universität innsbruck
Institut für Informatik

Software Engineering

1. Einführung und Überblick

Software ist überall

Software ist überall

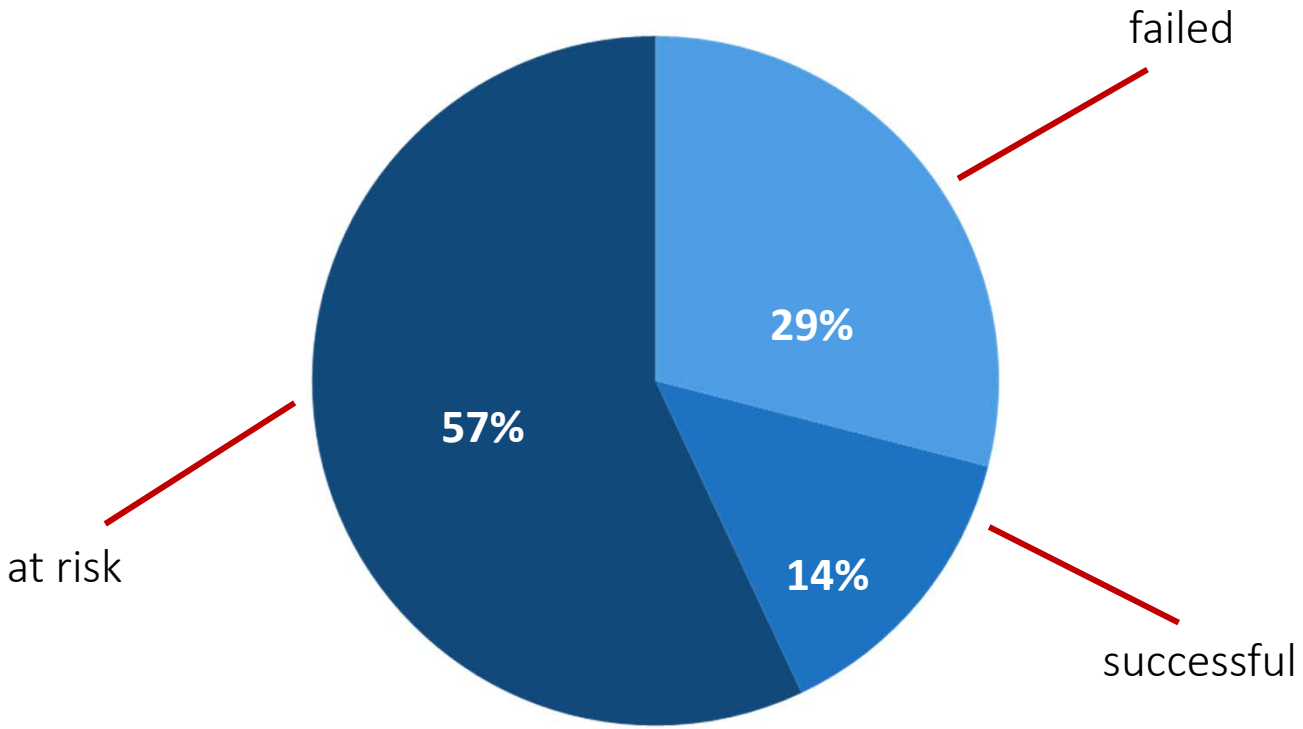
Android ~ Ca. 12 MLoc

Boeing 787~Ca. 13 MLoc

Ca. 100 MLoc

MLoC = Mio Lines of Code

IT-Projekte



of which:
- deadline missed (84 %)
- budget exceeded (56 %)
- incomplete functionality (36 %)

Source: Standish Group

Besonderheiten von Software

- Software ist immateriell.
- Software ist schwer zu vermessen.
- Software gilt als relativ leicht änderbar
  (im Vergleich zu materiellen technischen Produkten).
- Software unterliegt einem ständigen Anpassungsdruck.
- Software altert.

Fehlerursachen

- Fehlende Projektplanung und –kontrolle
- Vernachlässigung der Entwurfsphase
- Mangelnde Erfassung der Benutzerbedürfnisse und der Anwendungsumgebung
- Fehlende Dokumentation
- Uneingespielte Entwicklungsteams
- Fehlerhafte Aufwandsabschätzungen
- zu spätes Reagieren auf Entwicklungsrisiken technischer Natur

Was ist Software?

Software: Die Programme, Verfahren, zugehörige Dokumentation und Daten, die mit dem Betrieb eines Rechnersystems zu tun haben

(IEEE 610.12)

⇨ … mehr als Programme

Softwaretechnik

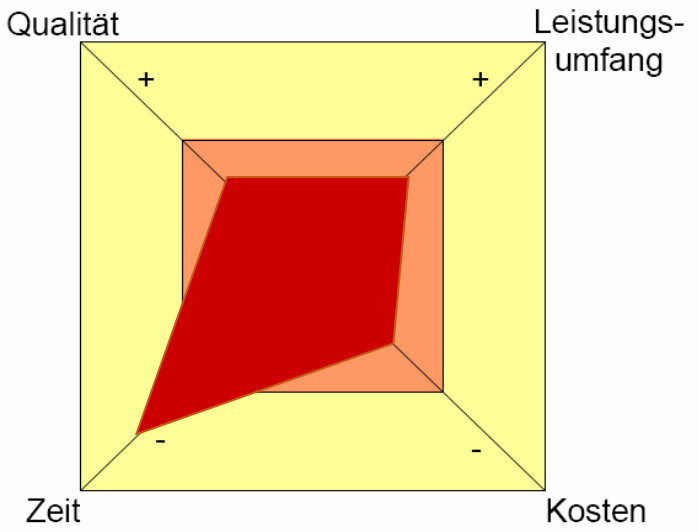
> software engineering: The establishment and use of sound engineering principles in order to obtain economically software that is reliable and runs on real machines.
>
> (F.L. Bauer, NATO-Konferenz Software-Engineering 1968)

