Requirements / Design- und Test-Dokumentation

Version 0.1

ESEP - Praktikum - Sommersemester 2021

Abrams, Lasse (acs227) – *Lasse.Abrams@haw-hamburg.de*
Hoffmann, Justin (act752) – *Justin.Hoffmann@haw-hamburg.de*
Ohsten, Moritz (acs356) – *Moritz.Ohsten@haw-hamburg.de*
Protsch, Hugo (acs521) – *Hugo.Protsch@haw-hamburg.de*
Stoltz, Jendrik (acs357) – *Jendrik.Stoltz@haw-hamburg.de*

1 TEAMORGANISATION

{ch:tean

1.1 Verantwortlichkeiten

{sec:ver

Die Verantwortlichkeiten gehen aus Tabelle **??** hervor.

Responsibility	Vorname	Name
Software Architekt	Lasse	Abrams
Projektleitung	Justin	Hoffmann
Requirements Manager	Moritz	Ohsten
Testleitung, Configuration Manager	Hugo	Protsch
Programmierleitung	Jendrik	Stoltz

Tabelle 1.1: Verantwortlichkeiten

{tab:ver

1.2 Repository-Konzept

{sec:rep

1.2.1 Ordner Organisation

{subsec:

Momentan bestehen zwei toplevel Ordner: *doc* und *esep_ss2021_Gruppe2_3*. Im ersteren wird das Pflichtenheft und jegliche weitere Dokumentation, inklusive Diagramme abgelegt. Im letzteren befindet sich momentan der Code für das Projekt. Der Name dessen ist jedoch noch temporär und wird noch angepasst, da wahrscheinlich zwei Ordner für den Code benötigt werden – einer pro Förderband.

1.2.2 Git Practices

{subsec:

Branching Strategie Für unsere Branching-Strategie verwenden wir als Grundlage GitLab Flow. Der default Branch ist der *master*-Branch, es können keine Commits direkt auf diesem gepusht werden. Um Änderungen vorzunehmen wird ein Feature-Branch erstellt. In diesem werden die nötigen Änderungen vorgenommen. Sobald das Feature abgeschlossen ist, wird dieser wieder in den *master*-Branch gemerged. Die Feature-Branches sind vom Umfang jeweils möglichst klein zuhalten, dies ermöglicht einen genaueren Review Prozess, der weiter in Abschnitt **??** ausgeführt wird.

Für das Pflichtenheft besteht außerdem ein eigener Branch mit dem Namen *latex*, um den *master*-Branch übersichtlicher und Code-basiert zu halten. In diesem können direkt Änderungen vorgenommen werden, dies ist jedoch nur für kleine Fixes, wie z. B. Rechtschreibung, Formatierung usw., vorgesehen. Für neue Abschnitte, oder für das Ändern des Inhaltes bestehender Abschnitte ist ein neuer Branch anzulegen, sodass die Änderung reviewt werden können.

Commits Commits sind möglichst atomar zu halten: So stellt jeder Commit genau eine vollständige Änderung, zum Beispiel das Beheben eines Fehlers, das Hinzufügen einer Funktion o. Ä. dar. Der Zustand des Repositories bzw. Codes soll optimalerweise nach jedem Commit funktional sein. Dies hat den Vorteil, dass Änderung leicht rückgängig gemacht und mithilfe von *git cherry-pick* ausgewählt werden können.

Commit-Messages sind aussagekräftig zu wählen. Es soll beschrieben werden was geändert wurde bzw. warum es geändert wurde. Wie etwas geändert wurde, geht hingegen aus dem Inhalt des Commits selber hervor und soll somit nicht erwähnt werden. Wir habens uns darauf geeinigt die Commit-Messages in Englisch im Imperativ zu verfassen, sodass diese mit Standard Commit-Messages von Git konsistent sind.

{subsec:

Auslieferungen Für Auslieferungen benutzten wir Git Tags in Kombination mit GitLab Releases, das genaue Vorgehen muss noch festgelegt werden. Die zwei möglichen Varianten hierbei sind einmal wie in Gitflow beschrieben ein Production-Branch, in den der *master* gemerged wird und der den Stand der

aktuellen Auslieferung widerspiegelt. Alternativ kann auch direkt im *master*-Branch ein Commit getaggt werden.

