Einführung in die Praktische Informatik

WS 09/10

Prof. Dr. Michael Gertz

Christian Sengstock

Institut für Informatik
Neuenheimer Feld 348
69120 Heidelberg
http://dbs.ifi.uni-heidelberg.de
sengstock@informatik.uni-heidelberg.de
gertz@informatik.uni-heidelberg.de

RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

Kapitel 1: Einführung

- 1.1 Was ist Informatik?
  - Definition, Historie und Bedeutung
- 1.2 Disziplinen der Informatik
  - Teilgebiete, Beziehung zu anderen Wissenschaften
- 1.3 Aufgaben in der Informatik
  - Mehr als nur Programmieren…
- 1.4 Grundkonzepte
  - Daten, Algorithmus, Datenstruktur, Prozess, Maschine, Modell (siehe hierzu auch Skript 1.5-1.10)

Was ist Informatik ?

„Informatik ist die Wissenschaft von der systematischen Verarbeitung von Informationen, insbesondere der automatischen Verarbeitung mit Hilfe von Rechenanlagen.“ (Wikipedia)

„Wissenschaft von der systematischen Darstellung, Speicherung, Verarbeitung und Übertragung von Informationen, besonders der automatischen Verarbeitung mithilfe von Computern.“ (Schülerduden)

Was ist Informatik ?

- Wort „Informatik“ wurde 1962 von dem Franzosen Phillipe Dreyfus vorgeschlagen („Informatique“)
- Wortschöpfung aus
  - „Information“ und „Elektrotechnik“
  - „Information“ und „Mathematik“
- Klar sind die Wurzeln in der Mathematik, Physik und Elektrotechnik.
- Im englisch-amerik. Sprachraum: „Computer Science“ (= „Computerwissenschaften“) oder „Information Science“.

Was ist Informatik ?

„Computer Science is no more about computers than astronomy is about telescopes.“ (Edsger Wybe Dijkstra, niederländischer Informatiker, Wegbereiter der Programmierung)

- Der Computer dient als „universelle Rechenmaschine“.
- Durch Programmierung können Computer für verschiedenste Aufgaben angepasst werden.
- Informatik konzentriert sich darauf, die Lösung zu einem Problem so zu formulieren, so dass es für einen Computer ausführbar ist.

Historisches

(Mechanische) Maschinen, mit deren Hilfe mathematische Berechnungen ausgeführt werden können.

- 1632, Wilhelm Schickard, Rechenmaschinen, um astronomische Rechnungen zu erleichtern (+ und – für bis zu 6-stellige Zahlen)

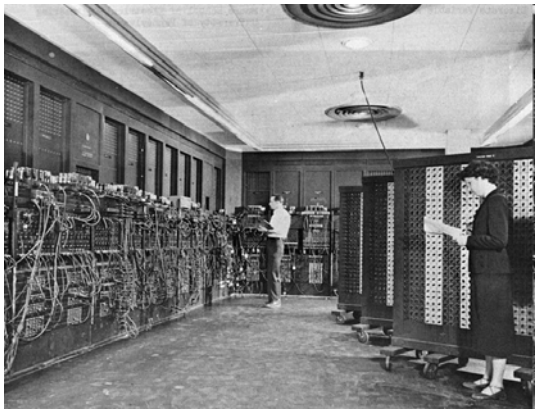
- 1642, Blaise Pascal, „Pascaline“ (Addition und Subtraktion)

- 1673, Gottfried Wilhelm Leibniz, Rechenmaschine für alle 4 Grundrechenarten

Historisches (2)

- 1838, Charles Babbage, Steuerung von Rechenoperationen mittels Lochkarten; weiterentwickelt von Hollerith u.a. zur Auswertung einer Volkszählung in den U.S.A.

- 1941, Konrad Zuse, Z3 verfügt als erster Computer über Befehls/Datenspeicher und Ein/Ausgabegerät
- 1944, Howard Aiken, Mark I erster Computer der U.S.A.
- 1946, Röhrenrechner ENIAC
- 1948, IBM beginnt Entwicklung von Computern



Bedeutung der Informatik

„Die Informatik zusammen mit der Informationstechnik bildet das zentrale Innovationsgebiet des 21. Jahrhunderts: Sie wird Lebensweise, Ausbildung, Arbeit und Freizeit verändern, indem sie die Methoden des Geschäftslebens, der Forschung und der Kommunikation revolutioniert. Die Allgegenwärtigkeit der aus der Verschmelzung von Rechnern, Telefon und Medien entstehenden multimedialen Kommunikationsplattformen befreit von Ortsbindungen, ermöglicht kontinentübergreifende Kooperationen und macht neueste Erkenntnisse unmittelbar weltweit verfügbar." (Schmid, Broy 2000)

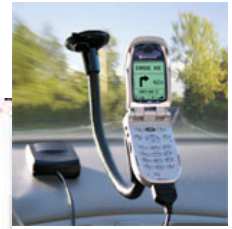
Bedeutung der Informatik (2)

- Informatik ist allgegenwärtig
  - Kommunikation
  - Gesundheit
  - Entertainment
  - Mobilität
  - Sicherheit
  - Wohnen
  - Sport
  - Kultur
  - …













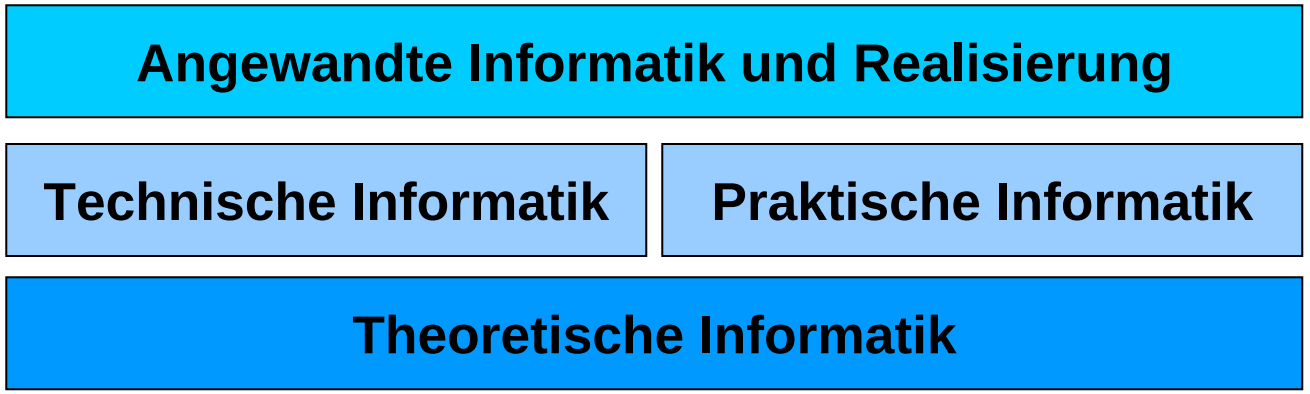



Bedeutung der Informatik (3)

- Software als Wirtschaftsgut
- Komplexität von Prozessoren verdoppelt sich alle 18 Monate (Moore‘s law)
- Beispiel: BMW 7er Serie
  - 96 Prozessoren (!!!), mehrere Kilometer Kabel
  - 10 Millionen Lines of Code (im Jahr 2003)
  - 100 Millionen Lines of Code in 6 Jahren
- ADAC: 50% der Pannen in 2002 waren E-Fehler

Disziplinen in der Informatik

- Hauptsäulen: Theoretische Informatik, Praktische Informatik und Technische Informatik

