering Prozesse.



Softwareentwicklung als Prozess

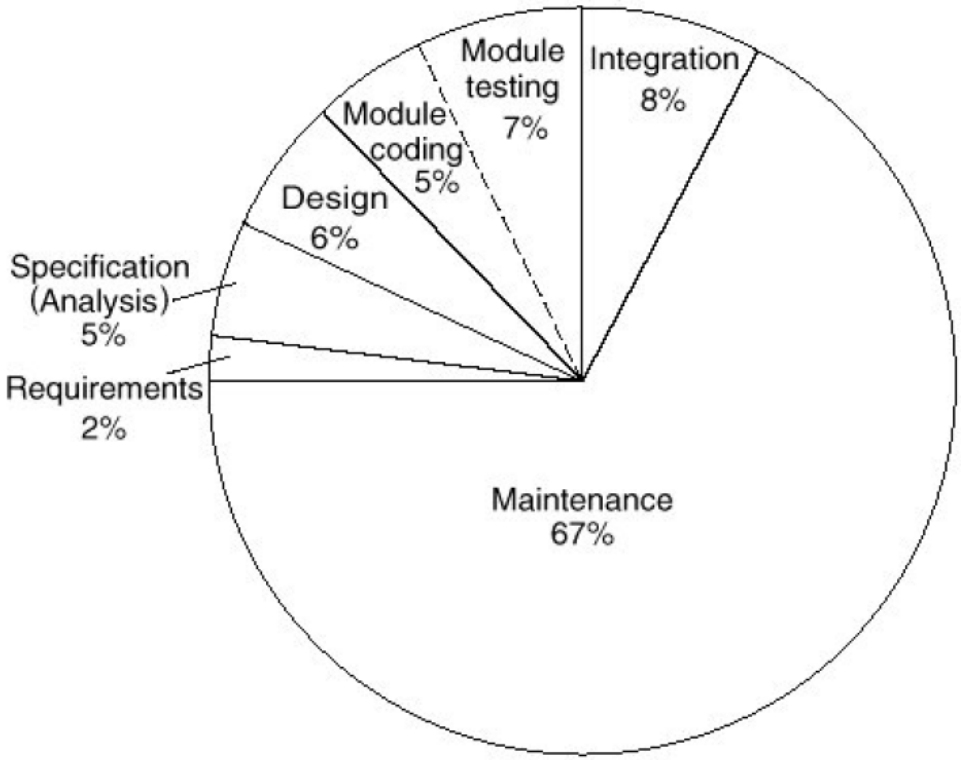
- Beginnt mit der Idee eine Aufgabe mittels Computer und Software zu lösen.
- Endet nicht mit der Auslieferung der Software.
- Software Engineering ist der Versuch diesen Prozess systematisch anzugehen.
- Am Prozess beteiligte Personen (Auszug):
  - (Potentielle) Benutzer der Software
  - Auftraggeber
  - Projektleiter
  - Softwareentwickler
  - GUI Designer
  - Tester
  - Systemadministrator
  - Qualitätsbeauftragter

