Software Engineering

Informatik II.

11. Software-Entwicklung

– Produktivitätsfaktoren –

Dipl.-Inform. Hartmut Petters

Vorwort – was ich noch zu sagen hätte ...

Basis dieser Vorlesung sind vor allem die folgenden Ausarbeitungen

- Vorlesungsskript „Software Engineering“
  von Prof. Dr. Martin Glinz Universität Zürich
  http://www.ifi.unizh.ch/groups/req/courses/ses/
- Vorlesungsskript „Informatik II – Software Engineering“
  von Frau Prof. Dr. Kühn FH Karlsruhe FB W
  http://www.home.fh-karlsruhe.de/~kuin0001/inhalt.htm
- Das Buch „Software Engineering“ 6. Auflage/2001
  von Prof. Ian Sommerville University of Lancaster (UK)
  Addison Wesley ISBN 3-8273-7001-9

Konkret entnommene Beiträge sind i.d.R. mit einem Quellen-Verweis gekennzeichnet – sollte dieser fehlen bitte ich um Nachsicht.

Den „roten Faden“ durch die Vorlesung habe ich dem Skript der Vorlesung von Prof. Dr. Martin Glinz entnommen und um eigene Beiträge erweitert bzw. aus den beiden anderen Quellen ergänzt.

Für die Möglichkeit der Verwendung der wesentlichen Inhalte möchte ich mich an dieser Stelle bei den Autoren herzlich bedanken.

Einflussfaktoren

- Produktivität wird durch viele Faktoren beeinflusst

- Hauptfaktoren
  - Werkzeuge
  - Wiederverwendung / Beschaffung
  - Menschen
  - Methoden / Prozesse

Werkzeuge – 1te

Zum Schnitzen braucht man ein Messer

... und zum Entwickeln von Software braucht man geeignete Werkzeuge!

Aber:

Die besten Messer sind *nutzlos*, ...

... wenn der Schnitzer *nicht* mit ihnen *umgehen* kann

... wenn er *nicht weiß, was* er schnitzen soll

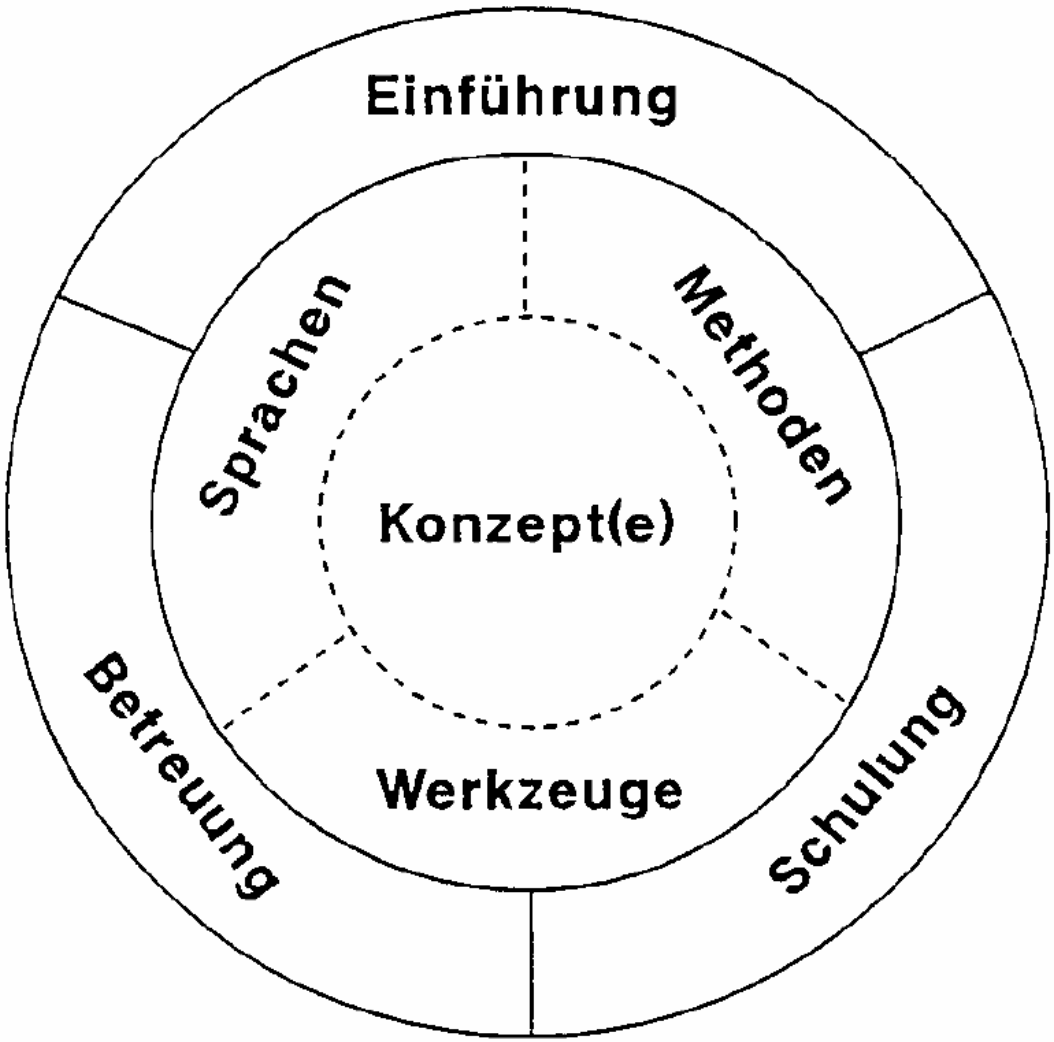
- Werkzeug (tool) – bzgl. Software-Engineering rechnergestützte Hilfsmittel für die Entwicklung von Software.
  Auch CASE (Computer Aided Software Engineering) genannt.

