Universität Bremen
FB 3 – Informatik
Dr. Karsten Hölscher
TutorIn: Karsten Hölscher

Software–Projekt 2 2014

VAK 03-BA-901.02

Architekturbeschreibung

Patrick Hollatz	phollatz@tzi.de	2596537
Tobias Dellert	tode@tzi.de	2936941
Tim Ellhoff	tellhoff@tzi.de	2520913
Daniel Pupat	dpupat@tzi.de	2703053
Olga Miloevich	halfelv@tzi.de	2586817
Tim Wiechers	tim3@tzi.de	2925222

Abgabe: 06.Juli 2014 — Version 1.7

Inhaltsverzeichnis

Wichtiger Hinweis: Diese Architekturbeschreibung wurde zu einem Teil aus verschiedenen Dokumententeilen der Architekturbeschreibung unserer Gruppenmitglieder aus dem Wintersemester 2013/14 erstellt (Gruppe *IT_R3V0LUTION*). Einige Teile wurden komplett übernommen, andere überarbeitet bzw. angepasst.
Diese Vereinbarung haben wir in der Kick-Off-Veranstaltung für RE SWP 2014 mit dem Veranstalter Dr. Karsten Hölscher getroffen.

Version und Änderungsgeschichte

Version	Datum	Änderungen
1.0	28.06.2014	Einleitung
1.1	02.07.2014	Globale Analyse
1.2	02.07.2014	Konzeptionelle Sicht
1.3	03.07.2014	Modulsicht
1.4	03.07.2014	Datensicht
1.5	04.07.2014	Ausführungssicht
1.6	04.07.2014	Zusammenhänge zwischen Anwendungsfällen und Architektur
1.7	05.07.2014	Evolution

1 Einführung

1.1 Zweck

Dieses Dokument ist die Architekturbeschreibung der von uns zu entwickelnden Software. Sie dient der Kommunikation zwischen allen Interessenten. Dies ist unerlässlich für die Entwicklung des Systems, da die Entwickler der Architekturbeschreibung die Funktionalität einzelner Komponenten entnehmen. Sie dient der Aufteilung der Arbeit in unabhängig bearbeitbare Teile, besitzt anfangs einen hohen Abstraktionsgrad, der von vielen verstanden werden kann und wird in den Schichten weiter unten in diesem Dokument präziser ausgearbeitet. Die präzise Ausarbeitung der Architektur ist wichtig, um Möglichkeiten und Probleme der Entwicklung auszuloten und präventive Strategien und Maßnahmen zu entwickeln.
Die Architektur des Systems ist daher das Fundament unserer Implementierung, die direkt aus der Architektur resultiert.

1.2 Status

Dies ist der erste Architekturentwurf vom 06.07.2014.

1.3 Definitionen, Akronyme und Abkürzungen

1.4 Referenzen

- https://elearning.uni-bremen.de/scm.php?cid=2b323f34b16a84e8dce31dcdfc0be6ad&show_scm=4c88951a202b2543c96de2c8a476d471 Die Mindestanforderungen für die Quiz-App
- http://www.elearning.uni-bremen.de Plattform der Universität Bremen. Zugriff auf Folien der Veranstaltung Software Projekt 1 des Sommersemesters 2013 und Übungen des Software Projekts 2 des Wintersemesters 13/14 nur eingeschränkt möglich.
- Vorlage dieses Dokuments - Stud.IP - 3-Architekturbeschreibung-Vorlage.tex
- Hinweise zu diesem Dokument - Stud.IP 3-Hinweise-Abgabe-Architektur.pdf

1.5 Übersicht über das Dokument

Dieses Dokument basiert auf der Vorlage des IEEE P1471 2002 Standards. Der Inhalt dieses Dokuments ist wie folgt aufgegliedert:

1. Einführung Die Einführung beschreibt den Nutzen dieses Dokuments. Sie erläutert Definitionen, Akronyme und Abkürzungen und listet die benutzten Referenzen auf, sowie eine Übersicht über dieses Dokument.

2. Globale Analyse In diesem Abschnitt werden die relevanten Einflussfaktoren aufgezeigt und bewertet, sowie Strategien entwickelt, um Probleme bzw. interferierende Einflussfaktoren zu behandeln und auf diese entsprechend zu reagieren.