Software Engineering Aktivitäten, Rollen und Artefakte



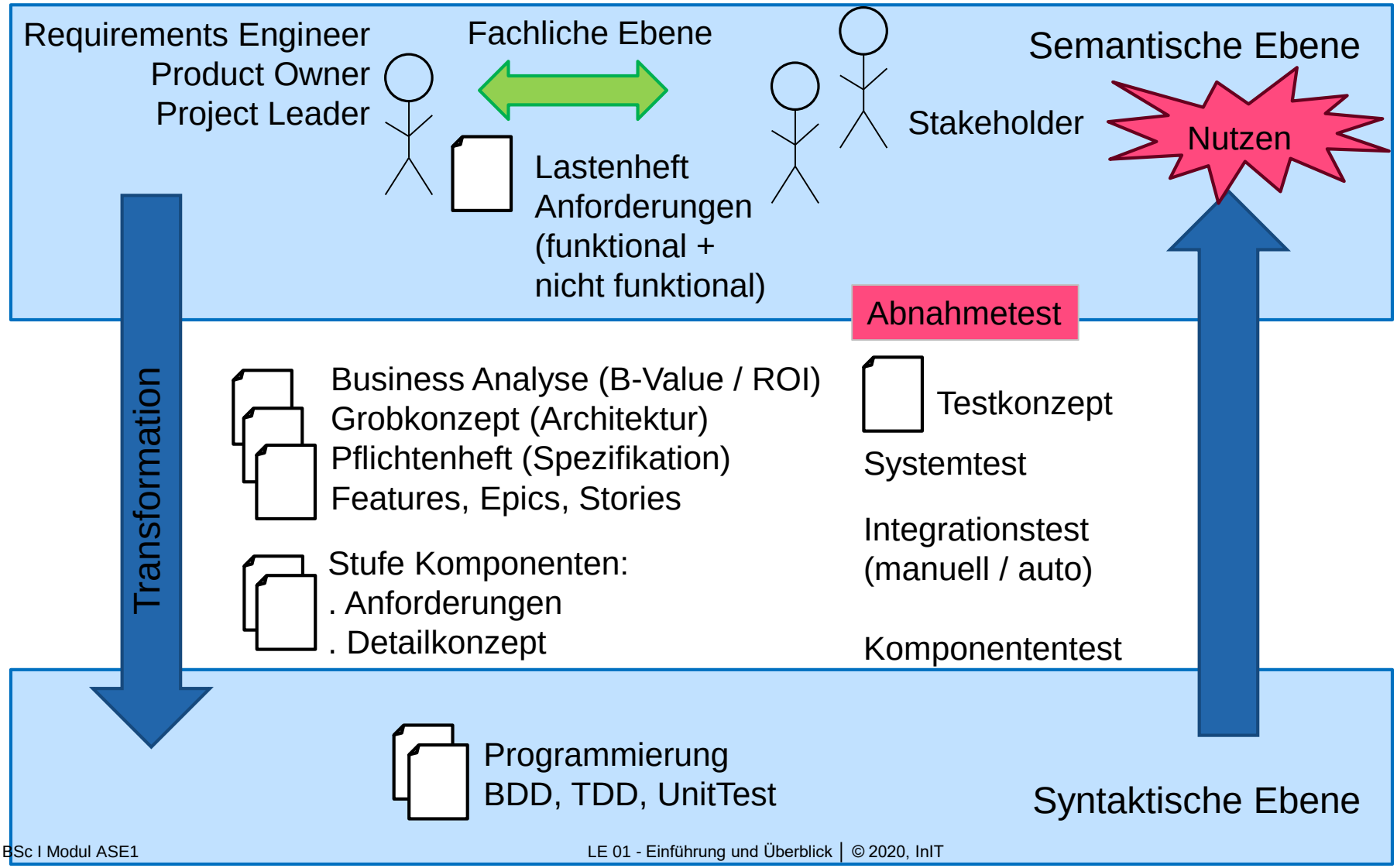
Was ist die «wichtigste» Phase?

- Wie hoch sind die Wartungskosten? Der Wartungsaufwand beeinflusst die Wartungsgebühren.
  (die Zahlen weichen je nach Anwendungstyp stark ab).



