Requirements and Design Documentation

Version 6.01

SE2P – Praktikum – WS2012

Jan-Tristan Rudat, 2007852, jan-tristan.rudat@haw-hamburg.de

Martin Slowikowski, 1999166, martin.slowikowski@haw-hamburg.de

Chris Addo, 2010200, christopher.addo@haw-hamburg.de

Jens Eberwein, 2007797, jens.eberwein@haw-hamburg.de

Changelog:

Version	Author	Datum	Anmerkungen
0.01	Rudat	14.10.2012	RDD erstellt + Requirements
0.02	Slowikowski	16.10.2012	UML
1.00	Slowikowski	17.10.2012	Aufgabenplan, Milestone 1
1.01	Rudat	17.10.2012	Klassen- + Komponentendiags
2.00	Rudat	22.10.2012	PSP grob
3.01	Slowikowski	07.11.2012	Regressionstests
3.02	Eberwein, Rudat	10.11.2012	Fertigstellung PSP
3.03	Eberwein, Addo	13.11.2012	Erstellung Zustandsautomaten
4.01	Addo, Slowikowski	28.11.2012	Regressionstest State Pattern, Ablauf eingefügt
4.02	Addo, Slowikowski	01.12.2012	FSMs aktualisiert im RDD
4.03	Rudat, Eberwein	01.12.2012	Klassendiagramme aktualisiert
5.00	Addo, Slowikowski	01.12.2012	Timerkonzept eingefügt
6.01	Alle	12.12.2012	Bedienkonzept, Lessons Learned

Inhaltsverzeichnis

1 Motivation

Im Rahmen des Studienganges “Technische Informatik” an der HAW Hamburg, soll im Rahmen des vierten Semesters ein Kurs namens Software Engineering 2 absolviert werden.
In diesem Kurs werden vertiefende Grundlagen des Software Engineering vermittelt, sowie eine Aufgabe erteilt, in welcher die Software für eine Werkstücksortieranlage entwickelt werden soll.
Die Werkstücksortieranlage besteht aus zwei Förderbandmodulen, welche über zwei GEME Rechner gesteuert werden. Beide Rechner sind über eine serielle Schnittstelle miteinander verbunden.
Aus diesem Kontext ist das vorliegende Dokument entstanden.

2 Randbedinungen

2.1 Entwicklungsumgebung

- Visual Paradigm 10.0 Enterprise
- Momentics 4.70 IDE
- QNX 6.5

2.2 Werkzeuge

- GIT
- TortoiseGIT
- Notepad++

2.3 Sprachen

- C
- C++
- Shellscript

3 Requirements und Use Cases (Sequenzdiagramme)

3.1 Allgemeine Anforderungen

Werkstücke werden in gewissen Zeitabständen aufs Band gelegt und Sensoren sortieren bestimmte Werkstücke aus.

Diese Werkstücke können auf das Band gelegt werden:

- mit richtiger Höhe
    - Mit Metalleinsatz
        - Öffnung nach oben
        - Öffnung nach Unten
    - Ohne Metalleinsatz
        - Öffnung nach oben
        - Öffnung nach Unten
- mit falscher Höhe

Folgende sollen aussortiert werden:

Höhe	Metall	Öffnung	Aussortieren
richtige Höhe	Mit Metalleinsatz	Öffnung nach oben	Ja (Band 2)
		Öffnung nach Unten	Ja (Band 2, wenden Band 1)
	Ohne Metalleinsatz	Öffnung nach oben	Nein
		Öffnung nach Unten	Nein (wenden Band 1)

mit falsche Höhe			Ja (Band 1)

3.2 Anforderungen

3.2.1 Durchlauf akzeptierter Werkstücke

Akteur: Arbeiter am Förderband
Ziel: Akzeptiertes Werkstück erreicht Ende des zweiten Förderbandes
Auslöser: Arbeiter legt ein Werkstück an den Anfang des ersten Förderbandes

Vorbedingung:
- Förderband 1 in Betrieb und bereit (Ampel: Grün)
- Anfang des ersten Förderbandes ist frei (Schranke 1 nicht unterbrochen)

Erfolgsszenario 1:
1. Ermittlung der Höhe des Werkstückes mit Höhenmessung
2. Werkstück hat akzeptierte Höhe und Bohrung zeigt nach **oben**, Werkstück wird angenommen
3. Öffnen der Weiche, Werkstück wird durchgelassen
4. Weiche wird geschlossen
5. Werkstück erreicht Ende des ersten Förderbandes
6. Transport des Werkstück auf Förderband 2, da dieses frei ist
7. Bohrung des Werkstücks zeigt nach oben
8. Werkstück enthält kein Metallkern
9. Weiche wird geöffnet, Werkstück wird durchgelassen
10. Weiche wird wieder geschlossen
11. Werkstück erreicht das Ende von Band 2, Band zwei bleibt stehen
12. Arbeiter nimmt das Werkstück vom Band 2

Erfolgsszenario 2:
1. Ermittlung der Höhe des Werkstückes mit Höhenmessung
2. Werkstück hat akzeptierte Höhe und Bohrung zeigt nach **unten**, Werkstück wird angenommen
3. Öffnen der Weiche, Werkstück wird durchgelassen
4. Weiche wird geschlossen
5. Werkstück erreicht Ende des ersten Förderbandes (Zustandsanzeige: Gelb blinkend)
6. Arbeiter nimmt Werkstück aus der Lichtschranke, wendet es und legt es zurück
7. Transport des Werkstück auf Förderband 2, da dieses frei ist
8. Bohrung des Werkstücks zeigt nach oben
9. Werkstück enthält kein Metallkern
10. Weiche wird geöffnet, Werkstück wird durchgelassen
11. Weiche wird wieder geschlossen
12. Werkstück erreicht das Ende von Band 2, Band zwei bleibt stehen
13. Arbeiter nimmt das Werkstück vom Band 2

Nachbedingung: Werkstück wird nach Erreichen der Lichtschranke am Ende von Band 2 entnommen

Fehlerfälle: siehe Fehlerszenarien:
- Werkstück wurde vom Band genommen

- Werkstück wurde mitten auf dem Band hinzugefügt

3.2.2 Aussortieren von zu flachen Werkstücken

Akteur: -
Ziel: Werkstücke, deren Höhe kleiner XXmm beträgt, werden von Band 1 aussortiert
Auslöser: Sensor meldet die Höhe des Werkstücks

