Script **generated by TTT**

Title: Einf_HF (22.06.2015)

Date: Mon Jun 22 14:14:43 CEST 2015

Duration: 88:13 min

Pages: 32

In diesem Kapitel werden einige Klassen von Algorithmen vorgestellt, insbesondere Suchverfahren und Sortierverfahren.

- Fragestellungen des Abschnitts:
  - Welche Möglichkeiten gibt es, Datenmengen im System darzustellen?
  - Welche Möglichkeiten gibt es, in Datenmengen zu suchen?
  - Welche Möglichkeiten gibt es, Datenmengen zu sortieren?
  - Was versteht man unter der Komplexität eines Algorithmus?

Algorithmus kann Aufwand erfordern, der "nicht vertretbar" ist. Bei algorithmischer Lösung eines Problems ist daher auch die Effizienz wesentlich.

Komplexität von Algorithmen

Ein Algorithmus ist umso effizienter, je geringer der Aufwand zu seiner Abarbeitung ist.

Aufwand bezieht sich auf bestimmte Ressourcen, z.B. Rechenzeit, Speicherplatz, Anzahl der Geräte

Je nach Ressource verschiedene Komplexitätsmaße. Wichtigste: Zeitkomplexität, Speicherplatzkomplexität.

Zu unterscheiden: Komplexität eines Algorithmus / eines Problems. Problem: zu erreichendes Ziel. Algorithmus: Vorgehen. Komplexität Problem = Komplexität effizientester bekannter Algorithmus.

Komplexitätsmaß für Algorithmus: Funktion abhängig von Größe der Eingabe. Misst Aufwand der Verarbeitung relativ zur zu verarbeitenden Information.

Beispiel

Liste Sortieren: Komplexitätsmaß abhängig von Anzahl der zu sortierenden Elemente; Brechen eines Schlüssels: Komplexitätsmaß meist abhängig von Schlüssellänge.

Algorithmen bewertet man relativ zu ihrer Komplexität.

Komplexität

Algorithmus kann Aufwand erfordern, der "nicht vertretbar" ist. Bei algorithmischer Lösung eines Problems ist daher auch die Effizienz wesentlich.

Komplexität von Algorithmen

Ein Algorithmus ist umso effizienter, je geringer der Aufwand zu seiner Abarbeitung ist.

Aufwand bezieht sich auf bestimmte Ressourcen, z.B. Rechenzeit, Speicherplatz, Anzahl der Geräte

Je nach Ressource verschiedene Komplexitätsmaße. Wichtigste: Zeitkomplexität, Speicherplatzkomplexität.

Zu unterscheiden: Komplexität eines Algorithmus / eines Problems. Problem: zu erreichendes Ziel. Algorithmus: Vorgehen. Komplexität Problem = Komplexität effizientester bekannter Algorithmus.

Komplexitätsmaß für Algorithmus: Funktion abhängig von Größe der Eingabe. Misst Aufwand der Verarbeitung relativ zur zu verarbeitenden Information.

Beispiel

Liste Sortieren: Komplexitätsmaß abhängig von Anzahl der zu sortierenden Elemente; Brechen eines Schlüssels: Komplexitätsmaß meist abhängig von Schlüssellänge.

Algorithmen bewertet man relativ zu ihrer Komplexität.

Komplexitätsklassen

Generated by Targeteam

Komplexitätsklassen

Relative Größenordnung (abzüglich konstanter Faktor) in Abhängigkeit von Zahl n der Wertelemente (bestimmt durch Kontext, z.B. Anzahl der Eingabeelemente, Anzahl der zu untersuchenden Datenelemente).

O steht hier für Ordnung: z.B. Wachstum der Laufzeit für einzelne Komplexitätsklassen, und zwar in Abhängigkeit der Anzahl n der Eingabedaten.

Definition

Sei f: N → N eine Funktion. Die Komplexitätsklasse O(f) ist definiert durch:

$O(f) := \{ t: N \rightarrow N \mid \exists\ c > 0, m \geq 0: \forall\ n > m: 0 \leq t(n) \leq c*f(n) \}$

O-Notation dient dazu, das asymptotische Wachstum einer Funktion abzuschätzen.