3. Konzeptionelle Sicht Die konzeptionelle Sicht zeigt grob die einzelnen Komponenten und deren Zusammenspiel des zu entwickelnden Systems auf. Dies geschieht auf einer hohen Abstraktionsebene und wird im weiteren Verlauf des Dokuments und den folgenden Sichten konkretisiert und verfeinert.

4. Modulsicht Im Abschnitt Modulsicht dieser Architekturbeschreibung geht es um eine tiefere Ebene der Abstraktion. Hier werden die Komponenten in einzelne Pakete zerlegt und diese wiederum in Module, welche eine Einheit bilden, die ein Entwickler in einer Arbeitswoche implementieren kann.

5. Datensicht Die Datensicht beschreibt das zugrundeliegende Datenmodell und das Zusammenspiel der einzelnen Daten der Datenbank. Dies wird in Form eines erklärenden Textes und UML-Diagrammen realisiert.

6. Ausführungssicht Die Ausführungssicht zeigt im Prinzip das System in "Aktion", d.h. es zeigt auf, welche Prozesse laufen, welche Module hierfür gebraucht werden und wie diese zusammenspielen.

7. Zusammenhänge zwischen Anwendungsfällen und Architektur Hier werden die Zusammenhänge zwischen Architektur und den Anwendungsfällen der Anforderungsspezifikation beschrieben.

8. Evolution In diesem Teil der Architekturbeschreibung wird beschrieben, welche Änderungen vorgenommen werden müssen, wenn sich Anforderungen und oder Rahmenbedingungen ändern. Ein besonderes Augenmerk liegt hierbei auf die in der Anforderungsspezifikation unter "Ausblick" genannten Punkte.

2 Globale Analyse

Hier werden Einflussfaktoren aufgezählt und bewertet sowie Strategien zum Umgang mit interferierenden Einflussfaktoren entwickelt.

2.1 Einflussfaktoren

Hier sind Einflussfaktoren gefragt, die sich auf die Architektur beziehen. Es sind ausschließlich architekturrelevante Einflussfaktoren, und nicht z.B. solche, die lediglich einen Einfluss auf das Projektmanagement haben. Fragt Euch also bei jedem Faktor: Beeinflusst er wirklich die Architektur? Macht einen einfachen Test: Wie würde die Architektur aussehen, wenn ihr den Einflussfaktor E berücksichtigt? Wie würde sie aussehen, wenn Ihr E nicht berücksichtigt? Kommt in beiden Fällen dieselbe Architektur heraus, dann kann der Einflussfaktor nicht architekturrelevant sein.

Es geht hier um Einflussfaktoren, die

1. *sich über die Zeit ändern,*
2. *viele Komponenten betreffen,*
3. *schwer zu erfüllen sind oder*
4. *mit denen man wenig Erfahrung hat.*

Die Flexibilität und Veränderlichkeit müssen ebenfalls charakterisiert werden.

1. *Flexibilität: Könnt Ihr den Faktor zum jetzigen Zeitpunkt beeinflussen?*

2. *Veränderlichkeit: ändert der Faktor sich später durch äußere Einflüsse?*

Unter Auswirkungen sollte dann beschrieben werden, wie *der Faktor* was *beeinflusst. Das können sein:*

- *andere Faktoren*
- *Komponenten*
- *Operationsmodi*
- *Designentscheidungen (Strategien)*

Verwendet eine eindeutige Nummerierung der Faktoren, um sie auf den Problemkarten einfach referenzieren zu können.

2.2 Probleme und Strategien

Aus einer Menge von Faktoren ergeben sich Probleme, die nun in Form von Problemkarten beschrieben werden. Diese resultieren z.B. aus

- *Grenzen oder Einschränkungen durch Faktoren*
- *der Notwendigkeit, die Auswirkung eines Faktors zu begrenzen*
- *der Schwierigkeit, einen Produktfaktor zu erfüllen, oder*
- *der Notwendigkeit einer allgemeinen Lösung zu globalen Anforderungen.*

Dazu entwickelt Ihr Strategien, um mit den identifizierten Problemen umzugehen.

Achtet auch hier darauf, dass die Probleme und Strategien wirklich die Architektur betreffen und nicht etwa das Projektmanagement. Die Strategien stellen im Prinzip die Designentscheidungen dar. Sie sollten also die Erklärung für den konkreten Aufbau der verschiedenen Sichten liefern.

