Ferhan Aydin

Hochschule Hannover
University of Applied Sciences and Arts

Software Engineering 1
Übungsblatt 2

Ausgabe: 18.03.2022 **Besprechung: 24.03.2022**

Ablauf der Übungen

- Erfolgreiches Bearbeiten von 2/3 der Pflicht-Übungsaufgaben und persönliche Vorstellung mindestens einer Aufgabenlösung (auf Nachfrage) berechtigen zur Klausurteilnahme.
- Eine Aufgabe gilt als erfolgreich bearbeitet, wenn eine korrekte Lösung bzw. ein nachvollziehbarer Lösungsversuch termingerecht abgegeben bzw. vorgestellt wurde.
- Die Aufgaben sollen in 2er Gruppen bearbeitet werden.
- Die Abgabe durch **jedes Gruppenmitglied** erfolgt per Moodle (bitte als eine PDF-Datei).
- Das Ergebnis soll in der Übung vorgestellt und diskutiert werden.

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?

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

(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.

(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?

a) · Gute Unterteilung des Programms in Funktionen, um redundanz zu vermeiden
- Code sollte gut dokumentiert sein.
- Die Funktionen sollten die Anforderungen im vollen Umfang erfüllen
- Software sollte auf verschiedenen Systemen laufen.
- Bedienungsfreundlich

b)

Systemsoftware ist die Software die den Betrieb eines Rechners steuert. Stellt verbindung zur Hardware her und steuert dessen Ressourcen (Betriebssysteme)

Anwendungssoftware werden vom Benutzer genutzt um nützliche oder gewünschte Funktionalitäten zu nutzen, um Aufgaben zu bewältigen.

c)

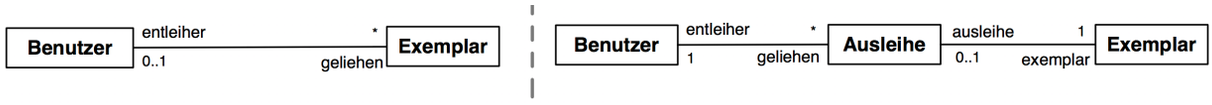
Softwareprodukte werden von vielen Käufern genutzt. So können Fehler und Bugs schnell gefixt werden, aber der Fokus auf Individualität geht verloren und steht immer im Konkurrenzdruck

Individualsoftware konzentriert sich nur auf den Kunden und soll in erster Linie nur bei ihm laufen.

d) Dem Auftraggeber ist in erster Linie die Konkurrenz im Markt wichtig und möchte so viele Funktionen wie möglich abdecken ohne zu wissen was eigentlich möglich und nicht möglich ist.
Der Entwickler hingegen weiß was implementierbar ist und muss dem Auftraggeber erklären was realisierbar ist.
Fehlt jetzt beiderseits ein gewisses Vorwissen, so kann schnell ein Konflikt entstehen.

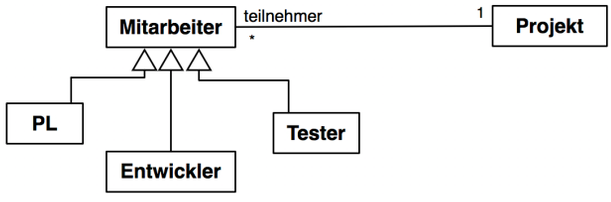
Aufgabe 2.2: UML (Fortsetzung)

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



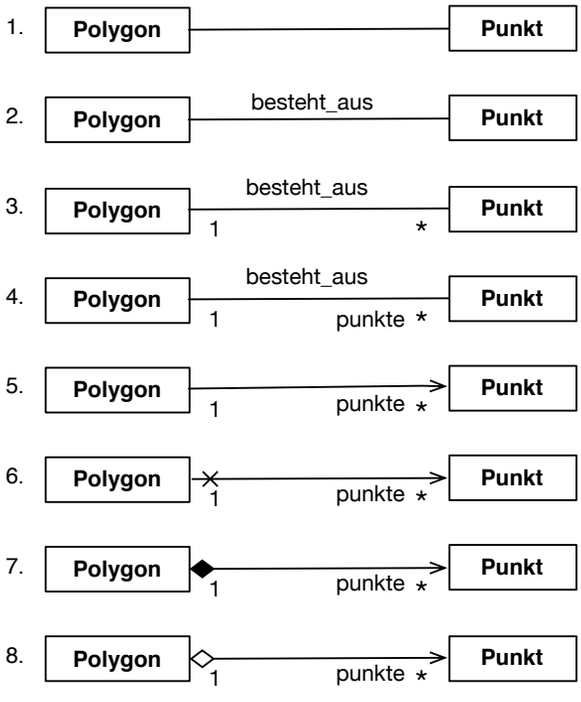
Welcher der beiden Ansätze ist tragfähiger? Nennen Sie die Vor- und Nachteile der Ansätze!

