Komplex 4 – Modellweben

Ziel dieser Teilaufgabe ist es, einen Einblick in die neuartige Technik des Modellwebens in der Modellgetriebenen Entwicklung zu gewinnen. Der Software Technologie Lehrstuhl entwickelt zur Zeit das *Reuseware* Tool, welches zum Modelweben eingesetzt werden kann und in dieser Übung zum Einsatz kommt. In der Übung werden UML Modelle verwebt. Um diese zu betrachten und zu erstellen, wird der OpenSource UML Editor TOPCASED verwendet.

Vorbereitung A2:

Installieren sie Reuseware, sowie den TOPCASED UML Editor samt SDK, Environment und OCL Tools über den Eclipse Update Manager.

Verwenden sie Eclipse 3.4 (Ganymed) Modelling Tools und folgende Update Site:

- Reuseware: **http://reuseware.org/update**
  - Alles installieren

Eine Anleitung zur Installation des TOPCASED UML Editors finden sie hier:
http://gforge.enseeiht.fr/docman/view.php/52/3003/Installation%20Guide%20v2.pdf

Laden sie das Aufgaben-Material von der Webseite der Lehrveranstaltung und entpacken sie es in ein Projekt (*New Project > General > Project*) in ihrem Eclipse Workspace. Deklarieren sie den Hauptordner „SWT2“ als Fragment-Store, indem sie ihn selektieren und den blauen Reuseware Button in der Toolbar klicken.

Hinweise zur Verwendung von TOPCASED:

- Neue Diagramme können sie in Eclipse unter *New > Other... > Topcased > Topcased Diagrams > UML Model with TOPCASED* erstellen. Wählen sie unter *Template*, welche Art von Diagram erstellt werden soll.
- Wenn sie im UML Editor primitive Typen (String, Integer, Boolean) für Attribute setzen möchten, rechts-klicken sie in der Outline des Editors auf das Modell und wählen dort *Generate Primitive Types*.

Aufgabenkontext

Ein System für einen Pizzalieferanten (PizzaShop) soll modellgetrieben entwickelt werden. Die folgenden drei Anwendungsfälle wurden in der Analyse definiert:

1. **Pizzabestellung**: Die Pizzabestellung erfolgt telefonisch. Zuerst wird die Adresse des Kunden aufgenommen. Dann werden die Details zu einer Pizza erfragt. Der Kunde wird dann gefragt, ob er eine weitere Pizza bestellen möchte. Wenn ja, wird zur Detailerfragung zurückgekehrt. Wenn keine weitere Pizza mehr bestellt werden soll, ist der Anwendungsfall abgeschlossen.
2. **Pizzabacken**: Der Pizzabäcker wird von dem System bei der Zubereitung der Pizzen unterstützt. Zuerst wird eine Bestellung ausgewählt, die zu bearbeiten ist. Dann wird der Teig für die erste Pizza zubereitet. Danach werden die Zutaten auf den Teig gelegt.

Dann wir die Pizza gebacken. Wenn es noch weitere Pizzen in der Bestellung gibt, wird zur Teigzubereitung zurückgekehrt. Der Anwendungsfall ist abgeschlossen, wenn alle Pizzen der Bestellung gebacken wurden.

3. **Pizzaauslieferung**: Die Auslieferungen der Pizzen werden vom System erfasst. Zuerst werden alle Pizzen einer Bestellung zum Kunden gebracht. Danach werden die Pizzen übergeben und Geld wird entgegen genommen. Dann wird zurückgefahren. Zuletzt wird ein Bericht über die Lieferung in das System eingegeben.

Die verschiedenen Anwendungsfälle sollen nun unabhängig voneinander in UML Zustandsmaschinen und Klassendiagrammen modelliert werden. Solche Teilmodelle können auch aus Bibliotheken wiederverwendet werden. Durch Modellweben sollen die einzelnen Modelle zu einem Gesamtmodell des Systems verwoben werden.

Aufgaben

Machen sie sich mit den Funktionen von Reuseware vertraut. Informationen finden sie unter: **http://reuseware.org**.
(Hinweis: manche Informationen bezüglich der konkreten Bedienung des Tools sind eventuell nicht up-to-date. *http://st.inf.tu-dresden.de/reuseware/index.php/Starter_Guide* beschreibt die Bedienung der aktuellen Version und wird als Einstiegspunkt empfohlen.)

Schauen sie sich außerdem das *TicketOrder* System an (im Aufgabenmaterial enthalten: */reference/TicketOrder*), welches die Anwendungsfälle *Ticketbestellung* und *Ticketversand* mit UML modelliert.