> software engineering: the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
>
> IEEE

- Prinzipien einer Ingenieursdisziplin: Bereitstellung und systematische Verwendung von Methoden, Verfahren und Werkzeugen zur Lösung von Problemstellungen
- **Softwaretechnik**
  - Meist als deutsches Synonym für *software engineering* gebraucht.

Harry Sneeds Teufelsquadrat

- Software Qualität ist keine Insel



Die Produktivität (=Fläche) bleibt die gleiche, auch wenn einer der Parameter (Qualität, Zeit, Kosten, Umfang) sich ändert

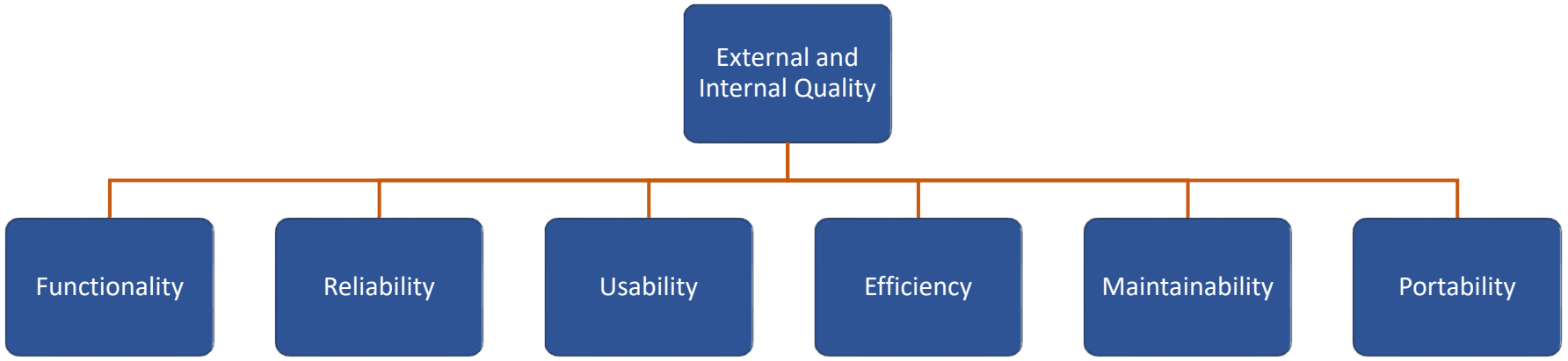
Themengebiet Softwaretechnik

- Entwurfsprozesse
  - Entwurfsaktivitäten, Dokumente
- Softwareentwicklungsmethoden
  - Techniken und Vorgehensweisen von der Erfassung von Anforderungen bis zum Code
- Beschreibungstechniken
  - graphische und textuelle Sprachen zur Beschreibung von Systemen
- Qualitätsmanagement
  - Maßnahmen zur Steigerung der Produktqualität

Themengebiet Projektmanagement

- Projektplanung
- Risikomanagement
- Kostenschätzung
- Projektanalyse
- Teamführung

Qualitätskriterien nach ISO 25010



- Basis für die Definition von Qualitätskriterien für ein spezifisches SW-System
- Externe Qualitätskriterien (beziehen sich auf User, Kunde)
  - Functionality, Reliability, Usability, Efficiency
- Interne Kriterien (beziehen sich auf Software Provider, Entwickler)
  - Maintainability, Portability

Funktionstüchtigkeit

- Die Software unterstützt die spezifizierten Funktionen unter definierten Bedingungen.
- Angemessenheit (Suitability)
  - *Primäres Kriterium – Angemessenheit der verfügbaren Funktionen.*
- Exaktheit (Accuracy)
  - *Z.B. Geldbeträge sollen auf 2 Dezimalstellen genau sein.*
- Interoperabilität
  - *E.g. Daten werden in Format XY exportiert.*
- Security
- Compliance
  - *E.g. Die Software erfüllt den Standard ISO 26262.*

Zuverlässigkeit (Reliability)

- Die Software ist in der Lage, ein gewisses Zuverlässigkeits-Level unter definierten Bedingungen in einer definierten Zeitperiode sicherzustellen.
- Korrektheit (Maturity)
  - *Z.B. geringe Fehlerrate*
- Robustheit (Fault Tolerance)
  - *Erwartete Funktionalität mit hoher Wahrscheinlichkeit*
  - *Auftretende Fehler mit geringen Auswirkungen*
- Wiederherstellbarkeit (Recoverability)
  - *Fähigkeit des Systems, nach Fehlern wieder in Betrieb zu gehen (einschließlich Datenrettung und Inbetriebnahme der Kommunikation mit anderen Systemen)*

