Labor Software-Engineering WS1112

Prof. Dr. Th. Fuchß
Hochschule Karlsruhe – Technik und Wirtschaft
Fakultät für Informatik und Wirtschaftsinformatik
Fachgebiet Informatik

Entwicklung eines Lern-Quiz-Computer-Spiels

Aufgabe 1: Analyse
Erstellen Sie in **Together ein UML-2-Projekt**. Analysieren Sie die Aufgabenstellung und definieren Sie den Umfang des ersten Entwicklungszyklus. Führen Sie hierzu folgende Schritte durch:

- **Funktionalität und Eigenschaften**
  - Spielen Sie das Spiel, lesen Sie die Spielanleitung.
  - Erarbeiten Sie die Funktionalität und Anforderungen, die Ihre Software bereitstellen soll. Benennen Sie jede funktionale und nicht funktionale Anforderung eindeutig, beschreiben Sie kurz was Sie mit den einzelnen Anforderungen verbinden, priorisieren und gruppieren Sie diese. **Erstellen Sie ein entsprechendes Excel-Dokument**.

- **Use-Cases, Diagramme und ihre Details**
  Bestimmen Sie die Use-Cases des Lern-Quiz-Computer-Spiels.
  - Beschreiben Sie jeden Use-Case mit eigenen Worten und ordnen Sie jedem Use-Case die zuvor definierten Anforderungen zu. Priorisieren Sie Ihre Use-Cases (essentiell, wichtig, unwichtig) und begründen Sie Ihre Entscheidung.
  - Skizzieren Sie das Use-Case-Diagramm mit allen Akteuren und Abhängigkeiten
  - Erstellen Sie für die wichtigsten (zwei bis drei) Use-Cases Beschreibungen in Form eines Activity-Diagramms.

- **Use-Cases, Details, Objektmodell und Schnittstellen**

  - Extrahieren Sie aus den erstellten Dokumenten die Konzepte des Lern-Quiz-Computer-Spiels und ihre Beziehungen. Stellen Sie diese in Form eines Klassendiagramms dar (Objektmodell).
  - Bestimmen Sie aus den Activity-Diagrammen Ihrer Use-Cases mögliche Systemoperationen. Erstellen Sie zur besseren Übersicht System-Sequenz-Diagramme und beschreiben Sie jede Operation mit eigenen Worten. Fassen Sie diese in einem entsprechenden Excel-Dokument zusammen.

- **Bedienkonzept**
  Entwerfen Sie ein Bedienkonzept, das die Umsetzung der bisher analysierten Use-Cases aus Sicht des Anwenders beschreibt.

Aufgabe 2: Design
Erstellen Sie in **Together ein Java-Modelling-Projekt** und entwickeln Sie ein Designmodell auf der Basis des Analysemodells. Führen Sie hierzu in Ihrem Java-Modelling-Projekt zwei neue Packages ein. Ein Modell-Package und ein „src"-Package.

- **Aufbereitung der Analyse und Systemoperationen**
  - Entscheiden Sie sich für eine Menge von Use-Cases, die Sie in der ersten Phase realisieren möchten. Begründen Sie kurz Ihre Entscheidung.
  - Skizzieren Sie die grobe Architektur des zu entwickelnden Systems. Ergänzen Sie für jede Schicht und jede Partition ein geeignetes Package. Verschieben Sie diese Packages ins „src"-Package
  - Beschreiben Sie die zu realisierenden Use-Cases in Form von System-Use-Cases.
    - Use-Case-Beschreibung
    - Use-Case-Diagramm
    - detaillierte Beschreibung in Form von Activity-Diagrammen

    Orientieren Sie sich hierzu an Ihren Analyseergebnissen und berücksichtigen Sie die gewählte Architektur und Ihr Bedienkonzept.
  - Bestimmen Sie die Schnittstelle der Applikationsschicht, wie sie für die Realisierung gebraucht wird.
    - Skizzieren Sie die System-Operationen in Form von System-Sequenz-Diagrammen.
    - Beschreiben Sie die System-Operationen.
    - Achten Sie auf Parameter, Ausnahmen, Vor- und Nachbedingungen.
    - Wählen Sie geeignete Use-Case-Controller (oder entsprechende Klassen aus dem Analyse-Modell) und ordnen Sie diesen die System-Operation zu. Erstellen Sie diese Klasse neu im Design-Package und verschieben Sie sie in ein entsprechendes Sub-Package im „src"-Package. Erstellen Sie für jeden Use-Case-Controller ein entsprechendes Java-Interface mit der gleichen Funktionalität.

- **Zustandsautomaten**
  Erstellen Sie einen Automaten und ordnen Sie jeder System-Operation mindestens einen Zustand zu, in dem sie ein gültiges äußeres Event darstellt. Beachten Sie insbesondere die Vor- und Nachbedingungen, sowie die gefundenen Ausnahmen aus dem vorherigen Designschritt.
  Erstellen Sie ein Sub-Package "States" und ordnen Sie jedem Zustand eine eigene Klasse zu. Vereinigen Sie alle Zustandsklassen unter einem Interface "State".

- **Detailed Design und das Objektmodell**
  - Designen Sie die zu realisierenden System-Operation und die Initialisierung des Systems. Erstellen Sie hierzu für jede System-Operation und Initialisierungsroutine ein detailliertes Sequenz- oder Kommunikationsdiagramm. Beachten Sie hierbei die Vor- und Nachbedingungen der System-Operationen (die Zustände)
  - Erstellen Sie das Design-Objektmodell (nur Klassen und ihre Beziehungen, keine Texte). Fügen Sie eine Klasse immer dann dem Objektmodell hinzu, wenn beim Entwickeln der Sequenzdiagramme der Bedarf besteht. Verbinden Sie Klassen über Assoziationen, Aggregationen oder Kompositionen immer dann, wenn dies für die Kommunikation erforderlich ist. Verschieben Sie diese Klassen in entsprechende Sub-Packages im „src"-Package.
  - Sorgen Sie dafür, dass alle zu visualisierenden Ergebnisse in entsprechenden Attributen abgelegt werden und versehen Sie die Klassen mit geeigneten Selektoren, damit später auf diese Attribute zugegriffen werden kann. Ergänzen Sie geeignete Interfaces, die diese Operationen sinnvoll gruppieren.

- **Anbindung der Oberfläche**
  - Ordnen Sie die Zustände Ihres Zustandsautomaten Ihrem Bedienkonzept zu.
  - Skizzieren Sie die Anbindung der System-Operationen an die Benutzerschnittstelle (gemäß MVC-Pattern) und entwerfen Sie die Fassadenklasse, die die Schnittstelle zum Modell verkörpert und die eingehenden Systemoperationen hinsichtlich des Zustandes und ggf. korrekter Parameter überprüft und an die entsprechenden, für die System-Operationen zuständigen Objekte, delegiert. (Verschieben Sie neue Klassen in entsprechende Packages im „src"-Package)

Aufgabe 3
Implementieren Sie die Funktionalität der ersten Entwicklungsphase in Java, halten Sie sich dabei möglichst an Ihr Design. Orientieren Sie sich hierzu an folgendem Java-Code-Fragment.

```
public class Game{
 public static void main (String[] args){
   IPersistenceLayer pl     = persistenceLayer.PersistenceLayer.make();
   IApplication      model  = application.ApplicationFacade.make(pl);
   GUI               g      = new GUI(model);
   model.initGame();
  ...}
}
```

Hinweis:
Zur Erstellung der Methodenrümpfe bietet Together die Möglichkeit, Sequenzdiagramme in Code-Fragmente zu verwandeln.