Beschreibt möglichst mehrere Alternativen und gebt an, für welche Ihr Euch letztlich aus welchem Grunde entschieden habt. Natürlich müssen die genannten Strategien in den folgenden Sichten auch tatsächlich umgesetzt werden!

Ein sehr häufiger Fehler ist es, dass SWP-Gruppen arbeitsteilig vorgehen: die eine Gruppe schreibt das Kapitel zur Analyse von Faktoren und zu den Strategien, die andere Gruppe beschreibt die diversen Sichten, ohne dass diese beiden Gruppen sich abstimmen. Natürlich besteht aber ein Zusammenhang zwischen den Faktoren, Strategien und Sichten. Dieser muss erkennbar sein, indem sich die verschiedenen Kapitel eindeutig aufeinander beziehen.

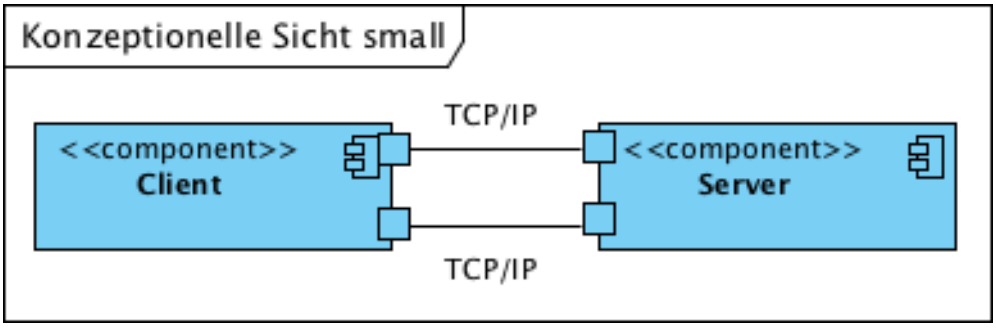
3 Konzeptionelle Sicht

Wir haben mithilfe von UML-Diagrammen die konzeptionelle Sicht realisiert. Im Folgenden werden die einzelnen Diagramme aufgezeigt und beschrieben und in nachfolgenden

Sichten zusätzlich verfeinert und konkretisiert.

3.1 Überblick

Abbildung 1: Konzeptionelle Sicht (Klein)