Bewertung der Zuverlässigkeit

- Einige Metriken (Maße):
  - „rate of failure occurrence“ (ROFOC)
    Häufigkeit von nicht erwartetem Verhalten
    z.B. 2/100 = 2 Fehler pro 100 Zeiteinheiten
  - „mean time to failure“ (MTTF)
    Zeitabstand zwischen zwei Fehlern
  - Verfügbarkeit (engl. *availability*)
    z.B. 988/1000 = während 1000 Zeiteinheiten war System 988 Einheiten benutzbar

Korrektheit und Zuverlässigkeit

Zuverlässigkeit umfasst

- die Korrektheit des Systems
- die Robustheit bei Auftreten von Fehlern
- die Ausfallsicherheit

Bei 100% Korrektheit fällt Zuverlässigkeit mit Korrektheit zusammen

Benutzbarkeit (Usability)

- Usability bezieht sich auf den Aufwand, das System zu benutzen, und auf die individuelle Bewertung der Nutzer
- Verständlichkeit (Understandability)
  - *Das System nutzt Symbole und Begriffe, die von der User-Zielgruppe verstanden werden.*
- Lernbarkeit (Learnability)
  - *Vertretbarer Lernaufwand zur Nutzung des Systems.*
- Betriebsfähigkeit (Operability)
  - *Z.B. Konfigurierbarkeit*
- Attraktivität (Attractiveness)
  - *Z.B. Software ist zurechtgeschneidert für Teenager.*
- Usability compliance
  - *E.g. Software ist Microsoft zertifiziert.*

Effizienz

- Die Effizienz bezieht sich auf die Beziehung zwischen dem tatsächlichen Verhalten des Systems (in Bezug auf Zeit und Ressourcen) und dem spezifizierten Verhalten.
- Zeitverhalten
  - *Z.B. Das System antwortet bei 100 parallelen Usern in höchstens 2 ms.*
- Ressourcenverbrauch
  - *Das System verbraucht höchstens X MB Hauptspeicher.*

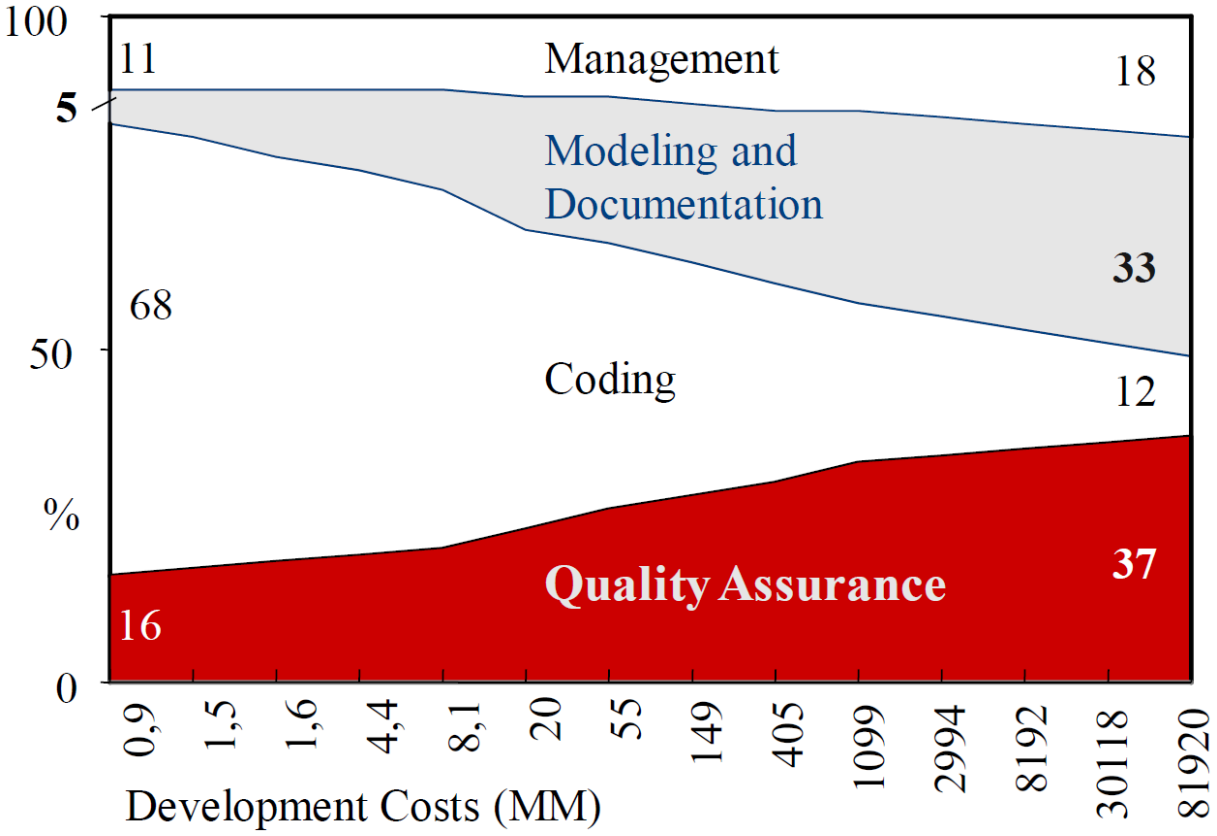
Wartbarkeit (Maintainability)

- Wartbarkeit bezieht sich auf den Aufwand, das System zu modifizieren.
- Analysierbarkeit (Analyseability)
  - *Z.B. das System unterstützt das Loggen von User-Aktionen.*
- Änderbarkeit (Changeability)
  - *Z.B. neue Funktionalität kann mit vertretbarem Aufwand implementiert werden.*
- Stabilität (Stability)
  - *Anfälligkeit für Fehler bei Änderungen*
- Testbarkeit (Testability)
  - *Z.B. vertretbarer Aufwand für die Überführung der Testumgebung in die Produktivumgebung*