- Sei t(n) die Funktion zur Bestimmung der Laufzeit des Programms.
- mit wachsendem n gewinnt die höchste Potenz von n in t(n) an Bedeutung.

Übliche Komplexitätsklassen

Nicht-polynomiale Probleme gelten als "hart".

Übliche Komplexitätsklassen

Hauptsächlich verwendete Komplexitätsklassen

konstant O(1)

logarithmisch O(log n)

linear O(n)

O(n) ⇒ Komplexität ist z.B. 3n + 10;

überlinear O(n log n)

quadratisch $O(n^2)$

polynomial $O(n^k)$, k > 2

exponentiell $O(k^n)$, k >= 2

Grafische Darstellung

Einführung in die Informatik für andere Fachrichtungen

- Prof. J. Schlichter
  - Lehrstuhl für Angewandte Informatik / Kooperative Systeme

    Fakultät für Informatik, TU München

    E-Mail: schlichter@in.tum.de

    Tel.: 089-289 18654

    URL: http://www11.in.tum.de/

Software-Entwicklung

"Vorgehen bei der Entwicklung von Softwaresystemen". Vorgehensmodelle für die Entwicklung von Programmen, Modelle für Analyse und Entwurf.

- Fragestellungen des Abschnitts:
  - Welche Kategorien von Programmiersprachen gibt es ?

    interpretierte und übersetzte Sprachen

  - Wie kann man bei der Konzeption und der Realisierung eines Software-Programms geeignet vorgehen?

    Modellierung der verschiedenen Aspekte, z.B. Daten, Abläufe und Interaktion mit dem Benutzer.

Programmiersprachen

Software-Engineering

Bereitstellung von höheren Programmiersprachen, wie z.B. C, Java, die die Erstellung von Programmen erleichtern.

Programme in höherer Programmiersprache müssen in Maschinensprache transformiert werden.

Interpretierer und Übersetzer

Scriptsprachen

Auswahl einer Programmiersprache

"Interpretierer "(Interpreter): Programmiersystem, das Anweisungen schrittweise zergliedert und Schritte sofort ausführt. Jede Programmanweisung wird separat betrachtet und Maschinensprache-Unterprogramm für Realisierung der Anweisung aufgerufen.

Alternative: "Übersetzer"(Compiler): wandeln komplettes Programm in Maschinensprache um.

Vorteile von Interpretierern

- Einfacher zu realisieren
- Quellprogramm-bezogenes Testen

Nachteile von Interpretierern

- Ineffizient
- Fehler erst zur Laufzeit entdeckt

Beispiele für Interpretierer(sprachen)

- Basic
- Kommandosprachen von Betriebssystemen, z.B. Shell in DOS Eingabefenster
- Skriptsprachen (z.B. JavaScript)

Programmiersprachen

Bereitstellung von höheren Programmiersprachen, wie z.B. C, Java, die die Erstellung von Programmen erleichtern.

Programme in höherer Programmiersprache müssen in Maschinensprache transformiert werden.

Interpretierer und Übersetzer

Scriptsprachen

Auswahl einer Programmiersprache

Generated by Targeteam

Rechenblätter mit Zellen, die in Zeilen und Spalten organisiert sind. Zellen enthalten

- Daten verschiedener Sorten (Zahl, Währung, Datum, Text, ...)
- Formeln, die aus vordefinierten Funktionen zusammengesetzt sind (Summe, Mittelwert, Runden, ...)

variable Daten werden mit Hilfe von Verweisen auf andere Zellen in Formeln eingegeben, z.B. Summe (A1:A4), Summe(A1;B1;C1)

nur Funktionen mit einer Datenausgabeleitung, d.h. es gibt keinen Datenspeicher

⇒ nicht jeder Algorithmus kann in einer Tabellenkalkulation dargestellt werden.

Auswertung von Formeln bei Zellenänderung.

Programmierung von Kommandobuttons, Textfelder und Dialogboxen.

Programmiersprachen

Bereitstellung von höheren Programmiersprachen, wie z.B. C, Java, die die Erstellung von Programmen erleichtern.

Programme in höherer Programmiersprache müssen in Maschinensprache transformiert werden.

Interpretierer und Übersetzer

Scriptsprachen

Auswahl einer Programmiersprache

Auswahl einer Programmiersprache

Prinzipiell Programmiersprachen gleichwertig: alle Algorithmen formulierbar. Praktische Unterschiede helfen bei Auswahl für Entwicklungsprojekt.