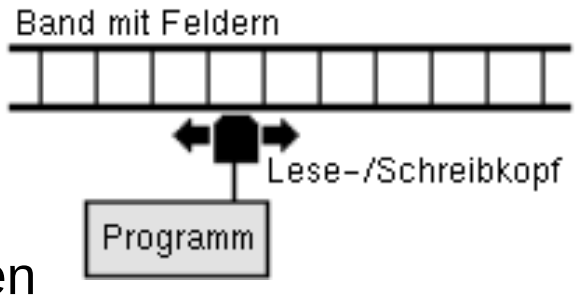


- Verwandte Disziplinen
  - Didaktik der Informatik
  - Künstliche Intelligenz
  - Informatik und Gesellschaft

Disziplinen in der Informatik (2)

- Theoretische Informatik
  - Mathematische Formalisierung von Problemen
  - Leistungsfähigkeit von Algorithmen
  - Grenzen des Computers beim Lösen von Problemen
  - Automatentheorie und formale Sprachen: gedachte Maschinen, die sich nach bestimmten Regeln, dem Programm, verhalten
  - Berechenbarkeitstheorie: welche Probleme sind mit welchen gedachten Maschinen lösbar
  - Turingmaschine

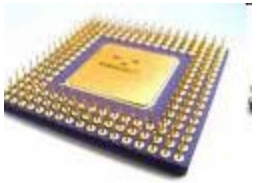
    Von dem britischen Mathematiker Alan Turing 1936 entwickeltes Modell, um eine Klasse von berechenbaren Funktionen zu bilden



Disziplinen in der Informatik (3)

- Praktische Informatik
  - Lösung von konkreten Problemen
  - Entwicklung von Computerprogrammen
  - Algorithmen spielen wichtige Rolle
  - Systematische Erstellung von Software
  - Werkzeuge zur Softwareentwicklung (z.B. Compiler, die eine Computersprache in eine andere übersetzen)
  - Betriebssysteme
  - Datenbanksysteme
  - Software Engineering
  - Computergraphik und Visualisierung

Disziplinen in der Informatik (4)



- Technische Informatik
  - Hardwareseitige Grundlagen der Informatik
  - Mikroprozessortechnik
  - Rechnerarchitekturen (Konzepte für den Bau von Computern)

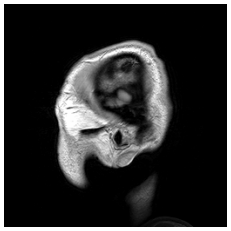
  - Rechnerkommunikation (technische Grundlage des Internets, inklusive entsprechender Softwarekomponenten)

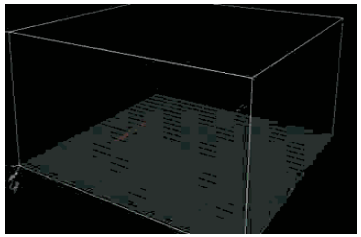
  - Verteilte Systeme (Zusammenschluss von Computern, Cluster, Grid-Computing, Middleware)

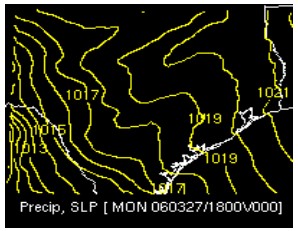
  - High-Performance Computing

Disziplinen in der Informatik (5)

- Informatik als Experimentalwissenschaft
  - „Informatik ermöglicht das Experimentieren in einem virtuellen Labor, das auf Modellierung und Simulation beruht, auf der Formalisierung des Untersuchungsraums und dem Durchrechnen von Modellen.“ (GI 2006)
  - Scientific Computing: Kombination von Methoden der Mathematik und Informatik mit einer Anwendungswissenschaft.





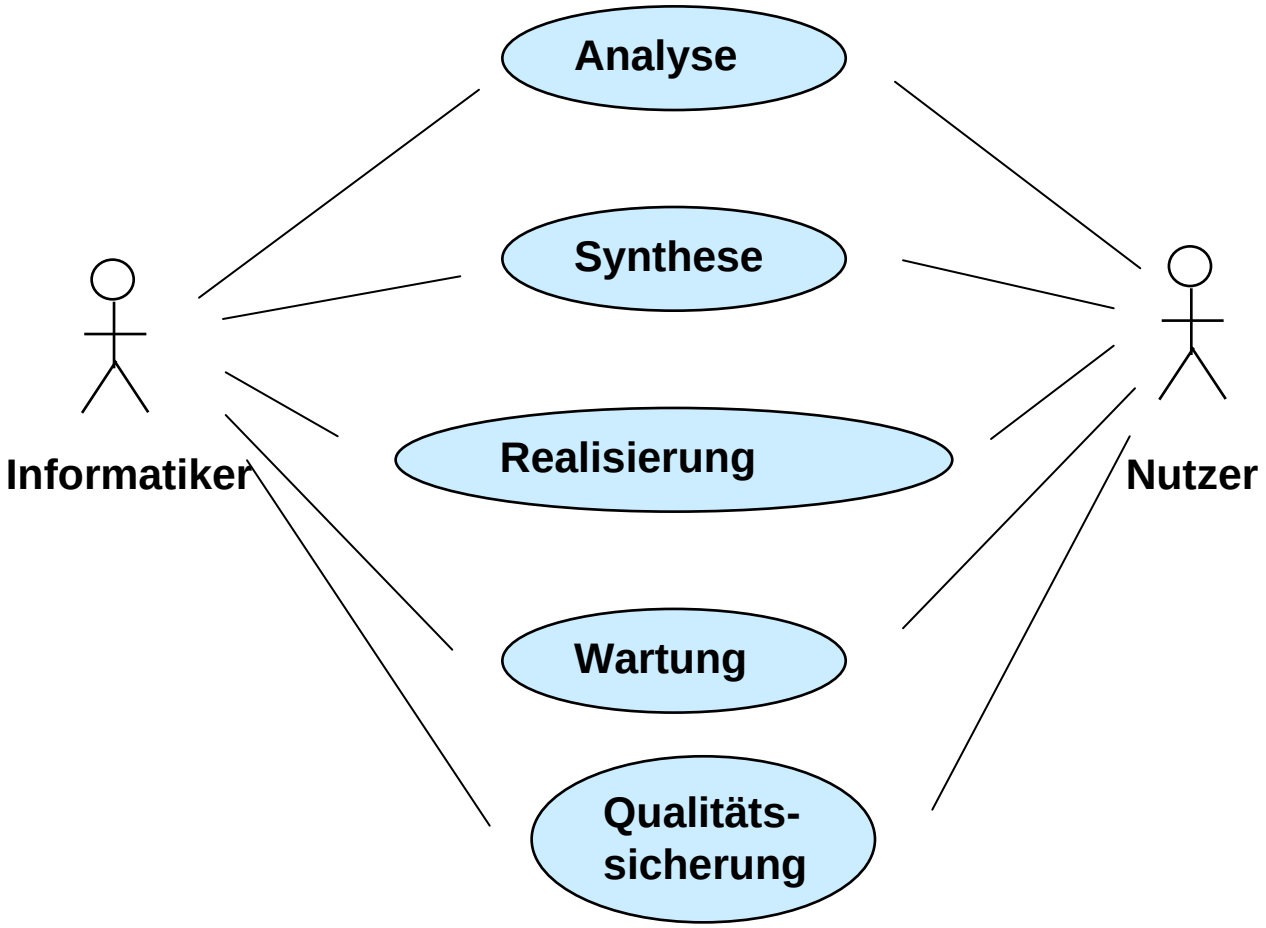


  - Biologie, Klimaforschung, Medizin, Geologie, Physik, Wirtschaft…. „X-Informatik“ (Bioinformatik, Geoinformatik)

Aufgaben von InformatikerInnen (1)

- Entwurf, Realisierung (~Implementierung) und Wartung von Informatiksystemen, die vorgegebene Probleme lösen.
- Informatiksystem: Elemente heißen Komponenten und bestehen aus Hardware und Software.
- Beispiele:
  - Interaktive Systeme, bestehend aus Daten/Information/ Wissensaufnahme und Bereitstellung (z.B. Abwicklung aller Geschäftsprozesse, inkl. Buchhaltung, Logistik, Personalwesen → SAP)
  - Eingebettete Systeme (z.B. Gerätesteuerung)
  - Kommunikationssysteme (z.B. Handy)
  - Prozessüberwachungssysteme (z.B. Verkehrsfluss-kontrolle)