Vorbedingung:
- Bandanlage in Betrieb (Zustandsanzeige: Grün)
- Werkstücke befinden sich auf dem Förderband

Erfolgsszenario:
1. Höhe des Werkstückes wird mit Hilfe der Höhenmessung erkannt
2. Werkstück zu flach
3. Weiche bleibt zu
4. Werkstück wird aussortiert

Nachbedingung: Aussortiertes Werkstück befindet sich auf der Rutsche

Fehlerfälle: siehe Fehlerszenarien:
- Rutsche ist voll

3.2.3 Aussortieren von Werkstücken mit Bohrung nach oben mit Metalleinsatz

Akteur: -
Ziel: Werkstücke mit Metalleinsatz, deren Bohrung nach oben liegt, werden von Band 2 aussortiert
Auslöser: Sensor meldet Metall im Werkstück

Vorbedingung:
- Bandanlage in Betrieb (Zustandsanzeige: Grün)
- Ein Werkstück befindet sich auf dem Förderband

Erfolgsszenario:
1. Metallsensor auf Band 2 erkennt Metallkern im Werkstück
2. Weiche bleibt geschlossen
3. Werkstück wird aussortiert

Nachbedingung: Aussortiertes Werkstück befindet sich auf der Rutsche

Fehlerfälle: siehe Fehlerszenarien:
- Rutsche ist voll

3.3 Fehlerszenarien

3.3.1 Fehlermeldung „Rutsche voll“

Akteur: Arbeiter am Förderband
Ziel: Behebung des Fehlers: Rutsche entleeren

Auslöser: Sensor meldet Rutsche voll

Vorbedingung:
- Bandanlage in Betrieb (Zustandsanzeige: Grün)
- Ein Werkstück wurde auf die Rutsche geschoben

Ablauf der Fehlerbehebung:
1. Bandstopp, Zustandsanzeige blinkt rot (Fehlerzustand: „anstehend unquittiert“, schnelles Blinken 1 Hz)
2. Arbeiter sieht den Fehler
3. Arbeiter drückt die Quittierungstaste
4. Zustandsanzeige: rotes Dauerlicht (Fehlerzustand: „anstehend quittiert“)
5. Arbeiter nimmt Werkstücke von der Rutsche
6. Arbeiter betätigt die Starttaste
7. Rote Leuchte erlischt

Nachbedingung: Bandanlage wieder in Betrieb (Zustandsanzeige: Grün)

3.3.2 Werkstück wurde vom Band genommen

Akteur: Arbeiter am Förderband
Ziel: Das entnommene Werkstück wird an den Anfang von Band eins gelegt
Auslöser: Sensor meldet, dass ein Werkstück fehlt

Vorbedingung:
- Bandanlage in Betrieb (Zustandsanzeige: Grün)
- Ein Werkstück wird vom Band genommen

Ablauf der Fehlerbehebung:
1. Bandstopp, Zustandsanzeige blinkt rot (Fehlerzustand: „anstehend unquittiert“, schnelles Blinken 1Hz)
2. Arbeiter sieht den Fehler
3. Arbeiter drückt die Quittierungstaste
4. Zustandsanzeige: rotes Dauerlicht (Fehlerzustand: „anstehend quittiert“)
5. Arbeiter legt das vom Band genommene Werkstück an den Anfang von Band 1
6. Arbeiter betätigt die Starttaste
7. Rote Leuchte erlischt

Nachbedingung: Bandanlage wieder in Betrieb (Zustandsanzeige: Grün)

3.3.3 Werkstück wurde mitten auf dem Band hinzugefügt

Akteur: Arbeiter am Förderband
Ziel: Werkstück wird wieder vom Band genommen
Auslöser: Sensor meldet, dass ein Werkstück zu viel auf dem Band ist

Vorbedingung:
- Bandanlage in Betrieb (Zustandsanzeige: Grün)
- Ein Werkstück wird mitten auf dem Band hinzugefügt

Ablauf der Fehlerbehebung:

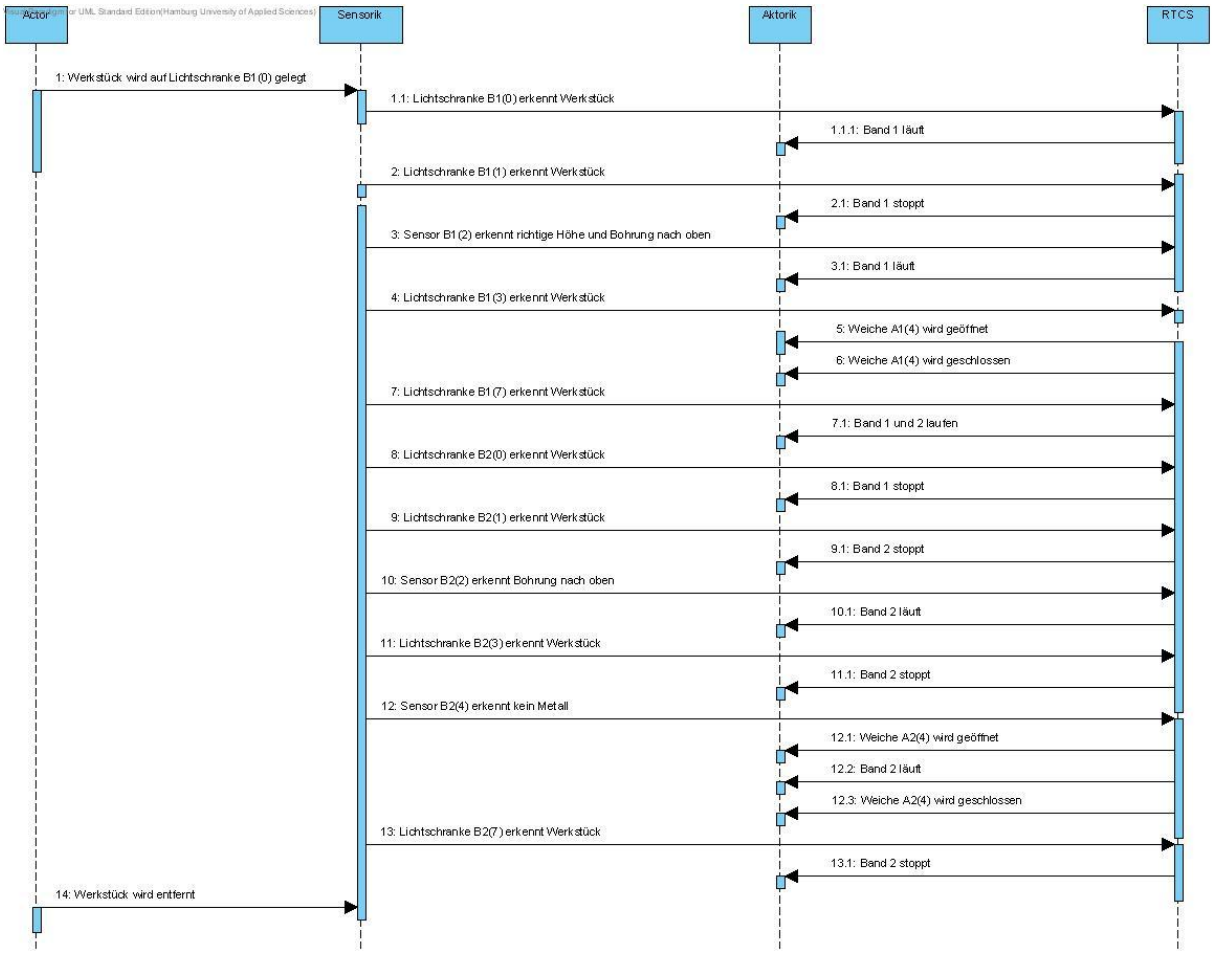
1. Bandstopp, Zustandsanzeige blinkt rot (Fehlerzustand: „anstehend unquittiert“, schnelles Blinken 1Hz)
2. Arbeiter sieht den Fehler
3. Arbeiter drückt die Quittierungstaste
4. Zustandsanzeige: rotes Dauerlicht (Fehlerzustand: „anstehend quittiert“)
5. Arbeiter entfernt das hinzugefügte Werkstück wieder vom Band
6. Arbeiter betätigt die Starttaste
7. Rote Leuchte erlischt