Übertragbarkeit (Portability)

- Portabilität bezieht sich auf die Fähigkeit, ein System auf eine andere Umgebung zu übertragen.
- Anpassbarkeit (Adaptability)
  - *E.g. System unterstützt Android and iOS.*
- Installierbarkeit (Installability)
  - *Aufwand zur Inbetriebnahme.*
- Coexistence
  - *Es gibt keine Nebeneffekte zu anderen Systemen.*
- Austauschbarkeit (Replaceability)
  - *Aufwand zum Ersetzen von (Teilen des) Systems, z.B. durch neue Versionen*

Verteilung von Aufwänden in SW-Projekten



Liggesmeyer, P.: "Software-Qualitätssicherung"

Entwurfsprozesse

- Welche Ziele verfolgt ein definierter Entwurfsprozess?
  - möglichst große Planungssicherheit
  - frühes Erkennen und Vermeiden von Risiken
  - einheitliches Vorgehen über Projektgrenzen hinweg
  - klare Abgrenzung von Aufgaben und Verantwortlichkeiten
  - kontrollierter Umgang mit Kundenwünschen
  - standardisierte Qualitätssicherung durch definierte Dokumente mit entsprechenden Abnahmen
- Wir sprechen von **Entwurfsprozessen** oder **Vorgehensmodellen**.

Vorgehensmodelle

Ein Vorgehensmodell definiert

- die **Aktivitäten**, die im Laufe der Softwareentwicklung durchlaufen werden
  - z.B. Angebotserstellung oder Testen
- die **Produkte** (**Dokumente, Modelle**), die innerhalb der Aktivitäten erstellt werden
  - z.B. Pflichtenheft oder lauffähiges System
- die **Beschreibungstechniken** und **Methoden**, die in den Produkten bzw. zur Erstellung der Produkte verwendet werden
  - z.B. Klassendiagramme
- die für die Aktivitäten und Produkte **Verantwortlichen**
  - z.B. Projektmanager, Entwickler
- den **Prozess**, d.h. mögliche Abfolgen von Aktivitäten

Die wichtigsten Aktivitäten der Softwareentwicklung

Was?

Anforderungs-spezifikation

Beschreibung der fachlichen Anforderungen an das System, Systemgrenzen

Design

Grobskizze der Systemstruktur

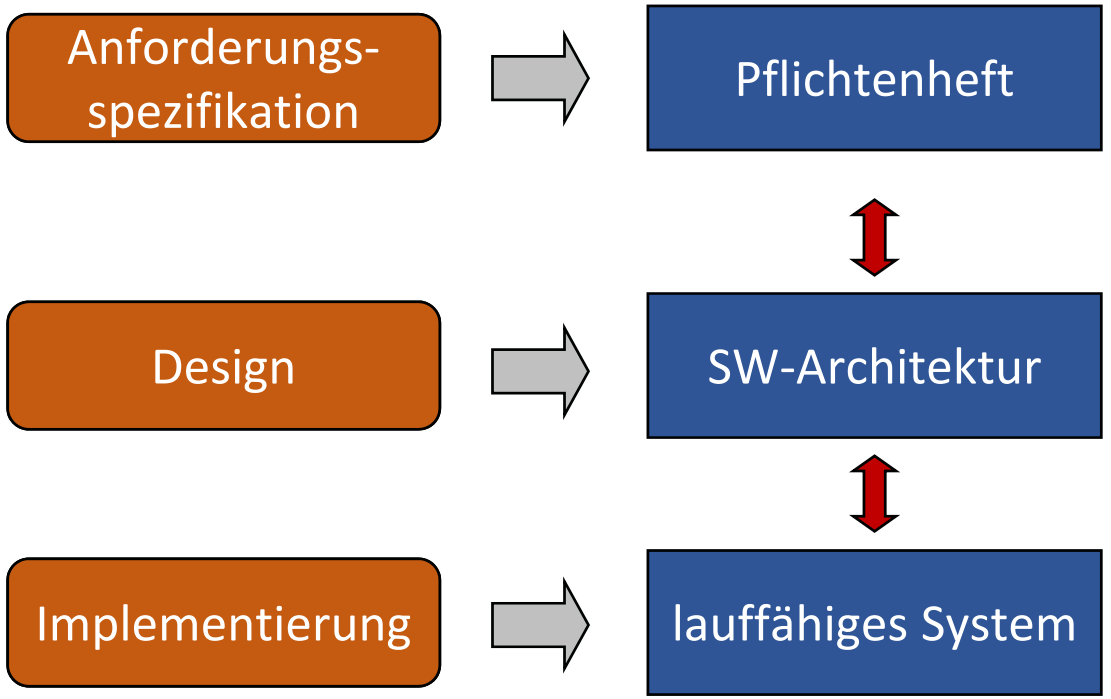
Wie?