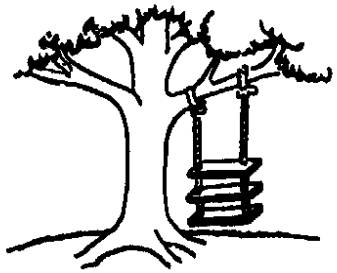
Aufgaben von InformatikerInnen (2)



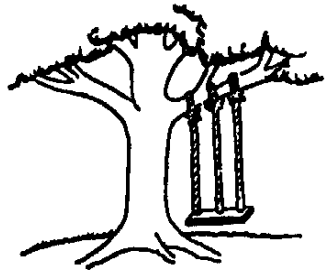
1. Die Analyse

- Das Verständnis von Problemen aus der Realität
- Realität ist dabei ein
  - soziales System
  - natürliches System
  - künstliches System
- Achtung: InformatikerInnen bearbeiten alle Systeme, haben aber immer künstliches System als Lösung im Kopf
- Verwendung von Modellen
  - Z.B. Entity – Relationship – Diagramm
- Ergebnis: Anforderungsspezifikation

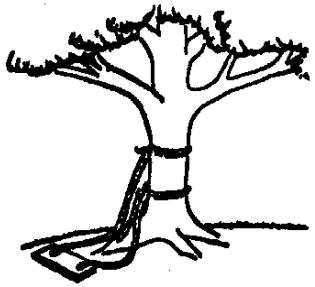
Analyse: Gemeinsames Verständnis nötig!



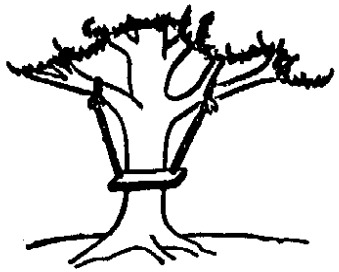
As proposed by the project sponsor



As specified in the project request

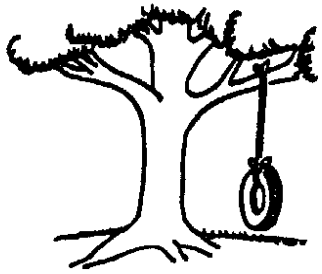


As designed by the senior analyst



As produced by the programmers

As installed at the user's site



What the user wanted

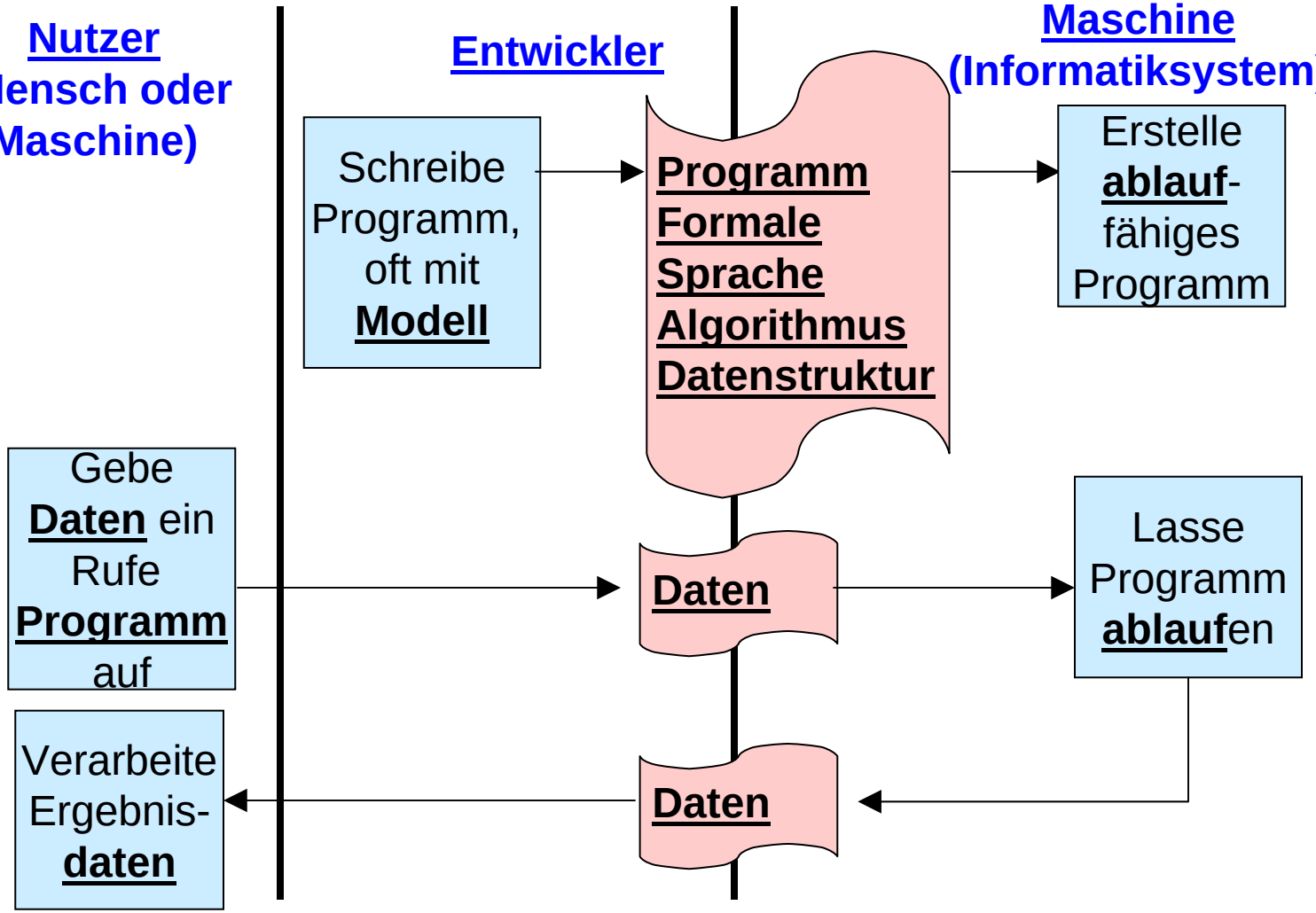
2. Die Synthese

- Die systematische Entwicklung einer Lösung des Problems (umgesetzt durch Informatiksystem)
  - Unter Verwendung existierender / neuer Hardware- und Softwarekomponenten
- Wahl der Technologie
- Komponenten: Benutzungsschnittstelle, Datenbasis, Anmeldungs-, Auswertungskomponente
- Feinentwurf
  - Schnittstellen der Komponenten
  - Funktionen und Datenstrukturen, z.B. Sortieralgorithmen, Listen oder Bäume
  - Verwendet Modelle (in der Vorlesung: Klassendiagramm und Sequenzdiagramm)
- Ergebnis: Entwurfsspezifikation

3. Die Realisierung

- Die Konstruktion der Lösung
  - In Form eines Informatiksystems
- Wahl der Programmiersprache
- Programmieren: Editieren – Kompilieren / Linken – Ausführen - Fehlerbehebung
- Ergebnis: Software (Code, Daten und Dokumentation)
- Entwicklung: Der Weg von der Analyse über die Synthese und Realisierung zum Informatiksystem

