technische universität dortmund

Fakultät für Informatik
Lehrstuhl 14 für Software Engineering
Prof. Dr. Falk Howar
Simon Dierl, M. Sc.
Softwarekonstruktion WiSe 18 / 19

Übungsblatt 3

Ausgabe: 12.11.2018 **Abgabe:** 22.11.2018, 12:00 Uhr

Organisatorisches

Da die Bearbeitung der Teilaufgabe 1.1.b) leider dadurch erschwert wurde, dass die Inhalte in der Vorlesung nicht bis zum Abgabedatum besprochen worden waren, haben wir uns entschlossen, die für die Zulassung notwendigen Punkte auf 58 herabzusetzen.

Ansonsten gelten die Hinweise von Übungsblatt 2.

Hausübung

Aufgabe 3.1 (Die Modellierungs-Hierarchie) 6 P.

Lesen Sie sich kurz in den XML-Dialekt GraphML[1] ein, der eine Sprache für Graphen beschreibt.

(a) Wir können GraphML als ein Metamodell betrachten. Was wären dann dazugehörende Modelle? 1 P.

(b) Benutzen Sie GraphML als Beispiel, um folgende Begriffe zu erläutern: 3 P.

(i) Abstrakte Syntax
(ii) Konkrete Syntax
(iii) Semantik

Umreißen Sie grob, welche Konzepte *von GraphML* zu den abstrakten Syntax, konkreten Syntax und Semantik gehören. Sie müssen *keine* allgemeinen Definitionen angeben.

Beispiel: „Die konkrete Syntax von Java definiert Schlüsselwörter und die Grammatik der Sprache, z.B. das Schlüsselwort `class`).“

(c) Zu GraphML werden (auf der Webseite) zwei formalisierte Metamodelle (als XML Schema und DTD) bereit gestellt. Welche Metamodelle würden sich ebenfalls mit diesen Sprachen beschreiben lassen? 1 P.

[1] `http://graphml.graphdrawing.org/`

(d) XML Schema[2] selbst ist ein Metametamodell. Was ist das Metametametamodell? 1 P.

Aufgabe 3.2 (Metamodellierung) **8 P.**

Kartenspiele wie Mau-Mau[3] und Uno[4] werden auf sehr ähnliche Weise gespielt: Reihum legen Spieler Karten von ihrer Hand auf einen Stapel ab. Dürfen sie keine Karten ablegen, müssen sie stattdessen eine Karte von einem Stapel ziehen. Gewonnen hat, wer als erstes alle Karten abgelegt hat. Zusätzlich haben einige Karten Sondereffekte (den nächsten Spieler zwei Karten ziehen lassen, den nächsten Spieler aussetzen lassen, eine neue Farbe wählen). Weitere Sondereffekte können Sie vernachlässigen.

(a) Definieren Sie ein Metamodell in Form eines UML-Klassendiagramms, welches gültige *Stellungen*, d.h. Kombinationen aus Spielern, Handkarten, Ablage- und Nachziehstapel für *beliebige* Decks und Regeln definiert. Zu den Regeln gehört: 4 P.

(i) welche Karten aufeinander gelegt werden dürfen und
(ii) welche Karte welchen Sondereffekt hat.

Das „Weiterreichen“ von Sondereffekten können Sie vernachlässigen. Es steht Ihnen frei, Ihre Antwort zur besseren Verständlichkeit zu kommentieren. Vergessen Sie nicht, die Multiplizitäten an *beiden* Enden *aller* Relationen anzugeben.

Hinweis: Sie müssen eine Sonderlösung finden, um Farbwahlen zu modellieren.

(b) Erstellen Sie ein zu Ihrem Metamodell konformes Modell in Syntax eines UML-Objektdiagramms, welches eine (vereinfachte) Partie Mau-Mau[3] mit drei Spielern und sechs Karten (7, 8, 10 in den Farben Herz und Karo) modelliert. Die Handkarten und die Stapel können Sie frei bestimmen. 4 P.