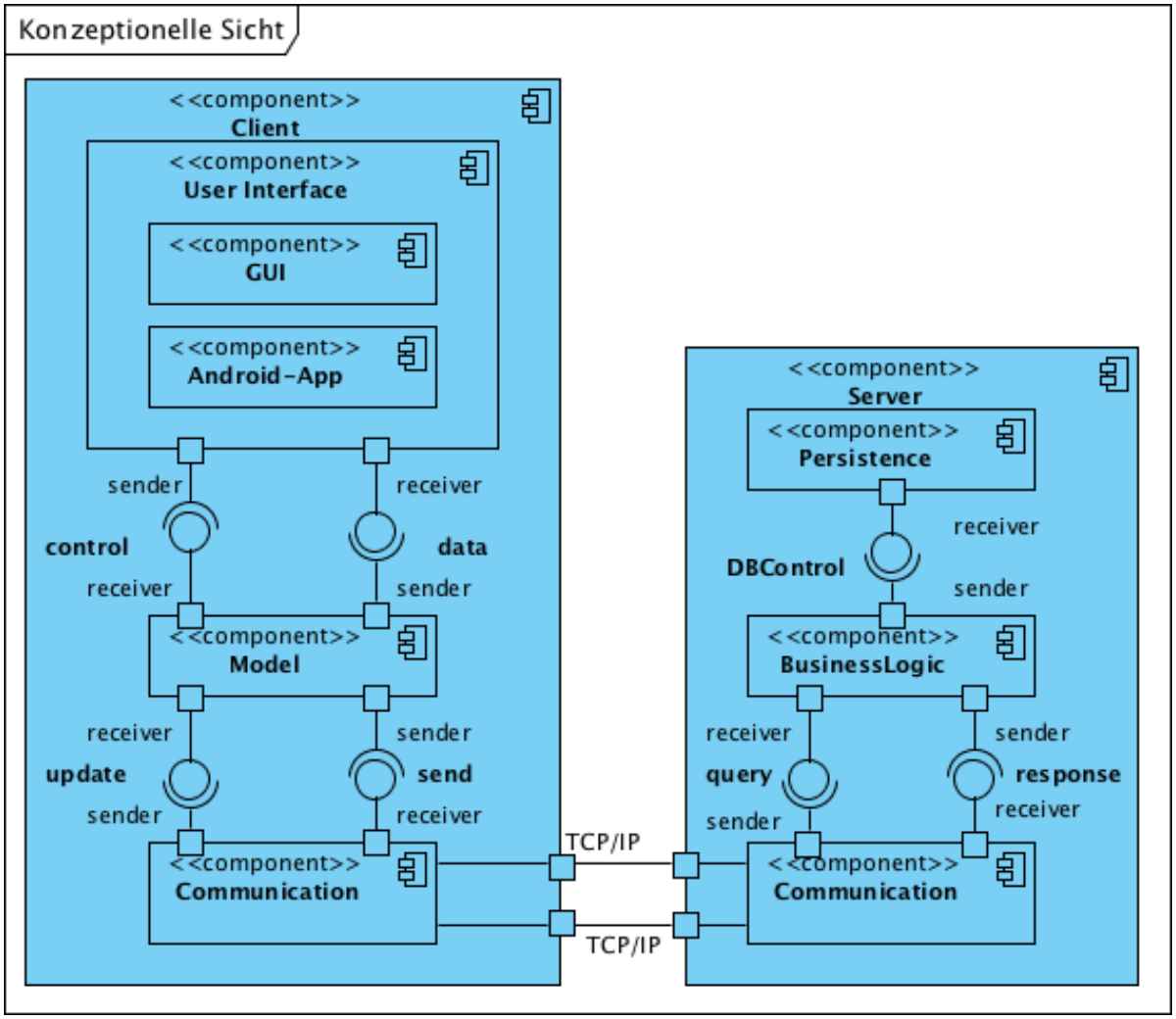


Unsere Architektur besteht aus zwei grundlegenden Komponenten, der Serverkomponente und der Clientkomponente (siehe Abb. 1 und Abb. 2 auf der nächsten Seite). Diese Komponenten beinhalten wiederum weitere Komponenten. Auf der einen Seite haben wir unsere Serverkomponente, die alle benötigten Daten für die Quizfragen, der Nutzer sowie z.B. Punktetänden usw. der App speichert.

Auf der anderen Seite, der Clientkomponente, muss zwischen zwei Komponenten unterschieden werden. Einmal der Komponente GUI-Client, welche sich in erster Linie an den Administrator, der über eine Webseite die Spiele konfigurieren und die Fragen und Antworten redaktionell bearbeiten kann, richtet und dann noch der mobile Android-Client, der sich ausschließlich an die Nutzer bzw. Spieler richtet.

Der GUI-Client stellt für die Verantwortlichen bzw. die Administratoren alle benötigten Funktionen bereit, um die App zu verwalten. Der Android-Client ermöglicht dem Spieler, die Quizapp zu spielen und Einstellungen vorzunehmen.

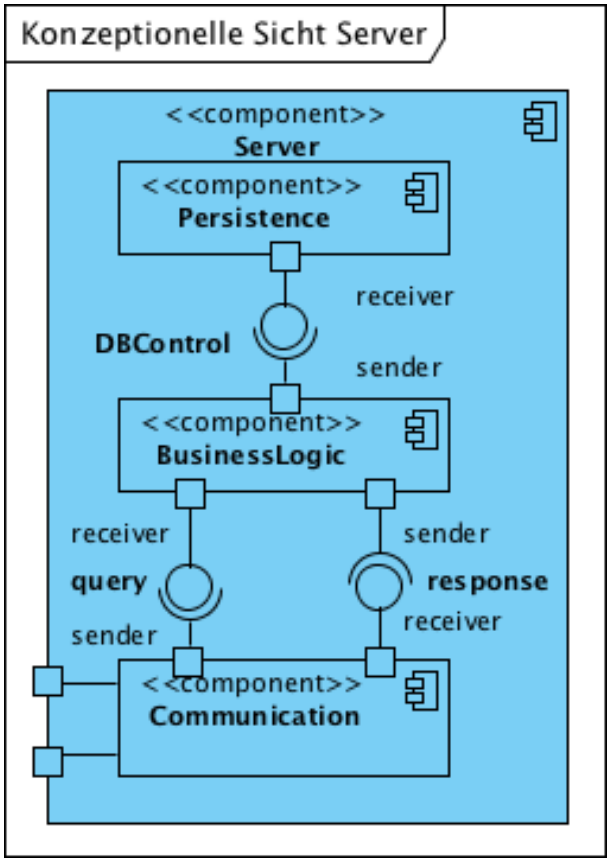
Abbildung 2: Konzeptionelle Sicht



Als Architekturstil verwenden wir das Model-View-Controller-Pattern.