3. Die Realisierung – Programmierung



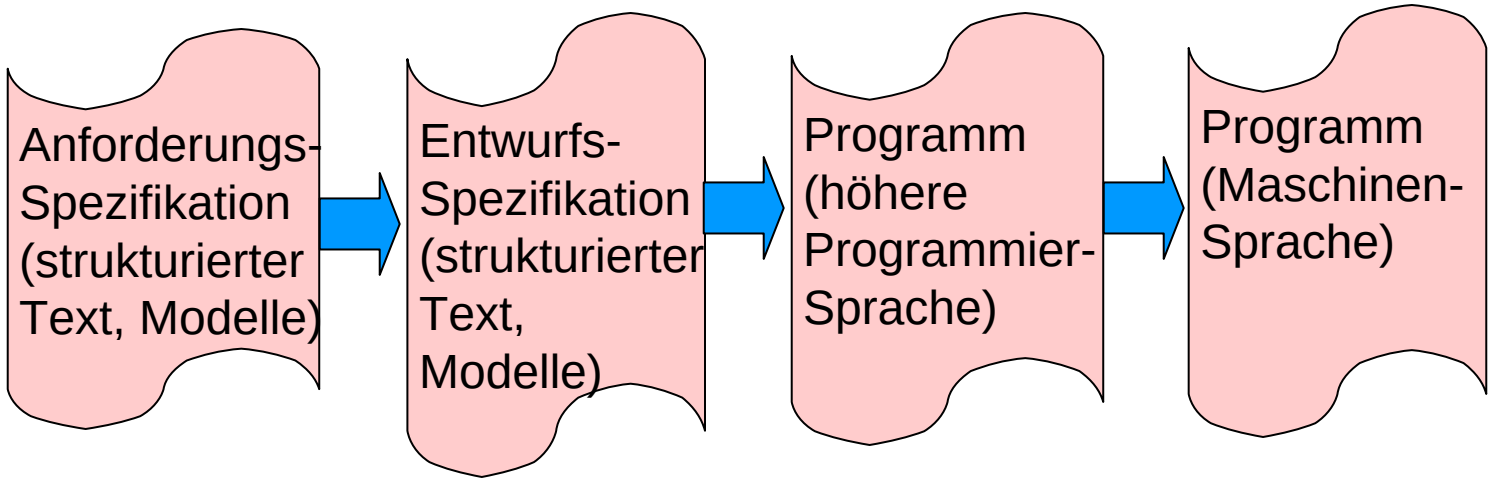
3. Die Realisierung – Programmierung (2)

- Editor: Erstellung des Programmtextes (Quelltext)
- Compiler: übersetzt menschenlesbares Programm in Maschinensprache
- Linker: Bindet Programmteile zusammen
- Weitere Möglichkeiten: Interpreter, virtuelle Maschine

- Integrierte Entwicklungsumgebung (IDE)
  - Integriert Editor, Compiler und Linker und Ausführung
  - Verwaltet Menge von Programmen
  - Editor: kann einfache Syntaxfehler erkennen

Formalisierung ⇔ Programmierung

- Formalisierung ist ein wichtiger Bestandteil der Entwicklung
- Maschine (Computer) „versteht nur“ künstliche Sprache (Maschinensprache, binär)
- EntwicklerInnen müssen ihr Lösungsverständnis in eine formale Sprache überführen



[Brügge03]

4.+5. Wartung und Qualitätssicherung

- Wartung: Die Bereitstellung und Betreuung des Informatiksystems über die gesamte Lebenszeit
  - Anpassung an neue Technologien
  - Anpassung an neue organisatorische Prozesse (z.B. Vorgaben aus dem Ministerium)

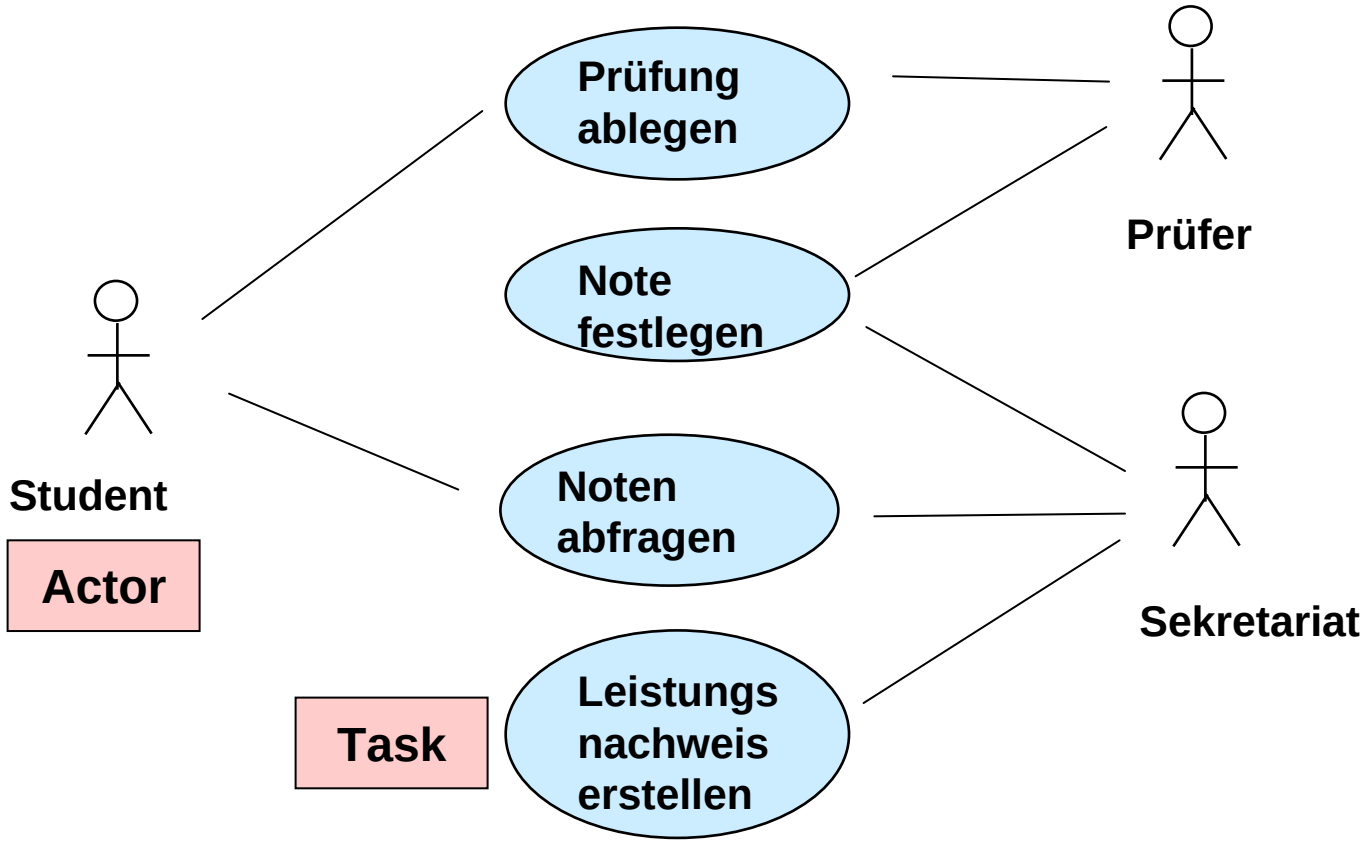
- Qualitätssicherung: Sicherstellung der Qualität während Entwicklung und Wartung
  - Validierung: Rückkopplung mit dem Benutzer
  - Verifikation: Überprüfung der Software gegen die Anforderungsspezifikation
    - Statische Analyse: Inspektion, Metriken, Beweis
    - Dynamische Analyse: Testen

