**Softwareprojekt Wintersemester 2019/2020**

**am Fachgebiet Software Engineering, Leibniz Universität Hannover**

**Anforderungsspezifikation**

***<Teamname>***

*SWP-WS1920-PROJEKTNAME-Spec-v02.docx*

**Vorgelegt**

am

von

**Ausführende:**

|  |  |  |
| --- | --- | --- |
| ***Nachname*** | ***Vorname*** | ***Rolle*** |
|  |  |  |
| **Verwendung des Templates:*** Deckblatt ausfüllen
* <Platzhalter> durch echten Text ersetzen
* Hinweise löschen, z.B. <bitte ankreuzen>
* Kapitelstruktur beibehalten, darf verfeinert werden
* Geben Sie der Datei eine höhere Versionsnummer, wann immer Sie sie geändert haben (v02, v03). Dies gilt auch in SVN! **Die explizite Versionsnummer ist im SWP vorgeschrieben.**
* Diesen Kasten löschen.
 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Das Dokument enthält** <Zutreffendes bitte ankreuzen>

Die Anforderungen aus Kundensicht (User Requirements)

x

x

Anforderungen, wie das zu System zu gestalten ist (System Requirements)

Datum, Unterschrift des Projektleiters, auch für die anderen Projektangehörigen

**Kunden-Bewertung**

Der Kunde, Prof. Dr. Kurt Schneider, bestätigt mit seiner Unterschrift, diese Anforderungsspezifikation erhalten, geprüft und für inhaltlich **□ in Ordnung | □ weitgehend in Ordnung | □ deutlich zu verbessernd | □ nicht akzeptabel** befunden zu haben.

Datum, Unterschrift des Kunden; evtl. Vermerk.

**Inhaltsverzeichnis**

[1 Mission des Projekts 3](#_Toc401321791)

[1.1 Erläuterung des zu lösenden Problems 3](#_Toc401321792)

[1.2 Wünsche und Prioritäten des Kunden 3](#_Toc401321793)

[1.3 Domänenbeschreibung 3](#_Toc401321794)

[1.4 Maßnahmen zur Anforderungsanalyse 3](#_Toc401321795)

[2 Rahmenbedingungen und Umfeld 4](#_Toc401321796)

[2.1 Einschränkungen und Vorgaben 4](#_Toc401321797)

[2.2 Anwender 4](#_Toc401321798)

[2.3 Schnittstellen und angrenzende Systeme 4](#_Toc401321799)

[3 Funktionale Anforderungen 4](#_Toc401321800)

[3.1 Use Case-Diagramm 4](#_Toc401321801)

[3.2 Use Case-Beschreibungen 4](#_Toc401321802)

[<Use Case 1: Gesamtsystem starten> 5](#_Toc401321803)

[4 Qualitätsanforderungen 6](#_Toc401321804)

[4.1 Qualitätsziele des Projekts 6](#_Toc401321805)

[4.2 Qualitäts-Prioritäten des Kunden 6](#_Toc401321806)

[4.3 Wie Qualitätsziele erreicht werden sollen 6](#_Toc401321807)

[5 Hinweise zur Umsetzung 7](#_Toc401321808)

[6 Probleme und Risiken 7](#_Toc401321809)

[7 Optionen zur Aufwandsreduktion 7](#_Toc401321810)

[7.1 Mögliche Abstriche 7](#_Toc401321811)

[7.2 Inkrementelle Arbeit 7](#_Toc401321812)

[8 Glossar 7](#_Toc401321813)

[9 Abnahme-Testfälle 8](#_Toc401321814)

# 1 Mission des Projekts

## 1.1 Erläuterung des zu lösenden Problems

<Hier schon einmal ein einigermaßen sinnvoller Text, den Sie aber noch löschen oder ändern dürfen und auf jeden Fall spezieller formulieren müssen. Viel länger muss er dabei nicht werden>

Im Projekt EyeTracker soll ein Spiel entwickelt werden. Mit Hilfe des EyeTrackers EyeX von Tobii sollen interessierte Laien in die Lage versetzt werden, ein Spiel allein mit den Augen zu steuern. Sie sollen davon das Gefühl mitnehmen, etwas Besonderes erlebt zu haben. Die Zielgruppe sind Besucher von Universitätsveranstaltungen wie der Nacht der Wissenschaft. Dazu gehören studieninteressierte Schüler als die relevanteste Zielgruppe, aber auch Eltern, Politiker, Presse und weitere Personen.