Empfehlungen

wenn ...	... dann
einfache Programme, Anwendungserweiterungen	Basic, Visual Basic, Python, Perl
Datenbank-Anwendung und komplexe Programmlogik	C, C++ , Java
Datenbank-Anwendung und einfache Programmlogik	SQL, Reportgenerator
Technisch-wissenschaftliche Anwendung und (Datenbank oder komplexe E/A-Strukturen) und Portabilität	C, C++, Java
(System-Software oder PC-Anwendung) und Portabilität	C, C++, Java
Künstliche Intelligenz-Anwendung	Prolog, LISP
Internet-Anwendung und Portabilität	Java, PHP, Python

Programmiersprachen

Bereitstellung von höheren Programmiersprachen, wie z.B. C, Java, die die Erstellung von Programmen erleichtern.

Programme in höherer Programmiersprache müssen in Maschinensprache transformiert werden.

Interpretierer und Übersetzer

Scriptsprachen

Auswahl einer Programmiersprache

Generated by Targeteam

Software-Engineering

Softwareerstellung als Ingenieurdisziplin.

Software/Engineering - Definition des Ideals

Die Aufstellung und Befolgung guter Ingenieur-Grundsätze und Management-Praktiken, sowie die Entwicklung und Anwendung zweckdienlicher Methoden und Werkzeuge, mit dem Ziel, mit vorhersagbaren Mitteln, System- und Software-Produkte zu erstellen, die hohe, explizit vorgegebene Qualitätsansprüche erfüllen (nach A. Marco & J. Buxton, 1987)

Komplexität von Software-Projekten

Vorgehensmodelle

Strukturierte Programmierung

Modellierung

Modelle für Analyse und Entwurf

Maße für den Umfang von Software

Zahl der Quelltextzeilen der Programme, aus denen das Softwareprodukt besteht (LOC = Lines of Code).

Zeit, die benötigt wird, um eine Programm zu erstellen (Messung in Bearbeiter-Jahre (BJ)).

Klassifikation von Software-Projekten

Projektklasse	Quelltext-Zeilen (LOC)	Bearbeitungsaufwand (BJ)
sehr klein	1 - 1.000	0 - 0,2
klein	1.000 - 10.000	0,2 - 2
mittel	10.000 - 100.000	2 - 20
groß	100.000 - 1 Mio.	20 - 200
sehr groß	1 Mio - ...	200 - ...

Beispiele

Projektklasse	Quelltext-Zeilen (LOC)
Windows XP	ca. 40 Millionen
Windows Vista	ca. 50 Millionen
Windows 7	ca. 45 Millionen
Linux Kernel 2.6	ca 5.2 Millionen
Mac OS X 10	ca. 85 Millionen

Komplexität von Software-Projekten

sehr klein	1 - 1.000	0 - 0,2
klein	1.000 - 10.000	0,2 - 2
mittel	10.000 - 100.000	2 - 20
groß	100.000 - 1 Mio.	20 - 200
sehr groß	1 Mio - ...	200 - ...

