Vertiefungsfach Software Engineering

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl für Informatik 11 (Software Engineering)

Francesca Saglietti

Problemdefinition

- Gegenstand des Software Engineering ist
  - die ingenieurmäßige Entwicklung
  - **komplexer, umfangreicher** Softwaresysteme
  - **hoher Qualität** unter Berücksichtigung
  - der einzusetzenden **Arbeits-** und **Zeitressourcen**.

- Randbedingungen

1. Problemkomplexität	Größe [SW] > 20 K LOC
2. Qualitätsvorgaben	Zuverlässigkeit, Wartbarkeit
3. Kostenlimits	Geld, Personal
4. Terminvereinbarungen	Lieferung, Genehmigung

Wissenschaft + Technik + Praxis

- Wie bei jeder klassischen **Ingenieurwissenschaft** geht es auch hier
  - - um die Erforschung wissenschaftlicher Erkenntnisse und
  - - um die Herleitung praktisch anwendbarer Techniken

- Software Engineering umfasst daher folgende Aspekte:
  - Allgemeingültige Prinzipien und wissenschaftliche Erkenntnisse auf der Basis von Theorien *(theoria [gr.] = Beobachtung)*, Experimenten und Fallstudien
  - Verfahren zur technischen Umsetzung *(techne [gr.] = Handwerk, Kunstfertigkeit)* der wissenschaftlich hergeleiteten Ansätze
  - Praktische Anwendung der Verfahren *(prassein [gr.] = tun, handeln)* mit Hilfe von CASE (Computer-Aided Software Engineering) Tools zur Automatisierung und Visualisierung der Verfahren

z. T. gegenläufige Anforderungen

- Begriff Ende der 60er Jahre geprägt, um auf Notwendigkeit einer über das Handwerk hinausgehenden Systematisierung des Entwicklungsprozesses hinzuweisen
- Seitdem rasantes Wachstum (z.B. in Verkehrs- und Medizintechnik)
  - des Umfangs,
  - der Komplexität und
  - der Relevanz der durch Software realisierten Funktionalitäten
- Andererseits: sehr variable, zum Teil einander gegenläufige nichtfunktionale Anforderungen an heutige Softwaresysteme:
  - die Zuverlässigkeit,
  - den Freigabezeitpunkt
  - die Änderungsfreundlichkeit
  - die Wiederverwendbarkeit
  - die Kosten
- In Abhängigkeit von der Priorisierung dieser Ziele
  - unterschiedliche Vorgehensweisen bei der Softwareerstellung

Keinen goldenen Weg

- Neueste internationale Standards (IEC / ISO)
  - unterschiedliche Vorgehensweisen empfohlen / vorgeschrieben
  - in Abhängigkeit vom Grad der Relevanz der Software.
  - eine einheitlich standardisierte Softwareentwicklung gibt es in diesem Sinne nicht.

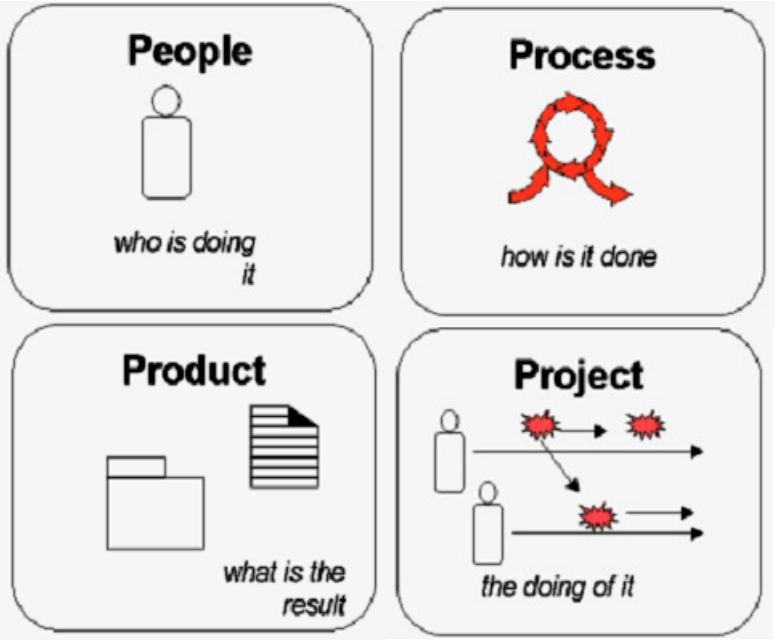
- „Kunst“ des Software-Ingenieurs
  - Beherrschung der einzelnen Techniken
  - Identifikation optimaler Vorgehensweisen (Qualität vs. Kosteneffizienz) anhand signifikanter Eigenschaften und Kenngrößen des aktuell vorliegenden Projekts
  - etwa im Hinblick auf Prozessmodellierung, Spezifikationssprache, Architektur, Nachweisverfahren, Personalorganisation, etc.)