Aufgabe A – Zustandsmaschinen wiederverwenden und verweben

1. Erstellen sie eine UML Zustandsmaschine für den Pizzashop unter Wiederverwendung vorgefertigter Modelle. Jeder Anwendungsfall soll darin als *Composite State* modelliert sein. Überprüfen sie, ob es bereits geeignete Modelle für einige der Anwendungsfälle in der Bibliothek (*/library/StateMachines*) gibt. Modellieren sie nur die Anwendungsfälle, die sie nicht in der Bibliothek finden konnten. Für die anderen Anwendungsfälle, setzen sie einen Platzhalter *State* ein. Platzhalter sind normale *States* ohne weitere Eigenschaften, deren Name mit „Reuse_“ beginnt
   (Vgl.: /reference/TicketSale/StateMachine/TicketSaleStateMachine.uml)
2. Erstellen sie ein Kompositionsprogramm, welches die Modelle aus der Bibliothek in ihr Modell einwebt.
   (Vgl.: /reference/TicketSale/StateMachine/TicketSaleStateMachineComposition.fcdi)
3. Überprüfen sie das Ergebnis und nehmen sie evtl. Korrekturen an ihrem Modell oder Kompositionsprogramm vor.
   Hinweis: Die Komposition des Diagramlayouts ist ein sehr junges Feature in Reuseware. Abgesehen davon, dass die Diagramme manchmal nicht „sehr schön“ sind, fehlen manchmal auch Elemente aus dem eigentlichen Modell. Wenn das Ergebnis einer Komposition seltsam ist, überprüfen sie die Outline im UML Editor. Dort ist auf jeden Fall das richtige Modell zu sehen.

Aufgabe B – Querschneidendes Verweben von Klassendiagrammen

In Klassendiagrammen lassen sich verschiedene Anwendungsfälle schwer trennen. In */exercise/PizzaShop/ClassDiagrams/PizzaShopDataModel.uml* finden sie das Kernklassendiagram des Pizzashops. Die einzelnen Anwendungsfälle erfordern nicht nur das Hinzufügen von neuen Klassen, sondern auch die Anreicherung der existierenden Klassen in diesem Modell.

1. Erweiterungen für die Anwendungsfälle Pizzabestellung und -backen sind bereits in */exercise/PizzaShop/ClassDiagrams* modelliert. Erstellen sie ein Kompositions-programm, welches diese beiden Erweiterungen in *PizzaShopDataModel.uml* einwebt. *Hinweis: Alle neuen Klassen können gebündelt über die* NewClasses *und* ExtensionPoint *Ports in ein Modell gewebt werden. Erweiterungen für einzelne Klassen* (ExtensionFor_?) *müssen direkt die entsprechende Klasse erweitern (Vgl.: /TicketSale/ClassDiagrams/).*
2. Erstellen sie eine Klassendiagramm für den Anwendungsfall Pizzaauslieferung, welcher folgendes berücksichtigt: Autos, die entweder gerade unterwegs oder in der Garage sind, sollen verwaltet werden. Drucker, zum Ausdrucken von Bestellungen, sollen zur Verfügung stehen. Zur Qualitätssicherung, soll das System Informationen über ausgelieferte Bestellungen und Pizzen halten. Dafür müssen Bestellungen als „ausgeliefert“ gekennzeichnet werden können. Außerdem soll in Bestellungen die Lieferungszeit und in Pizzen die Temperatur zum Auslieferungszeitpunkt festgehalten werden.
   Hinweis: Erweiterungen (Operationen und Attribute) für existierende Klassen werden in Erweiterungsklassen definiert. Diese sind dadurch gekennzeichnet, dass ihr Name mit „ExtensionFor_“ *beginnt. Wenn eine existierende Klasse referenziert werden soll (z.B. als Type eines Parameters einer Operation), kann man die entsprechende Erweiterungsklasse referenzieren.*
   (Vgl.: /TicketSale/ClassDiagrams/).
3. Erweitern sie ihr Kompositionsprogramm um die Einwebung ihres Pizzaauslieferungs-Klassendiagrams.

Übersicht Ergebnisse

Als Resultat des Aufgabenkomplexes sind die folgenden vier Artefakte im SVN einzureichen.

1. Die Zustandsmaschine für den Pizzashop
2. Kompositionsprogramm: Modelle aus der Bibliothek mit (1) verwebt
3. Das Klassendiagramm für Pizzaauslieferung
4. Das Kompositionsprogramm, welches alle 4 Klassendiagramme (1x Kernmodell + 3x Anwendungsfallmodelle) des Pizzashops verwebt.