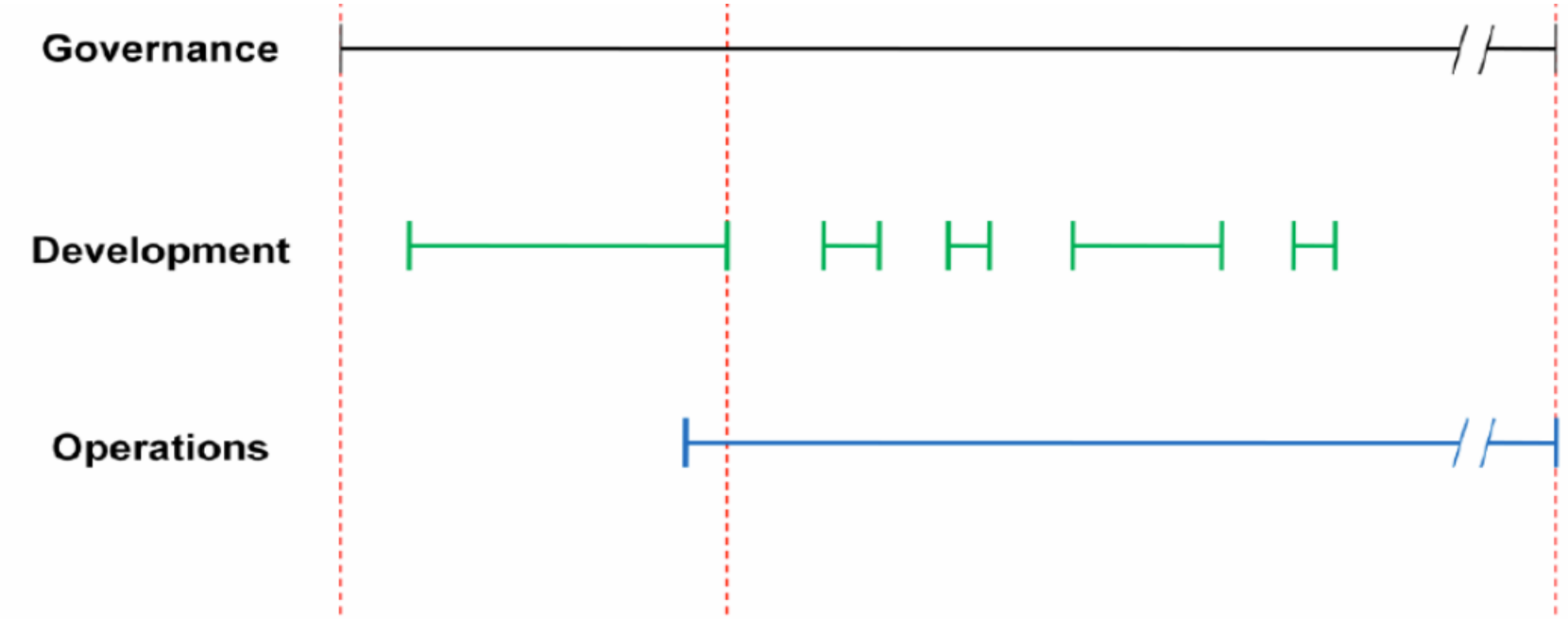
Maintenance:
. Anpassung (20%)
. Erweiterung (60%)
. Fehlerbehebung (20%)

Transformation zwischen