# Kapitel 01: Informatik und Software-Systeme

## Software Entwicklung 1

Annette Bieniusa, Mathias Weber, Peter Zeller

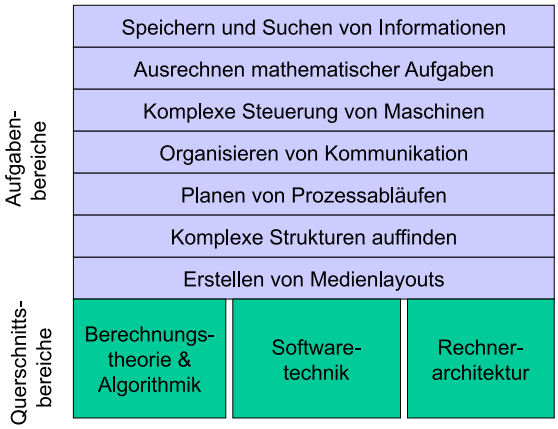
Das wichtigste Lernziel der Vorlesung "Software-Entwicklung 1" ist es einfache, sequentielle Software-Systeme eigenständig verstehen, entwickeln und implementieren zu können. In diesem einführenden Kapitel geben wir eine Einordnung der Software-Entwicklung in die Informatik und den historischen Kontext. Wir diskutieren den Begriff des Software-Systems und diskutieren die Herausforderungen bei der Entwicklung solcher Systeme.

**Lernziele dieses Kapitels:**

- Eine Definition des Begriffs "Informatik" zu nennen.
- Das Fach Informatik in den historischen, fachlichen und soziologischen Kontext einzuordnen.
- Softwaresysteme von anderen technischen Systemen abzugrenzen.
- Die Phasen der Software-Entwicklung zu beschreiben.
- Die Anforderungen an Software-Entwickler zu reflektieren.

## 1 Was ist Informatik?

Der Begriff "Informatik" wurde in den 1950er Jahren geprägt. Er umfasst nach einer Definition des Münchner Informatikers Manfred Broy die ***Wissenschaft, Technik und Anwendung der maschinellen Verarbeitung, Speicherung und Übertragung von Information***. Diese Begriffsprägung ist technisch orientiert.
Einen stärkeren Fokus auf die Aufgaben der Informatik hat die Definition des Begriffs als ***Wissenschaft von der Unterstützung und Automatisierung geistiger Tätigkeiten von Lebewesen und nicht-physikalischer Prozesse in Organisationen***.




Zu all diesen genannten Aufgabenbereichen hat der Mensch seit Tausenden von Jahren Lösungen vorgeschlagen. Sie umfassen beispielsweise das Lösen mathematischer Probleme, aber auch das Übersetzen natürlicher Sprache oder die Erkennung von handschriftlichen Texten, Gesichtern, Musikstücken. Erst durch die Entwicklung von Rechnern konnten diese Aufgaben automatisiert und auf immer größere Eingabedaten angewendet werden.

## 1.1 Zeitliches Umfeld

Die Geschichte der Rechentechnik beginnt mit der Entwicklung von ersten Algorithmen und Rechenvorschriften bereits in der Antike. Nach ersten mechanischen Anlagen führt der Wechsel zu elektronischen Rechnern zu einer Beschleunigung in der Entwicklung von programmierbaren System, die bis heute anhält und unser Leben nachhaltig prägt.

**ca. 5000 v.C.** Erste Zahlensysteme

**ab ca. 1100 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.

**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 konstruiert eine mechanische Anlage Z1 zur Berechnung von Baustatik

**1937** Alan Turing entwickelt ein Modell für einen Universalrechner

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

**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. Betty Jennings, Betty Snyder, Frances Spence, Kay McNulty, Marlyn Wescoff und Ruth Lichterman sind die ersten Programmiererinnen der ENIAC.