Nachbedingung: Bandanlage wieder in Betrieb (Zustandsanzeige: Grün)

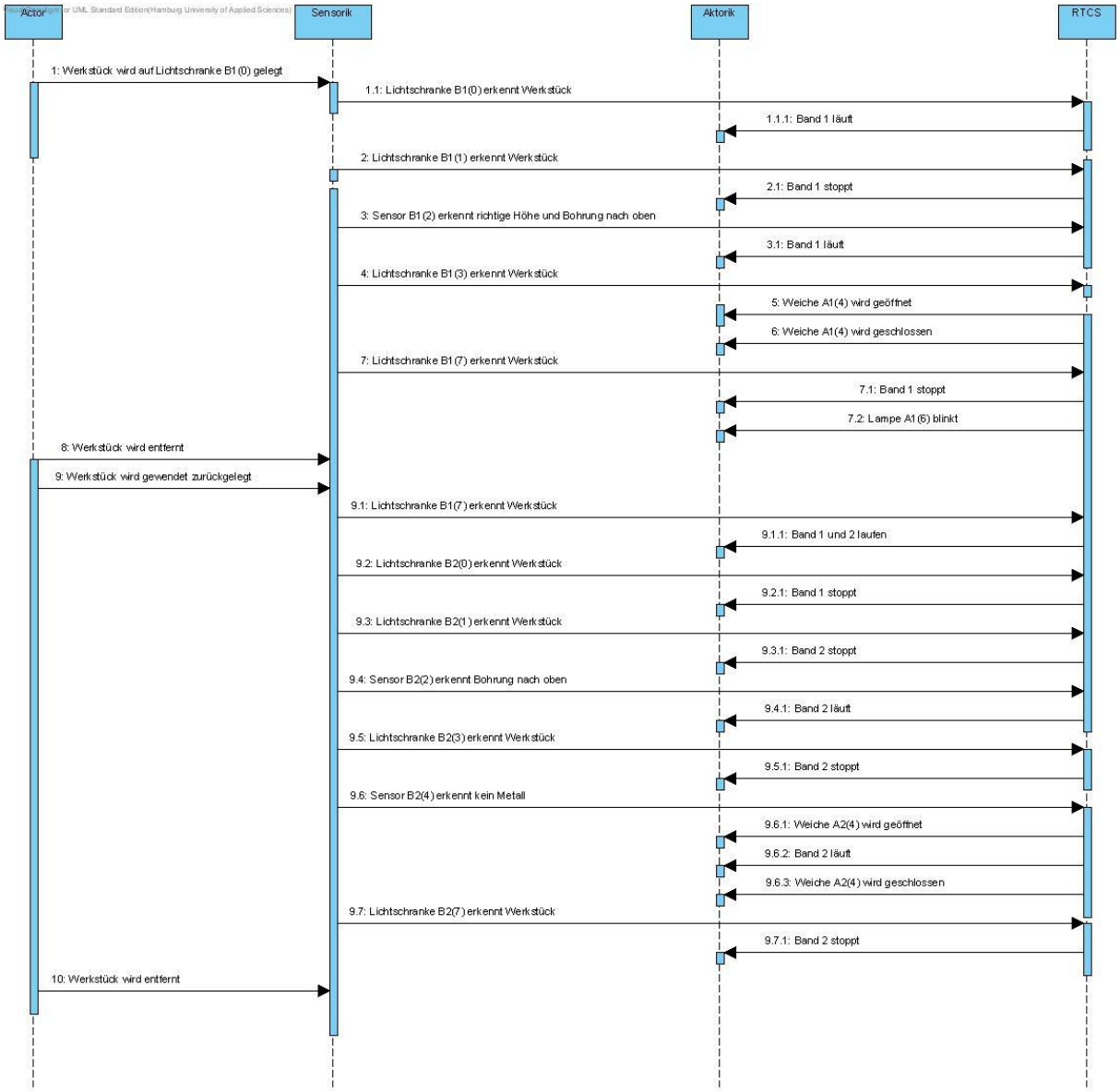
3.4 Diagramme

3.4.1 Durchlauf akzeptierter Werkstücke

