

## Aufgabe 1: Grundlagen des Software Engineering

Bitte beantworten Sie die folgenden Fragen, indem Sie in Software Engineering Büchern oder im Internet recherchieren.

a) Welche Merkmale zeichnen **gute** Software aus?

Software Engineering soll zum einen natürlich dazu führen, Software-Entwicklungsprojekte erfolgreich durchzuführen, zum anderen soll aber die dabei entstehende Software **qualitativ hochwertig** sein.

**Welches sind die wesentlichen Merkmale guter Software?**

- Robustheit, Korrektheit

Fehler (z.B. durch Falscheingaben) sollten vermieden werden oder von der Software "zu verkraften" sein

- Verständlichkeit, Benutzerfreundlichkeit

eindeutige Bezeichnungen, ergonomische Konzeption (Anordnung der Elemente)

- Wartbarkeit

logischer Aufbau, kein "Spaghetticode", möglicherweise modular

- Laufzeiteffizienz

Algorithmen mit niedriger Laufzeit

- Vollständigkeit

eingehaltenes Konzept ohne Lücken

b) Beschreiben Sie bitte den Unterschied zwischen **Systemsoftware** und **Anwendersoftware**.

Der Begriff **Systemsoftware** umfasst alle System- und Betriebsprogramme, die Schnittstellen zur Verfügung stellen und damit „nur“ den Betrieb ermöglichen, aber noch keinen sichtbaren „Nutzen“ bringen. Somit gehören dazu beispielsweise Betriebssysteme, Middleware, Programmierwerkzeuge und Datenverwaltungssoftware.

Die **Anwendersoftware** führt für Anwender nützliche Funktionen direkt aus, z.B. Buchhaltung, Informationssysteme, Textverarbeitung, Tabellenkalkulation.

c) Stellen Sie die **Interessengruppen (Stakeholder)** eines von Ihnen durchgeführten oder aber auch fiktiven Projektes gegenüber. Welche **Interessenkonflikte** bestehen zwischen den Gruppen?

Interessengruppen:

Management des Kunden

Mitarbeiter des Kunden

Management des Auftragnehmers

Mitarbeiter des Auftragnehmers

Konflikte:

Ablaufeffizienz, Kostensenkung

Arbeitsplatzsicherung, Erhalt der betrieblichen Abläufe

erfolgreiche Projektbeendigung, viel Einsatz der AN

erfolgreiche Erledigung der Aufgaben,

Selbstverwirklichung

D) Was versteht man unter Software-Entwicklung nach dem strukturierten Paradigma?  
 In den 80er Jahren wurden die so genannten strukturierten Methoden zur Software-Erstellung entwickelt (SA = Strukturierte Analyse, SADT = Structured Analysis And Design Technique).

**Worin bestehen die wesentlichen Ideen des strukturierten Paradigmas?**

**Worin unterscheidet sich das objektorientierte Paradigma vom strukturierten Paradigma?**

Strukturiertes Paradigma	Objekt-Orientiertes Paradigma
2. Spezifikations- (Analyse-) phase <ul style="list-style-type: none"> <li>• Bestimme, was das Produkt macht</li> </ul>	2*. Objekt-orientierte Analysephase <ul style="list-style-type: none"> <li>• Bestimme, was das Produkt macht</li> <li>• <b>Bestimme die Objekte</b></li> </ul>
3. Designphase <ul style="list-style-type: none"> <li>• Architektur Design</li> <li>• Detailentwurf</li> </ul>	3*. Objekt-orientierte Designphase <ul style="list-style-type: none"> <li>• Feinentwurf</li> <li>• <b>Entwerfe die Objekte</b></li> </ul>
4. Implementierungsphase <ul style="list-style-type: none"> <li>• Implementiere in geeigneter Programmiersprache</li> </ul>	4*. Objekt-orientierte Programmierphase <ul style="list-style-type: none"> <li>• Implementiere in geeigneter objekt-orientierter Programmiersprache</li> </ul>

## Strukturierte Phasen vs. OO-Phasen

Strukturiertes Paradigma	Objekt-Orientiertes Paradigma
1. Anforderungsphase	1. Anforderungsphase
2. Spezifikations-/Analyse-Phase	2*. <i>Objekt-orientierte Analysephase</i>
3. Designphase	3*. <i>Objekt-orientierte Designphase</i>
4. Implementierungsphase	4*. <i>Objekt-orientierte Programmierphase</i>
5. Integrationsphase	5. Integrationsphase
6. Wartungsphase	6. Wartungsphase
7. Rückzug	7. Rückzug