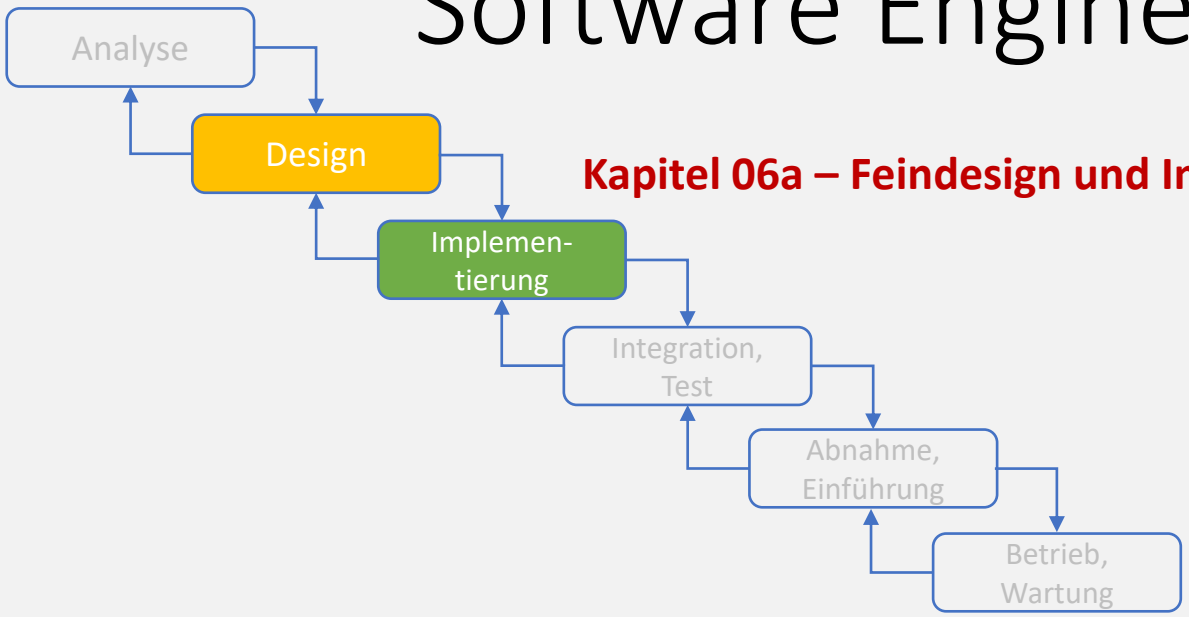
Software Engineering



Kapitel 06a – Feindesign und Implementierung

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

Gliederung

1. Motivation
2. Detailaspekte der Implementierungsphase
3. Design Patterns

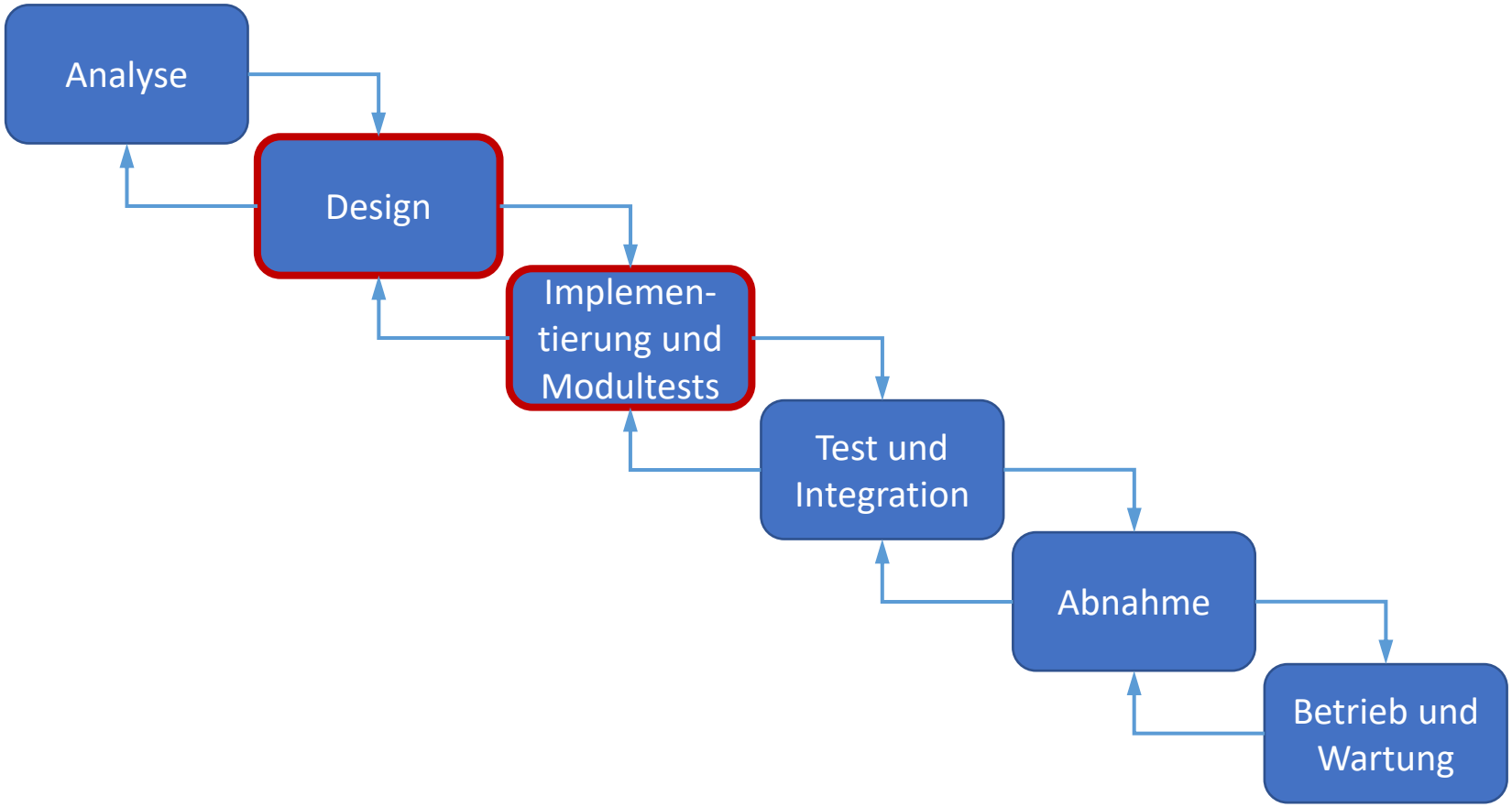
Motivation

Motivation

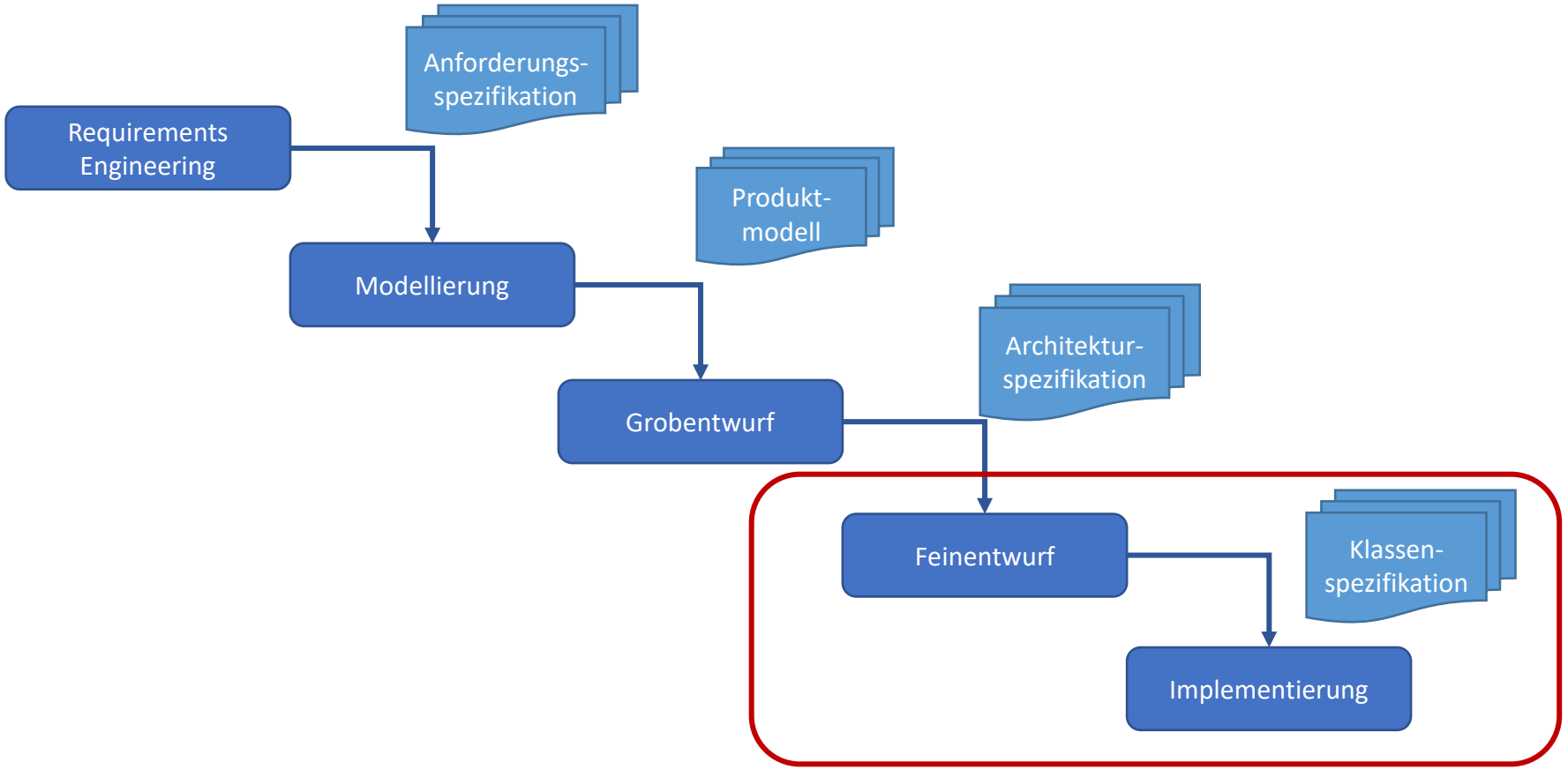
- Feindesign und Implementierungsphase sind die Teile eines Software Engineering Prozesses, in denen ausführbare Programme entstehen.