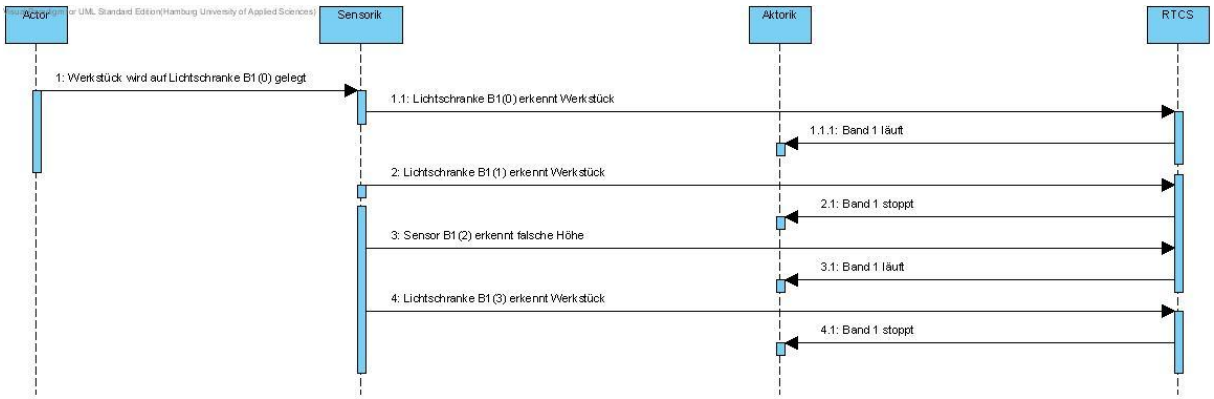
Szenario 1, richtige Höhe, Bohrung nach oben, kein Metall:



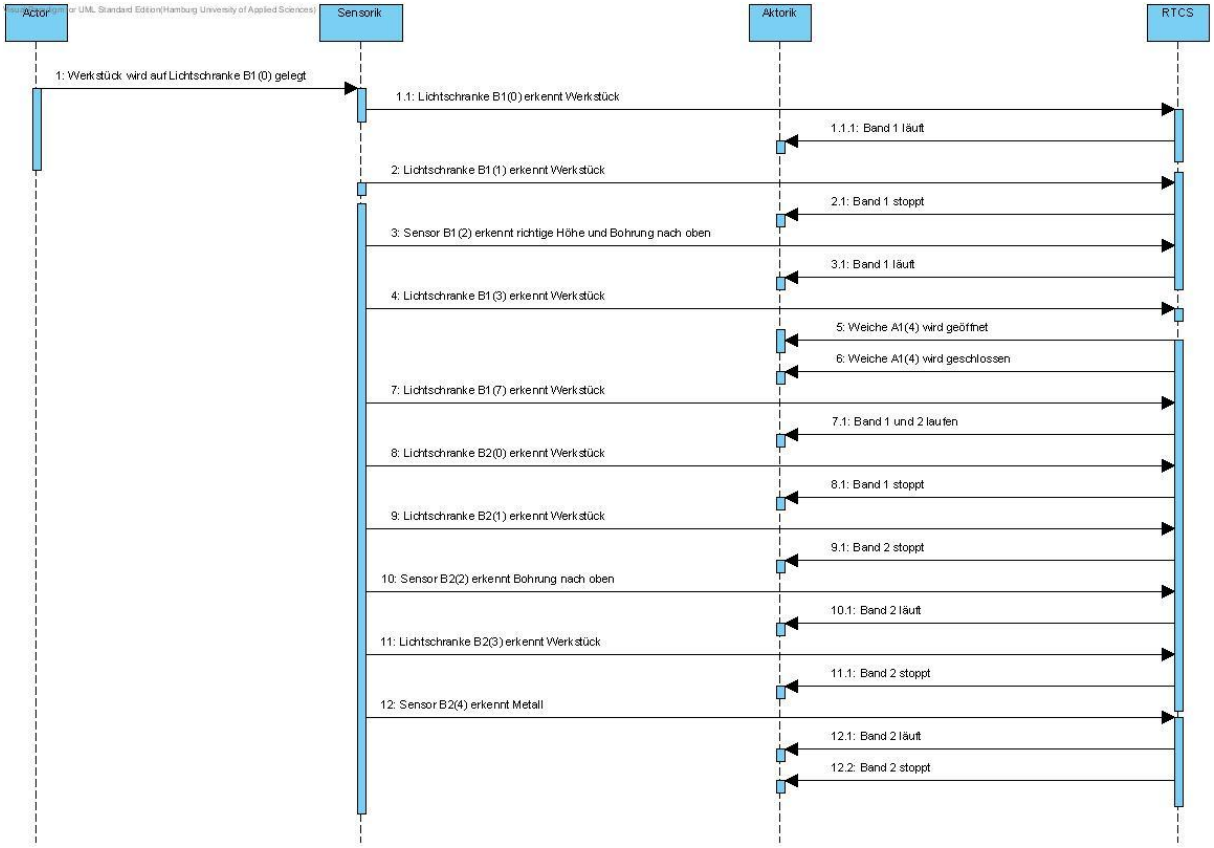
Szenario 2, richtige Höhe, Bohrung nach unten, kein Metall:



