Aufgabe 2.1: 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?

1. **Funktionalität (Functionality):** inhaltliche Übereinstimmung zwischen Anforderungen und Funktionalität der Applikation
   (Kriterien: Angemessenheit, Richtigkeit, Interoperabilität, Sicherheit, Ordnungsmäßigkeit)
2. **Zuverlässigkeit (Reliability):** Fähigkeit der Software, ihr Leistungsniveau unter festgelegten Bedingungen über einen festgelegten Zeitraum zu bewahren.
   (Kriterien: Reife, Fehlertoleranz, Robustheit, Wiederherstellbarkeit)
3. **Benutzbarkeit (Usability):** Software muss ohne unangemessene Anstrengungen vom Benutzer nutzbar sein
   (Kriterien: Verständlichkeit, Erlernbarkeit, Bedienbarkeit, Attraktivität)
4. **Effizienz (Efficiency):** Software sollte nicht verschwenderisch mit Systemressourcen wie Speicher und Prozessorkapazität umgehen. Effizienz umfasst u.a. Reaktionszeit, Verarbeitungszeit, Speichernutzung
   (Kriterien: Zeitverhalten, Verbrauchsverhalten)
5. **Änderbarkeit (Maintainability):** Software sollte so geschrieben werden, dass sie weiterentwickelt werden kann, um veränderten Kundenbedürfnissen Rechnung zu tragen (Kriterien: Analysierbarkeit, Modifizierbarkeit, Stabilität, Testbarkeit)
6. **Übertragbarkeit (Portability):** Übertragung auf andere Systemumgebung
   (Kriterien: Anpassbarkeit, Installierbarkeit, Koexistenz, Austauschbarkeit)

b) Beschreiben Sie den Unterschied zwischen **Systemsoftware** und **Anwendungssoftware**. Geben Sie jeweils Beispiele an.

- **Systemsoftware** wird üblicherweise eine Software bezeichnet, die im Hintegrund ohne direkte Benutzerinteraktion operiert. Systemsoftware wird benötigt, um mit der Hardware zu kommunizieren und in weiterer Folge auch, um Anwendungssoftware ausführen zu können.
  Beispiele sind: Betriebssystem, Treiber, ...

- **Anwendungssoftware** hingegen wird von Anwendern installiert und benutzt, um spezifische Aufgaben zu erledigen. Diese verfügen über eine GUI, mit dem Benutzer interagieren können.
  Beispiele sind: Microsoft Word, Google Chrome, ...

c) **Softwareprodukte (= Standardsoftware**) decken einen klar definierten Anwendungsbereich ab (z.B. Buchhaltung, Lagerverwaltung) und können als vorgefertigtes Produkt gekauft oder gemietet werden. **Individualsoftware (Custom Software)** wird vollständig neu speziell für einen Kunden bzw. ein Unternehmen entwickelt (z.B. Zugdisposition der DB).
Geben Sie jeweils die Vorteile der beiden Arten von Software an.

- **Vorteile der Softwareprodukte:**
  - Verfügbarkeit ist sofort da, bei IS kann dieses deutlich länger dauern.
  - Deutlich günstiger, man hat nur Lizenzkosten
  - Ständige Weiterentwicklung, regelmäßige Updates
  - i.d.R. weniger fehleranfällig
  - Support inklusive

- **Vorteile der Individualsoftware**
  - Sehr hohe Passgenauigkeit durch maßgeschneiderte Abbildung von Geschäftsprozessen, „richtiger“ Funktionsumfang (nicht zu viel, nicht zu wenig) Anpassungen können jederzeit angepasst werden
  - Nahtlose Integration
  - Nachträgliche Erweiterungen und Änderungswünsche sind jederzeit möglich.
  - Einmalige Projektkosten
  - Updates selber bestimmen
  - Unabhängigkeit
  - Es gibt nicht immer eine Softwareprodukt das benutzt werden kann daher IS

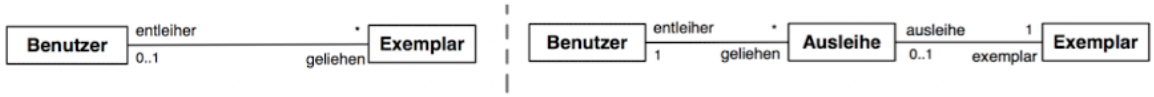
d) Stellen Sie die **Interessengruppen (Stakeholder**) eines von Ihnen durchgeführten oder aber auch fiktiven Softwareentwicklungsprojektes zwischen einem **Auftraggeber** (z.B. ein Verkehrsunternehmen) und einem **Auftragnehmer** (z.B. ein Softwarehaus) gegenüber. Welche **Interessenkonflikte** bestehen zwischen den Gruppen?