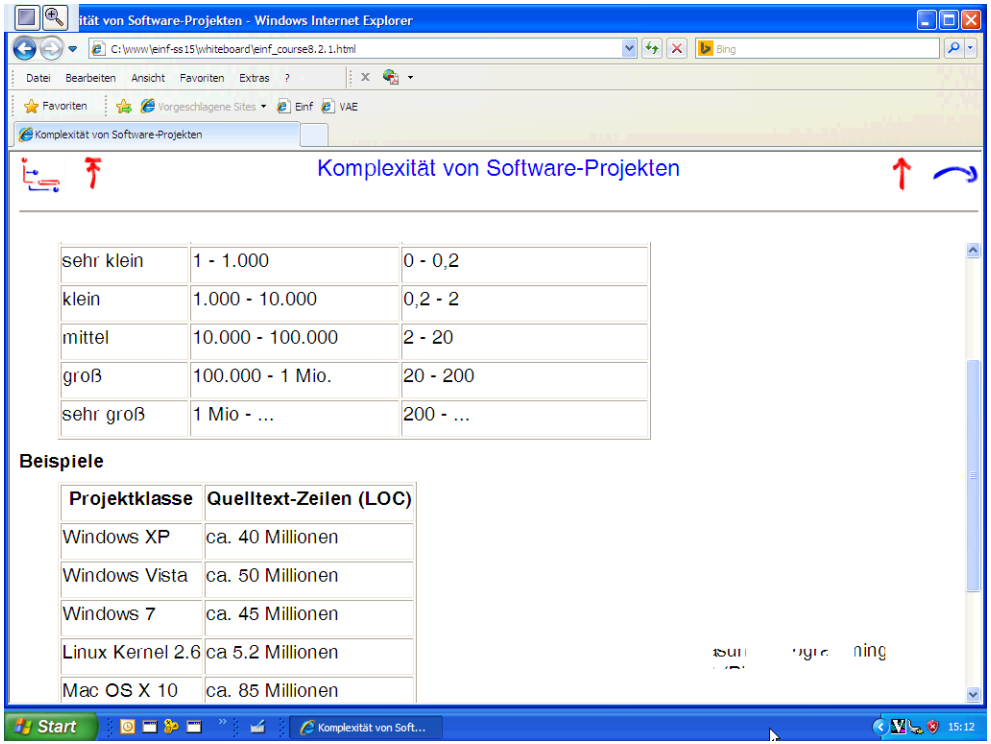
Projektklasse	Quelltext-Zeilen (LOC)
Windows XP	ca. 40 Millionen
Windows Vista	ca. 50 Millionen
Windows 7	ca. 45 Millionen
Linux Kernel 2.6	ca 5.2 Millionen
Mac OS X 10	ca. 85 Millionen

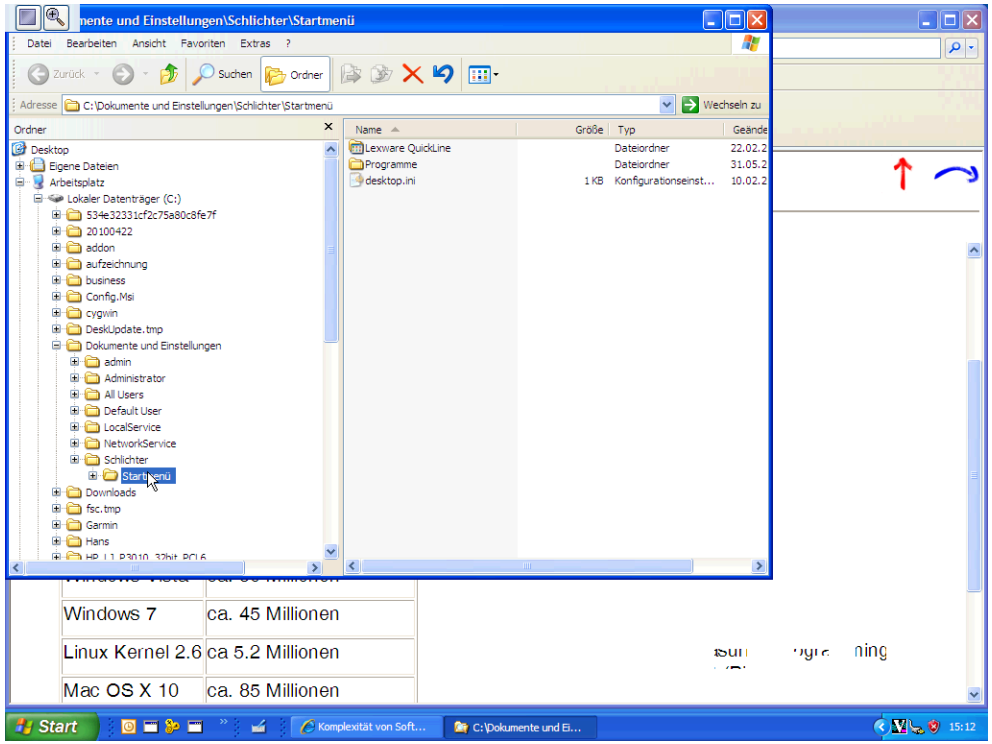
Nutzung von LOC, um Programmierfortschritt zu messen, ist umstritten:" Measuring programming progress by lines of code is like measuring aircraft building progress by weight (Bill Gates)."