Beispiel einer Analyse

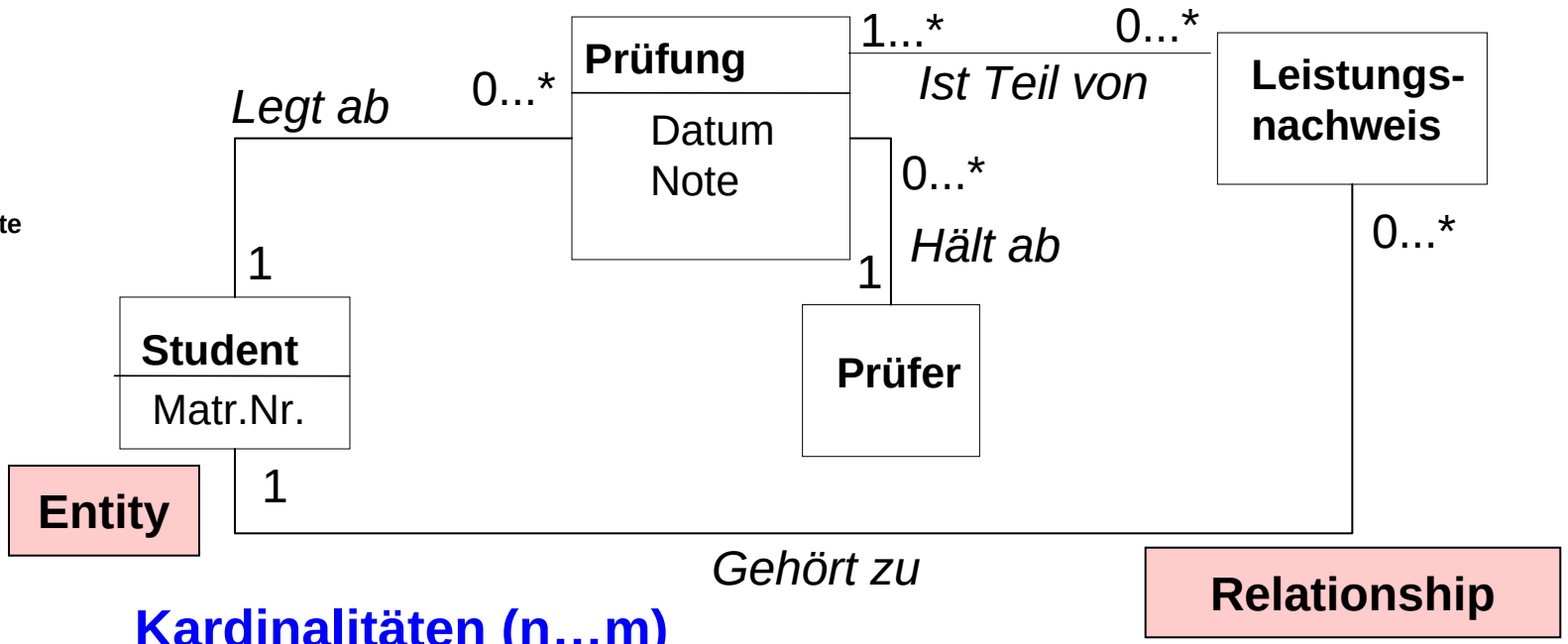
- Problem:
  - Bachelorstudierende brauchen viele Leistungs-nachweise; bisherige Erstellung und Verwaltung von Scheinen im Sekretariat passt nicht mehr
- Analyse:
  - Wer ist beteiligt? (z.B. Studierende, PrüferInnen, Prüfungsausschuss, Prüfungsamt, Studiensekretariat)
  - Welche Aufgaben haben die Beteiligten, welche Informationen erzeugen / benötigen sie wann? (z.B. Prüfungsnote vergeben, Überblick am Semesterende, Zeugniserstellung)
  - Wie kann ein Informatiksystem helfen?
    - Z.B. Übersicht für Studierende
      - Eingabe von Studierendendaten und Prüfungsnoten
  - Welche Qualitätsvorgaben, z.B. Performanz, Sicherheit?

Analyse – Aufgabenbeschreibung

AT-Modell



Entity Relationship (ER) Modell



Kardinalitäten (n…m)

n, m beliebige Zahlenwerte >= 0; * ist beliebiger Wert >= 0

Bsp: Eine Prüfung wird von genau einem Studenten abgelegt (1)

Ein Student kann zwischen 0 und beliebig viele (*) Prüfungen ablegen

Grundkonzepte

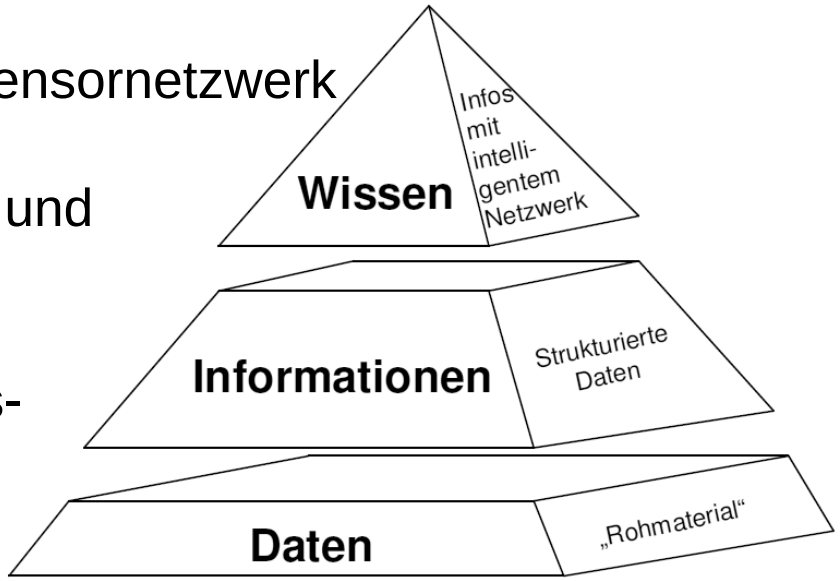
- Daten, Informationen, Wissen
- Algorithmus
- Datenstruktur
- Prozess
- Modell
- Maschine

Daten, Information, Wissen

Daten sind Gebilde aus Zeichen oder kontinuierliche Funktionen, die aufgrund bekannter oder unterstellter Abmachungen Informationen darstellen, vorrangig zum Zweck der Verarbeitung und als deren Ergebnis.

Bsp.: Daten aus einem Sensornetzwerk werden „interpretiert“ und entsprechend verarbeitet und analysiert.

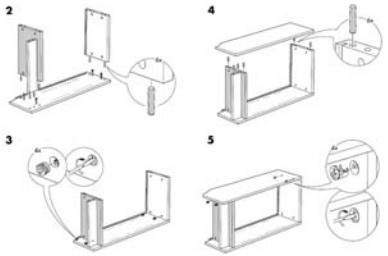
Wissen ist mit Erfahrungskontext „getränkte“ Information.



Wissenspyramide

Algorithmus

- Algorithmus ist eine genau definierte Handlungsvorschrift zur Lösung eines Problems oder einer Schar von Problemen in endlich vielen Schritten.
- Bsp.: Bauanleitungen, Kochrezepte, …



- Sichtweisen:
  - Funktional: welche Eingabe(n) und Ausgabe(n)
  - Operational: welche Schritte beinhaltet der Algorithmus (Steuerung des Kontrollfluss, Daten, elementare Schritte)

Algorithmen (2)

- Für Algorithmen gibt es unterschiedliche formale Repräsentationsformen (umgangssprachlich, …, Code in einer Programmiersprache)
- Weitere Beispiele:
  - Gegeben *n* Zahlen, gib die kleinste Zahl aus.
  - Gegeben *n* Zahlen, gib die Zahlen sortiert aus (→ Sortieralgorithmen werden später betrachtet).
  - Gegeben eine Menge von Web-Dokumenten und eine Anfrage *q*, gebe die Top-20 Dokumente aus, die hinsichtlich *q* am relevantesten sind (→ Suchmaschinen)
- Zu einer Problemklasse gibt es häufig verschiedene Algorithmen, mit verschiedenen Eigenschaften.

