Requirements and Design Documentation

(RDD)

Version 0.0

ESEP - Praktikum - Wintersemester 2016

Lüdemann	Mona	2212744	mona.luedemann1@haw-hamburg.de
Butkereit	Marvin	2247550	marvin.butkereit@haw-hamburg.de
Schumacher	Wilhelm	2245216	wilhelm.schumacher@haw-hamburg.de
Melkonyan	Anushavan	2243668	anushavan.melkonyan@haw-hamburg.de
Colbow	Marco	2177095	marco.colbow@haw-hamburg.de
Cakir	Mehmet	2195657	mehmet.cakir@haw-hamburg.de

18. Oktober 2016

Änderungshistorie:

Version	**Author**	**Datum**	**Anmerkungen/Änderungen**

Inhaltsverzeichnis

Inhaltsverzeichnis **2**

1 Teamorganisation **3**
1.1 Verantwortlichkeiten . . . 3
1.2 Absprachen . . . 3
1.3 Repository-Konzept . . . 4

2 Projektmanagement **4**
2.1 Prozess . . . 4
2.2 PSP/Zeitplan/Tracking . . . 4
2.3 Qualitätssicherung . . . 4

3 Randbedingungen **5**
3.1 Entwicklungsumgebung . . . 5
3.2 Werkzeuge . . . 5
3.3 Sprachen . . . 5

4 Requirements and Use Cases **5**
4.1 Systemebene . . . 5
4.1.1 Stakeholder . . . 5
4.1.2 Anforderungen . . . 6
4.1.3 Systemkontext . . . 6
4.1.4 Use Cases . . . 6
4.2 Systemanalyse . . . 6
4.3 Softwareebene . . . 7
4.3.1 Systemkontext . . . 7
4.3.2 Anforderungen . . . 7

5 Design **7**
5.1 System Architektur . . . 7
5.2 Datenmodellierung . . . 7
5.3 Verhaltensmodellierung . . . 7

6 Implementierung **7**

7 Testen **8**
7.1 Testplan . . . 8
7.2 Abnahmetest . . . 8
7.3 Testprotokolle und Auswertungen . . . 8

8 Lessons Learned **8**

9 Anhang **8**
9.1 Glossar . . . 8
9.2 Abkürzungen . . . 8

1 Teamorganisation

Grundsätzlich kann jedes Teammitglied eine Aufgabe seiner Wahl übernehmen. Die Aufgaben, dessen Verteilung bei jedem Meeting neu beschlossen wird, richten sich nach den zu bewältigenden Milestones(siehe [**?**]) zum jeweiligen Praktikumstermin. Allerdings wurden für die Projektleitung und die Pflege des RDD-Dokuments jeweils eine Person bestimmt, welche im Unterkapitel 1.1 eingesehen werden können.

1.1 Verantwortlichkeiten

Aufgabe	**Zuständige/r**	**Bemerkung**
Projektleitung	Mona	Die Projektleitung hält den Fortschritt im Auge und benachrichtigt insbesondere bei Nichteinhalten des Zeitplans alle Teammitglieder. Außerdem hat die Projektleitung bei Unstimmigkeiten immer das letzte Wort.
RDD-Pflege	Mehmet	Der Zuständige ist für die Gestaltung und für die Vollständigkeit des RDDs verantwortlich. Er kann andere Gruppenteilnehmer dazu auffordern Inhalte für das Dokument zu erarbeiten und ihm bereit zu stellen.
Protkollführung	Alle Teammitglieder	Die Protokollführung wird reihum von einem anderen Gruppenmitglied übernommen. Dabei wird folgende Reihenfolge eingehalten: $Mona \rightarrow Marvin \rightarrow Marco \rightarrow Wilhelm \rightarrow Mehmet \rightarrow Anushavan$

Tabelle 1: Zuteilung von Verantwortlichkeiten

1.2 Absprachen

Zur Kommunikation außerhalb der Praktikumstermine werden Slack und WhatsApp verwendet. Unstimmigkeiten, Fragen und Inkenntnissetzung können somit interaktiv geklärt bzw. mitgeteilt werden. Es wird erwartet, dass jedes Teammitglied so schnell wie möglich darauf reagiert. In folgender Abbildung 1 werden die Termine der Meetings dargestellt:

Terminplan für Meetings				
Oktober	Mi, 05.10. ab 16:00 Uhr	Do, 13.10. ab 12:00 Uhr	Mi, 19.10. ab 16:00 Uhr	Mi, 26.10. ab 16:00 Uhr
November	Do, 03.11. ab 12:00 Uhr	Do, 10.11. ab 12:00 Uhr	Mi, 16.11. ab 16:00 Uhr	Mi, 23.11. ab 16:00 Uhr
Dezember	Do, 01.12. ab 12:00 Uhr	Mi, 07.12. ab 16:00 Uhr	Mi, 14.12. ab 16:00 Uhr	Do, 22.12. ab 12:00 Uhr
Weitere Termine können/müssen je nach Bedarf in der Gruppe vereinbart werden.				

Abbildung 1: Terminplan der Meetings

1.3 Repository-Konzept

Das Projekt wird mit dem Versionskontrollsystem Git verwaltet. Zentral wurde ein Repository auf GitHub angelegt. Erreichbar ist das Repository unter `https://github.com/mbutkereit/conveyor`. Änderungen werden lokal auf einem anderen Branch als auf dem Master vorgenommen. Wenn die Änderungen gepusht werden sollen, muss zuvor auf den Master Branch gewechselt und gepullt werden. Nun kann ein Merge mit dem lokal anderen Branch durchgeführt und ggf. Mergekonflikte gelöst werden. Anschließend kann gepusht werden.

2 Projektmanagement

Für die Gewährleistung eines guten Managements, werden in den folgenden Kapiteln erklärt wie die Teammitglieder mit ihren Aufgaben umgehen bzw. wann eine gegenseitige Benachrichtigung über ihren Fortschritt spätestens stattfinden sollte.

2.1 Prozess

Das Projekt wird auf Grundlage der geforderten Milestones nach und nach umgesetzt. Für jede Implementierung ist zuvor ein geeignetes sowie größtenteils selbsterklärendes bzw. verständliches, aber auch möglichst vollständiges Diagramm anzufertigen. Bestenfalls sollte die Visualisierung vor der Implementierung allen anderen Teammitgliedern vorgestellt werden, um mögliche Verbesserungen einzuholen und ggf. Konflikte früh zu erkennen sowie sie zu lösen.

2.2 PSP/Zeitplan/Tracking

Zu jedem Praktikumstermin wird erwartet, dass die verteilten Aufgaben bzw. Milestones erfüllt werden. Um dies möglichst zu gewährleisten, muss jedes Teammitglied bei Schwierigkeiten andere Teammitglieder darüber sofort in Kenntnis setzen, damit frühzeitig ausgeholfen werden kann.

2.3 Qualitätssicherung

Hinsichtlich der Qualitätssicherung, werden die vier Punkte Team, Modellierung, Code und Förderband herangezogen.

1. **Team:** Jedes Teammitglied sollte über seine eigenen Fähigkeiten im Klaren sein und möglichst nur Aufgaben übernehmen, wofür es sich am besten geeignet fühlt. Darüber hinaus muss jedes Teammitglied bei Möglichkeit stets seine Unterstützung anbieten. Bei Problemen oder Überforderung müssen alle anderen Teammitglieder darüber unterrichtet werden und Aufgaben ggf. neu verteilt werden.
2. **Modellierung:** Vor der Implementierung muss eine geeignete Visualisierung erstellt, anderen Teammitgliedern vorgestellt und untereinander diskutiert werden.
3. **Code:** Der Code wird nach beschlossenen Konventionen gefertigt. Dabei werden bekannte Pattern eingesetzt und möglichst einfache sowie übersichtliche Realisierungen angestrebt.
4. **Förderband:** Um hohen Durchsatz sowie Effizienz bei der Aussortierung zu erzielen, werden die Komponenten mit der höchstmöglichen Leistung für die jeweilige Situation angetrieben, während die Sicherheit des Bedieners im Vordergrund steht. Dabei werden Fehler- bzw. Ausnahmezustände ggf. durch einfache Signalcodes mithilfe der Ampel dem Bediener mitgeteilt.