Implementierung

Systemrealisierung

Test

Die wichtigsten Dokumente der Softwareentwicklung



Terminologie

- Achtung: Die Terminologie ist in den einzelnen Vorgehensmodellen nicht einheitlich!
  - Requirements specification, analysis model, system specification, Pflichtenheft, Anwenderanforderungen, ….
  - Bezeichnung vom Vorgehensmodell abhängig
  - Inhalt oft ähnlich, aber im Detail Überschneidungen

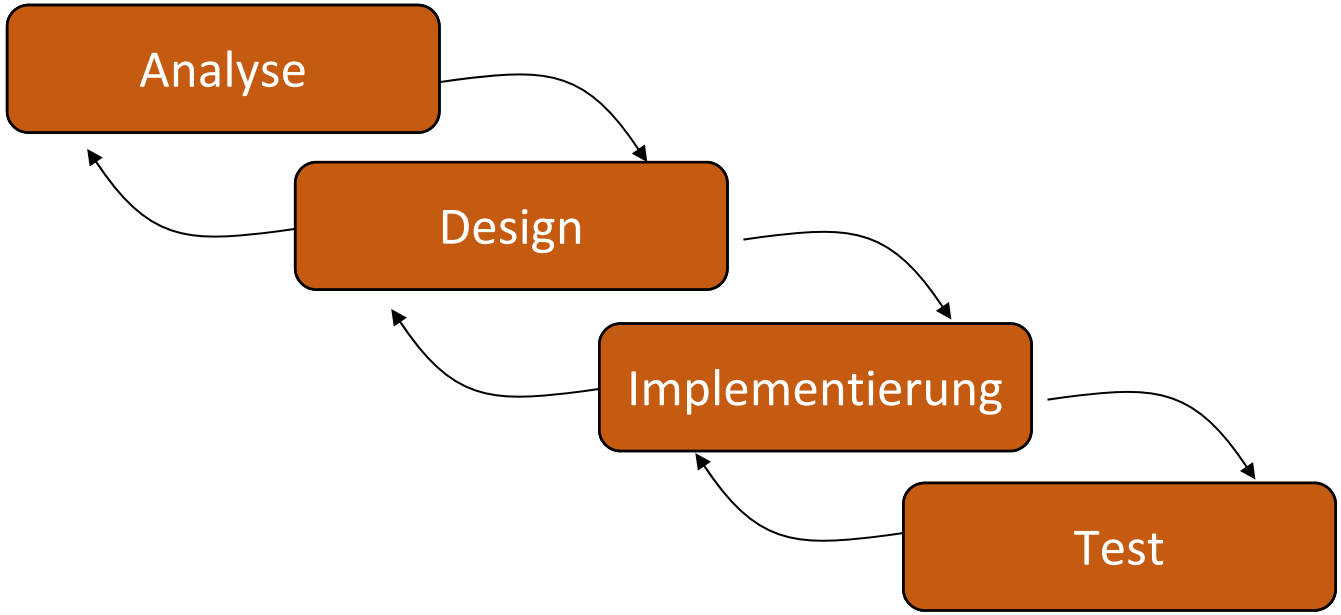
Weitere Aktivitäten von Vorgehensmodellen

- Weitere Aktivitäten befassen sich z.B. mit
  - Qualitätsmanagement
  - Projektmanagement
  - Konfigurationsmanagement
  - Auslieferung

Beispiele für Vorgehensmodelle

- Wasserfallmodell
- Unified Process
- V-Modell
- Agile Prozesse

Ursprung aller Vorgehensmodelle: Das Wasserfallmodell



Probleme des Wasserfallmodells

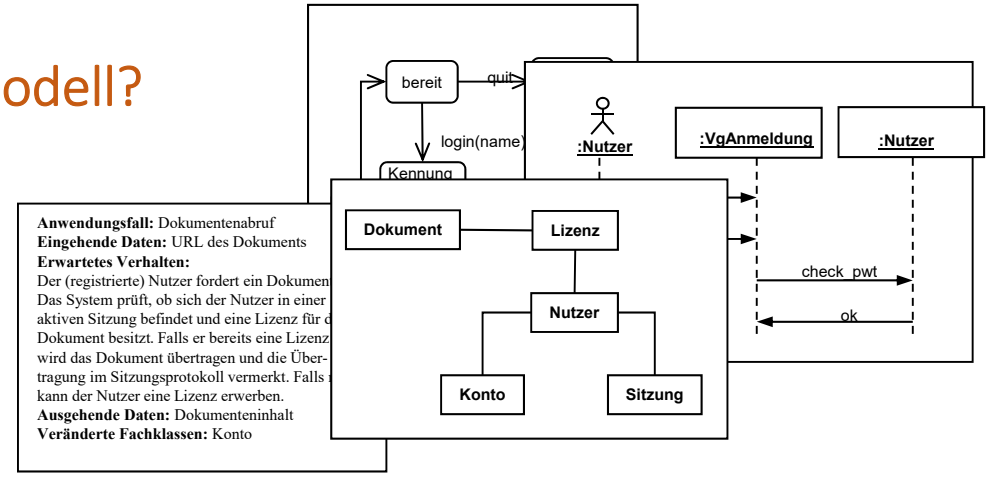
- geringe Planungssicherheit
- Kontrolle der Projektrisiken schwierig
- Implementierung zu spät verfügbar
  - „Look and feel“ der Kunden zu spät, deshalb Kundenwünsche zu spät
  - Grundsätzliche Probleme der Architektur zu spät sichtbar
  - Weitreichende Entscheidungen bei geringem Wissensstand
- keine Berücksichtigung von Wiederverwendung
  - problematisch: Integration von Altsystemen und eingekauften Komponenten

