TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Software Entwicklung 1

Annette Bieniusa / Arnd Poetzsch-Heffter

AG Softech
FB Informatik
TU Kaiserslautern

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Organisation

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Themen

- Überblick über die Software-Entwicklung
- Technische und formale Grundlagen der Programmierung
- Syntax und Semantik
- Grundkonzepte der Programmierung (Wert, elemtarer Datentyp, Prozedur, Bezeichnerbindung, ...)
- Datenstrukturen und Algorithmen
- Objektorientiertes Programmieren in Java
- Spezifikation, Testen

⇒ **Lernziel:** Einfache, sequentielle SW-Systeme selbstständig verstehen, entwickeln und implementieren können

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Literaturempfehlungen I

Objektorientierte Programmierung

R. Sedgewick, K. Wayne: Einführung in die Programmierung mit Java. 2011, Pearson Studium.

A. Poetzsch-Heffter: Konzepte objektorientierter Programmierung. 2009, Springer-Verlag

G. Krüger, H. Hansen: Handbuch der Java-Programmierung. 7. Auflage 2006, Addison Wesley (verfügbar unter www.javabuch.de)

B. Liskov: Program Development in Java: Abstraction, Specification, and Object-Oriented Design. 2000, Addison-Wesley

Literaturempfehlungen II

Datenstrukturen und Algorithmen

T. Ottmann, P. Widmayer Algorithmen und Datenstrukturen. 5. Auflage 2012, Spektrum Akademischer Verlag

R. Sedgewick, K.Wayne Algorithmen: Algorithmen und Datenstrukturen. 4. Auflage. 2014, Pearson Studium

Einführung in die Informatik (allg.)

G. Goos, W. Zimmermann Vorlesungen über Informatik. Band 1 u. 2. Springer-Verlag

M. Broy Informatik. Eine grundlegende Einführung. Band 1 u. 2. Springer-Verlag

SE1 Team

- Dr. Annette Bieniusa (Vorlesung)
- Mathias Weber und Peter Zeller (Übungsleitung)
- Studentische Tutoren (siehe Vorlesungswebpage)

Sprechstunden/Informationen/Materialien

A. Bieniusa mittwochs, 9:30-11:00

M. Weber mittwochs, 9:30-11:00

P. Zeller mittwochs, 9:30-11:00

- `http://softech.informatik.uni-kl.de/`
  > Lehre

Vorlesung

- Dienstags 11:45 - 13:15, Raum 46-220
- Donnerstag 11:45 - 13:15, Raum 46-215
- Besuch der Vorlesung:
  Zuhören, Mitdenken, Notizen machen, Fragen
- Nachbereiten der Vorlesung

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Übungen I

- Übungsblatt (1 pro Woche / 14 Blätter)
  - → Ausgabe in den Übungsstunden
- Präsenzaufgaben in der Übungsstunde
- Einreichaufgaben in Dreiergruppen

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Übungen II

- praktische Übungsaufgaben selbständig am Rechner durchführen
- Abgabe der Lösungen pro Gruppe
  - → bis 12:00 Uhr zwei Tage vor der eigenen Übungsstunde
  - → Online
  - → Korrektur durch studentische Tutoren mit Bepunktung
- Übungsstunde (1 x wöchentlich)
  - → Anwesenheitspflicht! (2 Fehlzeiten erlaubt)
  - → Diskussion von Fragen zur Vorlesung
  - → Behandlung der Einreichaufgaben
  - → Lösen der Präsenzaufgaben

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Modulprüfung

- schriftliche Abschlussklausur
- Termin (voraussichtlich!): März 2016
- Zulassungsvoraussetzungen: 50% der Punkte

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Übungsorganisation

- 10 Übungsgruppen bestehend aus je 8-9 Dreiergruppen
- Anmelden zu den Übungen
  - http://softech.informatik.uni-kl.de/stats
  - Account erstellen: Auf gültige Email-Adresse achten!
  - Maximal bei einer Übungsgruppe möglich!
  - Nur wer einer Übungsgruppe angehört, kann an den Klausuren/Prüfungen teilnehmen
  - Ersten Übungsgruppentreffen: Woche vom 03.11.-07.11.14
  - In erster Vorlesungswoche: Betreuung zu den Übungszeiten in den Terminalräumen in Gebäude 48

Was ist Informatik?

