



Technische Universität Berlin

Software and Embedded Systems Engineering Group

Prof. Dr. Sabine Glesner

www.sese.tu-berlin.de Sekr. TEL 12-4 Ernst-Reuter-Platz 7 10587 Berlin




Softwaretechnik und Programmierparadigmen WiSe 2022/2023

Prof. Dr. Sabine Glesner
Milko Monecke
Simon Schwan

Übungsblatt 12

Beispielösung



Hinweis: USE-OCL

Ihr könnt eure OCL Ausdrücke und Bedingungen mit USE-OCL der Uni Bremen testen. USE-OCL erlaubt zwar die graphische Darstellung von Klassen- und Objektdiagrammen, die Klassendiagramme müssen jedoch textuell im .use-Format definiert, Objekte werden mit Hilfe von Kommandozeilen-Befehlen angelegt. Um Befehle zum Anlegen und verlinken von Objekten wiederholt ausführen zu können, können Objekt-Skripte verwendet werden. Für diese Übung stehen euch auf ISIS ein vordefiniertes Klassendiagramm (`autowerkstatt.use`) sowie ein Objekt-Skript (`autowerkstatt.soil`), das ein Beispiel-Objektdiagramm erstellt, zur Verfügung. Dort findet ihr außerdem die Datei `autowerkstatt.default.clt`, mit der ein Default-Layout für das Klassendiagramm geladen wird. Modelldateien (`.use`) und die Skripte (`.soil`) können mit einem beliebigen Editor geöffnet und bearbeitet werden. Für das Bearbeiten von Modellen bietet die GUI von USE keine Möglichkeit. Weiteres im Video USE-OCL-Tool-Einführung. 

Um USE-OCL zu verwenden müsst ihr es nur herunterladen, entpacken und dann je nach Betriebssystem eins der Skripte `bin/use` oder `bin/start.use.bat` ausführen. Es öffnen sich dann sowohl eine GUI als auch die Kommandozeile zur Eingabe von Skript-Befehlen. Das Klassendiagramm könnt ihr in der GUI über den Dialog 'File → Open specification' laden, zum Laden des Objekt-Skripts müsst ihr in der Kommandozeile folgendes eingeben:

```
use> open autowerkstatt.soil
```

Schlüssel:

-  Ein ergänzendes Video wird zur Vor- oder Nachbereitung veröffentlicht.
-  Wird im Tutorium besprochen.

In der GUI könnt ihr über den Button “OCL” beliebige OCL-Bedingungen auswerten. Verwendet die in ISIS zur Verfügung gestellten Ressourcen zur Erstellung des Modells.