**1949** Grace Hopper entwickelt den ersten Compiler, der Anweisungen aus dem Englischen in Maschinensprache übersetzte.

**1952** Erste Rechner nach dem Prinzip von John von Neumann wird entwickelt.

**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.

**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).

**1971** Margaret Hamilton, u.a. Ingenieurin bei der NASA, prägt den Begriff “Software Engineering”.

**1972** B. Kernighan u. D. Ritchie entwickeln C für die Programmierung von UNIX.

**1973** Adele Goldberg u.a. entwickeln Smalltalk, die erste objekt-orientierte Programmiersprache.

**1974** Kahn und Cerf entwickeln das Netzwerkprotokoll 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 den i80386 Prozessor an.

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

**1993** Mosaic ist der erste graphische Web-Browser.

**1995** Die Programmiersprache Java wird vorgestellt.

**1998** Google Inc. wird von Lawrence Page und Sergey Brin gegründet.

**2004** Dustin Moskovitz, Chris Hughes, Eduardo Saverin und Mark Zuckerberg veröffentlichen TheFacebook

**2007** Markteinführung des iPhones

**2008** HTC Dream als erstes Android-Smartphone

Die genannten Ereignisse sind zum Zwecke der Illustration gewählt und spiegeln nicht unbedingt ihre Bedeutung und Einfluss dar. Entsprechend der Geschichte der Rechentechnik gibt es auch eine Geschichte der Speicherung und Wiedergabe von Information, Robotik und KI sowie der Nachrichtenübertragung.

## 1.2 Fachliches Umfeld

Nicht nur aus der Historie heraus, steht die Informatik mit einer Vielzahl von anderen Fachdisziplinen in engem Zusammenhang. Bezüglich der theoretischen Aspekte gibt es eine enge Verbindung mit der Mathematik, insbesondere bezüglich Numerik und formalen Logik; die technische Ausprägung der Informatik ist wiederum eng mit der Elektrotechnik verzahnt. Aber auch mit der Psychologie (Arbeitsprozesse, Kognition), Neurologie (Informationsverarbeitung), Linguistik und Philosophie bestehen Bezüge.
Im Anwendungsbereich gibt es vielfältige Anknüpfungspunkte an die Betriebs- und Verwaltungswirtschaft bezüglich Datenhaltung und -verarbeitung. Steuerung und Automation finden immer stärker Anwendung in der Produktionswirtschaft (Stichwort: Industrie 4.0). Auch die Beiträge der Informatik zur Kommunikation zwischen Mensch und Maschine, sowie zwischen maschinellen Systemen sind essentiell. Und dies stellt nur einen kleinen Ausschnitt dar.

## 1.3 Gesellschaftliches Umfeld

Die Ergebnisse und Produkte der Informatik dringen in zunehmendem Maße in die Gesellschaft ein. Der Einfluss auf Wirtschaft, Privatleben und auch öffentliches Leben hat in den letzten Jahren zu teilweise radikalen Veränderungen geführt.
Daraus resultiert vierfache Verantwortung für Informatiker:

1. Chancen für Innovation zu nutzen;
2. korrekte, verlässliche Systeme zu bauen;
3. die “richtigen” Systeme bauen, die Nutzen bringen und Schaden abwenden;
4. den Vermittlungs- und Veränderungsprozess in der Gesellschaft mitzutragen.

# 2 Was ist ein Softwaresystem?

Es ist nicht allein der physikalische Teil des Rechners, die Hardware, die zum Erfolg der Rechenmaschinen beigetragen hat. Bereits die ersten Rechenmaschinen waren darauf ausgelegt, verschiedene Aufgabenstellungen zu lösen – sie waren mit Hilfe von Software programmierbar. Softwaresysteme begegnen uns heute in vielfältiger Ausprägung. Um eine Definition dieses Begriff zu motivieren, betrachten wir zunächst einige Charakteristika der folgenden Beispiele:

- Betriebssystem, Fenstersystem, Dateisystem
- Anwendung zur Textverarbeitung, Analyse und Visualisierung von Daten, Computerspiele
- Internet, speziell das World Wide Web
- Informations- und Buchungssysteme

- Telekommunikationssysteme
- Eingebettete Softwaresysteme, z.B. Steuerungen für Motoren, Airbags, Roboter, Flugzeuge, Waschmaschinen, ...