Begriffsklärung I

Informatik aus technischer Sicht [Broy]

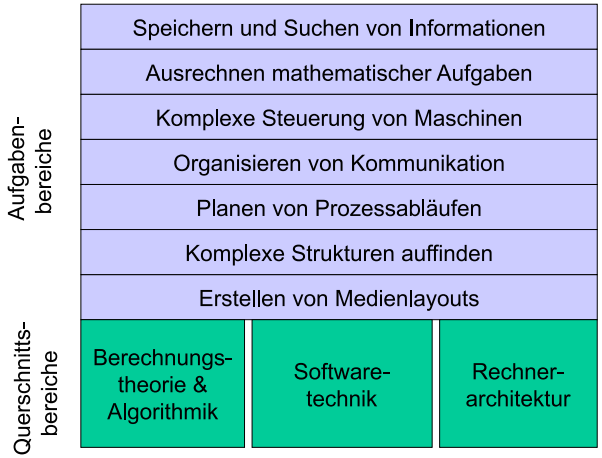
Die **Informatik** "umfasst Wissenschaft, Technik und Anwendung der maschinellen Verarbeitung, Speicherung und Übertragung von Information".

Informatik aus Aufgabensicht

Informatik ist die Wissenschaft von der Unterstützung und Automatisierung

- geistiger Tätigkeiten von Lebewesen und
- nicht-physikalischer Prozesse in Organisationen.

Begriffsklärung II



Bemerkungen

- Zu allen genannten Aufgabenbereichen hat der Mensch seit Tausenden von Jahren Lösungen vorgeschlagen.
- Es gibt etliche weitere Bereiche:
  - Übersetzen natürlicher Sprache
  - Erkennen von handschriftlichen Texten, Gesichtern, Musikstücken
  - Lösen mathematischer Probleme
  - ...

Einordnung der Informatik

- fachliches Umfeld
- zeitliches/historisches Umfeld
- gesellschaftliches Umfeld

Fachliches Umfeld der Informatik I

- Mathematik / Formale Logik
- Elektrotechnik
- Anwendungsbereiche
  - → Betriebs-, Verwaltungswirtschaft
  - → Steuerung, Automatisierung
  - → Kommunikation
  - → ...
- Weitere Schnittstellen:
  - → Psychologie (Arbeitsprozesse, Kognition)
  - → Neurologie (Informationsverarbeitung)
  - → Linguistik
  - → Philosophie

Fachliches Umfeld der Informatik II

Beziehung zu anderen Informatikvorlesungen des Grundstudiums:

- Software-Entwicklung 2 und 3
- Rechnersysteme
- Formale Grundlagen der Programmierung
- Logik
- Algebraische Strukturen
- Mathematische Algorithmen

Zur Geschichte der Rechentechnik I

ca. 5000 v.C.	Erste Zahlensysteme
ab 450 v. Chr.	Verwendung des Abakus
ca. 350 v.Chr.	Euklid: Algorithmus zur Berechnung des größten gemeinsamen Teilers.
5. Jhd.	Dezimalsystem wird in Indien entwickelt.
9. Jhd.	Ibn Musa Al-Chwarizmi (Persien): Lehrbuch “Regeln der Wiedereinsetzung und Reduktion”
1050	Dualsystem wird in China entwickelt.
1547	Adam Riese veröffentlicht Rechenbuch zur Rechnung im Dezimalsystem.
1623	Wilhelm Schickard (Tübingen) konstruiert eine Maschine für die Grundrechenarten.
1679	(Wieder-)Entdeckung der Dualzahlen durch Leibniz.

Zur Geschichte der Rechentechnik II

1801 Jacquard: mechanischer Webstuhl mit Steuerung von Mustern durch gestanzte Platten.

ab 1818 Rechenmaschinen nach Vorbild von Leibniz werden in Serie produziert und verbessert.

1886 Hollerith: elektrische Zählmaschinen für Lochkarten.

1924 Hollerith Tabulating Company wird zur IBM.

1937 Konrad Zuse: mechanische Anlage Z1

1937 Alan Turing: Modell für einen Universalrechner

1941 Die elektro-mechanische Anlage Z3 ist der erste funktionsfähige programmgesteuerte Rechner. Die Programmierung erfolgt mit Lochstreifen.