- Aktivitäten des Feindesigns und der Implementierung sind eng miteinander verbunden
  - Im Feindesign werden Klassen und ihre Relationen definiert
  - In der Implementierung werden Klassen realisiert

Phasen eines Software-Projektes



Von der Anforderung zur Implementierung

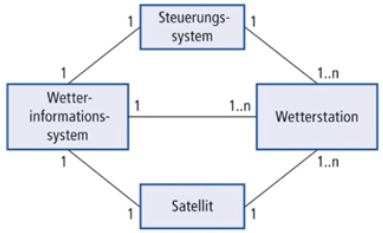


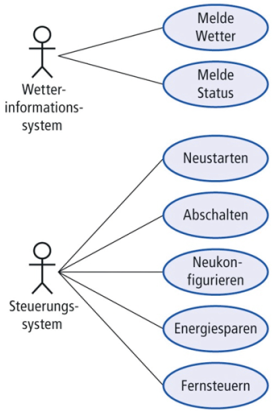
Wie entsteht die lauffähige Software?

Es gibt verschiedene Vorgehensweisen auf dem Weg von der Anforderung zur Implementierung.

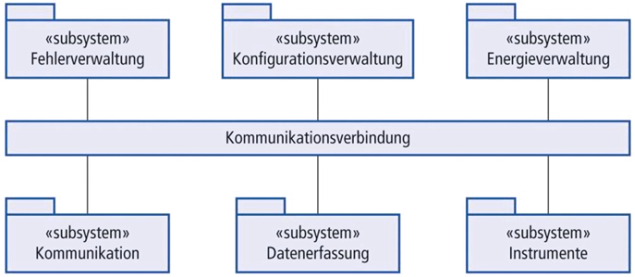
Objektorientierter Entwurf (wie bisher besprochen) beinhaltet:

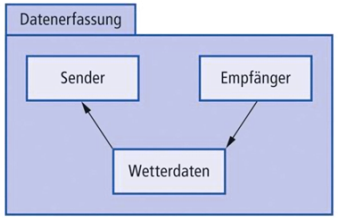
- Kontext + externe Interaktionen des Systems definieren:





- Systemarchitektur entwerfen:



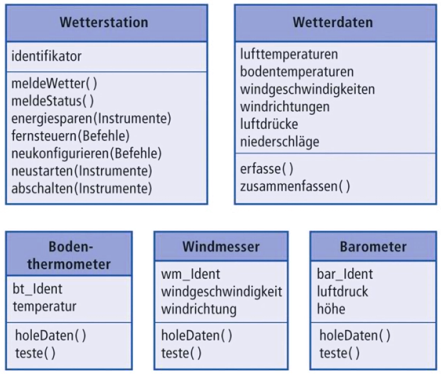


Wie entsteht die lauffähige Software?