semantischer und syntaktischer Ebene



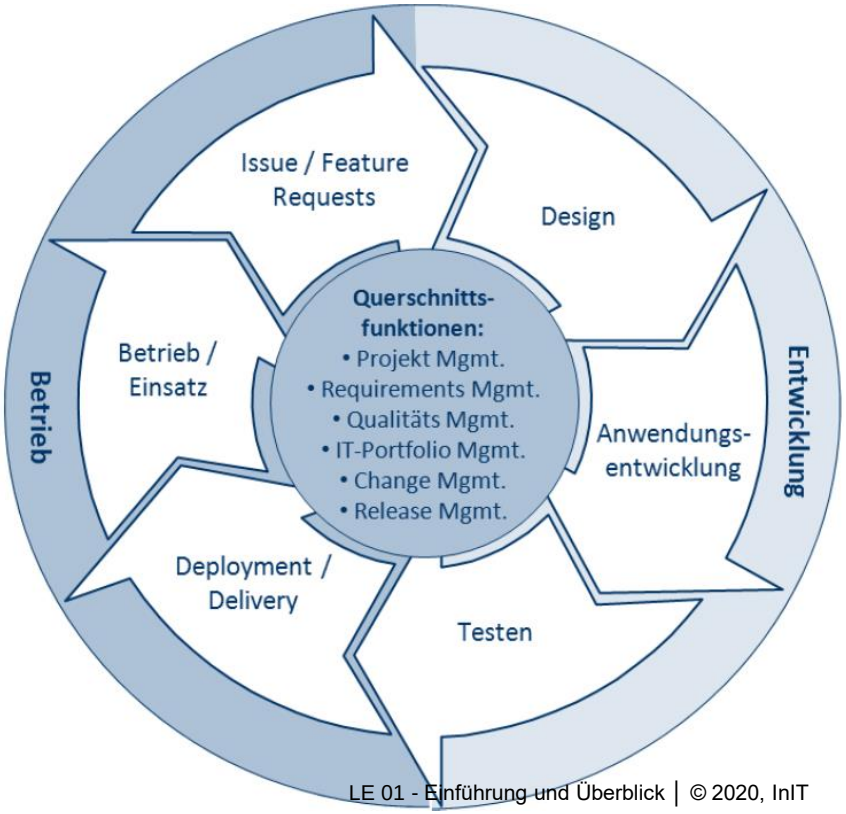
Drei Aspekte des Application Lifecycle Managements