Prinzipien moderner Vorgehensmodelle

- Inkrementelle Softwareentwicklung
  - sukzessive Entwicklung von Ausbaustufen des Systems
  - jede Ausbaustufe erweitert die Funktionalität des Systems
- Iterative Softwareentwicklung
  - wiederholtes Durchlaufen von Aktivitäten
  - sukzessive Verfeinerung von Arbeitsergebnissen
- Prototyping – schnelle Entwicklung von System(teilen)
- Wiederverwendung
  - Integration von Altsystemen oder bestehender Komponenten
  - Verwendung von Mustern

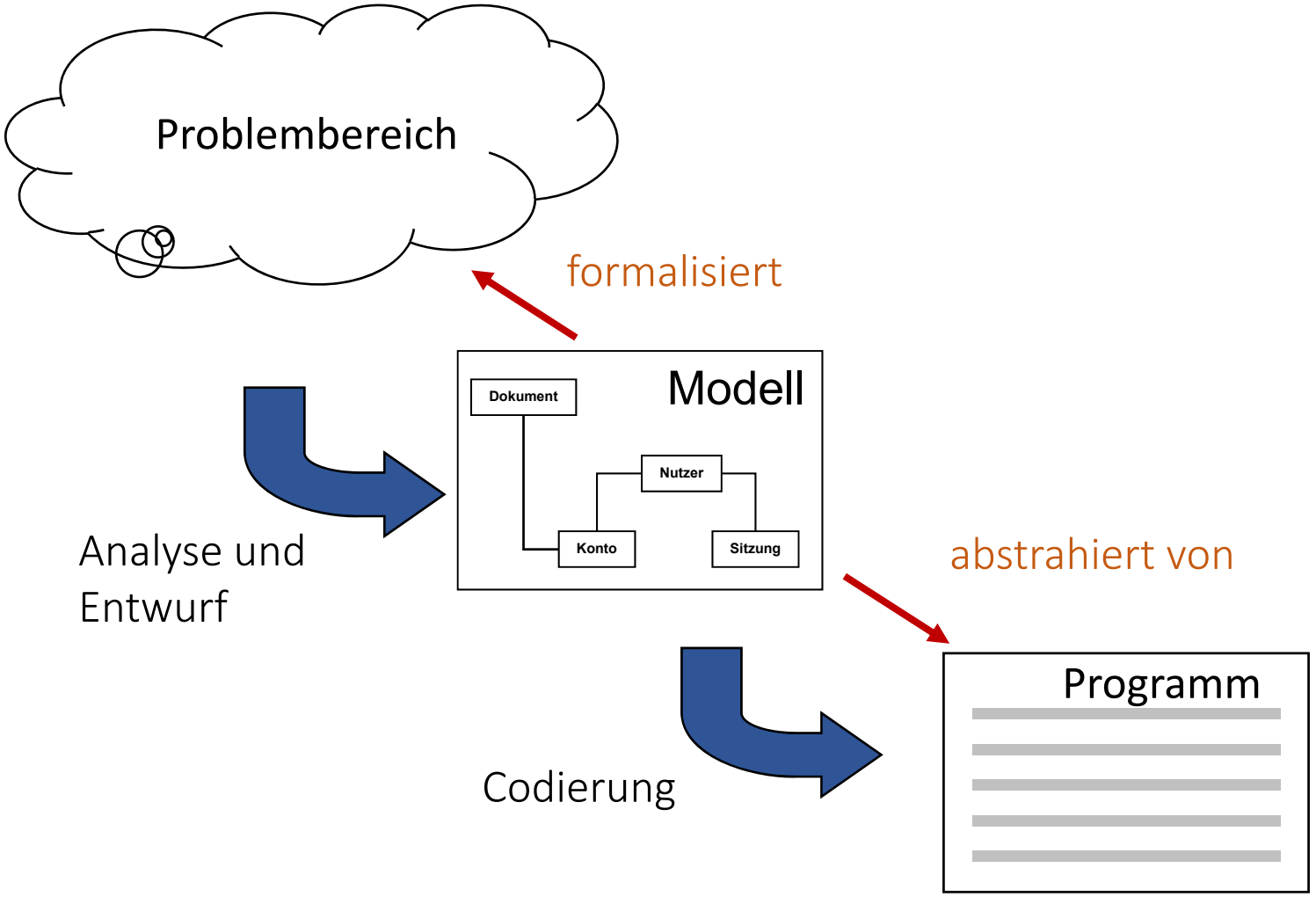
Modellierung

Was ist ein Modell?



- Ein Modell beschreibt die Eigenschaften eines Systems aus einer bestimmten Sichtweise heraus
  - z.B. aus der Sicht des Benutzers oder des Programmierers
- Ein Modell kann enthalten:
  - Text
  - Diagramme
  - ausführbaren Code

Grundidee

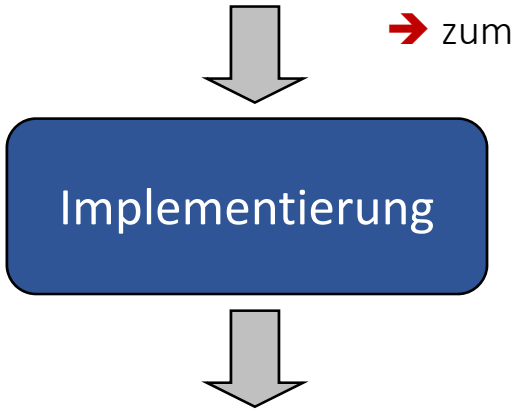

Problembereich
formalisiert
Modell
Dokument
Nutzer
Konto
Sitzung
Analyse und
Entwurf
abstrahiert von
Programm
Codierung

Warum werden Modelle erstellt?