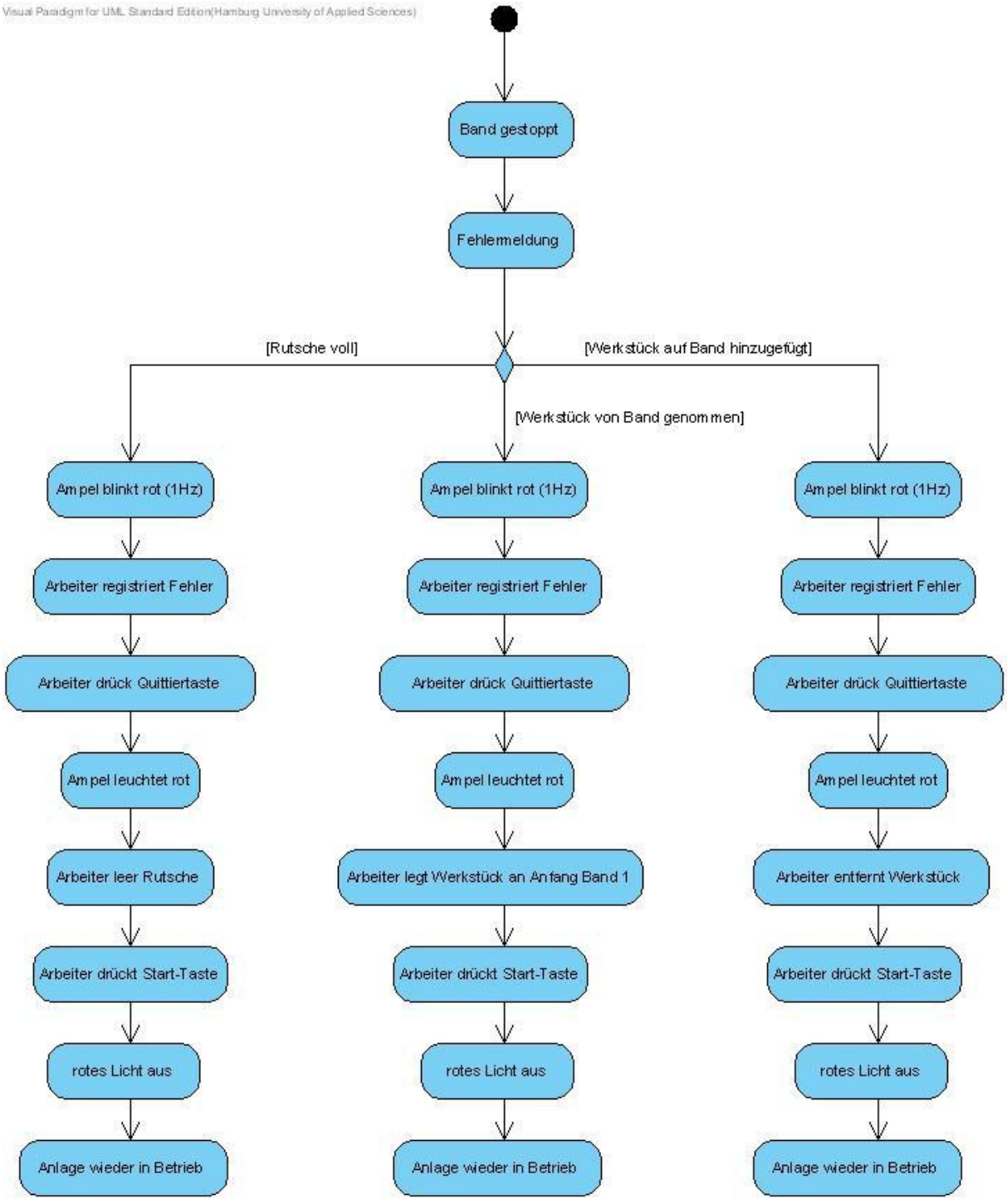
3.4.2 Aussortieren von zu flachen Werkstücken



3.4.3 Aussortieren von Werkstücken mit Bohrung nach oben mit Metalleinsatz



3.4.4 Fehlerbehandlung



3.5 Timer-Konzept

Timer werden benötigt, um gewisse Fehlerszenarien darstellen zu können, um einen geregelten Bandablauf sicherstellen zu können. Dafür werden an verschiedenen Stellen des Bandes Timer eingesetzt.

An folgenden Positionen sollen Timer zum Einsatz kommen:

1. Rutsche
    - Wurde die Lichtschranke unterbrochen, wurde aber nach x Sekunden nicht wieder geschlossen, ist die Rutsche voll

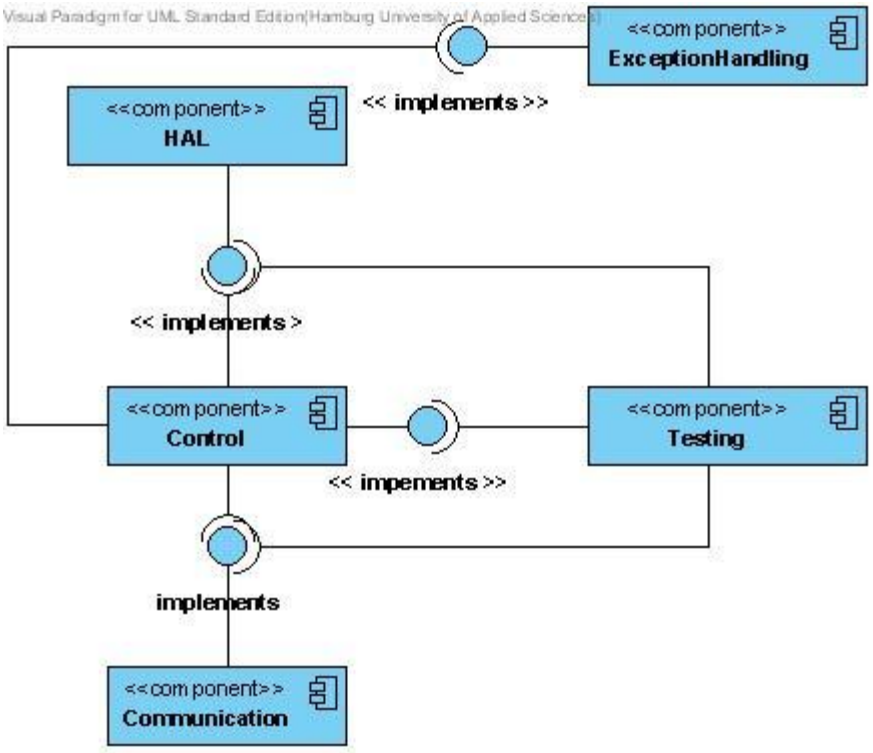
2. Weiche
    - Wurde die Weiche für einen Puck geöffnet, muss sie nach x Sekunden wieder geschlossen werden
3. Band 1 am Ende
    - Bei der Übergabe eines Pucks von Band 1 nach Band 2 läuft das Band noch x Sekunden, wenn der Puck nicht auf Band2 ankommt (LS(2)0 open), ist der Puck verschwunden.
4. Band 1 am Ende
    - Wartet am Ende von Band 1 ein Puck auf die Übergabe, und kein ACK wird nach x Sekunden empfangen, ist Band 2 vermutlich gestört (z.B. EStop, nicht aktiv)
5. Band 1 und Band 2 für jedes der drei Segmente
    - Min/Max Timer für die Fahrtzeit durch die drei Segmente pro Werkstück (z.B. vom Bandanfang (LS1 wieder geschlossen) bis zur Höhenkontrolle (LS2 unterbrochen)

Das Auslösen eines Timers hat das Absenden eines Pulses zur Folge. Je nachdem, welcher Timer ausgelöst wurde, sind für die oben genannten Timer verschiedene Ziele der Pulse zur Abarbeitung zu definieren:

1. Der Dispatcher erhält den Pulse, dass die Rutsche voll ist
2. Der Dispatcher erhält den Pulse, dass die Weiche geschlossen werden muss
3. Der Dispatcher von Band 1 erhält den Pulse, dass das Band noch weiterlaufen muss
4. Die Error FSM erhält den Pulse, dass am Ende von Band 1 kein ACK von Band 2 kam
5. Die Error FSM erhält den Pulse, dass ein Timer unter- oder überschritten wurde