3.2 Serverkomponente

Abbildung 3: Konzeptionelle Sicht Server



Die Serverkomponente (siehe Abb. 3 besteht aus insgesamt drei Teilkomponenten, welche sich wie folgt aufgliedern:

- Communication

  Die Komponente `Communication` nimmt Anfragen des Clients entgegen und leitet sie an die Komponente `BusinessLogic` weiter, wo die Anfragen verarbeitet werden und sendet die Ergebnisse zurück an den Client.

- BusinessLogic

  Die Komponente `BusinessLogic` dient zum Verarbeiten der Anfragen und leitet diese verarbeiteten Anfragen dann an die Komponente `Persistence` weiter.

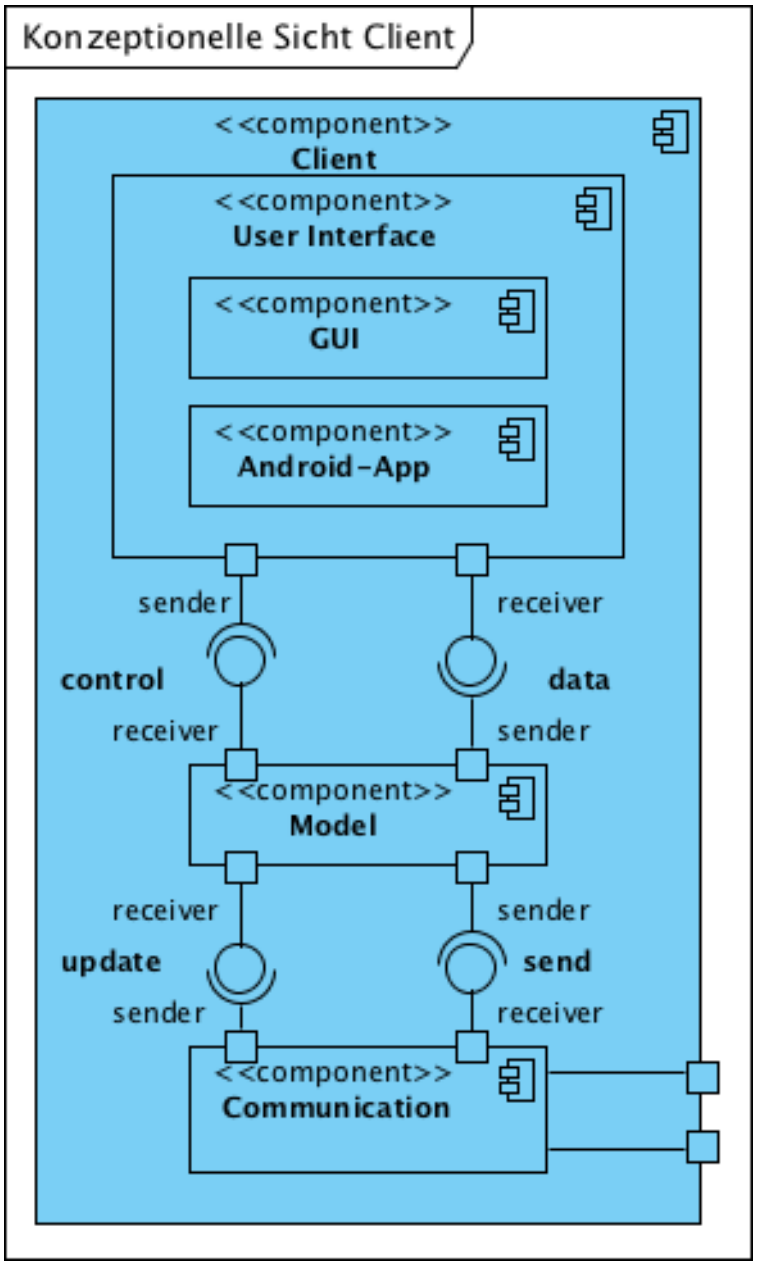
- Persistence

  Die Komponente `Persistence` ist die Schnittstelle zur Datenbank. Über das Interface `DBControl` werden die verarbeiteten Anfragen von der Komponente `BusinessLogic` empfangen und in Datenbankabfragen umgewandelt, welche dann

von der Datenbank entgegen genommen werden.