- Kommunikation vereinfachen
- Komplexität reduzieren
- Entwurf dokumentieren
- Vollständigkeit und Konsistenz des Entwurfs prüfen
- Modelle stellen eine personenunabhängige Wissensbasis dar

Wann werden Modelle erstellt?

- Modelle werden während des gesamten Entwurfsprozesses erstellt und bearbeitet



vor der Implementierung

➔ zum Festlegen von Eigenschaften des zu entwickelnden Systems

während der Implementierung

➔ Dokumentation

nach der Implementierung

➔ im Rahmen der Modifikation und Erweiterung des Systems

Beschreibungstechniken

- Informeller Text
- Pragmatische Methoden
  - meist graphikorientierte Methoden, die sich aus der Praxis heraus entwickelt haben
  - Beispiel: Klassendiagramme, Sequenzdiagramme
- Formale Methoden
  - Beschreibungstechniken mit einer formalen Syntax und Semantik
  - meist versehen mit Verifikationstechniken und/oder Transformatoren

Beispiele – Informelle Spezifikation

- Glossare
- Informelle Beschreibung der Systemanforderungen in Pflichtenheften

Use Case Create Meeting

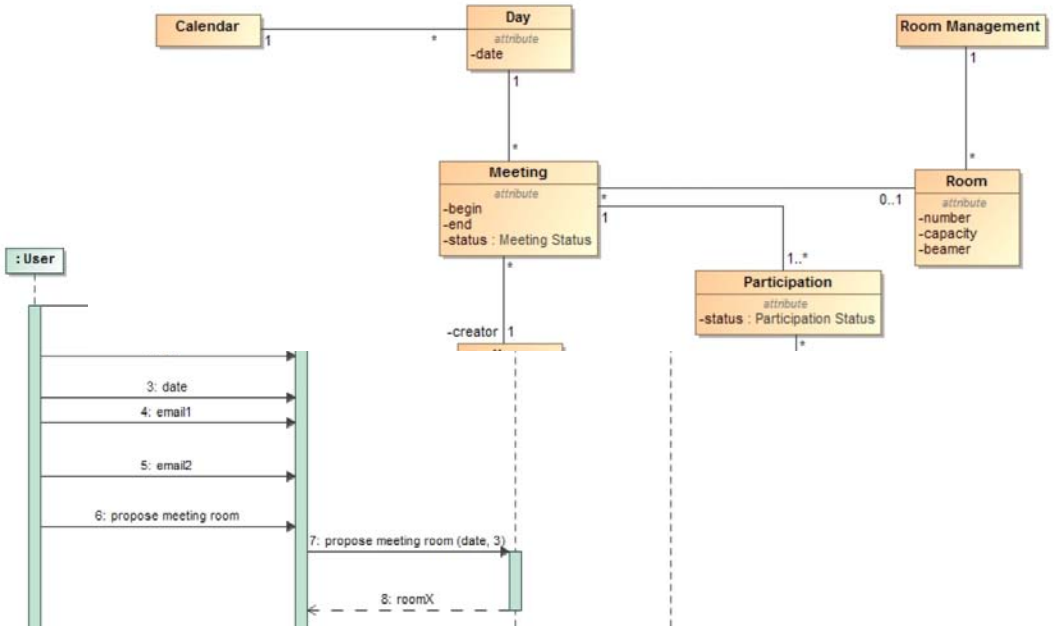
Actor User

Basic Flow

1. Upon start of this use case the user enters title, date, room and invited participants in any order.
    - The title is mandatory for persisting the meeting.
    - The date is pre-filled. The user enters day by selection from a calendar, and begin and end time. The system checks begin and end time for validity.
    - Upon choosing the “room” feature, the system shows the list of available rooms. The user can read the details of a meeting room and confirm one of the meeting rooms.
    - The user can enter e-mails of users to be invited. The user can choose the option to check if there is a time conflict of one of the invitees. The user can start the user case “meeting management assistant” to solve a time conflict.
2. The user triggers to save the meeting. The system enters the meeting into the calendar. If

Beispiele – Graphikorientierte Beschreibungstechniken

- Klassendiagramme
- Sequenzdiagramme
- Zustandsdiagramme
- Petrinetze
- u.v.m.



Beispiele – Formale Methoden

- Methoden zur Spezifikation sequentieller Systeme
  - prädikative Beschreibung von Vor- und Nachbedingungen, Invarianten
  - algebraische Spezifikationen
- Methoden zur Spezifikation verteilter Systeme
  - Zustandstransitionssysteme (z.B. Statecharts)
  - temporale Logik
  - Petrinetze
  - Prozessalgebren
  - Aktivitätsdiagramme
  - …

```
{ true} =>
{ 1 = 7⁰ ∧ 0 <= 4}
int i = 7; int j = 4;
int p = 1;
int k = 0;
{ p = iᵏ ∧ k <= j}
while (k < j) {
      { k < j ∧ p = iᵏ ∧ k <= j} =>
      {p*i = iᵏ⁺¹ ∧ k+1 <= j}
      p = p*i;
      {p = iᵏ⁺¹ ∧ k+1 <= j}
      k = k+1;
      {p = iᵏ ∧ k <= j}

}
{ j <= k ∧ p = iᵏ ∧ k <= j} =>

{ k = j ∧ p = i ᵏ}
```