Zur Geschichte der Rechentechnik III

1943 Der britische Röhrenrechner Colossus knackt den Enigma-Code der deutschen Wehrmacht.

1946 J. P. Eckert und J. W. Mauchly bauen ENIAC, den ersten voll elektronischen Rechner mit ca. 18.000 Elektronenröhren.

1952 Erste Rechner nach dem Prinzip von John von Neumann.

1957 Die Programmiersprache Fortran wird von John Backus vorgestellt.

1959 Texas Instruments entwickelt den ersten integrierten Schaltkreis.

1960 ALGOL-60, die erste Programmiersprache mit Blockstruktur und Rekursion, wird vorgestellt.

Zur Geschichte der Rechentechnik IV

1965 Doug Engelbart entwickelt die erste Maus.

1967 Dahl u. Nygaard entwickeln Simula-67.

1969 Das ArpaNet verbindet 4 Rechner an UCLA, UCSB, SRI und der University of Utah.

1971 Der Intel 4004 ist der erste kommerzielle Mikroprozessor (2100 Transistoren, 4-Bit).

1972 B. Kernighan u. D. Ritchie entwickeln C.

1973 Goldberg, Kay u.a. entwickeln Smalltalk.

Zur Geschichte der Rechentechnik V

1974 Kahn und Cerf entwickeln TCP.

1977 Beginn der PC-Ära mit dem Apple II und dem Radio Shack RTS-80 .

1980 Motorola entwickelt MC 68000 (32-Bit).

1985 Intel bietet i80386 an.

ab 1990 T. Berners-Lee entwickelt Grundlagen des WWW.

1993 Mosaic: erster graphischer Web-Browser.

1995 Die Programmiersprache Java wird vorgestellt.

Bemerkungen

- Die obigen Ereignisse sind zum Zwecke der Illustration gewählt und nicht nach Bedeutung.
- Entsprechend der Geschichte der Rechentechnik gibt es auch eine Geschichte der
  - Speicherung und Wiedergabe von Information, sowie der
  - Nachrichtenübertragung.

Gesellschaftliches Umfeld

Die Ergebnisse und Produkte der Informatik dringen in zunehmendem Maße in die Gesellschaft ein und verändern sie:

- Wirtschaft
- Privatleben
- öffentliches Leben

Daraus resultiert vierfache Verantwortung für Informatiker:

1. Chancen nutzen
2. korrekte, verlässliche Systeme bauen
3. die "richtigen" Systeme bauen
4. den Vermittlungs-, Veränderungsprozess mittragen

Beispiele: Softwaresysteme

- Betriebssystem
- Fenstersystem
- Dateisystem
- Übersetzer
- Computerspiel, Anwendung
- Internet
- World Wide Web
- Informations- und Buchungssystem der Bahn
- Telekommunikationssystem (z.B. eines Herstellers)
- Eingebettete Softwaresysteme, z.B. Steuerungen für Motoren, Airbags, Roboter, Flugzeuge, Waschmaschinen, ...

Beobachtungen I

- Softwaresysteme sind nicht physisch:
  - lassen sich nicht anfassen
  - gehorchen nicht physikalischen Gesetzen
  - lassen sich nicht direkt betrachten
  - sind konstruiert / technisch, nicht natürlich

Beobachtungen II

- Softwaresysteme dienen sehr unterschiedlichen Zwecken
  - Plattform für andere SW-Systeme (Betriebssystem, Internet, ...)
  - Kommunikation mit Menschen (Computerspiel, Informationssysteme)
  - Steuerung von Apparaten/Maschinen (Airbagsteuerung, Autopilot)
  - ...

Beobachtungen III

- Softwaresysteme
  - bauen auf anderen Systemen auf (z.B. Computerspiel auf Betriebssystem, WWW auf dem Internet, Betriebssystem auf Rechnerhardware, ...)
  - haben Schnittstelle zur Umgebung (insbesondere Mensch-Maschine-Schnittstelle, Human Computer Interface HCI)
  - sind Teil komplexerer Systeme (z.B. Auto)

Beobachtungen IV

- Softwaresysteme haben sehr unterschiedliche Eigenschaften, zum Beispiel bzgl.
  - Größe:
    - gemessen in Programmzeilen
    - gemessen in Personenjahren / Investitionen
  - Terminierung:
    - terminierend (z.B. Übersetzer)
    - nichtterminierend (z.B. Telekomsystem)
  - Persistenz:
    - persistent: Daten werden zwischen aufeinander folgenen Ausführungen aufbewahrt