Entsprechend dieser Einsatzsituation und -umgebung ist auf eine extrem einfache Bedienung und robuste Reaktion des Spiels zu achten. Die Spiele finden öffentlich auf einer Leinwand statt, so dass ein Spieler auch Publikum hat. Daher sollte das Spiel so kalibriert sein, dass man sich vermutlich nicht blamiert, aber auch nicht sehr schnell und perfekt abschließen kann.

## 1.2 Wünsche und Prioritäten des Kunden

Es folgen die geäußerten Wünsche des Kunden, nach absteigender Priorität geordnet:

* Die Schnittstelle zwischen Komponenten und ApplicationControl ist in einem eigenen Pflichtenheft spezifiziert. Dieses Pflichtenheft ist einzuhalten und geht in der Priorität den Festlegungen in der vorliegenden Spezifikation vor. Die Inhalte dieses Pflichtenhefts werden hier nicht noch einmal wiederholt, sondern vorausgesetzt.

<alles weitere und Projektspezifische bitte hier. Diese Angaben helfen Ihnen, sich später nicht in Details zu verheddern, sondern den Überblick darüber zu wahren, worauf es vor allem ankommt. Das ist besonders wichtig, wenn Sie nicht alles lösen können und Schwerpunkte setzen müssen>

## 1.3 Domänenbeschreibung

Das Produkt wird zu Vorführungen, z.B. bei Universitätsveranstaltungen wie den Hochschulinformationstagen (HIT) oder Erstsemesterbegrüßungen verwendet. Es soll typischerweise vor größeren Gruppen auf großflächigen Projektionen (Beamer, SmartBoard) eingesetzt werden, und zwar in Laborräumen, der InfoLOUNGE oder im Lichthof. Dementsprechend wird hoher Wert auf gute Lesbarkeit von Text und Erkennbarkeit von Symbolen und allen Anzeigen gelegt, auch in einem gewissen Abstand und unter nicht idealen Lichtverhältnissen.

Ziel dieser Veranstaltungen ist, die Zuschauer zu beeindrucken und zu fesseln. Daher sind auch Geräuscheffekte vorgesehen.

<Wie oben: das ist schon einmal für alle Projekte sinnvoll. Sie dürfen es gerne verbessern oder löschen und müssen es sicher noch verfeinern.>

## 1.4 Maßnahmen zur Anforderungsanalyse

<Bitte einfügen: Welche Mockups, Überlegungen, Prototypen wurden erstellt? Jeweils nur sehr kurz erläutern, wozu sie dienten, was dabei herausgekommen ist. Nicht alle Inhalte, nur die Fragen, die Sie klären wollten (und die Erkenntnisse daraus)>

# 2 Rahmenbedingungen und Umfeld

<Ab jetzt gilt generell, ohne dass es jedes Mal dabei stünde: Die Kapitelstruktur bis zur 2.Ebene ist fest, sie können aber hinten noch Kapitel/Unterkapitel anhängen. Die Inhalte sind hier Vorschläge. Sie sind aber alleine verantwortlich für den Text und können schreiben, was Sie für richtig halten.>

## 2.1 Einschränkungen und Vorgaben

Das Spiel erhält von ApplicationControl ein JPanel, auf dem das gesamte Spiel mit allen Oberflächenelementen ablaufen muss. Zusätzlich können akustische Ausgaben erfolgen, also Geräusche oder Musik. In allen Fällen ist darauf zu achten, ausschließlich freie Quellen (ohne Gebühren oder Verletzung des Urheberrechts) zu verwenden.

## 2.2 Anwender

Diese Komponente kommt zum Einsatz, nachdem das Gesamtsystem gestartet, die Komponente ausgesucht wurde. Anwender sind beispielsweise Oberstufenschüler, bei denen man gute Kompetenz in der Bedienung von Computern voraussetzen darf. Das Produkt soll jedoch auch von Personen mit hohem Interesse, aber eher durchschnittlichen Computerkenntnisse bedienbar sein.