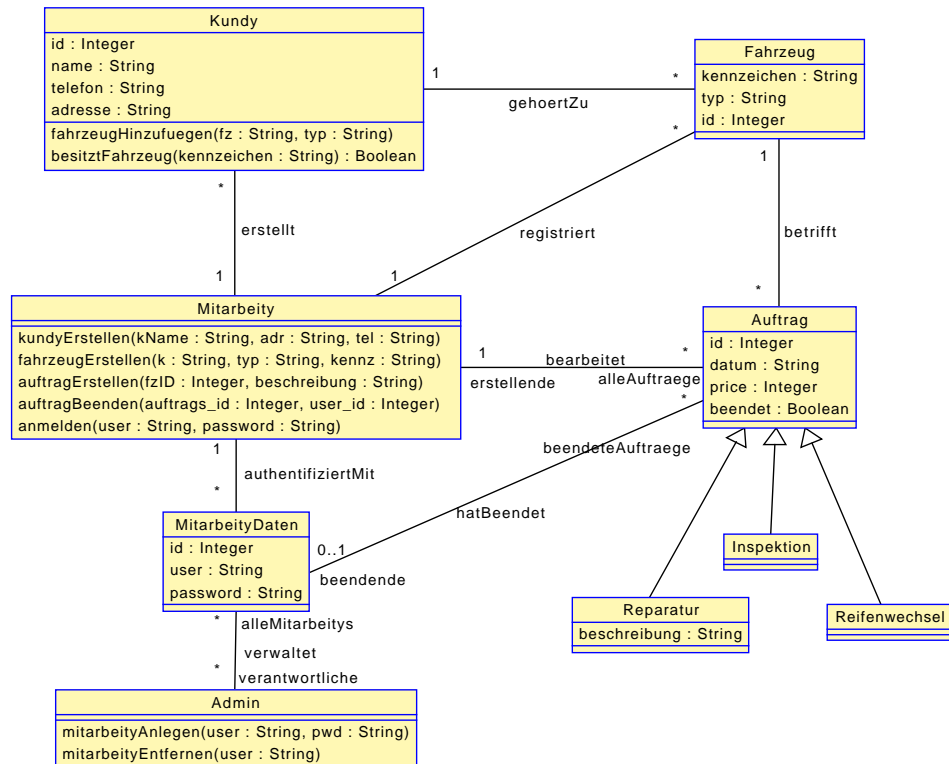


Abbildung 1: Klassendiagramm der Autowerkstatt

Aufgabe 1: Objektdiagramme

Öffnet das Klassendiagramm und den Systemzustand aus den Vorgaben. In der graphischen Oberfläche könnt Ihr das Klassendiagramm und das Objektdiagramm betrachten (Hinweis: zur besseren Übersicht können Controller-Klassen ausgeblendet werden).

- Wodurch unterscheidet sich das Objektdiagramm vom Klassendiagramm? Warum gibt es keine Multiplizitäten?
- Passt der Systemzustand zur Spezifikation? Falls nicht, nehmt entsprechend Änderungen vor.

Lösung: Siehe Abbildung 2. In der Vorgabe fehlten die Verbindungen i2,f2 und f1,c1.