Beobachtungen V

- Interaktionsverhalten:
  - batch-orientierte, transformierende Systeme
  - interaktive, reaktive Systeme
- Verteilung:
  - lokal, auf einem Rechner
  - verteilt auf vielen gleichartigen Rechnern
  - verteilt auf vielen unterschiedlichen Rechnern
- Komplexität
- Qualitätseigenschaften:
  - Korrektheit, Stabilität
  - Benutzerfreundlichkeit
  - Wartbarkeit

Beobachtungen VI

- Softwaresysteme
  - spielen eine bedeutende Rolle für die wirtschaftliche Entwicklung einer modernen Gesellschaft:
    - Effizienz, Automatisierung
    - konkurrenzfähige Produkte
    - Kommunikationsinfrastruktur
    - Zugriff auf Wissen / Information
  - können die medizinische und soziale Versorgung verbessern
  - können einen Beitrag zum gesellschaftlichen, privaten und kulturellen Leben leisten

(Technische) Systeme

Ein **System** besteht aus Teilen (Komponenten, Subsystemen), die in geordneter Weise miteinander in Beziehung stehen.

Technische Systeme bestehen üblicherweise aus Bauelementen, die auf Grund ihrer Eigenschaften ein bestimmtes Verhalten zeigen.

Beispiele: Systeme

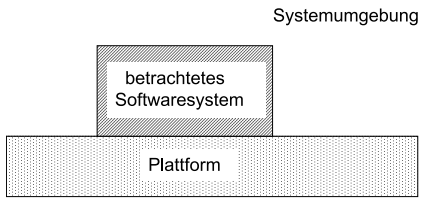
- Nichttechnische Systeme:
  - Gesellschafts-, Rechts-, Wirtschaftssystem
  - Ökosystem, Sonnensystem
  - Begriffssystem
- Teilweise technische Systeme:
  - Verkehrssystem
  - Bibliotheksverwaltungssystem
- Technische Systeme:
  - Auto, Kraftwerk, Telefonnetz
  - Rechensystem, Rechner

Softwaresystem

Ein **Softwaresystem** ist ein technisches System oder Subsystem, bei dem die Funktionalität mittels Software realisiert ist.

Softwaresysteme besitzen **Schnittstellen zur Umgebung**:

- Bedienschnittstellen
- weitere Ein-/Ausgabeschnittstellen
- Programmierschnittstellen



Jedes Softwaresystem läuft auf einer **Plattform** (Hardware- oder Softwareplattform).

Begriffserklärung: Software

Software im engeren Sinne umfasst

- Programme zur Beschreibung des Systemverhaltens
- Daten zur Beschreibung von Informationen

Um vom Rechner verarbeitet werden zu können, müssen die Beschreibungen in einer *formalen Syntax* abgefasst sein.

Software im weiteren Sinne umfasst neben Programmen und Daten auch Beschreibungen, die für die Erzeugung und Ausführung eines Softwaresystems nicht direkt notwendig sind, z.B.:

- Dokumentationen, Architekturbeschreibungen
- Daten für Anwendungs- und Testfälle
- Installations- und Wartungssoftware

Begriffserklärung: Programm

Ein **Programm** ist ein Text, der

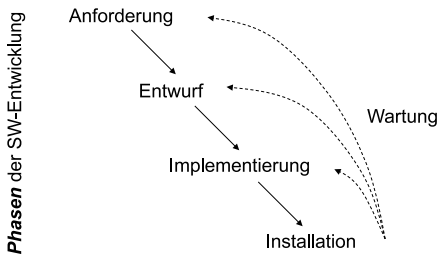
- in einer Programmiersprache verfasst ist und
- hinreichend vollständig ist, um auf einem Rechensystem ausgeführt zu werden.

Beispiele

- für Programme: Texte verfasst in Haskell, ML, C, Java, Scala, ...
- für Daten: Messwerte, Statistiken, Web-Seiten, Textdokumente, ...
- für Zusammenwachsen von Daten und Programmen:
  Web-Seiten mit eingebettetem Programmcode

Software-Entwicklung I

Die **Entwicklung eines Softwaresystems** umfasst die Schritte und Tätigkeiten von der Idee bis zur Implementierung und Installation des Systems sowie die Wartung.



Software-Entwicklung II

Betrachtet man den Ablauf der Schritte und Tätigkeiten in der Zeit, spricht man vom **Software-Entwicklungsprozess**.

Eine **SW-Entwicklungsmethode** beschreibt die Modelle, Prozesse, Sprachen und Werkzeuge zur Lösung einer Klasse von SW-Entwicklungsaufgaben.