Was Werkzeuge leisten

- Entlastung von *Routineaufgaben*
- Bearbeiten von *Sprachen*
- Unterstützen den Einsatz von *Methoden*
- Vereinfachen + Management von *Änderungen*

- Aber
  - Werkzeuge sind *keine Wunderwaffen*
  - *Keine* Produktivitätssteigerung um *Größenordnungen*
  - Ersetzen nicht eigenes *Denken* und *sorgfältiges Arbeiten + Disziplin!*
  - Machen das *Qualitäts-Management* nicht überflüssig

CASE (Computer Aided Software Engineering)



Klassifikation von Werkzeugen

- Editoren
- Spezifikations- und Entwurfs-Systeme
- Programm-Entwurfs-Systeme
- Compiler, Browser und Programmierumgebungen
- Programm-Generatoren
- Mess- und Test-Werkzeuge
- Konfigurationsverwaltung

Produktivitätsgewinn durch Werkzeuge

- *Substanzielle Produktivitäts- + Qualitäts-Steigerungen* sind durch Werkzeuge realisierbar!

- Aber
  - Bei der *Einführung* sinkt zuerst die *Produktivität*
    - Schulung
    - Lernkurve / Eingewöhnung
    - Verlagerung von Aufwendungen

- Der Gewinn – die Produktivitätssteigerung kommt erst *mittelfristig*
  - ↳ Werkzeug-Einführung ist eine Investition!

Planung des Werkzeugeinsatzes

- Was soll unterstützt werden?
- Wie wirtschaftlich ist der Einsatz?
- Welche Entwicklungskonzepte (Methoden, Sprachen) werden eingesetzt?
- Ist die Schulung sichergestellt?
- Wie sieht die Einführungsstrategie aus?
- Ist die Betreuung (im Betrieb) sichergestellt?

Mehrfachverwendung / Wiederverwendung

Die Masse bringt´s!

- Kostensenkung über hohe Stückzahlen
  - *Große Produktserien* mit identischer Software
  - Mehrfachverwendung der *gleichen Software in mehreren Projekten*

Rolle der Menschen im Software-Engineering

- Software wird von Menschen gemacht!

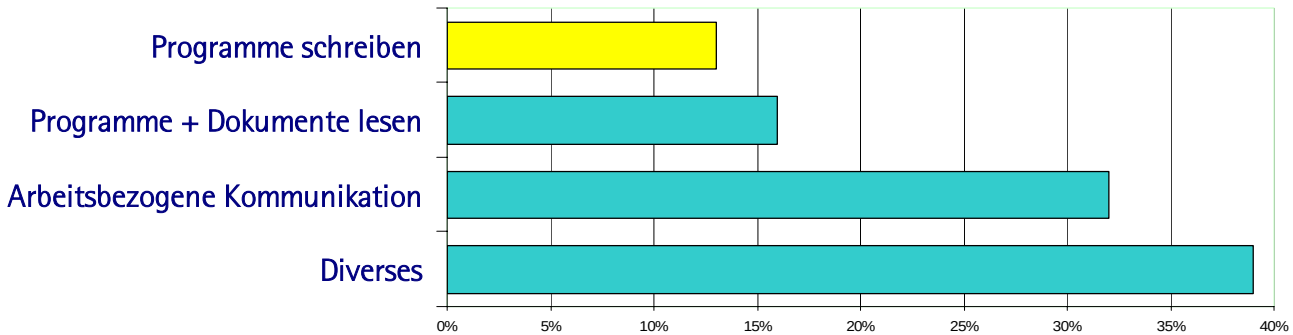
- Die Software-Leute sind ein entscheidender Produktivitätsfaktor
  - Können / Kompetenz
  - Motivation
  - Arbeitsumfeld
  - Tagesform