Eigenschaften von Algorithmen

- Jeder Algorithmus benötigt Zeit und Speicher, um ein Problem zu lösen (→ Effizienz, Themen im Rahmen der theoretischen Informatik, Komplexitätsanalyse)
- Deterministische Algorithmen: in jedem Schritt ist bekannt, welcher Schritt als nächster ausgeführt wird.
- Determinierte Algorithmen: bei jeder Ausführung mit gleicher Eingabe erhält man das gleiche Ergebnis.
- Terminierende Algorithmen: Der Algorithmus stoppt für jede zulässige Eingabe nach endlicher Zeit.

Algorithmus—Beispiel

- Problemstellung: zu zwei natürlichen Zahlen *a* und *b* soll der größte gemeinsame Teiler ggT(*a,b*) bestimmt werden.

ggT(a,b)

1. solange b <> 0
2. setze h = a mod b /* = entspricht einer Zuweisung
3. setze a = b
4. setze b = h
5. gebe a aus

- Eingabe: a = 1071 und b = 1029
  h = 1071 mod 1029 = 42
  a = 1029, b = 42
  h = 1029 mod 42 = 21
  a = 42, b = 21
  h = 42 mod 21 = 0
  a = 21, b = 0

Algorithmus—Beispiel (2)

- Alternativer Algorithmus, rekursiv, bei dem das Ergebnis durch Anwendung des gleichen Algorithmus auf veränderte Eingabe erreicht wird.

ggT(a,b)

1. wenn b = 0 gebe a aus
2. sonst gebe ggT(b, a mod b) aus

- Eingabe: a = 1071 und b = 1029

  b <> 0 → ggT(1029, 42)

  b <> 0 → ggT(42, 21)

  b <> 0 → ggT(21, 0) → gebe 21 aus

Datenstruktur

- Eine Datenstruktur ist ein (mathematisches) Objekt zur Speicherung von Daten; Struktur, da Daten auf bestimmte Art und Weise verknüpft sind, um Zugriff auf Daten und deren Verwaltung zu ermöglichen.
- Ein Algorithmus verarbeitet Ein- und Ausgabedaten und speichert diese in internen Datenstrukturen.
- Diese Strukturen sind notiert in der Programmiersprache.
- Typische (lesbare) Datenstrukturen:
  - Tabellen, Diagramme, Listen

Grundlegende Datenstrukturen

- Array (Feld): Daten eines einheitlichen Datentyps geordnet im Speicher eines Computers abgelegt, so dass ein Zugriff auf die Daten über einen Index möglich wird.

12	1	14	8	10	1	19	11

- Liste: Datenstruktur zur Speicherung von beliebig vielen Objekten gleichen Typs.

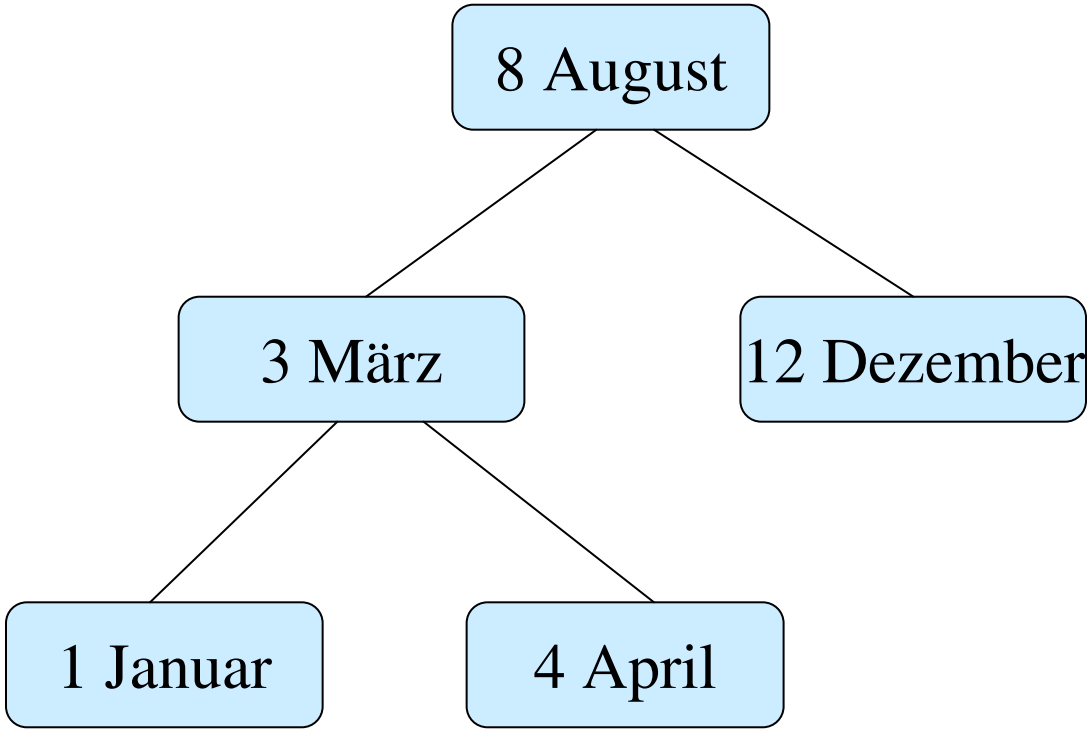
  HA→ DE→GE→FA→DD

- Record (Tupel): Liste von elementaren Datentypen.

⟨**Mustermann, 1.10.2007, Heidelberg, 1041**⟩

Grundlegende Datenstrukturen (2)

- Baum: spezielle Form von Graphen, in denen ausgehend von einer Wurzel mehrere gleichartige Objekte miteinander verkettet werden.



Prozess

- Ein Prozess ist ein ablaufendes Programm, welches aus einer Abfolge von (Verarbeitungs-) schritten besteht und auf einem Computer (Prozessor) ausgeführt wird.

- Die operationale Semantik eines Programms (eines Algorithmus) ist die Menge seiner Abläufe; (denotationelle Semantik: eine Funktion, die eine Eingabe auf eine Ausgabe abbildet)