Wie in anderen Disziplinen, hängt die Eignung einer SW-Entwicklungsmethode von den Anforderungen ab.

Phasen der SW-Entwicklung I

Ausgangspunkt der SW-Entwicklung sind die **Anforderungen** (engl. *requirements*):
Was soll das System leisten?

- *Funktionale* Anforderungen: Ein- und Ausgabeverhalten
- *Nicht-funktionale* Anforderungen: Effizienz, Portabilität, Bedienbarkeit, Stabilität, Zuverlässigkeit, Wartbarkeit, ...

Phasen der SW-Entwicklung II

Software-Entwurf (engl. *design*):

- Umsetzung der Anforderungen in ein Modell für das zu entwickelnde Softwaresystem
- Modell selbst, d.h. eine Beschreibung der Systemkomponenten und ihrer Beziehungen

Implementierung (engl. *implementation*):

- Verfeinerung des Entwurfs und Codierung, d.h. Formulierung mit Programmiersprachen (etc.)
- die resultierenden Programme

Installation (engl. *installation/deployment*):

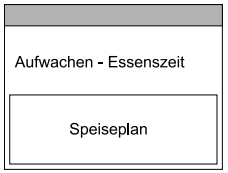
- Einrichten der Software auf den Plattformen, auf denen sie ablaufen soll
- das eingerichtete Software-System

Beispiel: Phasen der SW-Entwicklung I

- Anforderung an das zu entwickelnde System
  - Jeden Tag um 11:45 soll ein Bildschirmfenster aufgehen, das ans Mittagessen erinnert.
  - Auf Wunsch soll sich der Benutzer über den Speiseplan der Woche informieren können.
  - Das System soll auf der Linux-Plattformen laufen.

Beispiel: Phasen der SW-Entwicklung II

- Entwurf
  - Benutze den crontab-Mechanismus von Linux, um das zu entwickelnde System um 11:45 zu starten.
  - Benutze einen Web-Browser, um den Essensplan anzuzeigen (Annahme: Es gibt eine Web-Seite mit dem Essensplan)
  - Entwickle eine Komponente, die eine Meldung anzeigt und es erlaubt, per Mausklick den Browser zu starten. Name: LeckerWecker.
  - Die LeckerWecker-Komponente bekommt die Kommandozeile zum Aufruf des Browsers als Zeichenreihe übergeben.
  - Layout-Entwurf für das Fenster:



Beispiel: Phasen der SW-Entwicklung III

- Implementierung
  - Als Implementierungssprache benutzen wir Java.
  - Die Komponente wird als Klasse `LeckerWecker` implementiert und in einer Datei `LeckerWecker.java` abgelegt werden.
  - Übersetze das Programm.

```
import java.awt.*;
import java.awt.event.*;

public class LeckerWecker {
  public static void main( String[] a ){
    final String bkom = a[0];
    Label anzeigeLabel =
      new Label("Aufwachen - Essenszeit");
    Button b = new Button(„Speiseplan");
    b.addActionListener( ... );

    Frame  f = new BaseFrame();
    f.setLayout( new GridLayout(2,1) );
    f.add( anzeigeLabel );
    f.add( b );
    f.setVisible(true);
  }
}
```


Beispiel: Phasen der SW-Entwicklung IV

- Installation
  - Speichere die übersetzte Datei auf den Rechnern, auf denen das LeckerWecker-System laufen soll.
  - Trage den Aufruf von LeckerWecker in die Crontab ein. Dabei ist die richtige Zeit einzustellen und als Parameter der lokale Browser mit der URL zur Seite mit dem Essensplan anzugeben.

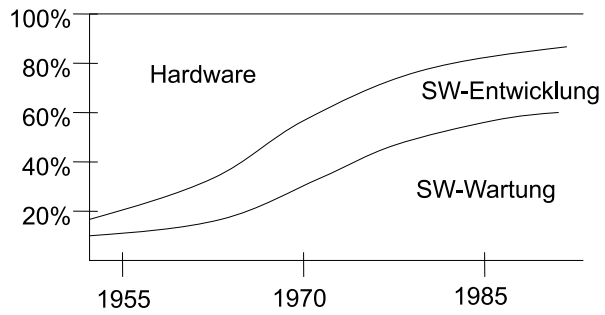
Wartung von Software

Software-Wartung umfasst die Pflege von Software-Systemen nach der Installation.
Gründe für Wartung:

- Anpassung an veränderte Umgebungsbedingungen (Hardware-, Software-Updates)
- Verbesserungen der Software, Fehlerkorrektur
- Anpassung an veränderte Anforderungen

Wartung basiert auf Verständnis der zu wartenden Software (Dokumentation wichtig!).