(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?



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!

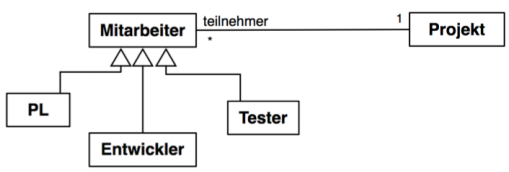
Vorteile:
- einfacher zu verstehen
- ausgeliehenes Exemplar erkennt man sofort.

Nachteile:
- Bei der Zuordnung der Exemplare zu dem Benutzer entsteht redundanz

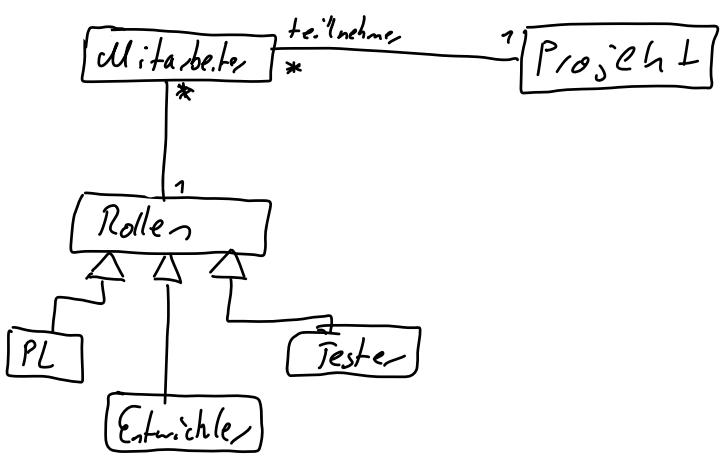
Vorteile: ausgeliehenes Exemplar kann dem Benutzer direkt zugeordnet werden
- Informationen wie z.B. Ausleihdatum usw. möglich, da Beziehung zur Entität wird.

Nachteile:
- komplexer
- Eine Tabelle mehr.

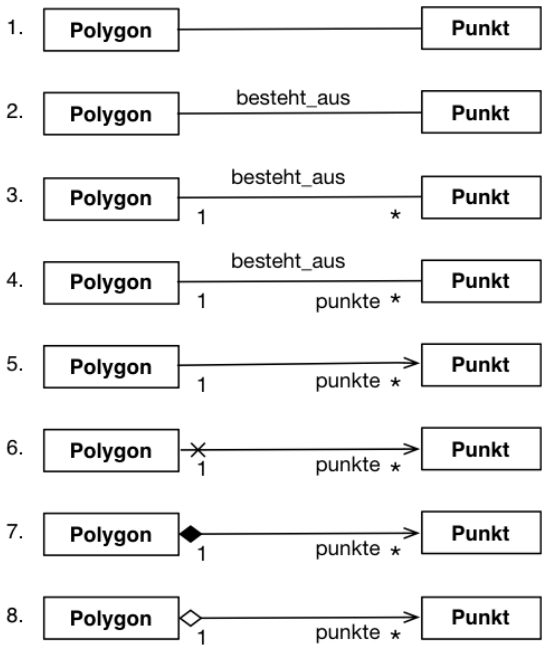
(b) In einem UML-Klassendiagramm soll ein Projekt mit seinen Beteiligten modelliert werden. Dazu wird zunächst das folgende Modell entwickelt, das insbesondere die verschiedenen Rollen im Projekt abbildet.



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. Polygone und Punkt stehen in Beziehung zu einander und kennen sich.

2. Ein Polygon besteht aus einem Punkt

3. Ein Polygon besteht aus beliebig vielen Punkten und beliebig viele Punkte gehören zu einem Polygon

4. Ein Polygon besteht aus beliebig vielen Punkten und beliebig viele Punkte gehören zu einem Polygon. Die Klasse Polygon speichert seine Punkte in einem Attribut namens punkte

5. Ein Polygon besteht aus beliebig vielen Punkten
Die Klasse Polygon speichert seine Punkte in einem Attribut namens punkte.

Durch die Navigierung kennt ein Polygon all seine Punkte, aber die Punkte nicht sein Polygon

6. Verdeutlicht die Navigierung von 5.

7. Punkte sind existenzabhängig von ihrem Polygon (Komposition) d.h. wird das Polygon gelöscht, dann werden auch die dazu gehörigen Punkte gelöscht

8. Ein Polygon besteht aus Punkten (Aggregation). Wird das Polygon gelöscht, dann können die Punkte noch existieren.