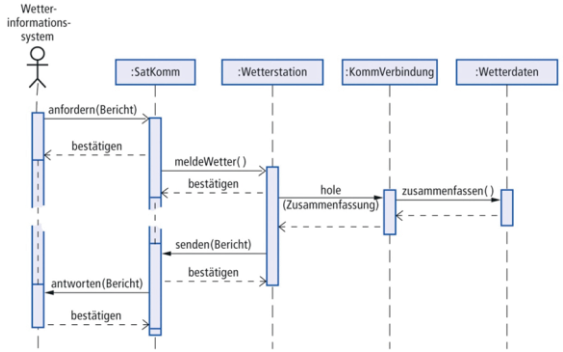
Es gibt verschiedene Vorgehensweisen auf dem Weg von der Anforderung zur Implementierung.

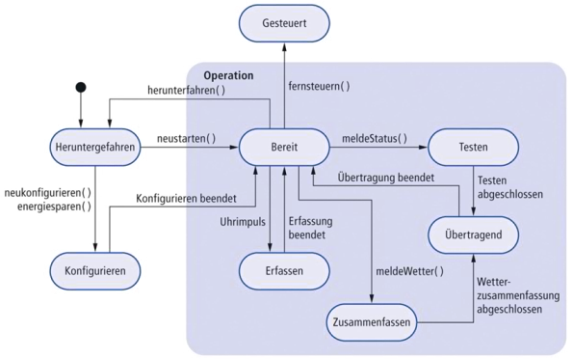
Objektorientierter Entwurf (wie bisher besprochen) beinhaltet:

- Bestimmung der Objektklassen:



- Entwurfsmodelle entwickeln (Architektursichten):





Wie entsteht die lauffähige Software?

Es gibt verschiedene Vorgehensweisen auf dem Weg von der Anforderung zur Implementierung.

Objektorientierter Entwurf (wie bisher besprochen) beinhaltet:

- Schnittstellenbestimmung:

- und anschließend?

```
            trashFound.AppendLine(line);
            linesProcessedSoFar.AppendLine(line);
            continue;


        f (trimmedLine.StartsWith(LaTeXSeparators.BEGIN_CHAPTER) || trimmedLine.StartsWith(LaTeXSeparators.BEGIN_CHAPTER_NO_NUMBER))

            // save all current theory under the old mainTopicID
            if (this.readingTheorySection)
            {
                this.theoryCollection.Add(Guid.NewGuid(), new Tuple<string, Guid>(this.theoryFound.ToString(), currentTopicID));
                this.theoryFound.Clear();
            }
            else if (this.readingExercisesSection)
            {
                this.exercisesCollection.Add(Guid.NewGuid(), new Tuple<string, Guid>(this.exercisesFound.ToString(), currentTopicID));
                this.exercisesFound.Clear();
            }

            // extract the chapter name here
            this.chapterName = trimmedLine.StartsWith(LaTeXSeparators.BEGIN_CHAPTER) ? trimmedLine.Replace(LaTeXSeparators.BEGIN_CHAPTER, string.Empty) : trimmedLine.Replace(
            var openBraces = 1;
            var closingIndex = 0;
            for (int i = 0; i < this.chapterName.Length; i++)
            {
                if (this.chapterName[i] == '{')
                {
                    openBraces++;
                }
                else if (this.chapterName[i] == '}')
                {
                    openBraces--;
                }

                if (openBraces == 0)
                {
                    closingIndex = i;
                    break;
                }
            }
            this.chapterName = chapterName.Substring(0, closingIndex);
        lse if (trimmedLine.StartsWith(LaTeXSeparators.BEGIN_THEORIE_NUMBERED_SUBSECTION) || trimmedLine.StartsWith(LaTeXSeparators.BEGIN_THEORIE_SUBSECTION))

            justBeganSubsection = true;
```