Hauptanforderungen bei der Softwareentwicklung

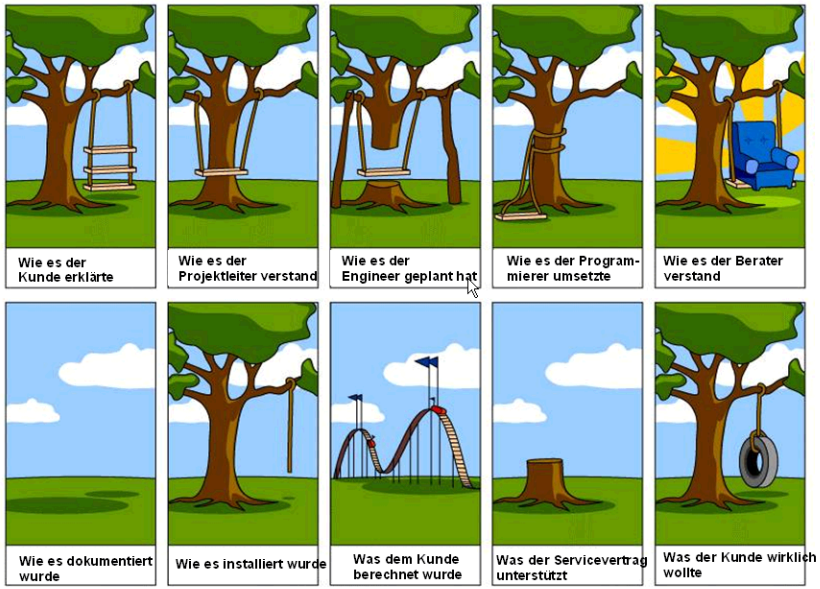
Ergebnis ist zuverlässig (fehlerfrei), verhält sich gemäß Anforderungen.

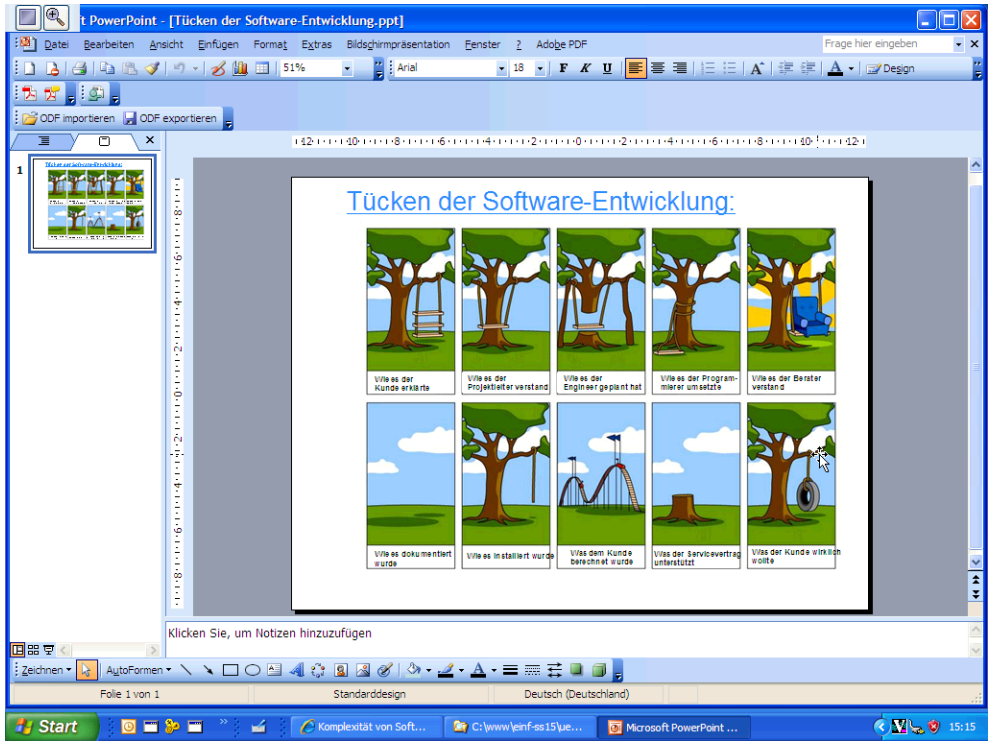
Kann später problemlos geändert werden.