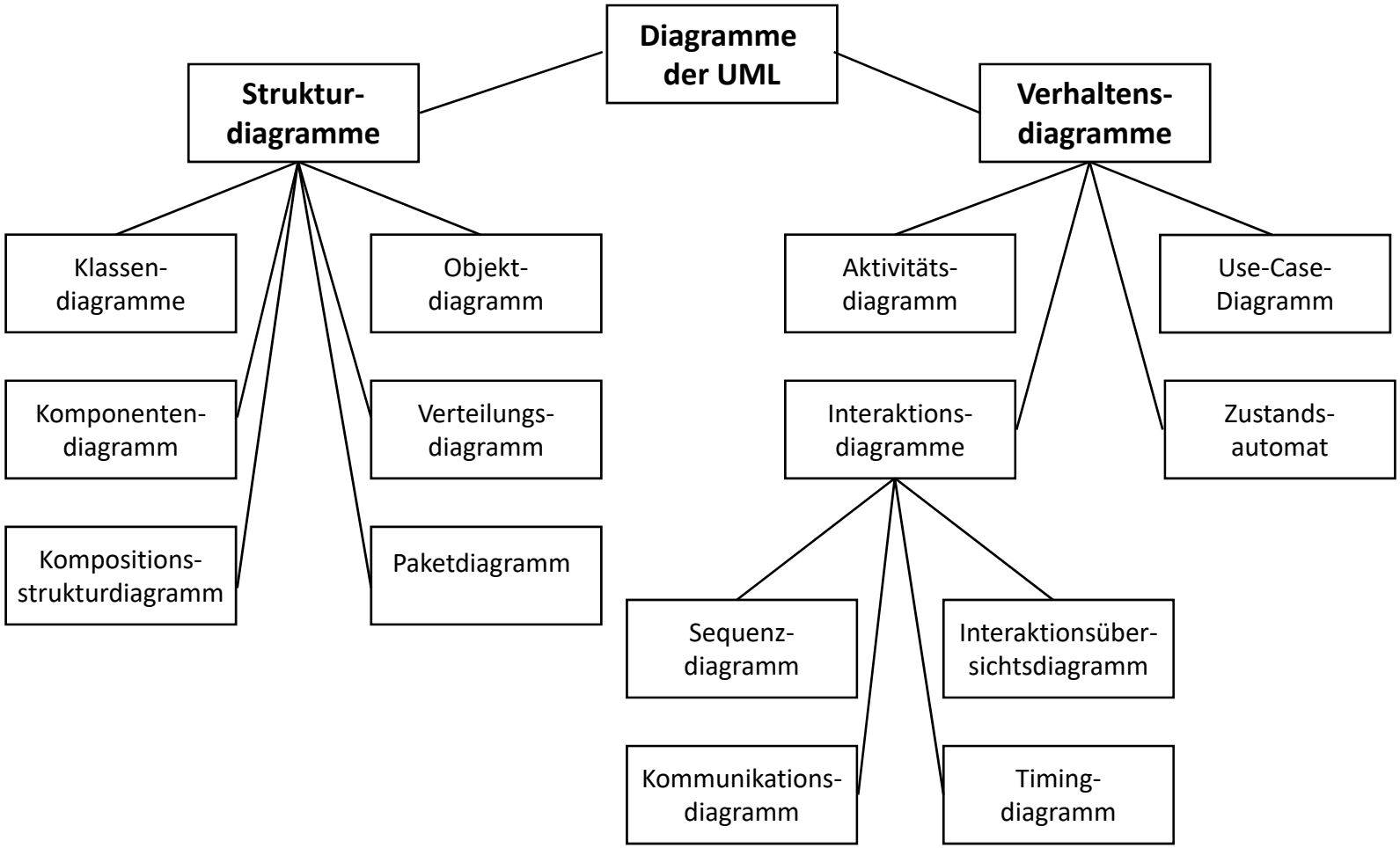
Objektorientierte Entwurfsmethoden

- UML (Unified Modeling Language) ist eine standardisierte Sprache zur objektorientierten Modellierung von Systemen
  - definiert eine Reihe von Diagrammtypen
  - enthält außerdem eine prädikative Sprache (OCL)
- Viele auf dem UML-Standard aufbauende weitere Sprachen
  - Modellierung von Hard-/Softwaresystemen, Geschäftsprozessen, IT-Architekturen, Softwareentwicklungsprozessen, …

Ansprüche von UML

- Anwendungsunabhängigkeit
  - Von Realzeitsystemen bis zu Business Information Systems
- Unabhängigkeit von Zielsprachen
  - UML definiert keine bzw. wenig programmiersprachliche Konzepte (z.B. Basistypen)
- Unabhängigkeit von Toolherstellern
  - UML ist eine Sprache mit einem öffentlich zugänglichen Metamodell
- Anpassbarkeit
  - UML beinhaltet Konzepte, mit denen Modellelemente an bestimmte Anforderungen angepasst werden können
  - Beispiel: spezielle graphische Symbole für bestimmte Arten von Klassen

Die wichtigsten Diagrammtypen



NB

- Nicht alle im folgenden besprochenen Notationen sind UML-konform
- Die vorgestellten Konzepte orientieren sich an allgemeinen Techniken und Prinzipien objektorientierter Modellierung