Aufteilung der Kosten von SW-Systemen



[Barry Boehm: Software Eng. IEEE Transactions on Computers, 25(12), Dec. 1976]

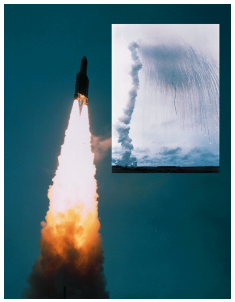
Aufteilung der Programmierer nach Aufgaben bei der SW-Entwicklung

Jahr	neue Projekte	Erweitern	Korrigieren	Gesamt
1950	90	3	7	100
1960	8500	500	1k	10k
1970	65k	15k	20k	100k
1980	1,2M	0,6M	0,2M	2M
1990	3M	3M	1M	7M
2000	4M	4,5M	1,5M	10M
2010	5M	7M	2M	14M
2020	7M	11M	3M	21M

[A. van Deursen, P. Klint und C. Verhoef: Research Issues in the Renovation of Legacy Systems, LNCS 1577, Springer 1999]

Beispiele: Fehlgeschlagene Projekte

- Stellwerk Hamburg-Altona: Fehler legt Betrieb für viele Stunden still und verursacht über Wochen erhebliche Behinderung (*Programmierfehler*).
- Ariane 5: Versagen des "Track Control System" resultiert in Selbstzerstörung der Rakete (*Integrationsfehler*).
- Flughafen Denver: Verspätete Lieferung der Software für das Gepäcksystem verzögerte die Flughafeneröffnung um 16 Monate (*Planungsfehler*).



Beobachtungen

- Kleine Fehler können verheerende Konsequenzen haben.
- Skalierbarkeit von SW-Entwicklungsmethoden:
  Methode für
  - kleine Projekte (< 10 kLOC, < 20 Entwickler)
    ist ggf. untauglich für
  - große Projekte (> 1 MLOC, > 100 Entwickler)?
- Die ingenieurmäßige Entwicklung von SW-Systemen hat auch Termin-, Budget- und Qualitätsanforderungen zu berücksichtigen.

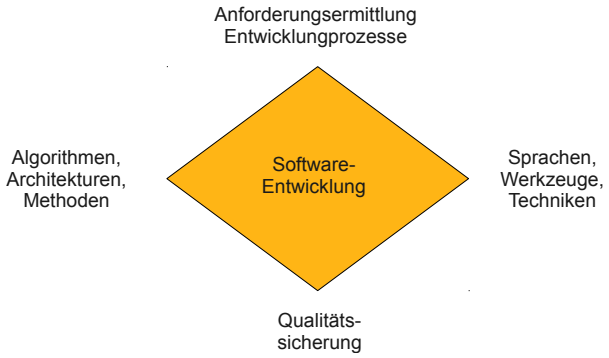
Problemfelder

- Anforderungsbeschreibung muss Kundenwunsch präzise widergeben (→ Validierung)?
- Entwurf
  - setzt Anforderungen um (→ Verifikation)
  - ist zeit- und termingerecht realisierbar
  - lässt sich effizient implementieren
- Implementierung und Installation
  - setzen Entwurf um (→ Verifikation)
  - bilden effizientes System

Was muss ein Software-Entwickler wissen bzw. können?

- Anforderungen und Aufgabenstellung verstehen (meist aus anderem Fachgebiet)
  → Domänenwissen
- Beschreibungsmittel und Werkzeuge detailliert beherrschen (Modellierung)
  → Abstraktionsvermögen
- Alternative Realisierungsmöglichkeiten kennen und bewerten können
- Schritte des Entwicklungsprozess kennen und durchführen können
- Wissen, was Korrektheit bei Entwurf, Implementierung und Installation bedeutet und wie man sie nachweist

Kompetenzraute: Software-Entwicklung



Zusammenfassung

- Begriffe Informatik und Softwaresysteme
- Entwicklung von Softwaresystemen

Ausblick:

- Beschreibungssprachen
- Syntax vs. Semantik
- Formale Grammatiken

Organisatorisches

Anmeldung zu den Übungsgruppen!