## 2.3 Schnittstellen und angrenzende Systeme

Durch die genaue Beachtung des Pflichtenhefts zu Anwendungskomponenten wird sichergestellt, dass die Einbettung dieser Komponente in das Gesamtprogramm funktioniert.

Direkte Schnittstellen zwischen Komponenten verschiedener Teams sind nicht vorgesehen.

<Sie müssen sich Beschreibungen zu technische Schnittstellen von Java-APIs selbst holen. Wenn Sie irgendwelche weiteren Bibliotheken oder Frameworks nutzen wollen, sprechen Sie mit Ihrem Tutor und Kunden darüber.>

# 3 Funktionale Anforderungen

## 3.1 Use Case-Diagramm

<Viele der folgenden Unterkapitel sind leer. Füllen Sie sie entsprechend Ihren SWT-Kenntnissen; bedenken Sie dabei auch, dass ein UML-Diagramm oder auch ein Use Case ohne Erläuterung oft nicht aussagekräftig ist. Scheuen Sie sich also nicht, noch Kommentare oder weitere Angaben zu machen, wenn das der Klärung dient>

## 3.2 Use Case-Beschreibungen

<Abläufe mit interessanter Interaktion zwischen einem Akteur und verschiedenen Systemteilen werden als Use Cases beschrieben. Zu jedem Use Case kann es weitere, detaillierende Anforderungen geben, die dort angegeben sind (z.B R412). Rein interne Abläufe werden unter Umständen anders notiert (z.B. Ablauf- oder Sequenzdiagramm). Hier handelt es sich um Anforderungen des Kunden. >

**<Der folgende UseCase ist ein Beispiel. Löschen Sie ihn am Ende! Sonst gibt es Ärger.**

**Setzen Sie hier Ihre eigenen UseCases mit kurzer Erläuterung ein.>**

### <Use Case 1: Gesamtsystem starten>

<Eine Komponente ist **in die Applikation integriert**, wenn sie als Komponente in der Applikation deklariert ist und gemeinsam mit der Applikation übersetzt worden ist.

|  |  |
| --- | --- |
| **Use Case 1**  | **Gesamtsystem starten**  |
| **Erläuterung**    | Indem die Applikation gestartet wird, wird auch die Konfiguration aus verfügbaren Komponenten geladen und gestartet.  Sie sind über ein Menü startbar. |
| **Status**    | -- |
| **Systemgrenzen (Scope)**    | Gesamtsystem |
| **Ebene**    | Hauptfunktion |
| **Vorbedingung**    | Zumindest eine Komponente ist in der Applikation integriert |
| **Mindestgarantie**    | Menü mit mindestens einer Komponente wird angezeigt |
| **Erfolgsgarantie**    | Ein Menü wird angezeigt, das der Konfiguration entspricht und von dem aus die verfügbaren Komponenten startbar sind. |
| **Stakeholder**    | Systembediener  |
| **Hauptakteur**    | Systembediener |
| **Auslöser**    | Systembediener doppelklickt auf das Systemlogo |
| **Hauptszenario**    | 1. Systembediener doppelklickt auf das Systemlogo
2. Applikation liest Konfigurationsdatei ein
3. Applikation zeigt Menü an, das der Konfiguration entspricht
4. Applikation wartet
 |
| **Erweiterungen** | 2a: WENN Konfigurationsdatei nicht korrekt, DANN zeige alle Komponenten als schlichte Liste an |
| **Priorität**    | unverzichtbar    |
| **Verwendungshäufigkeit**    | regelmäßig    |