*Softwaresysteme sind nicht physisch.* Sie lassen sich nicht anfassen oder direkt betrachten; sie gehorchen nicht physikalischen Gesetzen und sind nicht natürlich, sondern konstruiert / technisch.
*Softwaresysteme dienen sehr unterschiedlichen Zwecken.* Sie dienen beispielsweise der Kommunikation mit Menschen (Computerspiel, Informationssysteme) oder der Steuerung von Apparaten/Maschinen (Airbagsteuerung, Autopilot). Sie können aber auch Plattform für andere SW-Systeme sein (Betriebssystem, Internet, ...).
*Softwaresysteme bauen auf anderen Systemen auf.* Ein Computerspiel setzt auf dem Betriebssystem auf, das Betriebssystem wiederum auf der Rechnerhardware, das WWW auf dem Internet, .... Systeme bieten Schnittstellen zur Interaktion mit ihrer Umgebung an (insbesondere Mensch-Maschine-Schnittstelle, Human Computer Interface HCI). Sie sind häufig Teil komplexerer Systeme (z.B. Auto) und unterstützen bzw. erweitern deren Handhabung.
Softwaresysteme haben z.T. sehr unterschiedliche Eigenschaften, zum Beispiel bzgl.

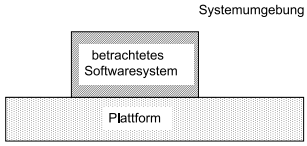
- Größe (gemessen in Programmzeilen, in Personenjahren / Investitionen, Features)
- Terminierung (einmaliger, abgeschlossener Arbeitsschritt (z.B. Übersetzer) vs. ständig verfügbarer Service (z.B. Telekomsystem))
- Persistenz (Daten werden zwischen aufeinander folgenden Ausführungen aufbewahrt vs. verworfen),
- Interaktionsverhalten (batch-orientierte, transformierende Systeme vs. interaktive, reaktive Systeme)
- Verteilung (lokal auf einem Rechner vs. verteilt auf vielen gleichartigen oder auch unterschiedlichen Rechnern)
- Komplexität
- Qualitätseigenschaften (Korrektheit, Stabilität, Benutzerfreundlichkeit, Wartbarkeit)

Softwaresysteme spielen eine bedeutende Rolle für die wirtschaftliche Entwicklung einer modernen Gesellschaft. Sie erlauben eine effiziente, automatisierte Verwendung von Ressourcen, liefern eine Kommunikationsinfrastruktur und erlauben schnellen und einfachen Zugriff auf Wissen und Information. Damit tragen sie essentiell zu Innovation und Fortschritt bei. Sie können die medizinische und soziale Versorgung verbessern und leisten einen Beitrag zum gesellschaftlichen, privaten und kulturellen Leben.

## 2.1 Zur Definition

Ein ***System*** besteht aus Teilen (Komponenten, Subsystemen), die in geordneter Weise miteinander in Beziehung stehen. ***Technische Systeme*** bestehen üblicherweise aus technischen Komponenten (Bauteilen, Geräten, usw.), die gemeinsam eine Einheit bilden.
Beispiele für nichttechnische Systeme sind Gesellschafts-, Rechts-, Wirtschaftssysteme, das Ökosystem und unser Sonnensystem. Technische Systeme hingegen sind Autos, Kraftwerke, Telefonnetze, Rechner, etc. Es finden sich aber auch zahlreiche hybride Varianten (z.B. Verkehrssystem, Bibliotheksverwaltungssystem).
Ein ***Softwaresystem*** ist ein technisches System oder Subsystem, dessen Funktionalität mittels Software realisiert ist. Softwaresysteme besitzen ***Schnittstellen zur Umgebung***:

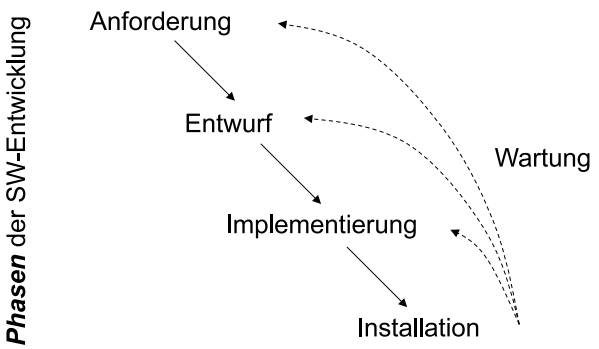
- Bedienschnittstellen
- weitere Ein-/Ausgabeschnittstellen
- Programmierschnittstellen




Jedes Softwaresystem läuft auf einer ***Plattform*** (Hardware- oder Softwareplattform).
*Software im engeren Sinne* umfasst dabei Programme zur Beschreibung des Systemverhaltens sowie Daten zur Beschreibung von Informationen. Ein ***Programm*** ist ein Text, der in einer Programmiersprache verfasst ist und hinreichend vollständig ist, um auf einem Rechensystem ausgeführt zu werden. Daten können gegeben sein in Form von Messwerten, Statistiken, Web-Seiten, Textdokumente, etc. 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.

## 2.2 Entwicklung von SW-Systemen

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




**Phasen der SW-Entwicklung** Ausgangspunkt der SW-Entwicklung sind die ***Anforderungen*** (engl. *requirements*):

Was soll das System leisten?

Man unterscheidet dabei *funktionale* Anforderungen (Ein- und Ausgabeverhalten) und *nicht-funktionale* Anforderungen (Effizienz, Portabilität, Bedienbarkeit, Stabilität, Zuverlässigkeit, Wartbarkeit, ...).

***Software-Entwurf*** (engl. *design*):

- Umsetzung der Anforderungen in ein Modell für das zu entwickelnde Softwaresystem
- Das resultierende Artefakt ist das 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.)
- Das resultierende Artefakt sind die Programme.

***Installation*** (engl. *installation/deployment*):

- Einrichten der Software auf den Plattformen, auf denen sie ablaufen soll
- Das resultierende Artefakt ist das eingerichtete Software-System.

***Software-Wartung*** (engl. *maintenance*):

- 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!).

Auf Grund der wachsenden Zahl an Software-Systemen, immer stärker diversifizierte Systemumgebungen und ständiger Neuerungen nimmt die Wartung einen immer größeren und wichtigen Anteil im Software-Entwicklungsprozess ein.

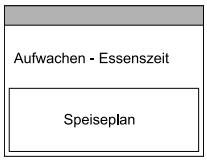
## 2.3 Beispiel: Der MensaWecker

Als Beispiel für eine Umsetzung der Phasen der SW-Entwicklung betrachten wir ein Software-System, das die im Terminalraum in die Arbeit vertiefte Studierende an das Mittagessen erinnert. Wir stellen die folgenden Anforderungen 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 Linux-Systemen laufen.

Wir entwerfen ein System, das die Anforderungen genügt, wie folgt:

- 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. Diese Komponente soll "MensaWecker" heißen.
- Die MensaWecker-Komponente bekommt die Kommandozeile zum Aufruf des Browsers als Zeichenreihe übergeben.
- Layout-Entwurf für das Fenster:



Wir verfeinern den Entwurf und implementieren ihn schließlich. Als Implementierungssprache benutzen wir Java. Die Komponente wird als Klasse `MensaWecker` implementiert und in einer Datei `MensaWecker.java` abgespeichert.
Das Java Programm hat die folgende Form:

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.net.URI;
4
5  public class MensaWecker extends Frame
6    implements ActionListener {
7
8     private Label lblWecker;
9     private Button btnSpeiseplan;
10    private final String webpage;
11
12    public MensaWecker(String webpage) {
13       this.webpage = webpage;
14       setLayout(new FlowLayout());
15       lblWecker = new Label("Aufwachen - Mittagessen!");
16       add(lblWecker);
17
18       btnSpeiseplan = new Button("Speiseplan");
19       add(btnSpeiseplan);
20
21       btnSpeiseplan.addActionListener(this);
22       setTitle("MensaWecker");
23       setSize(250,100);
24       setVisible(true);
25    }
26
27    public void actionPerformed(ActionEvent evt) {
28       try {
29          if(Desktop.isDesktopSupported()) {
30             Desktop.getDesktop().browse(new URI(webpage));
31          }
32       } catch (Exception e) {
33          System.err.println("A problem happened, can't open!");
34       }
35    }
36
37    public static void main(String[] args) {
38       MensaWecker mw = new MensaWecker(args[0]);
39    }
40 }
```

Nun übersetzen wir das Programm in eine vom Rechner ausführbare Version:

```
javac MensaWecker.java
```

Im letzten Schritt installieren wir das System. Dazu speichern wir die übersetzte Datei auf den Rechnern, auf denen das MensaWecker-System laufen soll. Zuletzt tragen wir den Aufruf von MensaWecker 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.

## 2.4 Herausforderungen bei der SW-Entwicklung

In SW-Projekte können kleine Fehler oft verheerende Folgen haben und hohe Kosten mit sich bringen.

- Ariane 5, Flug 501 (1996): Versagen des "Track Control System" resultiert in der Selbstzerstörung der Rakete (*Integrationsfehler*).
  Kosten $370.000.000 US-Dollar.
- Flughafen Denver (2005): Verspätete Lieferung der Software für das Gepäcksystem verzögerte die Flughafeneröffnung um 16 Monate (*Planungsfehler*).
  Kosten: $560.000.000 US-Dollar.
- Ein Programmierfehler in der OpenSSL-Bibliothek, welche von vielen anderen Systemen zur sicheren Übertragung von Daten verwendet wird, wurde 2012 eingeführt und 2014 entdeckt. Durch den Fehler konnten Angreifer vertrauliche Daten von betroffenen Systemen auslesen. Durch die große Zahl an betroffenen Systemen wird der Schaden auf über $500.000.000 US-Dollar geschätzt.

Einer der größten Herausforderungen ist, dass die Anforderungsbeschreibung den Kundenwunsch präzise wiedergeben muss. Dieser ist allerdings häufig unvollständig, inkonsistent und unbeständig auf Grund von spontanen Änderungswünschen.
Die Skalierbarkeit von SW-Entwicklungsmethoden ist in der Praxis ein großes Problem. Eine 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 außerdem Termin-, Budget- und Qualitätsanforderungen zu berücksichtigen.

## 2.5 Was muss ein Software-Entwickler können und wissen?

Software-Entwickler müssen über eine Reihe von Kompetenzen verfügen.

- Sie müssen Anforderungen und Aufgabenstellung verstehen, die meist aus anderem Fachgebiet entstammen. Dies erfordert Domänenwissen.
- Sie müssen Beschreibungsmittel und Werkzeuge zur Modellierung detailliert beherrschen. Dies erfordert Abstraktionsvermögen.
- Sie müssen verschiedene Realisierungsmöglichkeiten kennen und Alternativen bewerten können.
- Sie müssen Schritte des Entwicklungsprozess kennen und durchführen können.

- Sie müssen wissen, was Korrektheit bei Entwurf, Implementierung und Installation bedeutet und wie man sie nachweist.

Die Anforderungen sind in der Kompetenzraute zur Software-Entwicklung zusammengefasst.