

# Software Entwicklungsprozesse

## Prinzipien des Software Engineering

Dadurch soll eine **bessere Wartbarkeit** und eine **einfachere Anpassbarkeit** erreicht werden

- **Zufriedenheit** des Kunden hat höchste Priorität
- **Anforderungsänderungen** können auch in fortgeschrittenen Entwicklungsstadien auftreten.
- **Kommunikation** von Angesicht zu Angesicht ist der effiziente Weg, um

---

### 1. Abstraktion

Identifizierung und Trennung wichtiger und (zunächst) weniger wichtige Aspekte.

- Notwendig, um *Komplexität* handhaben zu können
- Anstatt ein konkretes Problem direkt zu lösen besser ein allgemeineres Problem lösen und dies als Grundlage zu verwenden
- Viele **Abstraktionen** erfordern eine weitere *Klassifizierung*
- *erhöht* die **Wiederverwendbarkeit**, aber auch die **Komplexität**
- **Arten:**
  - **Hierarchie:** Verwandten Elementen erlauben, Eigenschaften und Implementierung zu teilen
  - **Polymorphie:** Verwandten Elementen erlauben, ähnliches Verhalten über ähnliche Schnittstellen zu zeigen
  - **Muster:** Ausnutzen wiederkehrender Arten von Beziehungen zwischen Elementen

---

### 2. Modularisierung

beschreibt das Strukturieren eines Produkts oder Systems in Form von austauschbaren Funktionsbausteinen.

- **Strategien:** *Bottom up* und *Top Down*
- **Arten:**
  - **Teile und Herrsche:** Die *Wiederverwendung* von implementierten und verwendeten

Modulen erhöht Produktivität

- **High module cohesion:** Es sollte eine *enge Verbindung der Elemente* innerhalb eines Moduls geben
- **Low inter-modular coupling:** Zwischen separaten Modulen sollten *so wenig Verbindungen wie möglich* existieren

---

### 3. Kapselung

**Verbergen und Schützen** von (inneren) Details der Implementierung eines Moduls

**Nachteil:** Erschwert ggf. die Testbarkeit von Modulen und die Fehlersuche

---

### 4. Hierarchische Zerlegung

Zerlegung eines Problems in Teilprobleme, so dass sich eine Baumstruktur ergibt.

Funktioniert durch: **Abstraktion, Modularisierung** und **Trennung von Zuständigkeiten**.

---

### 5. Trennung von Zuständigkeiten

Aufteilung eines Problems in **unabhängige Teilaufgaben** zur Reduzierung der Gesamtkomplexität

Unter anderem wird dafür das Architekturmuster **MVC** genutzt

---

### 6. Einheitlichkeit

**Vereinheitlichung** von Strukturen, Schemata, Muster, Vorgehensweisen und Entwurfsentscheidungen

- *Komplexität* von Software beherrschbar machen und die *Wartbarkeit* erhöhen
- 

## Agile Softwareentwicklung

“ • **Individuals and interactions** over processes and tools

- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

---

## Scrum

Ein **agiler Prozess** für Softwareentwicklung und Projektmanagement

- Scrum-Projekte werden schrittweise entwickelt
- Jedes Inkrement ist ein Zeitrahmen von üblicherweise 30 Kalendertagen und wird Sprint genannt
- Während eines **Sprints** arbeitet das Team fokussiert und ohne äußere Störungen
- Innerhalb jedes Sprints entwickelt das Team ein Inkrement von potenziell an den Kunden auslieferbarer Funktionalität
- jeden Tag gibt es ein **Daily Scrum Meeting** und am Ende ein **Sprint Review Meeting**
- Rollen: **Product Owner, Scrum Master, Team** und **Stakeholder**
- Scrum erfordert hochmotivierte und allgemein erfahrene Softwareentwickler

---

## 3. Allgemeine Softwareentwicklungsprozesse

- **Wasserfallmodell**: einfach, strikt
- **iteratives Modell**: Rückwärts gehen, geht nichts gleichzeitig
- **Spiralmodell**: alle Parteien zusammen, nicht einheitlich
- **evolutionäres Modell**: Beteiligung aller, schwere Kostenschätzung

---

Revision #4

Created 3 September 2022 14:53:31 by Merith Holtmann

Updated 21 December 2022 09:42:10 by Merith Holtmann