4 Design

4.1 System (Komponentendiagramm)



4.2 Datenmodell (Klassendiagramm)

Das zugehörige Klassendiagramm befindet sich in der Datei „Klassendiagramm.jpg“

5 Implementierung

5.1 Algorithmen

5.2 Patterns

- State Pattern
- Facade Pattern
- Singleton Pattern
- Observer Pattern

6 Testen

6.1 Regressions- und Komponententests

Um Fehler in Modifikationen bereits getesteter Software innerhalb dieses Projektes zu finden, ist es unerlässlich, über einen Grundstock an Regressionstests zu verfügen. Es werden folgende Testfälle ausgeführt:

6.1.1 Aktorik HAL Testablauf

Für einen Test der gesamten Aktorik werden folgende Zeilen Code verwendet:

```
39      HALTest halTest;
40      halTest.testHal();
```

Folgende Ausgabe auf der Konsole wird erwartet, analog zur Ausgabe soll das Verhalten am Festo-System zu beobachten sein:

Debug Hal: New HAL instance created	
Debug Hal: red light on	rotes Licht an Ampel geht an
Debug Hal: red light off	rotes Licht an Ampel geht aus
Debug Hal: yellow light on	gelbes Licht an Ampel geht an
Debug Hal: yellow light off	gelbes Licht an Ampel geht aus
Debug Hal: green light on	grünes Licht an Ampel geht an
Debug Hal: green light off	grünes Licht an Ampel geht aus
Debug Hal: green light on	grünes Licht an Ampel geht an
Debug Hal: yellow light on	gelbes Licht an Ampel geht an
Debug Hal: red light on	rotes Licht an Ampel geht an
Debug Hal: all lights off	alle Lichter an Ampel gehen aus
Debug Hal: engine right with normal speed	Laufband fährt rechts
Debug Hal: engine right with slow speed	Laufband fährt langsam rechts
Debug Hal: engine stopped	Laufband hält an
Debug Hal: engine revert stop	Laufband fährt weiter rechts
Debug Hal: engine stopped	Laufband hält an
Debug Hal: gate open	Weiche öffnet sich
Debug Hal: gate closed	Weiche schließt sich
Debug Hal: start led on	LED Start geht an
Debug Hal: reset led on	LED Reset geht an
Debug Hal: Q1 led on	LED Q1 geht an
Debug Hal: Q2 led on	LED Q2 geht an
Debug Hal: start led off	LED Start geht aus
Debug Hal: reset led off	LED Reset geht aus

Debug Hal: Q1 led off Debug Hal: Q2 led off	LED Q1 geht aus LED Q2 geht aus

6.1.2 RS232 Testablauf

Für einen Test der seriellen Schnittstelle werden folgende Zeilen Code verwendet, weiterhin müssen die beiden COM-Ports des GEME-PC mit einem Null-Modem Kabel verbunden werden:

```
42      RS232Test rs232Test;
43      rs232Test.testRS232();
```

Folgende Ausgabe ist bei korrekter Funktion auf der Konsole zu erwarten:

Debug RS232_1: opening devfile1 SUCCESSED Debug RS232_1: New RS232_1 instance created Debug RS232_2: opening devfile2 SUCCESSED Debug RS232_2: New RS232_2 instance created Debug RS232_1: Unknown msg recved: b Debug RS232_1: Timeout or EAGAIN Debug RS232_1: Timeout recved Debug RS232_1: Timeout or EAGAIN Debug RS232_1: Timeout recved Testmessage recved on devfile1: a Debug RS232_1: Timeout or EAGAIN Debug RS232_1: Timeout recved Debug RS232_1: Timeout or EAGAIN Debug RS232_1: Timeout recved Debug RS232_2: Unknown msg recved: b Debug RS232_2: Timeout or EAGAIN Debug RS232_2: Timeout recved Debug RS232_2: Timeout or EAGAIN Debug RS232_2: Timeout recved Testmessage recved on devfile2: a Debug RS232_2: Timeout or EAGAIN Debug RS232_2: Timeout recved Debug RS232_2: Timeout or EAGAIN Debug RS232_2: Timeout recved	COM1 initialisiert und geöffnet COM2 initialisiert und geöffnet Lese auf COM1, schreibe auf COM2 Unbekannte Nachricht ‚b‘ empfangen Zyklisch generierter Timeout Gültige Nachricht ‚a‘ empfangen Lese auf COM2, schreibe auf COM1 Unbekannte Nachricht ‚b‘ empfangen Zyklisch generierter Timeout Gültige Nachricht ‚a‘ empfangen

6.1.3 Ampelkontrollthread Testablauf

Für einen Test der verschiedenen Funktionen der Ampel werden folgende Zeilen Code benötigt:

```
46      LightControllerTest lctest;
47      lctest.testLightController();
```

Bei korrekter Funktion ist folgender Ablauf auf der Konsole zu beobachten, analog dazu sind die beschriebenen einzelnen Zustände auch an der Ampel zu sehen:

Debug Hal: New HAL instance created Debug Hal: all lights off Debug Hal: green light on Debug Hal: all lights off Debug Hal: red light on Debug Hal: red light off	 Falls noch Lampen an sind, aus machen Normaler Betrieb Alle Lichter löschen Anstehend unquittiert

Debug Hal: red light on Debug Hal: red light off Debug Hal: red light on Debug Hal: red light off Debug Hal: red light on Debug Hal: red light off Debug Hal: all lights off Debug Hal: red light on Debug Hal: all lights off Debug Hal: red light on Debug Hal: red light off Debug Hal: red light on Debug Hal: red light off Debug Hal: all lights off Debug Hal: yellow light on Debug Hal: yellow light off Debug Hal: yellow light on Debug Hal: yellow light off Debug Hal: all lights off	 Alle Lichter löschen Anstehend quittiert Alle Lichter löschen Gegangen unquittiert Alle Lichter löschen Manuelle Drehung Alle Lichter löschen

6.1.4 Sensorik HAL Testablauf

Für einen Test der Sensorik mit Interrupts müssen folgende Zeilen Code verwendet werden:

```
54      ISRTest isrtest;
55      isrtest.start(0);
```

Beim folgenden Testablauf wurden der Reihe nach folgende Werkstücke auf das Band gelegt:

1. Zu kleines Werkstück (landet auf der Rutsche)
2. Akzeptiertes Werkstück, Bohrung nach oben (Band stoppt, wenn WS das Ende erreicht)
3. Akzeptiertes Werkstück, Bohrung nach unten (Band stoppt, wenn WS das Ende erreicht, Ampel blinkt gelb)
4. Werkstück mit Metall (WS landet auf der Rutsche)

Anschließend werden die Knöpfe bedient:

1. Reset
2. Stop
3. E-Stop drücken
4. Start
5. E-Stop lösen
6. Start

Diese Reihenfolge führt zu folgender Ausgabe auf der Konsole:

Debug SensorHAL: New SensorHAL instance created Debug Hal: New HAL instance created Debug LightController: New LC instance created Werkstueck im Einlauf Debug Hal: engine right with normal speed Debug Hal: engine revert stop Kein Werkstueck im Einlauf	 Werkstück auf Band gelegt Band fährt los

Werkstueck in Hoehenmessung	Werkstück ist in Höhenmessung
Werkstueck Hoehe: 2712	Messung
Debug Hal: engine stopped	Band stoppt
Debug Hal: engine revert stop	Nach Messung/Prüfung weiterfahren
Kein Werkstueck in Hoehenmessung	
Werkstueck in Weiche	Werkstück an der Weiche
Kein Werkstueck in Weiche	
Rutsche voll	Werkstück wird aussortiert
Rutsche nicht voll	
Debug Hal: engine stopped	
Werkstueck im Einlauf	Neues Werkstück auf Band gelegt
Debug Hal: engine right with normal speed	
Debug Hal: engine revert stop	
Kein Werkstueck im Einlauf	
Werkstueck im Toleranzbereich: 4040	
Werkstueck zu klein/gross: 2525	
Werkstueck in Hoehenmessung	
Werkstueck Hoehe: 3526	
Debug Hal: engine stopped	
Debug Hal: engine revert stop	
Werkstueck im Toleranzbereich: 4025	
Werkstueck zu klein/gross: 2548	
Kein Werkstueck in Hoehenmessung	
Werkstueck in Weiche	Werkstück erreicht Weiche
Debug Hal: gate open	Werkstück ok, daher Weiche öffnen
Weiche offen	
Kein Werkstueck in Weiche	
Werkstueck im Auslauf	Werkstück erreicht Ende des Bands
Debug Hal: gate closed	Weiche wieder schließen
Debug Hal: engine stopped	
Weiche geshlossen	
kein Werkstueck im Auslauf	
Debug Hal: all lights off	
Werkstueck im Einlauf	Neues Werkstück auf Band gelegt
Debug Hal: engine right with normal speed	
Debug Hal: engine revert stop	
Kein Werkstueck im Einlauf	
Werkstueck im Toleranzbereich: 4023	
Werkstueck in Hoehenmessung	
Werkstueck Hoehe: 2471	
Debug Hal: engine stopped	
Debug Hal: engine revert stop	
Werkstueck zu klein/gross: 2551	
Kein Werkstueck in Hoehenmessung	
Werkstueck in Weiche	
Debug Hal: gate open	
Weiche offen	
Kein Werkstueck in Weiche	
Werkstueck im Auslauf	Werkstück erreicht Ende des Bands
Debug Hal: gate closed	
Debug Hal: engine stopped	
manualTurnover: 1	Werkstück muss gewendet werden
Debug Hal: all lights off	Gelbes blinken
Debug Hal: yellow light on	
Weiche geshlossen	
Debug Hal: yellow light off	
Debug Hal: yellow light on	
kein Werkstueck im Auslauf	
Debug Hal: all lights off	
Werkstueck im Einlauf	Neues Werkstück auf Band gelegt

Debug Hal: engine right with normal speed	
Debug Hal: engine revert stop	
Kein Werkstueck im Einlauf	
Werkstueck im Toleranzbereich: 2663	
Werkstueck zu klein/gross: 2510	
Werkstueck in Hoehenmessung	
Werkstueck Hoehe: 3465	
Debug Hal: engine stopped	
Debug Hal: engine revert stop	
Werkstueck im Toleranzbereich: 3800	
Werkstueck zu klein/gross: 2553	
Kein Werkstueck in Hoehenmessung	
Werkstueck Metall	Werkstück enthält Metall
Werkstueck in Weiche	
Werkstueck kein Metall	
Kein Werkstueck in Weiche	
Rutsche voll	Werkstück aussortieren
Rutsche nicht voll	
Debug Hal: engine stopped	
Resettaste gedrueckt	Reset Taster
Resettaste losgelassen	
Stoptaste gedrueckt	Stop Taster
Stoptaste losgelassen	
E-Stoptaste gedrueckt	E-Stop Taster
System angehalten mittels EStop	
Debug Hal: red light on	Rotes Licht geht an
Starttste gedrueck	Start Taster gedrückt, rotes Licht
Starttste losgelassen	bleibt an
E-Stoptaste geloest	E-Stop Taster zurückgesetzt, Schalter
E-Stoptaste gedrueckt	kann prellen, an dieser Stelle mit
System angehalten mittels EStop	bool und Start Taster verbunden
Debug Hal: red light on	
E-Stoptaste geloest	
E-Stoptaste gedrueckt	
System angehalten mittels EStop	
Debug Hal: red light on	
E-Stoptaste geloest	
E-Stoptaste gedrueckt	
System angehalten mittels EStop	
Debug Hal: red light on	
E-Stoptaste geloest	
E-Stoptaste gedrueckt	
System angehalten mittels EStop	
Debug Hal: red light on	
E-Stoptaste geloest	
Starttste gedrueck	Nach rücksetzen von E-Stop Start
System rennt weiter, EStop geloest udn	Taster gedrückt, System läuft wieder,
start	rotes Licht geht aus, grünes Licht
Debug Hal: red light off	geht an
Debug Hal: green light on	
Starttste losgelassen	

6.1.5 State Pattern Testablauf

Für einen Test des State Patterns muss in der Datei „bandselection.h“ folgendes define gesetzt werden:

```
20  //valid defines are BAND_1 BAND_2 BAND_TEST
21  #define BAND_TEST
```

Weiterhin wird folgendes Setup in der „main.cpp“ benötigt:

```
64      ErrorFSM* errfsm = ErrorFSM::getInstance();
65      errfsm->start(0);
66
67      Dispatcher* disp = Dispatcher::getInstance();
68      disp->start(0);
69
70      ISRHandler* isrhandler = ISRHandler::getInstance();
71      isrhandler->start(0);
72
73      PuckHandler::getInstance()->initializePucks(disp);
```

