11.02.2021
Florian Ege

Meilenstein 6: Implementierungsentwurf

Abgabetermin: *[Abgabefristen werden wegen des Prüfungszeitraums jeweils in den Tutorien vereinbart]*

Aufgabe 1: Erstellen eines Implementierungsentwurfs

Im letzten Meilenstein wurde ein Architekturentwurf für *Marvelous Mashup* erstellt. Dieser modelliert in Form eines Komponentendiagramms den strukturellen Aufbau der einzelnen Teile des Systems. Eine solche Architektur ist aber immer noch eine verhältnismäßig grobe Sicht auf das System. Diese wollen wir nun durch einen detaillierteren **Implementierungsentwurf** verfeinern. Für einen solchen Feinentwurf, der als letzter Schritt vor dem Übergang zur eigentlichen (objektorientierten) Programmiertätigkeit steht, eignen sich **UML2-Klassendiagramme.** Im Gegensatz zum Domänenmodell beinhalten die Klassendiagramme jetzt nicht nur **Anwendungsklassen,** die Konzepte der Anwendungsdomäne repräsentieren, sondern auch **Implementierungsklassen,** mit allen dazugehörigen **Attributen**, **Methoden** und **Relationen** untereinander.

In diesem Meilenstein soll *nur für die Komponente Benutzer-Client* des verteilten Systems ein Implementierungsentwurf erstellt werden. Vor der tatsächlichen Implementierung des Spiels müsst ihr natürlich auch für die anderen von euch zu entwickelnden Komponenten die entsprechenden Implementierungsentwürfe angefertigen. Dies ist aber nicht Teil der Aufgabe in diesem Meilenstein.

- Erstellt einen Implementierungsentwurf in Form eines **UML2-Klassendiagramms.** Beschränkt euch dabei auf die **Komponente Benutzer-Client** von *Marvelous Mashup.* (Für die anderen Komponenten erstellt ihr nach der Korrektur dieses Meilensteines ebenfalls entsprechende Implementierungsentwürfe, bevor ihr mit der Programmierung beginnt.)
- Das Klassendiagramm sollte **Klassen der Anwendungsdomäne** beinhalten, die wenn nötig durch zusätzliche Attribute und Methoden erweitert werden sollten. Ihr könnt dazu die für Meilenstein 1 erstellten Diagramme wiederverwenden. Dazu kommen jetzt auch **Implementierungsklassen,** die sich um die technischen Aspekte der Anwendung kümmern (Laden und Speichern von Daten, graphische Darstellung, Netzwerkkommunikation etc.).
- Fügt allen Methoden, die nicht vom Namen her völlig selbsterklärend und trivial sind, eine kurze **textuelle Beschreibung** bei, was ihre Vor- und Nachbedingungen sind, was sie tun, und wie sich der Zustand des Systems durch den Aufruf verändert.
- Alle **funktionalen Anforderungen** an die Komponente, also alle möglichen Benutzerinteraktionen und Systemoperationen, müssen letztendlich bei einer objektorientierten Implementierung durch Methoden realisiert werden. Achtet darauf, dass jeder funktionalen Anforderung eine Methode (oder mehrere) zugeordnet werden können. Erstellt eine **Liste,** in der für jede funktionale Anforderung vermerkt ist, durch welche Methode(n) sie umgesetzt wird.

Hinweise und Tipps

- Achtet auf die **korrekte graphische Syntax** des Klassendiagramms, vor allem darauf, wie die einzelnen Relationstypen (Vererbung, Assozationen und Kardinaliäten, Komposition usw.) mit unterschiedlichen Pfeilendungen dargestellt werden.
- Wenn es Aspekte gibt, die sich schlecht im Diagramm graphisch ausdrücken lassen, kann man auch **Annotationen** oder einen **erklärenden Text** beifügen. Die Diagramme und Modelle, die vor der Implementierungsphase erstellt werden, sind kein Selbstzweck, sondern dienen später beim Programmieren zur Orientierung. Sie sollten darum möglichst klar und verständlich sein.