3 Randbedingungen

In diesem Kapitel werden die Bedingungen genannt unter denen das Projekt umgesetzt wird und die Mittel, die für die Umsetzung herangezogen werden.

3.1 Entwicklungsumgebung

Die beiden Förderbänder werden über zwei QNX Systeme gesteuert, die über eine serielle Schnittstelle verbunden sind. Als IDE wird QNX Momentics auf Windows 7 verwendet.

3.2 Werkzeuge

- QNX Momentics IDE 5.0
- Latex
- Git(GitHub)

3.3 Sprachen

Das System wird in C++ 11 programmiert. Die dazukommenden Bibliotheken sind in folgender Tabelle 2 aufgelistet:

Name	**Version**	**Autor**
HWaccess.h	Unknown	Prof. Dr. Stephan Pareigis
HAWThread.h	Unknown	Prof. Dr. Stephan Pareigis
Lock.h	0.1	Simon Brummer

Tabelle 2: Verwendete Programmierbibliotheken

4 Requirements and Use Cases

4.1 Systemebene

4.1.1 Stakeholder

Stakeholder	**Interessen**
Kunde	- fehlerfreie Umsetzung der Anforderungen - erfolgreiche Beendigung des Projektes
Designer	- übersichtliches, leicht erweiterbares Design - sorgfältige Dokumentation
Entwickler	- präzises Design - sinnvolle Kommentare - lesbarer Code
Tester	- übersichtliches, vollständiges Testkonzept
Bediener (Mitarbeiter, die das Laufband später bedienen sollen)	- einfache und intuitive Bedienung
Instanthalter	- robustes System
Andere Mitarbeiter	- Kenntnis über System und Funktionsweise

Tabelle 3: Tabellenunterschrift

4.1.2 Anforderungen

In der Aufgabenstellung sind Anforderungen an das System gestellt. Arbeiten sie diese hier auf und ergänzen sie diese entsprechend der Absprachen mit dem Betreuer. Achten sie auf die entsprechende Attribuierung. Berücksichtigen sie auch mögliche Fehlbedienungen und Fehlverhalten des Systems.

4.1.3 Systemkontext

Use Cases werden aus einer bestimmten Sicht erstellt. Dokumentieren sie diese mittels Kontextdiagramm oder Use Case Diagramm. Die Use Cases und Test Cases müssen zu der hier verwendeten Nomenklatur konsistent sein.

4.1.4 Use Cases

Dokumentieren sie hier, welche Use Cases sie auf der Systemebene implementieren müssen. Die Test Cases sollen später zu den Use Cases konsistent sein.

4.2 Systemanalyse

Ihr technisches System hat aus Sicht der Software bestimmte Eigenschaften. Was muss man für die Entwicklung der Software in Struktur, Schnittstellen, Verhalten und an Besonderheiten wissen? Wählen sie eine Kapitelstruktur, die am besten zur Dokumentation ihrer Ergebnisse geeignet ist.

4.3 Softwareebene

Sie sollen Software für die Steuerung des technischen Systems erstellen. Aus den Anforderungen auf der Systemebene und der Systemanalyse ergeben sich Anforderungen für Ihre Software. Insbesondere wird sich die Software der beiden Anlagenteile in einigen Punkten unterscheiden. Dokumentieren sie hier die Anforderungen, die sich speziell für die Software ergeben haben.

4.3.1 Systemkontext

Wie sieht der Kontext Ihrer Software aus? Wie erfolgt die Kommunikation mit Nachbarsystemen? Liste der ein- und ausgehenden Signale/Nachrichten.

4.3.2 Anforderungen

Welche wesentlichen Anforderungen ergeben sich aus den Systemanforderungen für ihre Software? Achten sie auf die entsprechende Attribuierung. Berücksichtigen sie auch mögliche Fehlbedienungen und Fehlverhalten des Systems.