Gesetzmäßigkeiten bei Software-Leuten – 1te

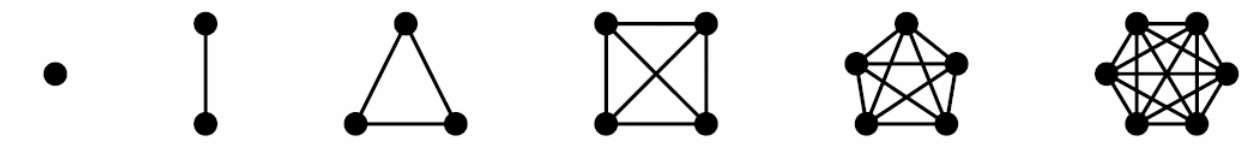
- Produktivität
  - Enorme Schwankungsbandbreiten – bis zu 20:1
  - Selbst bei Gruppen noch Schwankungen bis 4:1
- Disponibilität
  - Personalbestände sind nur langsam auf- und abbaubar
  - Zu jedem Aufwand eine optimale Personenzahl
  - Das Aufstocken des Personalbestands in einem verspäteten Projekt führt zu noch mehr Verzögerungen (Gesetz von Brooks)

Gesetzmäßigkeiten bei Software-Leuten – 2te

- Arbeitsverteilung
  - Programmierer schreiben nicht nur Programme



- Gruppengröße
  - Nicht produktiver Aufwand wächst überproportional zur Gruppengröße



Emotional vs. Rational

- *Fühlen* im Kleinen – *Arbeiten* im Großen
- Kreativität *wollen* – Disziplin *benötigen*
- *Lustbetonte* Arbeiten ≠ *Notwendige* Arbeiten
- *Trägheitseffekte* behindern Innovationen
- *Kurzfristiges Denken* führt zu Fehlsteuerungen

Der Einfluss der Arbeitsumgebung

- Ausbildung
  - *Gute Leute einstellen*
    - 1er stellen 1er ein
    - 2er stellen 3er ein
  - *Leute besser machen*
    - Fortbildung
    - Geeignete Gruppenprozesse / Team-Bildung
- Arbeitsplätze, an denen gearbeitet werden kann
  - Genug *Platz*
  - *Technisch adäquat* ausgerüstet
  - *Wenig Störungen*
  - *Kommunikations-Kultur*

Software-Management / Software-Kultur

- Realistische Planung
- Gegenseitiges Vertrauen
- Informationskultur
- Schaffung einer Software-Kultur
  - Wir sind kompetent, aber wir können nicht alles.
  - Wir sind schnell, aber wir versuchen nicht uns selbst zu überholen oder die Lichtgeschwindigkeit zu ändern
  - Wir tun die Dinge von Anfang an richtig
  - Wir haben Spaß an professioneller Arbeit
  - Entscheidungen so spät wie möglich, aber so früh wie nötig treffen.

Zusammenfassung

- Produktivität wird beeinflusst durch
  - Die *Menschen*
  - Die *Arbeitsumgebung*
  - Die *Arbeitskultur*
- Werkzeuge müssen
  - *zweckdienlich* sein
  - *einsetzbar* sein
    - ↳ die Bedienung *bekannt und vertraut*
- Produktivität wird erreicht durch
  - *Kontinuität*
  - *Disziplin*
  - *Professionalität*

Literatur – Software Engineering

- Skript Informatik II — Prof. Dr. Kühn / Fb W FH Karlsruhe
  http://www.home.fh-karlsruhe.de/~kuin0001/inhalt.htm
- Skript Software Engineering — Prof. Dr. Martin Glinz Universität Zürich
  http://www.ifi.unizh.ch/groups/req/courses/se_I/
- Skript Software Engineering II — Bernd Kahlbrandt FH Hamburg
  http://www.informatik.fh-hamburg.de/~khb/st4se2/
- Software Engineering
  Ian Sommerville — (ISBN3-82737-001-9)
- Software Engineering
  - Grundkurs für Praktiker –
  Roger S. Pressman — (ISBN 3-89028-163-X)
- Software Entwurf
  - Methoden und Werkzeuge –
  A. Schulz — (ISBN 3-486-21608-2)
- Software Engineering und Prototyping
  Thorsten Spitta — (ISBN 3-540-17542-3)
- CASE
  Helmut Balzert — (ISBN 3-411-03224-3)
- Software-Qualitätssicherung
  Ernest Wallmüller — (ISBN 3-446-15846-4)

Software Engineering

Informatik II.

Software-Entwicklung

Dipl.-Inform. Hartmut Petters