Prozess

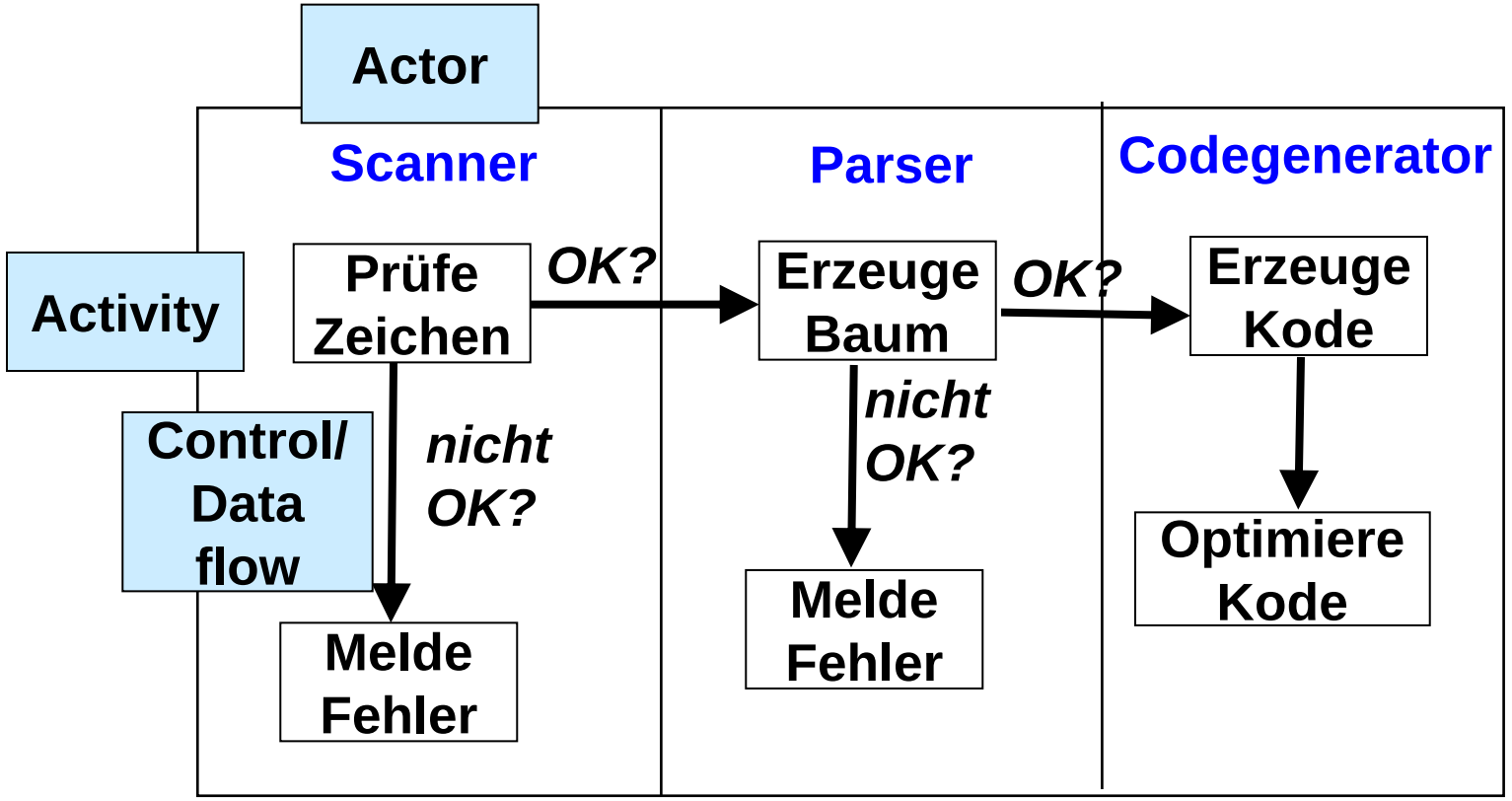
- Um ein Programm (einen Algorithmus) zu verstehen, muss man sich die Abläufe vorstellen (visualisieren).
  - Aktivitätsdiagramm: Abfolge der Verarbeitungsschritte
  - Zustandsdiagramm: Abfolge der internen Programmzustände
  - Sequenzdiagramm: Abfolge der Nachrichten

Beispiel Übersetzer

- Ein Übersetzer (Compiler) erzeugt aus einem Programmtext einen ablauffähigen Binärcode.
  - 1. Schritt: Lexikalische Analyse
    - Überprüfe Zeichen
  - 2.Schritt: Syntaxanalyse
    - Überprüfe Worte (leite Syntaxbaum ab)
  - 3.Schritt: Codeerzeugung und- optimierung

- Ein Interpreter übersetzt immer nur einen Befehl und führt ihn gleich aus.

Aktivitätsdiagramm Übersetzen

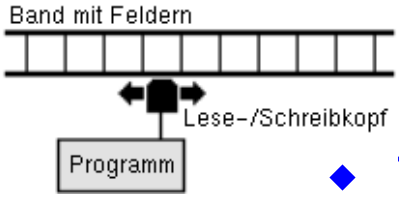


Maschine

- Eine Maschine verarbeitet Eingabedaten zu Ausgabedaten
- Neben dem traditionellen Computer/Rechner existieren in der Informatik aber noch abstraktere Arten von Maschinen
  - Turingmaschine
  - Von-Neumann Maschine

Turingmaschine (TM)

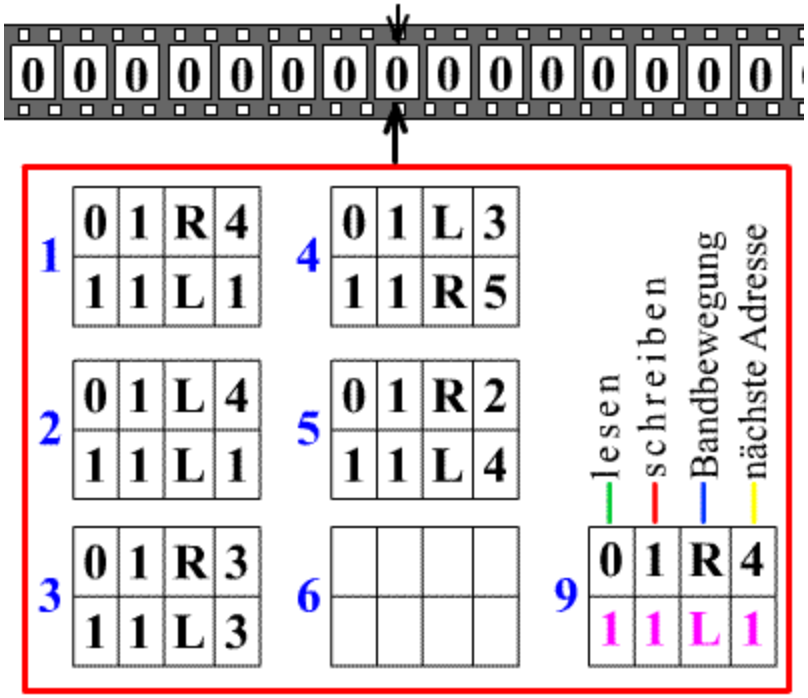
- 1936 von Allen Turing (1912-1954) zum Studium der Berechenbarkeit eingeführt.
  - eine Funktion ist berechenbar, wenn es einen Algorithmus gibt, der die Funktion berechnet.
  - Besteht aus Hardware (Band, Lese/Schreibkopf) und Software (Programm).
  - Eine TM ist nicht eine Maschine, die genau eine Sache macht, sondern ein allgemeines Konzept, welches eine ganze Menge verschiedener Maschinen definiert



- Turing-Äquivalenz: Zu jeder TM kann man ein äquivalentes Computerprogramm erzeugen, das das Verhalten der TM Schritt für Schritt nachvollzieht.

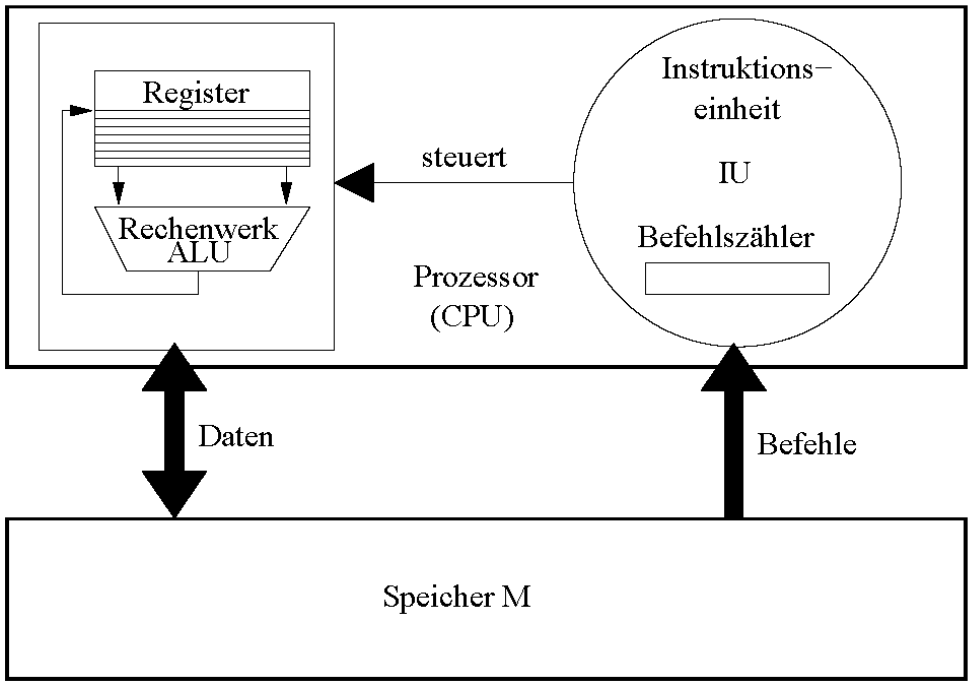
Turingmaschine—Beispiel

- Arbeitsweise einer TM



Von-Neumann Maschine

- John Von-Neumann (1903-1957) war bedeutender Mathematiker.
- Er entwickelte das Konzept der Von-Neumann Architektur für Computer.