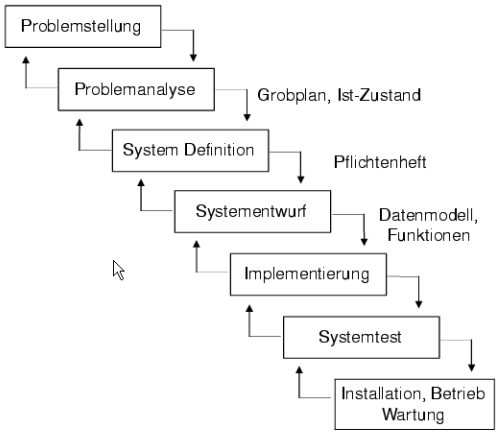


Tücken der Software-Entwicklung:





Graphische Darstellung



Validierungsprozess prüft Ergebnisse am Ende jeder Phase. Rücksprung zwischen Phasen nur wenn Fehler nicht in Phase erkannt; problematisch.

Wasserfall-Modelle

Entwicklungsprozess einteilen in aufeinander folgende Phasen. Jeweils festgelegt: Ausgangspunkt, Vorgaben, Tätigkeiten, Ergebnisse.

- Problemanalyse und Anforderungsdefinition
- System Definition: fachlicher Entwurf des Datenmodells und des Anwendungsmodells (Funktionen)
- Systementwurf: Festlegung der Struktur der zu entwickelnden Software
- Implementierung: Programmierung und Modultest
- System-Integration und Systemtest
- Installation, Betrieb und Weiterentwicklung

Graphische Darstellung

Vorgehensmodelle

Vorgehen im Projekt: Fortschrittskontrolle, Zusammenarbeit Entwickler untereinander und mit Management und Kunden.

Code and fix-Verfahren

Unsystematisch (Frühzeit der Programmiertechnik): Beginnt mit dem Schreiben von Code, endet mit Test und Zusammenfügen der Programmteile.

Wasserfall-Modelle

Prototyping und Spiralmodelle

Systematische Abfolge von Prototyp-Entwicklungen. Lineare Abfolge der Phasen Analyse, Design und Realisierung. Jeweils: Zielbestimmung, Bewertung der Alternativen, Prototypentwicklung, Verifikation.

Software-Engineering

Softwareerstellung als Ingenieurdisziplin.

Software/Engineering - Definition des Ideals

Die Aufstellung und Befolgung guter Ingenieur-Grundsätze und Management-Praktiken, sowie die Entwicklung und Anwendung zweckdienlicher Methoden und Werkzeuge, mit dem Ziel, mit vorhersagbaren Mitteln, System- und Software-Produkte zu erstellen, die hohe, explizit vorgegebene Qualitätsansprüche erfüllen (nach A. Marco & J. Buxton, 1987)

Komplexität von Software-Projekten

Vorgehensmodelle

Strukturierte Programmierung

Modellierung

Modelle für Analyse und Entwurf

Strukturierte Programmierung

"Top-Down-Entwurf" Schrittweise Verfeinerung zerlegt komplexes Problem in Teilprobleme. Rekursiv fortsetzen, bis Teilprobleme überschaubar.

Eigenschaften

Generelle Vorgehensweise bei Entwurf von Softwaresystemen; auch: "divide-and-conquer"

zunächst gesamtes Programm als eine Operation, nach und nach aufteilen in Teilfunktionen.

das Problem wird solange zerlegt, bis es beherrschbar wird;

späte Festlegung der Datendarstellung.

Vorteile

Schnittstellenprobleme (Festlegung der Parameter von Funktionen/Methoden) zuerst gelöst. Interne Realisierung interessiert noch nicht.

vermeidet widersprechende Schnittstellendefinitionen

erzwingt klärende Strukturierung

Wozu Modellierung?

Softwarefehler oder Fehlplanungen

1992: Rettungsleitstelle in London fällt 2-mal komplett aus: Schaden ca. 9 Mio Euro, mehrere Todesfälle

1993: Das Taurus-Projekt an der Londoner Börse (automatische Transaktionsabwicklung) wird nach 5 Jahren Laufzeit wieder eingestellt; Verlust 450 Mio Pfund

1996: Ariane 5 muss wegen plötzlichen Neigens 39sec nach dem Start gesprengt werden. Verlust der Sonnensatelliten (850 Mio DM).