Auftraggeber:
- wenig IT-Kosten
- schnelle Projektergebnis
- Endanwender*innen:
  - Angst um Arbeitsplatz;
  - gewohnte Abläufe beibehalten
  - effizient arbeiten
- Management der Anwender:
  - Kosten verringern
  - Prozesse optimieren
  - Termine halten

Auftragnehmer:
- Gewinn erzielen
- Entwickler*innen:
  - „gute“ Software bauen
  - interessante Technologien verwenden
  - ohne Druck arbeiten
- Management des Softwarehauses:
  - Umsatz-/Gewinnziele
  - Termine halten
  - Kosten gering
- Vertriebsmitarbeiter*innen:
  - neue Projekte akquirieren

Aufgabe 2.2: UML (Fortsetzung)

(a) Im Bibliotheksbeispiel aus der vorigen Übung lässt sich die Beziehung zwischen Benutzer und Exemplar auf zweierlei Weise modellieren.

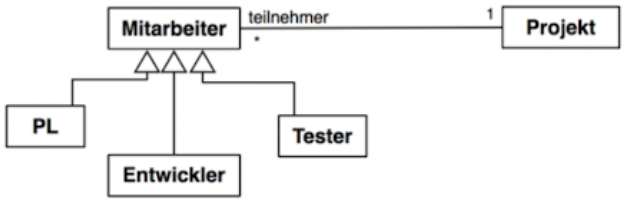


Welcher der beiden Ansätze ist tragfähiger? Nennen Sie die Vor- und Nachteile der Ansätze!

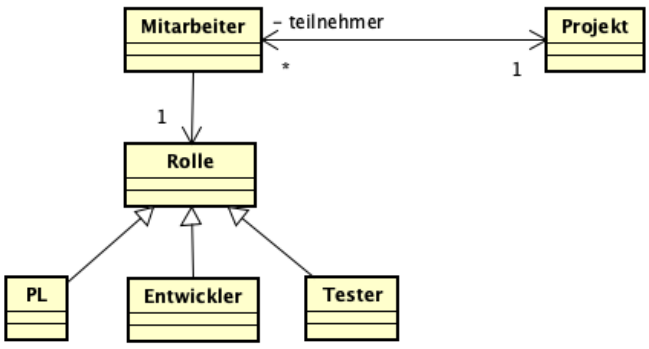
1. Im ersten Exemplar haben wir das Ausleihen nur als eine Beziehung implementiert:
    a. Vorteil:
        i. Klassenmodell simpler
    b. Nachteil:
        i. Details zur Ausleihe fehlen z.B. Datum, Ausleihzeitraum, Versäumnisse usw.

2. Im zweiten Exemplar sehen wir noch ein extra Klasse für das Ausleihen:
    a. Vorteil:
        i. Andere Meta-Daten können in die Klasse geschrieben werden mittels Attribute
            - z.B. Ausleihzeitpunkt, Rückgabedatum, ...
    b. Nachteil:
        i. Komplizierter: Mehr Klassen und Attribute
        ii. Größerer Speicherplatzverbrauch

(b) In einem UML-Klassendiagramm soll ein Projekt mit seinen Beteiligten modelliert werden. Dazu wird zunächst das folgende Modell entwickelt, das insbesondere die verschiedenen Rollen im Projekt abbildet.



Nun soll es aber möglich sein, dass die Rollen im Projekt dynamisch wechseln können. Zum Beispiel kann dieselbe Person während des Projektes zeitweise als Entwickler*in aber auch als Tester*in eingesetzt werden. Wie lässt sich das sinnvoll im Modell abbilden?



(c) Eine Assoziation dient dazu, Objekte zweier Klassen in Beziehung zu setzen. Welche Informationen werden durch nachfolgende Assoziationen ausgedrückt?

1. Polygon und Punkt kennen sich gegenseitig
2. “
3. Ein Polygon besteht aus beliebig vielen Punkten und kennen sich gegenseitig
4. “, Zusätzlich besitzt Polygon ein Attribut von der Klasse Punkt.
5. Polygon kennt beliebig viele Punkte und besitzt ein Attribut von der Klasse Punkt
6. Das Durchkreuzen dient nur der Verdeutlichung das dort keine Beziehung ist. (Selbe wie 5.)
7. Punkte sind existenziell abhängig von einem Polygon.
   (Falls das Polygon gelöscht wird werden die Punkte auch gelöscht!)
8. Das Polygon besteht aus beliebig vielen Punkten.
   (Hier werden die nicht geloescht!)