Modell

- Ein Modell ist seinem Wesen nach eine in Maßstab, Detailliertheit und/oder Funktionalität verkürzte beziehungsweise abstrahierte Darstellung des originalen Systems. [Stach. 73]
- Ein Modell ist eine Abstraktion eines Systems mit der Zielsetzung das Nachdenken über ein System zu vereinfachen, indem irrelevante Details ausgelassen werden [Brügge 00].
- Beispiele für Modelle: Schaltbild, Baupläne, Architekturmodell, Anatomie Gerippe, Modelleisenbahn, *AT-Diagramm, ER-Diagramm, Programm*

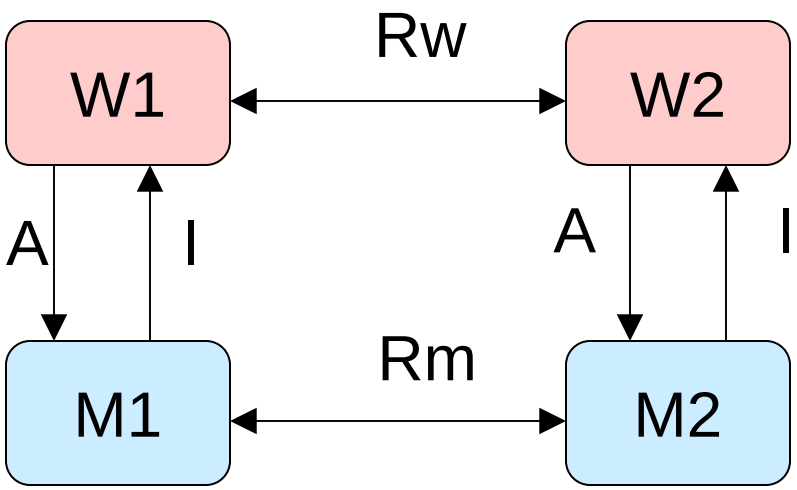
Modell—Bestandteile einer Notation

- Syntax (formale Sprache)
  - Welche Zeichenfolgen / Bildelemente sind für die Modelldarstellung erlaubt?
- Semantik
  - Was ist gemeint?
  - Interpretation des Modells in der realen Welt!
- Pragmatik (Methodik des Gebrauchs)
  - Wozu gebraucht man Modelle?
- Auch eine Programmiersprache ist eine Notation

Wozu braucht man Modelle?

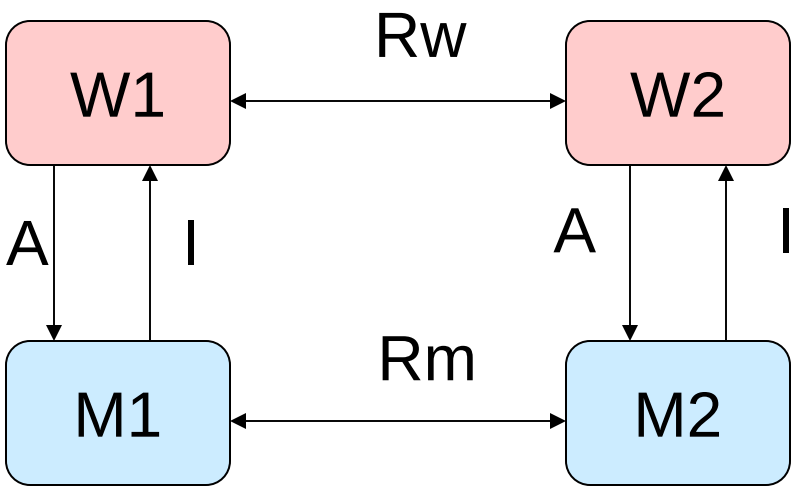
- Modelle helfen komplexe Zusammenhänge zu verstehen.
- Bildhafte Darstellung und Abstraktion unterstützen Kommunikation zwischen verschiedenen Beteiligten
  - Skizze zur Diskussion
  - Vorgabe zur Entwicklung
- Exakte Notation und Abstraktion ermöglicht Verwendung von Werkzeugen
  - zur Analyse von Eigenschaften (des Originals)
    - Größe, Komplexität, Zeitverbrauch, Vorhersage
  - Zur Transformation von Modellen
    - Bezug zwischen verschiedenen Modellen, Änderung von Modellen

Modell und Wirklichkeit



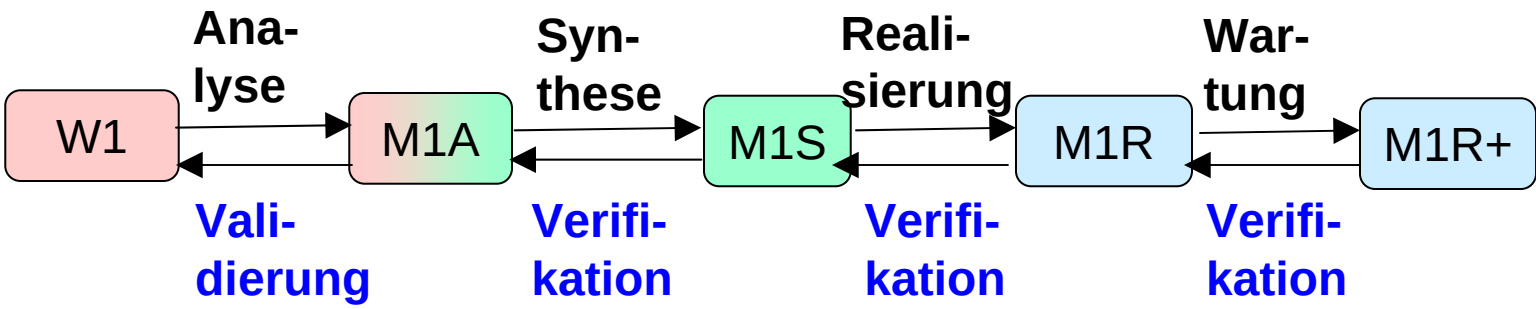
- W1, W2: Elemente der realen Welt
- Rw: Relation in der realen Welt
- M1, M2: Elemente im Modell
- Rm: Relation im Modell
- A: **Abstraktion** (Abbildung reale Welt -> Modell)
- I: **Interpretation** (Abbildung Modell -> reale Welt)

Was ist ein gutes Modell?



- Ein Modell ist gut, wenn das Diagramm kommutativ ist, d.h.
Beziehungen im Modell gelten genau dann, wenn sie auch in der Wirklichkeit gelten

Modelltransformation



- Programm ist auch ein Modell !
- Softwareentwicklung kann als Modelltransformation angesehen werden
- Verifikation: ist die Transformation von M1x nach M1y korrekt?
- Validation: entspricht Modell M1x der Wirklichkeit W1?

Literatur

- H.-P. Gumm, M. Sommer: Einführung in die Informatik, 8. Auflage, Oldenbourg, 2008.
- G. Pomberger, H. Dobler: Algorithmen und Datenstrukturen, Pearson Studium, 2008.
- A. K. Dewdney: New Turing Omnibus (Taschenbuch), Henry Holt, 2003.
- F. Naumann: Vom Abakus zum Internet – Die Geschichte der Informatik, Primus Verlag, 2001.
- H. Abelson, G.J. Sussman, J. Sussman: Structure and Interpretation of Computer Programs, MIT Electrical Engineering and Computer Science Series, 1996.
- von E. Yourdon, L.L. Constantine: Structured Design – Fundamentals of a Discipline of Computer Program and System Design, Prentice Hall, 1979.