- ALM kann in **drei** eindeutige Bereiche aufgeteilt werden:
  - Governance (Steuerung),
  - Development (Entwicklung) und
  - Operations (Betrieb).



ALM - DevOps

- ALM: Application Life Cycle Management
- DevOps: Kopplung von Entwicklung und Betrieb



Agenda

- Was ist Software Engineering
  - Geschichte Software Engineering
  - Software Engineering vs. Programmieren
- Software Entwicklung als Prozess
  - Phasen bzw. wichtigste Phase
- Qualitätsaspekte
  - Cost to Fix
  - Spektakuläre Fehler
  - Qualitätsmodell
- Teilgebiete des Software Engineerings
  - IEEE SWEBOK
- Was sind die Herausforderungen im Software Engineering?

ISO/IEC Qualitätsmodell

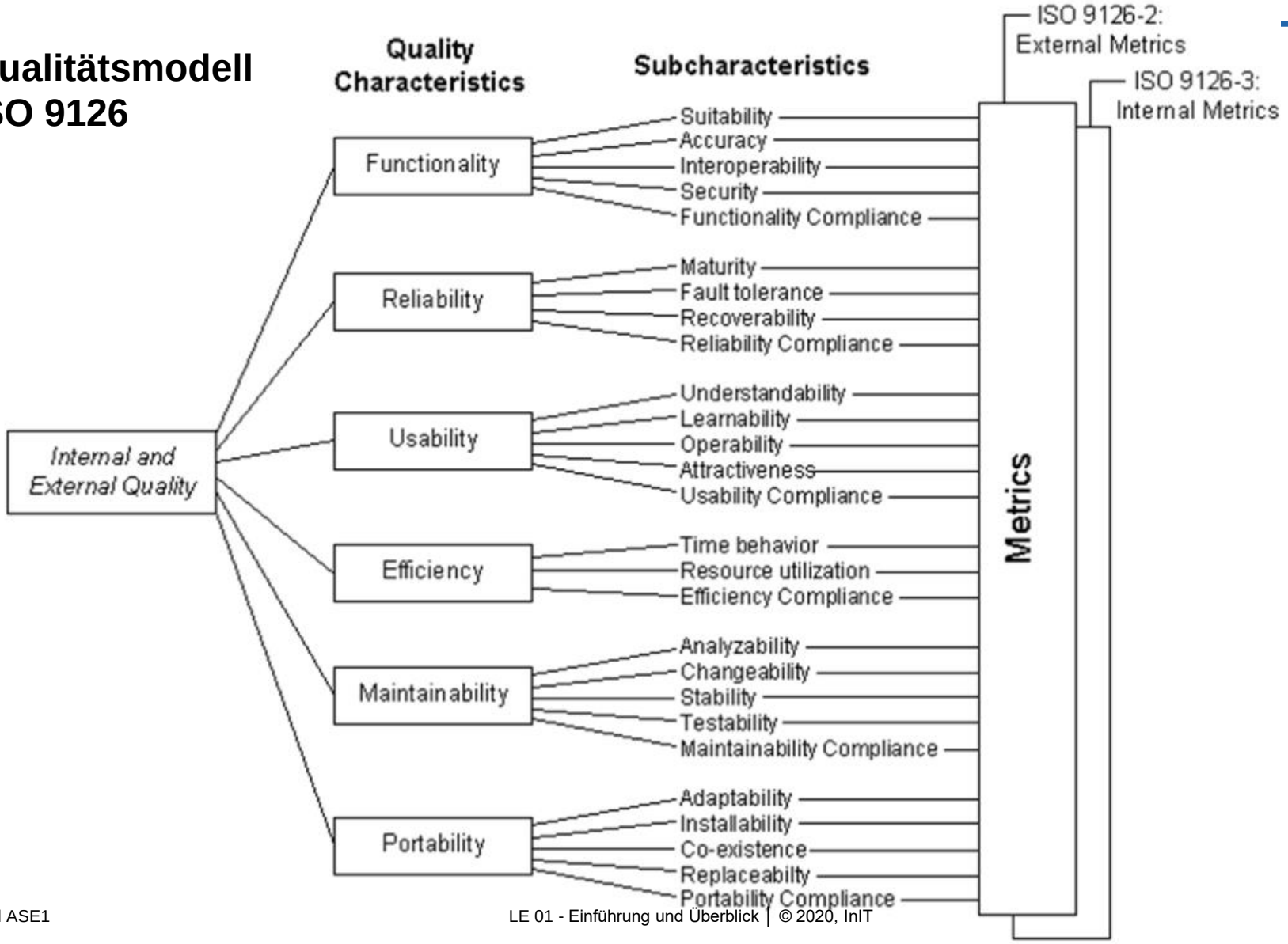
- Unterscheidung zwischen:
  - Produktqualitätsmodell
  - Quality in use Modell