Für den Test müssen nach und nach alle Lichtschranken unterbrochen und wieder geschlossen werden. Folgende Reihenfolge ist für den Test vorgesehen:

1. Lichtschranke am Bandanfang unterbrechen/schließen
2. Lichtschranke an der Höhenmessung unterbrechen/schließen
3. Lichtschranke an der Weiche unterbrechen/schließen
4. Lichtschranke an der Rutsche unterbrechen/schließen
5. Lichtschranke am Bandende unterbrechen/schließen

Dieser Ablauf bewirkt folgende Ausgabe auf der Konsole:

Entering State 1: LS1	LS am Bandanfang unterbrochen (Konstruktor)
Leaving State 1: LS1	LS am Bandanfang geschlossen (new State 2)
Entering State 2: LS2	LS an Höhenkontrolle unterbrochen (Konstruktor)
Leaving State 2: LS2	LS an Höhenkontrolle geschlossen (new State 3)
Entering State 3: LS3	LS an der Weiche unterbrochen (Konstruktor)
Leaving State 3: LS3	LS an Weiche geschlossen (new State 4)
Entering State 4: LS4	LS an Rutsche unterbrochen (Konstruktor)
Leaving State 4: LS4	LS an Rutsche geschlossen (new State 5)
Entering State 5: LS5	LS am Bandende unterbrochen (Konstruktor)
Leaving State 5: LS5	LS am Bandende geschlossen (Test beendet)
time to kill test program ...	

6.2 Abnahmetest

6.3 Testplan

6.4 Testprotokolle und Auswertungen

7 Bedienung der Anlage

7.1 Funktionsweise des Laufbands

Nach Start der Anlage ist diese noch nicht betriebsbereit. Es muss zunächst der „Start“-Knopf betätigt werden. Anschließend zeigt die Ampel die Betriebsbereitschaft durch grünes Dauerlicht an.

Es können nun Pucks in die Lichtschranke am Anfang des Förderbands gelegt werden, solange der Anfang frei ist. Für einen reibungslosen Ablauf und für eine korrekte Fehlererkennung dürfen Pucks nur mit einem Mindestabstand von einer Sekunde auf das Band gelegt werden.

Akzeptierte Werkstücke müssen am Ende von Band 2 entnommen werden. Dazu stoppt das Band, sobald ein akzeptiertes Werkstück das Ende des Bandes erreicht hat.

7.2 Funktionsweise der Buttons

- Start
  - initial, zum versetzen der Maschine in den betriebsbereiten Zustand
  - im normalen Betrieb, um den pausierten Ablauf fortzusetzen
  - Nach der Fehlerquittierung zum Wechsel in den betriebsbereiten Zustand
- Stop
  - Zum Pausieren des normalen Betriebs
- Reset
  - Quittieren von aufgetretenen Fehlern, außer E-Stop
  - Zum Neustart des normalen Betriebs nach Quittierung des E-Stops
- E-Stop
  - Stoppt den Ablauf beider Bänder (drücken)
  - Quittiert den E-Stop Fehlerzustand (herausziehen)

7.3 Verhalten im Fehlerfall

Für das Verhalten im Fehlerfall siehe Kapitel 3.

8 Projektplan

8.1 Verantwortlichkeiten

Innerhalb der Gesprächssitzung zur Organisation des Teams wurde sich darauf verständigt, eine demokratische Grundordnung zu verfolgen. Entscheidungen werden gemeinsam im Team getroffen.

8.2 Projektstrukturplan

Der fertige Projektstrukturplan liegt in der Datei „se2psp.pdf“ vor.

9 Lessons Learned

9.1 Implementierung

9.1.1 HAL Simulation

```
out8(PORT_A, (val | ENGINE_RIGHT));
out8(PORT_A, val & ~(ENGINE_SLOW));
```

In der Simulation funktionieren zwei out8 Befehle direkt hintereinander, am echten Laufband bekommen wir das Problem, dass bit1 und bit3 gesetzt sind, obwohl bit3 gelöscht sein sollte. Abhilfe schafft die Kombination der beiden Befehle zu einem einzigen:

```
out8(PORT_A, (val | ENGINE_RIGHT) & ~(ENGINE_SLOW));
```

9.1.2 RS232: Initialisierung

Die Verwendung eines termios Struct, um die serielle Schnittstelle zu konfigurieren ist sinnvoll, damit eine garantierte Konfiguration besteht, da nicht zwingend bekannt ist, wie die Schnittstelle (z.B. nach einem laufenden Betrieb) konfiguriert ist. Weiterhin könnten sich auch noch Daten im Lese- oder Schreibpuffer befinden. Diese werden ebenfalls vorher geflushed.

9.1.3 RS232: read

Die Funktion „read“ blockiert, somit hat man keine Möglichkeit den Thread von außen geregelt zu beenden, ausser diesen zu killen. Alternativ wird das „readcond“ benutzt, welches einen zyklischen timeout generieren kann, um so die Schleifenbedingung des Threads regelmäßig zu prüfen. Somit kann sichergestellt werden, dass threads regulär beendet werden können und somit das abschließende join() auf den Thread nicht blockiert.

9.2 Was lief gut

- Die Implementierung der Aufgaben mit der „Pair-Programming“-Methode funktionierte sehr gut
- Die Meilensteine konnten alle im vorgegebenen Zeitrahmen eingehalten werden

9.3 Was lief weniger gut

- Die Praktikumstermine waren der zugehörigen Vorlesung stets eine Woche voraus, so dass ein zeitlicher Mehraufwand für die Planung und Realisierung auftrat
- Es traten häufiger Probleme bei der Implementierung mangels C++ Kentnisse auf
- Die Dokumentation erfolgte teilweise zu spät und kostete daher im weiteren Verlauf Zeit
- Wegen unzureichender Planung/Analyse zu Beginn des Projekts konnte das Designprinzip „Seperation of Concerns“ im späteren Verlauf teilweise nur noch unzureichend erfüllt werden, was zu einer Verkomplizierung des Codes geführt hat

9.4 Was wurde gelernt

- Anforderungen so genau wie möglich zu Beginn des Projekts festhalten und vereinbaren
- Datenmodell vorab besser designen, um z.b. zyklische Abhängigkeiten zu vermeiden
- Dokumentation des Codes und Aktualisierung der Diagramme sofort durchführen