***Erläuterungen und Details***

* **Liest Konfigurationsdatei** ein: Siehe Use Case zum Ändern der Konfiguration, dort steht auch das Format der Einträge und ihre Bedeutung
* **Zeigt Menü** ergibt sich ebenfalls aus den Einträgen in der Konfigurationsdatei
* **Schlichte Liste** enthält jeweils den Namen, z.B. auf einem Button. Keine Strukturierung.
* **Systemlogo** wird noch festgelegt
* **„In die Applikation integriert“** kann unterschiedlich umgesetzt werden.
	+ **R110:** Irgendwo muss die Komponente deklariert werden und dann über einen Namen (String) innerhalb der Applikation ansprechbar sein.
* **Mockups des Menüs** siehe weiter unten.
* **Wartet** ist ein anderer Use Case, siehe unten
* **Die Erweiterung ist wichtig:**
	+ R120: In keinem Fall darf eine fehlende oder inkorrekte Konfigurationsdatei zum Absturz oder Abbruch führen.
	+ R121: Stattdessen ist es immer erlaubt, eine einfache, unstrukturierte und beispielsweise alphabetisch nach Namen der Komponente sortierte Liste von Links oder Buttons zu präsentieren. Es wird also notfalls die Funktion **gerettet, auch wenn das Menü nicht zu haben ist.>**

# 4 Qualitätsanforderungen

## 4.1 Qualitätsziele des Projekts

<Im Grunde sollte hier ein einfaches Qualitätsmodell stehen (s. Vorlesung), in dem Sie klären, welche Qualitätsanforderungen bestehen und an welchen Kriterien/Maßen die Erfüllung gemessen wird>

## 4.2 Qualitäts-Prioritäten des Kunden

Die Qualitätsziele sind wie folgt absteigend priorisiert:

## 4.3 Wie Qualitätsziele erreicht werden sollen

# 5 Hinweise zur Umsetzung

<Hier sollen alle weiteren, wichtigen Angaben stehen, wie zum Beispiel:

- Oberflächen-Mockups. Bitte unbedingt mit kurzer Erläuterung, was man da sieht und was Sie mit dem Mockup ausdrücken wollen!

- Angaben zur Umsetzung, also eher Systemanforderungen, in Form von Klassendiagrammen, Sequenzdiagrammen oder irgendwelchen anderen Modellen (Petri-Netzen, was immer). Auch hier reicht ein unkommentiertes Diagramm in der Regel nicht aus. Machen Sie sich verständlich.

- Sie können hier auch noch ganz andere Sachen einbauen, die Sie für wichtig halten. Das Dokument soll aber nicht unnötig aufgebläht werden. Kopieren Sie nicht irgendwelche allgemeinen Texte hinein>

# 6 Probleme und Risiken

<Halten Sie sich an den Stil aus der Vorlesung>

**WENN** .. DANN … . Konsequenz: …

**Abhilfe**: ..

**<Es kann sein, dass sich auch daraus noch Anforderungen ergeben, z.B.**

* R910 **Es muss unbedingt …**.>

# 7 Optionen zur Aufwandsreduktion

## 7.1 Mögliche Abstriche

<Sehr wichtig: Klären Sie gleich zu Anfang, welche Funktionen und Anforderungen für den Kunden möglicherweise nicht ganz so wichtig sind. Dort kann man angreifen, wenn nicht alles zu machen ist.>

## 7.2 Inkrementelle Arbeit

<Im SWP wird nun inkrementell gearbeitet. Das heißt, Sie dürfen noch Änderungswünsche aufnehmen. Um am Ende den besten Kundennutzen zu erzielen, werden Sie versuchen, die wichtigsten Dinge immer vorzuziehen, also in die erste Iteration. Diese Reihenfolge sollen Sie hier beschreiben. Sie entspricht in etwa dem Backlog, der anschließend aus Ihrer Spezifikation abgeleitet wird>

# 8 Glossar

|  |  |
| --- | --- |
| **Komponente** | Java-Klasse, mit der sich das Resultat eines Teams starten lässt. Muss Instanz einer Unterklasse der hier erstellten Klasse Component sein. |
|  |  |
|  |  |

# 9 Abnahme-Testfälle

<Müssen zumindest enthalten:

* Wie baut man Startvoraussetzung auf? (setup)
* Was sind die Eingaben (in)?
* Welche Ausgaben oder Reaktionen werden erwartet (soll)?

Die Testfälle sollen sehr spezifisch sein. Gehen Sie aber davon aus, dass Kunden manchmal leicht davon abweichen werden.>