- Produktqualitätsmodell: Externe und interne Qualität haben eine spezifische Bedeutung im ISO/IEC 9126/25010 Framework:
  - Externe Qualität betrachet das Produkt von aussen
  - Interne Qualität misst die Charakteristiken basierend auf dem Wissen der internen Strukturen

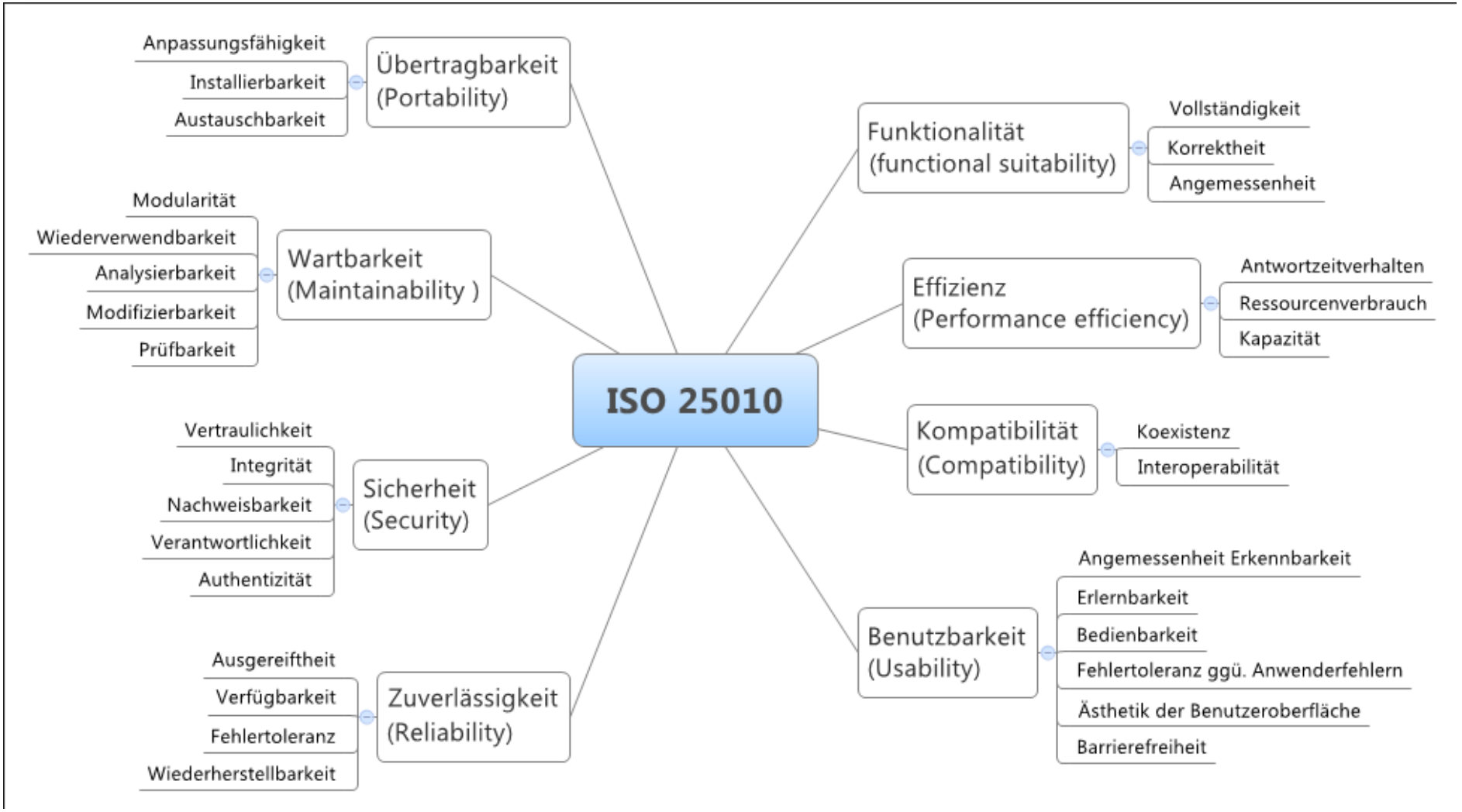
- Viele grosse Firmen haben ESTA (Entwicklungsstack)-Vorgaben mit Musterprojekten incl Unit- und e2e-Tests mit definierter Testcoverage

Alter Standard ISO 9126

Qualitätsmodell ISO 9126



Aktueller Standard ISO 25010



Ursachen für schlechte Software-Qualität

- Organisation und Management
  - unklare Zielvorstellung zw. unklare Verantwortlichkeit
  - nicht adäquate Projektplanung und Projektsteuerung
  - unzulängliche Projektkostenschätzung (Business Value vs. ROI)
  - Mitarbeiter (Qualifikation, Fluktuation, Motivation)
- Technologie
  - keine Verwendung von Vorgehensmodellen
  - keine Strategie für Entwicklungsprozessverbesserungen
  - geringe Nutzung von CASE-Tools
  - kein systematisches Testen (keine analytischen Massnahmen QM)
- Methodik
  - fehlende / ungenügende Business-Analyse und Anforderungserfassung
  - keine Nutzung von Entwicklungsmethoden
  - unvollständige Dokumentation bzw. keine Nutzung von Standards
  - keine Qualitätssicherung (keine konstruktiven Massnahmen QM)

Agenda

- Was ist Software Engineering
  - Geschichte Software Engineering
  - Software Engineering vs. Programmieren
- Software Entwicklung als Prozess
  - Phasen bzw. wichtigste Phase
- Qualitätsaspekte
  - Cost to Fix
  - Spektakuläre Fehler
- Teilgebiete des Software Engineerings
  - IEEE SWEBOK
- Was sind die Herausforderungen in Software Engineering?