2 PROJEKTMANAGEMENT

{ch:proj

2.1 Absprachen

{sec:abs

- Die Kommunikation läuft über unseren Discord Server, für dringenden Angelgelegenheiten wird eine WhatsApp-Gruppe genutzt
- Zweimal pro Woche wird ein Meeting gehalten. Die Agenda wird in GitLab in je einem Issue geführt. Es ist ein tabellarisches Protokoll zu führen, welches die Ergebnisse festhält, dieses wird in das Issue gestellt.
- Wir verwenden den Google C++ Coding Style Guide

2.2 Projektplan

{sec:pro

Wir verwenden einen agilen Ansatz für das Umsetzten des Projektes und setzten diesen auf GitLab um. Dabei orientieren wir uns an dem Guide zur Verwendung von GitLab für agile Software Entwicklung:

- Die Tasks aus dem Projektplanung-Template wurden auf GitLab als Epics angelegt. Es wird jeweils eine Verantwortlichkeit mithilfe eines Labels zugeordnet
- Falls Epics größere Unteraufgaben beinhalten, werden diese als Subepics formuliert
- Für die festen Phasen werden Milestones verwendet
- Für einzelne Aufgaben werden Issues verwendet, die, soweit möglich, einem Epic und Milestone zugeordnet werden. In einem Issue kann mit Kommentaren über das Issue selber diskutiert werden. Über die Assignee Funktion können einem Issue beliebig viele Personen zugeteilt werden, die für für das Lösen dieses verantwortlich sind.

Sobald an einem Issue gearbeitet wird, wird, falls ein eigener Branch nötig ist, ein Draft MR erstellt. In diesem sammeln sich alle Änderungen für dieses Issue. Jegliche implementierungsspezifische Kommentare können direkt in Commits an den jeweiligen Codezeilen hinterlassen werden und werden im MR angezeigt. Bei Änderungsvorschlägen ist ein neuer Thread statt eines Kommentars zu eröffnen, sodass diese im MR auf den ersten Blick über „unresolved threads“ zu sehen sind. Sobald der Branch gemerged werden soll, wird der MR als *ready* markiert und das Label *workflow::pending review* zugewiesen, was den anderen Teammitgliedern signalisiert, dass sie mit dem Reviewprozess wie in Abschnitt **??** beschrieben beginnen können.

2.3 Qualitätssicherung

{sec:qua

Die Qualität wird mithilfe eines Review bzw. Approval Prozesses bei MR sichergestellt. Für einen Merge in den Master-Branch werden 3 Approvals benötigt. Für einen Merge in den Latex-Branch werden 2 Approvals benötigt. Die Anzahl an nötigen Approves kann bei jedem MR bearbeitet werden. Wir haben uns darauf geeinigt, diese nicht herunter zu setzten, sondern im Einzelfall zu erhöhen, falls dieses nötig erscheint. Jeder aus dem Team kann einen MR approven.

Wenn ein MR approved wird, übernimmt der Approver neben dem Implementierer die volle Verantwortung über die Korrektheit und Fehlerfreiheit der Änderungen. Somit ist der Code Zeile für Zeile durchzugehen und auf Fehler sowie Abweichung von unseren Codingstyle-Guidelines zu überprüfen.

3 REQUIREMENTS UND USE CASES

{ch:requ

3.1 Systemebene

{sec:sys

3.1.1 Stakeholder

{subsec:

Externe Stakeholder

- Auftraggeber
  - Erfüllung aller spezifizierten Anforderungen
  - Pünktliche Lieferung zum vorgegebenen Termin
  - Verwendung der vorgegebenen Hard- und Software
- Betreuer
  - Erreichen der Ziele am Ende jeder Phase
  - Möglichst vollständige Dokumentation als Rückmeldungsgrundlage
- Benutzer
  - System stellt keine Gefahr dar
  - Information über Fehlerzustände
  - Möglichst selten Eingreifen erforderlich
  - Hoher Durchsatz
  - Einfache Bedienung und Inbetriebnahme (Dokumentation)
- Verwaltung TI-Labor
  - Keine Beschädigung der Anlagen

Interne Stakeholder

- Entwickler
  - Gute Testbarkeit
  - Einfache Erweiterbarkeit und Modularität
  - Einheitliche Schnittstellen und Benennungen
  - Dokumentation (im Code) für Fehlersuche und Teamarbeit

3.1.2 Anforderungen

{subsec:

3.1.3 Systemkontext

{subsec:

3.1.4 Use Cases / User Stories

{subsec:

UC1: UseCase Name

{uc:1}

siehe **??**.