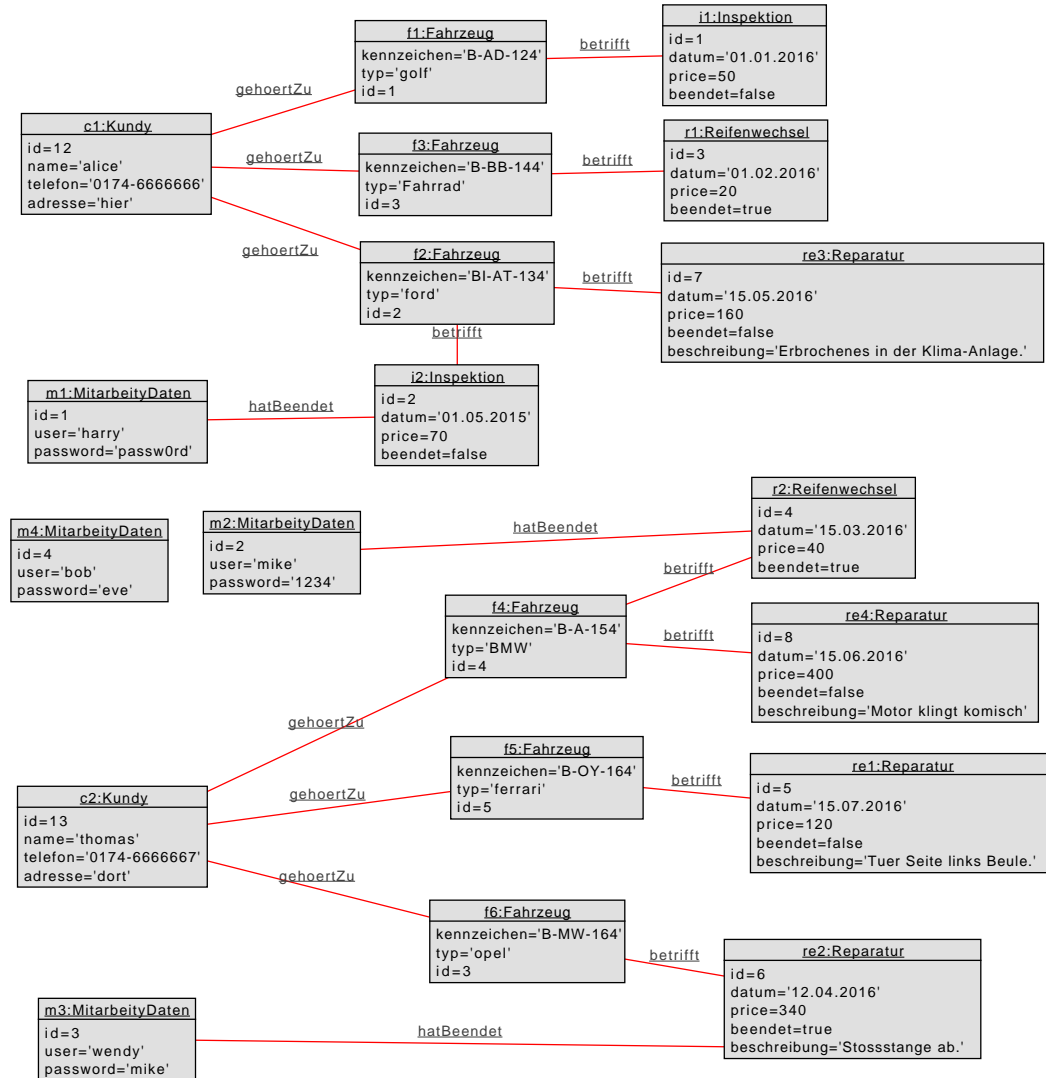










Abbildung 2: Beispiel-Objektdiagramm (nur Entity-Objekte)

Aufgabe 2: OCL Werte

Verwendet OCL um folgende Werte des Systemzustands zu erhalten.








- a) Die Id des Kundys `c1` 
- b) Das Kennzeichen des Fahrzeugs `f1`  
- c) Der Name des Besitzers des Fahrzeugs `f1` 
- d) Das Mitarbeiter, das den Auftrag `r2` beendet hat 
- e) Diejenige Reparatur aus `re3` und `re4` mit dem größeren Preis 
- f) Den ersten Buchstaben des Kennzeichens von Fahrzeug `f1`  

Lösung:

- a) `c1.id`
- b) `f1.kennzeichen`
- c) `f1.kundy.name`
- d) `r2.beendende` (siehe Rollenbezeichner / Assoziationsenden im Klassendiagramm)
- e) `if re3.price > re4.price then re3 else re4 endif`
- f) `f1.kennzeichen.substring(1,1)`

Aufgabe 3: OCL Bedingungen

Logische Ausdrücke sind statisch verifizierbar und werden auf verschiedene Weise eingesetzt. Formalisiert die folgenden logischen Aussagen für das gegebene Klassenmodell.













- a) Sind die Namen der Kundys `c1` und `c2` gleich? 
- b) Gehört das Fahrzeug `f1` dem Kundy `c1`?  
- c) Hat das Kennzeichen des Fahrzeugs `f1` einen nicht-leeren Wert? 
- d) Ist der Auftrag `i1` eine Inspektion? 
- e) Ist das Objekt `i1` ein Auftrag?  