Teilgebiete des Software Engineering (SWEBOK V3) – Stand 2013

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering

KA01 - Requirements	KA02 - Design	KA03 - Construction	KA04 - Testing	KA05 Maintenance
KA06 – Software Configuration Management				
KA07 – Software Engineering Management				
KA08 – Software Engineering Process				
KA09 – **Software Engineering Model and Methods**				
KA10 – Software Quality				
KA11 – **Software Engineering Professional Practice**				
Educational Requirements for Software Engineering				
KA 12 – Software Engineering Economics Foundations				
KA 13 – Computing Foundations				
KA 14 – Mathematical Foundations				
KA 15 - Engineering Foundations				

http://www4.ncsu.edu/~tjmenzie/cs510/pdf/SWEBOKv3.pdf

Agenda

- Was ist Software Engineering
  - Geschichte Software Engineering
  - Software Engineering vs. Programmieren
- Software Entwicklung als Prozess
  - Phasen bzw. wichtigste Phase
- Qualitätsaspekte
  - Cost to Fix
  - Spektakuläre Fehler
- Teilgebiete des Software Engineerings
  - IEEE SWEBOK
- Was sind die Herausforderungen in Software Engineering?

Was sind die Herausforderungen im Software Engineering?

- Altsysteme
  - Alte, wertvolle Systeme müssen gewartet und aktualisiert werden … (von Mitarbeitern welche das System nicht erstellt haben)
- Heterogenität
  - Systeme sind verteilt und bestehen aus einer Mischung von Hardware und Software
- Lieferzeiten
  - Es gibt einen zunehmenden Druck zu einer immer schnelleren Auslieferung der Software
- Produktverwaltung
  - Eine Vielzahl von Artefakten mit Versionen und Releases
- Teamarbeit
  - Nicht alle Mitarbeiter möchten in einem Team (… SCRUM Team) arbeiten. Trotzdem… grosse Aufgaben können nur in einem Team bewältigt werden.
  - Schnittstelle Fachdomäne zu IT (wie kann effizient kommuniziert werden)

Das Ende der Software-Krise?

Die Software-Krise überwunden? Nein.

- Wir haben nach wie vor ...
  - ... häufige Termin- und Kostenüberschreitungen
  - ... ganz oder teilweise gescheiterte Projekte
  - ... zu wenig konsequent gelebtes Software Engineering
- Also nichts erreicht? Doch, sehr viel sogar.
  - Wir schaffen heute Software einer Grösse und Komplexität, die vor 20 Jahren weit jenseits des Machbaren lag
  - Viele Routineprobleme werden heute gut beherrscht
- Sind wir schlechter als andere? Nein.
  - Software Engineering ist im Mittel schwieriger als das Engineering klassischer Systeme und Produkte
  - In der Beherrschung grosser Non-Standardsysteme sind wir heute teilweise besser als klassische Ingenieure

Was tun?

- Auf der Erzeugerseite!
  - Software Engineering systematisch lernen ...
    ... und konsequent in der Praxis umsetzen
  - Weg von der Kultur des Bastelns und Heimwerkens ...
    ... hin zu einer Kultur professionellen Arbeitens
- Auf der Kundenseite!
  - Bewusstsein für den Zusammenhang zwischen Kosten, Terminen und Qualität stärken
  - Governance und Application Portfolio Management fordern
  - Professionelle Business Analyse durchführen und nicht denken «die IT machts schon» => Integration der Business Analysten im SCRUM Team
  - Professionelles Software Engineering als Selbstverständlichkeit fordern
  - ALM und DevOps Einführung fordern...
  - ... und bereit sein, den damit erzielten Mehrwert zu bezahlen

Agenda

1. Motivation und Leitidee
2. Ziele, Inhalte und Ablauf des Moduls
3. Einführung ins Software Engineering
4. **Wrap-up und Ausblick**

Wrap-up

- Was ist Software Engineering
  - Geschichte Software Engineering
  - Software Engineering vs. Programmieren
- Software Entwicklung als Prozess
  - Phasen bzw. wichtigste Phase
- Qualitätsaspekte
  - Cost to Fix
  - Spektakuläre Fehler
- Teilgebiete des Software Engineerings
  - IEEE SWEBOK
- Was sind die Herausforderungen in Software Engineering?