Detailaspekte der Implementierungsphase

Detailaspekte der Implementierungsphase

- Wiederverwendung
- Konfigurationsverwaltung
- Host-Ziel-Entwicklung
- Werkzeuge und Entwicklungsplattformen

Wiederverwendung

- Zwischen 1960 und 1990:
  - Viel Neuentwicklung, kaum Wiederverwendung
  - Höchstens in Programmiersprachenbibliotheken
- Dieser Ansatz wurde immer weniger tragfähig:
  - Wegen zunehmenden Kosten, Termindruck
- Aktuell:
  - Immer mehr Wiederverwendung
  - Auf ganz unterschiedlichen Ebenen

Wo überall?

Wiederverwendungsebenen

- Abstraktionsebene:
  - Keine direkte Wiederverwendung von Software
  - Sondern Nutzung erfolgreicher Entwurfs- und Architekturmuster
- Objektebene:
  - Verwendung von Objekten aus Bibliotheken
  - Dazu: Auffinden von Bibliotheken, die gewünschte Funktionalität bieten
- Komponentenebene:
  - Verwendung von Frameworks
- Systemebene:
  - Wiederverwendung von gesamten Anwendungssystemen
  - Üblich: Neukonfiguration des Systems

Vorteile und Kosten von Wiederverwendung

- Vorteile: Systeme können:
  - Schneller
  - Mit weniger Entwicklungsrisiko
  - Kostengünstiger
  - Zuverlässiger (da bereits in anderen Anwendungen getestet)

  entwickelt werden

- Mögliche Kosten bei wiederverwendbaren Systemen:
  - Kosten durch Suchen und Evaluierung wiederverwendbarer Software
  - Kaufkosten wiederverwendbarer Software
  - Kosten für Customizing und Konfiguration
  - Kosten für Integration von Komponenten verschiedener Hersteller

Konfigurationsverwaltung/ Konfigurationsmanagement

Definition: Konfigurationsverwaltung:

- Ist der Prozess, ein sich veränderndes Softwaresystem zu verwalten

Ziel:

- Unterstützung des Systemintegrationsprozesses, so dass:
- Alle Entwickler greifen auf kontrollierte Art auf Code und Dokumentation zu
- Alle können herausfinden, welche Änderungen wann gemacht wurden
- …

Was gehört alles dazu?

Konfigurationsverwaltung/ Konfigurationsmanagement

Grundlegende Aktivitäten:

- Versionsmanagement:
  - Verwaltung und Kontrolle verschiedener Versionen der erstellten Artefakte
  - Koordinierung der Entwicklung von mehreren Programmierern
  - Verhindert Codeüberschreibungen von unterschiedlichen Programmierern
- Systemintegration:
  - Unterstützung von Entwicklern bei Festlegung, welche Softwareversion welche Komponenten verwendet
  - Soll automatisierte Erzeugung eines Systems ermöglichen (Buildsysteme)
- Problemverfolgung:
  - Tracking von Programmierfehlern und anderen Problemen
  - Verfolgung, wer gerade welche Probleme löst, wann sie korrigiert sein werden, etc.

Host-Ziel-Entwicklung

- Üblicherweise Entwicklung von Systemen auf einem Computer (Host) und Betrieb auf einem anderen Computer (Ziel/Target)

Welche Probleme können auftreten?

- **Allgemein:**
  - Development Platform
  - Production/Execution Platform
  - Dabei ist Plattform mehr als nur Hardware (z.B. OS, DBMS, IDE, …)

Werkzeuge von Entwicklungsplattformen

Aktuelle Entwicklungsplattformen beinhalten in der Regel:

- Syntaxorientierter Editor (Code-Erstellung, -Editierung)
- Integrierter Compiler (Code-Kompilierung)
- Debugger
- Graphische Bearbeitungstools (z.B. UML-Tools)
- Testwerkzeuge (z.B. CppUnit/JUnit zur Unterstützung automatisierter Entwicklertests)
- Weitere Werkzeuge, z.B.:
  - Analyseprogramme
  - Profiler,
  - Werkzeuge für Softwaremetriken und zur Projektunterstützung
  - Dokumentationswerkzeuge (Javadoc, Doxygen, …)

Literatur

Bilder aus:

- Software Engineering, I. Sommerville, Pearson 2014