3.3 Clientkomponente

Abbildung 4: Konzeptionelle Sicht Client



Die Clientkomponente (siehe Abb. 4 besteht so wie die Serverkomponente aus drei Teilkomponenten, welche sich wie folgt aufgliedern:

- Communication

  Die Komponente `Communication` sendet Anfragen des Clients an den Server, welche dort verarbeitet werden und nimmt die Ergebnisse entgegen, um diese an die Komponente `Model` zu übergeben, wo die Ergebnisse der Anfrage weiter verarbeitet werden.

- Model

  Die Komponente `Model` nimmt Ergebnisse von der Komponente `Communication` entgegen und schickt diese an die Komponente `User Interface`.

- User Interface

  Die Komponente `User Interface` muss in zwei unterschiedliche Komponenten zerlegt werden:

  – GUI

    Die GUI richtet sich in erster Linie an den Administrator bzw. die Verantwortlichen und nimmt alle möglichen Aktionen des Administrators entgegen und schickt diese an die Komponente `Model`, um weiter verarbeitet zu werden. Sie bildet ebenfalls eine eigenständige Komponente.

  – Android-App

    Die Android-App richtet sich ausschließlich an die Nutzer bzw. Spieler der App und bietet ihnen die Funktionen, die in der Anforderungsspezifikation erarbeitet wurden. Zu diesen gehören z.B. das Registrieren beim Server sowie das Spielen von Fragerunden oder das Anzeigenlassen von Ranglisten oder sonstigen spielerelevanten Statistiken.

4 Modulsicht

Diese Sicht beschreibt den statischen Aufbau des Systems mit Hilfe von Modulen, Subsystemen, Schichten und Schnittstellen. Diese Sicht ist hierarchisch, d.h. Module werden in Teilmodule zerlegt. Die Zerlegung endet bei Modulen, die ein klar umrissenes Arbeitspaket für eine Person darstellen und in einer Kalenderwoche implementiert werden können. Die Modulbeschreibung der Blätter dieser Hierarchie muss genau genug und ausreichend sein, um das Modul implementieren zu können.

Die Modulsicht wird durch UML-Paket- und Klassendiagramme visualisiert.

Die Module werden durch ihre Schnittstellen beschrieben. Die Schnittstelle eines Moduls M ist die Menge aller Annahmen, die andere Module über M machen dürfen, bzw. jene Annahmen, die M über seine verwendeten Module macht (bzw. seine Umgebung, wozu auch Speicher, Laufzeit etc. gehören). Konkrete Implementierungen dieser Schnittstellen sind das Geheimnis des Moduls und können vom Programmierer festgelegt werden. Sie sollen hier dementsprechend nicht beschrieben werden.

Die Diagramme der Modulsicht sollten die zur Schnittstelle gehörenden Methoden enthalten. Die Beschreibung der einzelnen Methoden (im Sinne der Schnittstellenbeschreibung) geschieht allerdings per Javadoc im zugehörigen Quelltext. Das bedeutet, dass Ihr für alle Eure Module Klassen, Interfaces und Pakete erstellt und sie mit den Methoden der Schnittstellen versehen. Natürlich noch ohne Methodenrümpfe bzw. mit minimalen Rümpfen. Dieses Vorgehen vereinfacht den Schnittstellenentwurf und stellt Konsistenz sicher.

Jeder Schnittstelle liegt ein Protokoll zugrunde. Das Protokoll beschreibt die Vor- und Nachbedingungen der Schnittstellenelemente. Dazu gehören die erlaubten Reihenfolgen, in denen Methoden der Schnittstelle aufgerufen werden dürfen, sowie Annahmen über Eingabeparameter und Zusicherungen über Ausgabeparameter. Das Protokoll von Modulen wird in der Modulsicht beschrieben. Dort, wo es sinnvoll ist, sollte es mit Hilfe von Zustands- oder Sequenzdiagrammen spezifiziert werden. Diese sind dann einzusetzen, wenn der Text allein kein ausreichendes Verständnis vermittelt (insbesondere bei komplexen oder nicht offensichtlichen Zusammenhängen).