Aufgabe 3.3 (Anforderungs-DSLs) **6 P.**

Um Anforderungen zu modellieren, können wir eine domänenspezifische Sprache definieren. Hierbei deutet (a | b | c) ein Wahl zwischen a, b und c an, (... | ε), dass kein Element gewählt werden muss. *<Dies>* ist ein Platzhalter und `das` ein Schlüsselwort. Whitespace kann beliebig gesetzt werden, die Reihenfolge der Elemente ist aber fest.

Die Bedeutung der Elemente orientiert sich an den Sophisten-Satzschablonen bzw. an der Vorlesung bekannten Untergliederung von Anforderungstypen, fügt jedoch Unterstützung für Unteranforderungen hinzu.

[2] `https://www.w3.org/standards/xml/schema`
[3] `https://de.wikipedia.org/wiki/Mau-Mau_(Kartenspiel)`
[4] `https://de.wikipedia.org/wiki/Uno_(Kartenspiel)`

```
requirement <Anforderungsname>
( user | system | organization | product | external ) {
  ( required-by <Anforderungsname> ; | ε )
  implemented-in ( component <Komponente> | developers | global ) ;
  urgency ( must | should | may ) ( not | ε ) ;
  ( usecase-for <Benutzer oder Rolle> ; | automatic ; | ε )
  description " <Beschreibung> ";
  action <Prozesswort> ;
};
```

Beispiele:

```
requirement GDPRCOMP external { implemented-in global; urgency must;
  description "Bestimmungen der DSGVO"; action einhalten; };
requirement GDPRDEL user { required-by GDPRCOMP;
  implemented-in component Web-UI; urgency must; usecase-for admin;
  description "Stammdaten eines Benutzers"; action löschen; };
```

Versetzen Sie sich jetzt wieder in die Rolle des Projektleiters vom 1. Übungsblatt, der nun ein Personalverwaltungssystem programmieren soll. Benutzen Sie die DSL, um folgende sechs Anforderungen zu formalisieren:

(i) Der Kunde hat einen Implementierungsleitfaden vorgelegt, der Vorgaben zum Entwicklungsprozess macht. Dieser muss ausnahmslos eingehalten werden. Zwei spezifische Anforderungen werden aus dem Leitfaden hergeleitet:

  (i.i) Die Anwendung muss in einer Sprache entwickelt werden, zu der eine quelloffene Implementierung vorliegt. Hier ist der Kunde nicht kompromissbereit.

  (i.ii) Der Kunde würde es vorziehen, wenn auch alle verwendeten Bibliotheken quelloffen sind. Sollte es jedoch keine quelloffenen Bibliotheken für einen Use Case geben und eine Eigenentwicklung zu teuer sein, ist der Kunde ausnahmsweise bereit, eine nicht-quelloffene Bibliothek zu lizenzieren.

(ii) Nach einem Servercrash ist der Kunde *sehr* vorsichtig geworden. Jede Stunde soll das System selbsttätig einen vollständigen Datenexport durchführen. Dem Kunden ist egal, welcher Teil des Systems dafür verantwortlich ist.

(iii) Zur Verwaltung seiner zahllosen Backups hat der Kunde bereits ein grafisches Werkzeug entwickeln lassen, welches Sie nur erweitern müssen. Hier legt der Kunde aber Wert darauf, dass der Backup-Administrator unter keinen Umständen das neueste Backup löschen darf.

(iv) Zu guter Letzt informiert der Kunde Sie, dass einer seiner Mitarbeiter, Tobias Tunichtgut, möglicherweise nicht so vertrauenswürdig wie erhofft ist. Daher wäre es schön, wenn er über das Webfrontend keinen Zugriff auf Gehaltsinformationen bekäme. Sollten Sie es vorziehen, so etwas nicht zu implementieren, hat der Kunde aber volles Verständnis dafür.

Achten Sie dabei darauf, unwichtige Details aus den Texten zu entfernen.