Use Case					
ID	UC1	Name	UseCase Name	Priorität	hoch
Mainflow	1) do something 2) do this 2a) or tihs				
Alternate flow	1) do something 2) do this 1) Very long text: Lorem ipsum dolor sit amet, consectetur adipiscing elit, 2) do this 2a) or tihs				
Description	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum				

UC10: UseCase Bootup Configuration

{uc:10}

3.2 Systemanalyse

{sec:sys

3.3 Softwareebene

{sec:sof

3.3.1 Systemkontext

{subsec:

3.3.2 Anforderungen

{subsec:

Use Case					
ID	UC10	Name	UseCase Bootup Configuration	Priorität	hoch
Actors	$\text{FTS}_2 Control_P anel$				
Preconditions	System ist mit Strom versorgt und Hochgefahren				
Mainflow	1) HAL wird initialisiert 2) Herstellen einer Verbindung zu $\text{FTS}_2 Kontrolleuchte1 auf Control_P anel Blink_S lowaktivieren$ 3) Warten auf Drücken des Startknopfes auf dem ControlPanel 4) $\text{EVNT}_S W_C OMM_P RI_R EQ an FTS_2 schicken Warten auf EVNT_S W_C OMM_S EC_A KC von$ 5) Eigenen Betriebsmodus auf PRIMARY setzen 6) $\text{EVNT}_S W_C OMM_P RI_A KC an FTS_2 schicken Kontrolleuchte1 auf Control_P aneldauerha$				
Alternate flow 1	9) at step 4 of Mainflow 1) Event $\text{EVNT}_S W_C OMM_P RI_R EQ von FTS_2 Empfangen Eigenen Betriebsmodus au$ 2) $\text{EVNT}_S W_C OMM_S EC_A KC an FTS_2 sende auf EVNT_S W_C OMM_P RI_A KC von FTS_2 wa$ 3) Eigenen Betriebsmodus auf SECONDARY setzen 4) Kontrolleuchte 1 auf $\text{Control}_P aneldauerhaft aktivieren$				
Alternate flow 2	at step 2 of Mainflow 1) Verbindung schlägt fehl 3) Zurück in Schritt 2 Mainflow				
Postconditions	1) HAL ist initialisiert 2) Verbindung mit $\text{FTS}_2 hergestellt Betriebsmodus der Anlage ist festgelegt$				
Exceptional flow 1	3) at step 1 of Mainflow 1) Initialisierung der HAL schlägt fehl 2) Fehlermeldung ausgeben 3) System abschalten				
Exceptional flow 2	at step 6 of Mainflow 1) Timeout beim Empfangen von $\text{EVNT}_S W_C OMM_S EC_A KC Fehler COMM_E RROR aus lös$				
Exceptional flow 3	2) at step 4 of Alternate flow 1) Timeout beim Empfangen von auf $\text{EVNT}_S W_C OMM_P RI_A KC Fehler COMM_E RROR aus$				
Description	Dieser Use Case beschreibt die Initialisierung des Systems, sowie die Bestimmung von Primary und Secondary-Anlage. Das Primary System wird bestimmt Verwendete Signale sind $\text{EVNT}_S W_C OMM_P RI_R EQ(Sendendes System will Primary sein), EVNT_S W_C OMM_S EC$				

4 DESIGN

{ch:desi

4.1 Systemarchitektur

{sec:sys

4.2 Datenmodellierung

{sec:dat

4.3 Verhaltensmodellierung

{sec:ver

5 IMPLEMENTIERUNG

{ch:impl

6 TESTEN

{ch:test

6.1 Testplan

{sec:tes

6.2 Testszenarien

{sec:tes

6.3 Abnahmetest

{sec:abn

6.4 Testprotokolle und Auswertungen

{sec:tes

7 LESSONS LEARNED

{ch:less

A GLOSSAR

{ch:glos

B ABKÜRZUNGEN

{ch:abku