Der Bezug zur konzeptionellen Sicht muss klar ersichtlich sein. Im Zweifel sollte er explizit erklärt werden. Auch für diese Sicht muss die Entstehung anhand der Strategien erläutert werden.

5 Datensicht

Hier wird das der Anwendung zugrundeliegende Datenmodell beschrieben. Hierzu werden neben einem erläuternden Text auch ein oder mehrere UML-Klassendiagramme verwendet. Das hier beschriebene Datenmodell wird u.a. jenes der Anforderungsspezifikation enthalten, allerdings mit implementierungsspezifischen Änderungen und Erweiterungen. Siehe die gesonderten Hinweise.

6 Ausführungssicht

Die Ausführungssicht beschreibt das Laufzeitverhalten. Hier werden die Laufzeitelemente aufgeführt und beschrieben, welche Module sie zur Ausführung bringen. Ein Modul kann von mehreren Laufzeitelementen zur Laufzeit verwendet werden. Die Ausführungssicht beschreibt darüber hinaus, welche Laufzeitelemente spezifisch miteinander kommunizieren. Zudem wird bei verteilten Systemen (z.B. Client-Server-Systeme) dargestellt, welche Module von welchen Prozessen auf welchen Rechnern ausgeführt werden.

7 Zusammenhänge zwischen Anwendungsfällen und Architektur

In diesem Abschnitt sollen Sequenzdiagramme mit Beschreibung(!) für zwei bis drei von Euch ausgewählte Anwendungsfälle erstellt werden. Ein Sequenzdiagramm beschreibt den Nachrichtenverkehr zwischen allen Modulen, die an der Realisierung des Anwendungsfalles beteiligt sind. Wählt die Anwendungsfälle so, dass nach Möglichkeit alle Module Eures entworfenen Systems in mindestens einem Sequenzdiagramm vorkommen. Falls Euch das nicht gelingt, versucht möglichst viele und die wichtigsten Module abzudecken.

8 Evolution

In diesem Abschnitt geht es um mögliche Änderungen, Anpassungen bzw. Erweiterungen, die vorgenommen werden müssten, wenn sich Anforderungen des Systems ändern. Dabei ist wichtig, dass sich solche Änderungen möglichst modular realisieren lassen, ohne die bestehende Architektur komplett zu verändern, was sehr aufwändig und somit nicht wünschenswert wäre.
Im Folgenden werden einige wichtige mögliche neue Anforderungen bzw. Erweiterungen aufgelistet und deren jeweiligen zu implementierenden Änderungen an der Architektur beschrieben.

Erweiterungsmöglichkeiten

Da sich aus der Anforderungsspezifikation im Abschnitt "Ausblick" noch keine genauen absehbaren Änderungen ergeben haben, werden im Folgenden potenzielle Änderungen aufgezeigt, die näher beschrieben werden.

1. Erweiterung des GUI-Layouts

Es wäre möglicherweise wünschenswert, wenn man als Benutzer nicht nur ein GUI-Design in der Quiz-App verwenden könnte, sondern mehrere. Dafür müsste eine Funktionalität hinzugefügt werden, um zwischen verschiedenen Benutzeroberflächen wählen zu können (z.B. verschiedene Themes oder Farbwahlen).
Dazu müssten entsprechende Referenzierungen von neuen GUI-Style-Änderungen mit Bilddateien stattfinden sowie für Textanpassungen die XML-Dateien im entsprechendem Paket verändert bzw. erweitert werden.

2. Mehrsprachigkeit

Auch wenn die Mindestanforderungen keine Mehrsprachigkeit für die Quizapp vorschreibt, wäre es ggf. wünschenswert, dass die Möglichkeit besteht, mehrere Sprachen für die Quiz-App zu unterstützen, um einen größeren bzw. internationaleren Spielerkreis anzusprechen. Insofern wäre es denkbar, dass neben Deutsch eine weitere Sprache eingebaut werden könnte. Englisch würde sich natürlich anbieten.