2002: Ariane 5 gerät außer Kontrolle und muss in 96 km Höhe mitsamt zweier Satelliten gesprengt werden (600 Mio. Euro)

2003: Toll Collect konnte wegen Unterschätzung der Komplexität der notwendigen Software nicht wie geplant in Betrieb gehen; Verlust mehr als 1 Milliarde Euro.

Modellierung zwingt zu sauberer Planung des Systems.

Modellierung vor Programmierung

Modellierung vor Programmierung

Modellierung dient zur Strukturierung komplexer Systeme

Modelle strukturieren Systeme unabhängig von speziellen (zufälligen) Rahmenbedingungen der Implementierungsplattform

durch Abstraktion Konzentration auf relevanten Teile; Ausblenden von Details

intuitive Darstellung ermöglicht Lösung komplexer Probleme

kompakte Beschreibung des Systems; 5 - 10 Diagramme statt 20 Seiten Text

Übersichtlichkeit und Verständlichkeit erleichtern Realisierung, Wartung und Kommunikation über das System.

Was ist Modellierung in der Informatik

Definition: Ein **Modell** ist eine abstrahierte Beschreibung eines realen oder geplanten Systems, welche die für eine bestimmte Zielsetzung wesentlichen Eigenschaften des Systems wiedergibt.

Informatische Modellierung besteht aus

Abgrenzen: Identifikation der Grenzen.

Abstrahieren: Weglassen von nicht oder wenig bedeutsamen Details, Sonderfällen, speziellen Ausprägungen.

Idealisieren: Korrigieren kleiner Abweichungen von idealen Eigenschaften in Richtung einer leichteren Beschreibung.

Beschreiben: Anwendung spezieller Techniken zur Darstellung der wesentlichen Eigenschaften des Systems.

Beispiel: Bankwesen

Beispiel: Bankwesen

Aufgabe: Gesucht sind die Adressen und Betreuer von Kunden, deren Kredit den Betrag von 10.000 Euro übersteigt.

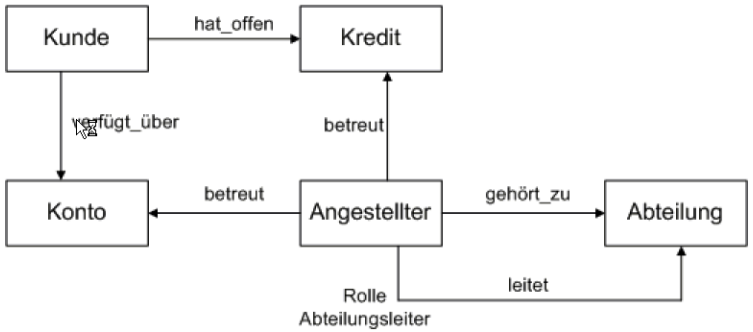
Abgrenzen: EDV Ausstattung, Werbeaktionen werden ignoriert

Betrachtung der Vorgänge zwischen Kunden und Betreuer

Abstrahieren: Erscheinungsbild, Alter des Betreuers sind nicht relevant

wichtig sind seine Abteilung, seine Kunden

Beschreiben: durch Diagramm



Realisierung: mit Hilfe von Tabellen in einer Datenbank; Abruf von Information

SELECT Kunde.Name FROM Kunde, Kredit

Modellierung

Die Erstellung von geeigneten Modellen vor der Realisierung des Softwaresystems ist ein zentraler Aspekt der Softwareentwicklung

Wozu Modellierung?

Was ist Modellierung in der Informatik

Softwareerstellung als Ingenieurdisziplin.

Software/Engineering - Definition des Ideals

> Die Aufstellung und Befolgung guter Ingenieur-Grundsätze und Management-Praktiken, sowie die Entwicklung und Anwendung zweckdienlicher Methoden und Werkzeuge, mit dem Ziel, mit vorhersagbaren Mitteln, System- und Software-Produkte zu erstellen, die hohe, explizit vorgegebene Qualitätsansprüche erfüllen (nach A. Marco & J. Buxton, 1987)