5 Design

Anmerkung: Die Implementierung MUSS zu Ihrem Design-Modell konsistent sein. Strukturen, Verhalten und Bezeichner im Code müssen mit dem Modell übereinstimmen. Daher ist ein wohlüberlegtes Design wichtig.

5.1 System Architektur

Erstellung sie eine Architektur für Ihre Software. Geben sie eine kurze Beschreibung Ihrer Architektur mit den dazugehörenden Komponenten und Schnittstellen an. Dokumentieren sie hier wichtige technische Entscheidungen. Welche Pattern werden gegebenenfalls verwendet? Wie erfolgt die interne Kommunikation?

5.2 Datenmodellierung

Bestimmen sie das Datenmodell und dokumentieren sie es hier mit Hilfe von UML Klassendiagrammen unter Beachtung der Designprinzipien. Die Modelle können mit Hilfe eines UML-Tools erstellt werden. Hier ist dann ein Übersichtsbild einzufügen. Geben sie eine kurze textuelle Beschreibung des Datenmodells und deren wichtigsten Klassen und Methoden an.

5.3 Verhaltensmodellierung

Ihre Software muss zur Bearbeitung der Aufgaben ein Verhalten aufweisen. Überlegen sie sich dieses Verhalten auf Basis der Anforderungen und modellieren sie das Verhalten unter Verwendung von Verhaltensdiagrammen. Sie können für die Spezifikation der Prozess-Lenkung entweder Petri-Netze oder hierarchische Automaten verwenden. Die Modelle können mit Hilfe eines UML-Tools erstellt werden. Hier sind dann kommentierte Übersichtsbilder einzufügen.

6 Implementierung

Anmerkung: Nur wichtige Implementierungsdetails sollen hier erklärt werden. Code-Beispiele (snippets) können hier aufgelistet werden, um der Erklärung zu dienen. Anmerkung: Bitte KEINE ganze Programme hierhin kopieren!

7 Testen

Machen sie sich auf Basis ihrer Überlegungen zur Qualitätssicherung Gedanken darüber, wie sie die Erfüllung der Anforderungen möglichst automatisiert im Rahmen von Unit-Test, Komponententest, Integrationstest, Systemtest, Regressionstest und Abnahmetest überprüfen werden.

7.1 Testplan

Definieren sie Zeitpunkte für die jeweiligen Teststufen in ihrer Projektplanung. Dazu können sie die Meilensteine zu Hilfe nehmen.

7.2 Abnahmetest

Leiten sie die Abnahmebedingungen aus den Kunden-Anforderungen her. Dokumentieren sie hier, welche Schritte für die Abnahme erforderlich sind und welches Ergebnis jeweils erwartet wird (Test Cases).

7.3 Testprotokolle und Auswertungen

Hier fügen sie die Test Protokolle bei, auch wenn Fehler bereits beseitigt worden sind, ist es schön zu wissen, welche Fehler einst aufgetaucht sind. Eventuelle Anmerkung zur Fehlerbehandlung kann für weitere Entwicklungen hilfreich sein. Das letzte Testprotokoll ist das Abnahmeprotokoll, das bei der abschließenden Vorführung erstellt wird. Es enthält eine Auflistung der erfolgreich vorgeführten Funktionen des Systems sowie eine Mängelliste mit Erklärungen der Ursachen der Fehlfunktionen und Vorschlägen zur Abhilfe

8 Lessons Learned

Führen sie ein Teammeeting durch in dem gesammelt wird, was gut gelaufen war, was schlecht gelaufen war und was man im nächsten Projekt (z.B. im PO) besser machen will.

Listen sie für die Aspekte jeweils mindestens drei Punkte auf. Weitere Erfahrungen und Erkenntnisse können hier ebenso kommentiert werden, auch Anregungen für die Weiterentwicklung des Praktikums.

9 Anhang

9.1 Glossar

Eindeutige Begriffserklärungen

9.2 Abkürzungen

Listen sie alle Abkürzungen auf, die sie in diesem Dokument benutzt haben.