Lösung:

- a) `c1.name = c2.name`
- b) `f1.kundy = c1` (hier noch auf Mengen verzichten)
- c) `f1.kennzeichen <> ''` oder `f1.kennzeichen.size() > 0`
Hier auch möglich Undefined (null) zu berücksichtigen:
`f1.kennzeichen <> Undefined and f1.kennzeichen <> ''` oder
`f1.kennzeichen <> null and f1.kennzeichen <> ''`
- d) `i1.oclIsTypeOf(Inspektion)`
- e) `i1.oclIsKindOf(Auftrag)`

Aufgabe 4: OCL Collections

Definiert die folgenden Mengen:









- a) Alle Fahrzeuge im System 
- b) Die Fahrzeuge von c1 
- c) Die Fahrzeuge von c1 und das Fahrzeug f4 
- d) Alle Fahrzeuge von c1 und c2 
- e) Die Fahrzeuge die gleichzeitig c1 und c2 gehören.  
- f) Die Typen aller Fahrzeuge des Kundys c1. 
- g) Die Preise aller Aufträge im System.  
- h) Die Anzahl der Aufträge für das Fahrzeug f4. 
- i) Die Anzahl der Kundys mit der ID f12  

Lösung:

- a) `Fahrzeug.allInstances()`
- b) `c1.fahrzeug`
- c) `c1.fahrzeug->including(f4)`
- d) `c1.fahrzeug->union(c2.fahrzeug)`
- e) `c1.fahrzeug->intersection(c2.fahrzeug)`
- f) `c1.fahrzeug.typ`
- g) `Auftrag.allInstances().price`
- h) `f4.auftrag->size()`
- i) `Kundy.allInstances.id->count(12)`

Aufgabe 5: OCL Aussagen

Überprüft mithilfe von OCL ob folgende Aussagen über den Systemzustand stimmen. Beginnt mit der Navigation immer bei der Mitarbeiter-Controller Instanz **m**.








- a) Alle Aufträge sind beendet. 
- b) Mindestens ein Auftrag ist noch nicht beendet. 
- c) Alle Aufträge, die keine Inspektionen sind, sind beendet.  
- d) Alle beendeten Aufträge haben einen Mitarbeiter als Beender vermerkt. 
- e) Alle Aufträge die einen Mitarbeiter als Beender vermerkt haben sind auch beendet. 
- f) Die IDs der Fahrzeuge sind eindeutig.  

Lösung:

- a) `m.alleAuftraege->forall(a:Auftrag | a.beendet)` oder
`m.alleAuftraege->select(not beendet)->size() = 0`
- b) `m.alleAuftraege->exists(not beendet)` oder
`m.alleAuftraege->select(not beendet)->size() > 0` oder
`m.alleAuftraege.beendet->count(true) > 0`
- c) `m.alleAuftraege->select(not oclIsTypeOf(Inspektion))->forall(beendet)` oder
`m.alleAuftraege->forall(not oclIsTypeOf(Inspektion) implies beendet)`
- d) `m.alleAuftraege->select(beendet = true)->forall(beender <> null)`
- e) `m.alleAuftraege->select(beender <> null)->forall(beendet = true)`
- f) `m.fahrzeug.id->asSet()->size() = m.fahrzeug.id->size()`

Aufgabe 6: Rekursion in OCL

Extrahiert folgende Information mit `iterate` oder `closure` aus dem Systemzustand.

- a) Wie viel Geld bringen alle Aufträge zusammen? 
- b) Wie groß ist der Anteil der Inspektionen am gesamten Umsatz in Prozent?  
- c) Erstellt einen String, in dem die Kundys mit den Fahrzeugtypen aller ihrer Autos aufgelistet werden.  
- d) Erstellt ein Set mit allen geraden positiven Zahlen bis 100.  

Lösung:

- a) `m.alleAuftraege->iterate(a;s : Integer = 0 | s + a.price)`
- b) `m.alleAuftraege->select(oclIsTypeOf(Inspektion))->iterate(i;s : Integer = 0 | s + i.price) / m.alleAuftraege->iterate(a;s : Integer = 0 | s + a.price) * 100`
- c) `m.kundy->iterate(k; s : String = '' | s + k.name + ' hat:' + k.fahrzeug->iterate(f; s2 : String = '' | s2 + ' ein ' + f.typ) + '.')`
- d) `Set{0}->closure(i | if i < 100 then (i+2) else (i) endif)`