Lehrangebot

- Bedarf
  - wissenschaftliche Kenntnis alternativer technischer Optionen
  - wissenschaftlich geprägtes Analyse- und Mess-Instrumentarium

- Vertiefungsfach Software Engineering
  - über die Erlernung zahlreicher Techniken zur Softwareentwicklung und zum Eignungsnachweis hinaus
  - systematische Untersuchung der Stärken und Schwächen der einzelnen Verfahren
  - sowie Bewertung ihres Eignungspotentials in Abhängigkeit von konkreten Problem- und Anwendungsklassen zum Ziel.
  - aus theoretischer und anwendungsorientierter Sicht

4 Aspekte des Software Engineering



- **Personal**
  Wer entwickelt?
- **Prozess**
  Wie wird entwickelt?
- **Produkt**
  Was ist das Ergebnis?
- **Projekt**
  Gesamte Koordination

Alle 4 Aspekte kontrollieren, also **messen**!

"You cannot control what you cannot measure!" (**Tom De Marco**)

Unterschiede zu bisherigen Veranstaltungen

- Erste 2 Semester
  - Programmieren im Kleinen
  - durchaus anspruchsvolle Aufgaben
  - überschaubare, vorgegebene Fragestellungen
  - Schwerpunkt: Implementierung von Lösungen

- 3. Semester: Software-Entwicklung in Großprojekten
  - ausgewählte Themen für das „Programmieren im Großen“
  - Requirements Engineering: zu ermittelnde Aufgabenstellung
  - Phasen des Lebenszyklus: strukturiertes Vorgehen
  - Einzelne Prozessmodelle
  - beispielhafte Vorgehensweisen für jede Phase

Vertiefungsfach Software Engineering

- Ab 5. Semester:
  - Software Engineering als Ingenieurwissenschaft
  - pro Phase und Tätigkeit des Software-Lebenszyklus:
    mehrere **alternative** Vorgehensweisen vergleichend bewertet
  - Instrument zur **Entscheidungsfindung** für optimales Vorgehen im Einzelfall (Preisleistungsverhältnis: Einfamilienhaus oder Schloß?)
  - Betrachtung **menschlicher** Faktoren (kognitiv-psychologisch)
    Denkfallen (Scheinwerferprinzip, Pregnanztendenz, etc.)
    Personalführungsstil (chief programming, democratic teams, etc.)
  - **Messung** von Kenngrößen:
    Entwurfsqualität (Kohäsion, Kopplung)
    Programmkomplexität (Wortschatz, Struktur)
    Testfortschritt (Testüberdeckungsmaße, mutation score)
    Kostenmodelle (Personalkosten, Projektrestzeit)
  - Weitere spezielle Themen: Wiederverwendung, Integrationstest

Module zu übergeordneten Themen

- Grundlagen des Software Engineering 7,5 ECTS

  Vorlesung und Übungen (6 SWS), Sommersemester

- Erweiterte Grundlagen des Software Engineering 10 ECTS

  Vorlesung und Übungen (6 SWS), zu ergänzen durch
  - Option A: “Software Engineering in der Praxis” (Übungen, 3 SWS) oder
  - Option B: “Seminar Software Engineering” (Seminar, 2 SWS)

- Konstruktives Software Engineering 5 ECTS
  - Vorlesung “Grundlagen des Software Engineering” (erste 2 Monate) mit Übungen (insgesamt 4 SWS)

