Meilenstein 5: Architekturentwurf

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

Aufgabe 1: Erstellen einer Systemarchitektur

Die Funktionalität des verteilten Systems von *Marvelous Mashup* und seine Benutzerschnittstellen sind seit dem letzten Meilenstein im Pflichtenheft beschrieben. Damit ist jetzt detailliert spezifiziert *was* bei der Implementierung realisiert werden soll. Als nächstes muss nun präzisiert werden, *wie* genau diese Anforderungen umgesetzt werden sollen.

Der **Architekturentwurf** gliedert ein System in **Komponenten** bzw. Teilsysteme, und diese wiederum hierarchisch in (Unter-)Teilsysteme. Zur Systemarchitektur gehören dabei auch die **Schnittstellen** der Komponenten, und die vorkommenden **Artefakte.** Diese strukturelle Gliederung eines Systems kann mit **Komponentendiagrammen** dargestellt werden.

Sinn und Zweck des Architekturentwurfs ist eine weitgehend programmiersprachenunabhängige Festlegung der zu realisierenden Softwareeinheiten. Diese einzelnen, voneinander abgegrenzten Komponenten mit ihren definierten Schnittstellen können dann später von verschiedenen Entwicklern unabhängig voneinander realisiert werden. Somit ergeben sich automatisch einzelne Arbeitspakete. Das ist eine Voraussetzung für effiziente Arbeitsteilung während der Implementierung.

Erstellt ein Dokument, das die Systemarchitektur beschreibt und folgendes beinhaltet:

- Erstellt ein **UML2-Komponentendiagramm,** welches die Struktur des gesamten verteilten Systems für *Marvelous Mashup* zeigt. Dabei solltet ihr natürlich die grundlegende Architektur, die im Lastenheft vorgegeben ist, berücksichtigen. Das ist notwendig, damit z.B. Clients und Server, die von verschiedenen Teams implementiert wurden, miteinander interagieren können. Diese grobe Architektur solltet ihr dabei bis zu einem angemessenen Grad an Granularität verfeinern. Modelliert für jede **Komponente** die **angebotenen und benötigten Schnittstellen** zu anderen Teilen des Systems. Überlegt euch auch, welche **Artefakte** (Datenobjekte, Konfigurationsdateien, etc.) in welchen Komponenten stecken, oder welche Komponenten darauf zugreifen.
- Fügt jeder Komponente eine **textuelle Beschreibung** bei, die informell erklärt, was die Komponente macht bzw. wozu man sie braucht. Begründet dabei, warum ihr euch entschieden habt, das System gerade so zu strukturieren und nicht anders. Die Gründe für die Aufteilung in Komponenten und die Verteilung der Komponenten auf die Subsysteme (Server, Benutzer-Client, KI-Client, Editor) sollten nachvollziehbar sein.
- Gebt für jede Komponente eine **Liste der funktionalen Anforderungen** an, die von dieser Komponente abgedeckt werden. Da alle Komponenten zusammen letztendlich das Gesamtsystem bilden, muss es natürlich für jede funktionale Anforderung mindestens eine Komponente geben, die sich darum kümmert! Nur so kann das System als Ganzes alle Anforderungen erfüllen. Umgekehrt sollte für jede Komponente rückverfolgbar sein, durch welche Anforderungen ihre Implementierung motiviert wurde. Ihr könnt diese gegenseitige Abbildung darstellen,

indem ihr auf die Bezeichner der Anforderungen in früheren Dokumenten verweist, oder Anforderungen ggf. überarbeitet oder verfeinert und passend neu aufteilt.

Hinweise und Tipps

- Achtet auf das **Wechselspiel zwischen Anforderungen und den sie erfüllenden Systemkomponenten.** Jetzt, da man die Architektur konkretisiert, kann es sinnvoll sein, eine Anforderung in feingranularere aufzuteilen, und diese auf mehrere Komponenten zu verteilen. Umgekehrt sollte sich natürlich auch die gewählte Architektur an den Anforderungen orientieren. Vielleicht muss man auch zusätzliche Komponenten einführen, um Anforderungen zu erfüllen. Gegebenenfalls solltet ihr also beim Erstellen der Systemarchitektur nochmal die Gliederung der Anforderungen überarbeiten.
- Es kann vorkommen, dass eine bestimmte Funktionalität in mehreren Teilsystemen benötigt wird. Sie sollte dann entsprechend in eine Komponente gekapselt werden, die an mehreren Stellen **wiederverwendet** werden kann. Das vermeidet unnötige Arbeit und code duplication.
- In der Softwaretechnik gibt es die Begriffe der **Kohäsion** (möglichst starker logischer Zusammenhang der Bestandteile einer Komponente) und **Kopplung** (präzise definierte, möglichst minimale Schnittstellen zwischen Komponenten) als Qualitätsmetriken für eine modulare Struktur. Eine gute Architektur sollte eine hohe Kohäsion innerhalb der Komponenten besitzen, während die Komponenten untereinander eine geringe bzw. lose Kopplung haben sollten. Behaltet diese Prinzipien bei Entscheidungen über Aufteilungen oder Zuordnungen von Funktionalitäten im Blick.
- Achtet auf die **korrekte graphische Syntax** beim Komponentendiagramm für die angebotenen bzw. benötigten Schnittstellen, also auf die "Richtung" der Schnittstellen. Es ist wichtig zu wissen, welche Komponenten provider, und welche consumer sind.