- Organisation u. Qualitätskontrolle im modernen Software Engineering 5 ECTS
  - Vorlesung “Grundlagen des Software Engineering” (dritter Monat) mit Übungen (insgesamt 2 SWS)

  zu ergänzen durch
  - Option A: “Software Engineering in der Praxis” (Übungen, 3 SWS) oder
  - Option B: “Seminar Software Engineering” (Seminar, 2 SWS)

Lerninhalte zu übergeordneten Themen

- Konstruktive Aspekte
  - Prozessmodelle, Requirements Engineering, Spezifikationssprachen
  - Architekturen, Integration, Wiederverwendung, Entwurfsmuster
- Analytische Aspekte
  - informal (Inspektion, Durchsicht, Durchgang, Schreibtischprüfung)
  - statische (Analysatoren, Komplexitätsmetriken, Beweise)
  - dynamisch (Testen, on-line checks, Auswertung der Betriebserfahrung)
- Organisatorische Aspekte
  - Projektmanagement, Personalführung
  - Prozessverbesserung, Kostenmodelle, menschliche Faktoren
- Praktische Erprobung von CASE-Werkzeugen
  - individuelle praktische Erprobung der vorgestellten Verfahren
  - unter möglichst realen Randbedingungen
  - Einsatz automatisierter Hilfsmittel für Entwicklung, Analyse und Management

Software Engineering in der Praxis

- Unterstützung folgender Tätigkeiten des Software-Lebenszyklus
  - Modellierung und Simulation des Systemverhaltens, Analyse erreichbarer Zustände durch **klassische und zeit-behaftete Petri Netze**
  - Anforderungsanalyse durch vollautomatisches Nachweisen bzw. Widerlegen von Sicherheits- bzw. Lebendigkeitseigenschaften mittels **Model Checkers**
  - Verifikation durch maschinenunterstützte interaktive Korrektheitsbeweisführung mittels **Theorem Provers**
  - objektorientierte Analyse und Design durch graphische Editierung von **UML**-Modellen und automatische Generierung von Programmskeletten
  - Ermittlung quantitativer Indikatoren projektspezifischer **Komplexität** mittels automatischer Bestimmung prozess- und produktbasierter Software**metriken**
  - Evaluierung struktureller Testphasen durch automatische Ermittlung kontrollflussbasierter **Testüberdeckungsmaße** mittels Codeinstrumentierung
  - Configuration Management mittels maschineller **Verwaltung** zeitlich aufeinanderfolgender Versionen, Releases und kundenspezifischer Varianten
  - Dokumentation der Wartungs- und Pflegemaßnahmen durch Aufzeichnung, Kategorisierung und Weitergabe eingehender **Fehlermeldungen**, Änderungswünsche sowie getätigter Aktionen bei Fehlersuche und -behebung

Module zu Spezialgebieten

- Test- u. Analyseverfahren zur Software-Verifikation und –Validierung
  5 ECTS
  Vorlesung mit Übungen (4 SWS) Wintersemester

- Fehlertolerierende Softwarearchitekturen 5 ECTS
  Vorlesung mit Übungen (4 SWS) Wintersemester

- Bewertung der Softwarezuverlässigkeit 5 ECTS
  Vorlesung mit Übungen (4 SWS) Sommersemester

Lerninhalte zu Spezialgebieten

- Software Test, -Verifikation und –Validierung
  - Analyse der technischen Sicherheit softwaregesteuerter Anwendungen
  - Analyse der Fehlerarten und –effekte (FMEA), Fehlerbaumanalyse (FTA)
  - Risiko-Analyse und Klassifikation, Sicherheitsstandards, Safety Integrity Levels
  - Kontroll- und datenflussbasierte Teststrategien
  - Mutationstesten
  - Model Checking und axiomatische Beweisverfahren

- Bewertung der Softwarezuverlässigkeit
  - Schätzung optimaler Time-to-market für kommerzielle Softwarepakete
  - Bewertung hochzuverlässiger, sicherheitskritischer Software
  - Bewertung hochverfügbarer Software für Telekommunikationssysteme

- Fehlertolerierende Softwarearchitekturen
  - Entwurf und Bewertung redundanter Softwarearchitekturen
  - Erkennung, Behebung, Beherrschung sporadischer Softwarefehler im Betrieb