

Requirements / Design and Test Documentation (RDT)

Version 0.7

ESEP – Praktikum – Sommersemester 2023

Lorenz, Maik, 2542513, maik.lorenz@haw-hamburg.de

Schukow, Dominik, 2441109, dominik.schukow@haw-hamburg.de

Malik, Sulaiman, 2441151, sulaiman.malik@haw-hamburg.de

Änderungshistorie:

Version	Erstellt	Autor	Kommentar
0.1	2018-03-12	LMN	Initiale Version des Templates.
0.2	2020-03-15	DAI	Überarbeitung wegen Corona.
0.3	2022-02-24	LMN	Anpassungen für Sommersemester. Anforderungen an Requirements reduziert auf Ergänzungen.
0.4	2022-11-22 ff.	CHRS	Neustrukturierung des Templates, Schriftgrößen vereinheitlicht, Erweiterungen: Hinweise am Anfang des Dokuments, Unterkapitel Hardware und technische Gegebenheiten, Unterkapitel Analyse des Kundenwunsches , Unterkapitel Nachrichten und Signale, allg. Abnahmetest Text + Tabelle, Unterkapitel Abbildungsverzeichnis
0.5	2023-03-28	DOM, SUL, MAI	Absprachen, Requirement Analysis, Project management, System context diagram, Abkürzungen, Glossar
0.6	2023-04-13	DOM, SUL, MAI	Requirement Analysis (Absprachen), Requirement (Software and Hardware perspective) Uses Cases, Warning, Hardware Analysis, Software Architecture
0.7	2023-04-27	DOM, SUL, MAI	Software-Architektur als Komponentendiagramm, Sequenzdiagramme und erste Verhaltensmodellierung (FSMs) ergänzt

Inhaltsverzeichnis:

1	Teamorganisation.....	5
1.1	Verantwortlichkeiten	5
1.2	Absprachen.....	5
1.3	Repository-Konzept	5
2	Projektmanagement	6
2.1	Prozess.....	6
2.2	Projektplan	6
2.3	Risiken.....	7
2.4	Qualitätssicherung.....	7
3	Problemanalyse	8
3.1	Analyse des Kundenwunsches.....	8
3.1.1	Stakeholder	8
3.1.2	Systemkontext des Systems	8
3.1.3	Anforderungen	9
3.1.4	Fehler und Warning.....	11
3.1.5	Use Cases / User Stories.....	12
3.1.6	Absprachen	15
3.2	Hardware: Analyse der technischen Gegebenheiten.....	17
3.2.1	Technischer Aufbau und Hardwarekomponenten	17
Die Festo-Transfersysteme verfügen über unterschiedliche Sensoren und Aktoren, mit dessen Hilfe unsere Sortieranlage realisiert wird.		17
3.2.2	Werkstücke.....	18
3.2.3	Anforderungen aus dem Verhalten und technischen Besonderheiten	18
3.3	Softwareebene	20
3.3.1	Systemkontext der Software	20
3.3.2	Resultierende Anforderungen an die Software	20
3.3.3	Nachrichten und Signale	22
3.3.4	Warnungen und Fehler	25
4	Software-Design.....	28
4.1	Software Architektur	28
4.2	Software Struktur	30
4.3	Verhaltensmodellierung.....	33
5	Implementierung	35
6	Qualitätssicherung.....	36
6.1	Teststrategie	36
6.2	Testszenarien/Abnahmetest.....	37
6.3	Testprotokolle und Auswertungen	38
7	Technische Schulden.....	39
8	Lessons Learned.....	39

9	Anhang.....	39
9.1	Glossar	39
9.2	Abkürzungen.....	39
9.3	Abbildungsverzeichnis	40

1 Teamorganisation

1.1 Verantwortlichkeiten

Verantwortlichkeit	Person/en
Projektmanager	Lorenz, Maik
Implementierung, Test	Schukow, Dominik
Requirements Analyst	Malik, Sulaiman

1.2 Absprachen

Dokumentation

- Source-Code auf [GitHub](#)
- Arbeitsversion RDT im SharePoint
- Abgabefertiges RDT im MS-Teams Raum für Gruppe 2.1

Kommunikation

- Feste Zeiten für Meetings im Labor: Donnerstag ab 12:00 Uhr bzw. nach dem Praktikum
- Freitag 13:00 Sprint-Planung / Standup
- Montag 19:00 online
- Meetings je nach aktuellen Themen in MS-Teams – Absprache über WhatsApp
- Besprechungsprotokolle werden in [Confluence](#) dokumentiert
- Anfragen und Absprachen über WhatsApp (max. Reaktionszeit 1 Stunde). Wenn jemand verhindert ist oder nicht weiterhelfen kann, wird das kommuniziert und ein Termin für eine Antwort genannt oder die Anfrage delegiert

Aufgabenverteilung

- Über das [Scrum-Board von JIRA](#) werden zu erledigende Aufgaben in Issues definiert. Es ist immer eine "Definition of Done" (vorweggenommener Endzustand) anzugeben, die spezifiziert wann eine Aufgabe als abgeschlossen gilt

1.3 Repository-Konzept

Der Source-Code liegt auf GitHub im Projekt [ESEP-2023SoSe-Team-2-1](#).

Wir arbeiten nach dem [GitFlow Workflow](#). Im main-Branch dürfen nur funktionsfähige Versionen liegen.

Im develop-Branch ist der Arbeitsstand für die nächste Version. Neue Features werden in eigenen feature-Branches implementiert und danach in den develop-Branch gemerged. Auslieferungsbereite Versionen werden mit Versionsnummern getaggt.

2 Projektmanagement

2.1 Prozess

Da wir nach dem Scrum-Modell arbeiten, werden zu bearbeitende Aufgaben immer in Sprints eingeplant. Sprints finden immer jeweils zwischen zwei Praktikumsterminen statt, dauern also in der Regel zwei Wochen.

Ein Review des gerade abgeschlossenen sowie die Planung eines neuen Sprints findet immer am Freitag 13:00 nach einem Praktikumstermin statt. In Wochen ohne Praktikum wird dieser Termin dazu genutzt, um den Stand der zu bearbeitenden Aufgaben des aktuellen Sprints zu besprechen.

Besprechungen werden immer schriftlich in Confluence dokumentiert. Sich daraus ergebende Absprachen werden ebenso dokumentiert und falls notwendig direkt in JIRA-Tasks eingeplant.

Wichtige Absprachen mit den Betreuern werden in diesem Dokument festgehalten.

2.2 Projektplan

Meilensteine

Zeitpunkt	Ziele
Praktikum 1	Organisation innerhalb des Teams definiert. Anforderungsanalyse und Systemkontextdiagramm erstellt. Projektplan und Projektstruktur erstellt. Momentics und Repository ist eingerichtet. Ein Programm kann auf die Anlage geladen werden und diese ansteuern
Praktikum 2	Vollständige Anforderungsanalyse und Abmachungen sind dokumentiert. Die Aktorik der HAL ist implementiert. Beispiel zur Datenübertragung via QNET ist implementiert. Abnahmetests sind formuliert. Erstes Dokument der Software Architektur ist ausgearbeitet.
Praktikum 3	Überarbeitetes Dokument mit dem Entwurf der Software Architektur liegt vor. FSMs sind grob modelliert. Die Sensorik der HAL ist implementiert. Konzept der Übergabe der Daten von HAL zu FSM liegt vor. Präsentation der Architektur als Vortrag.
Praktikum 4	Das Dokument der Software Architektur ist final und kann implementiert werden.
Praktikum 5	FSMs sind ohne Fehlerbehandlung modelliert. Grundfunktionalität ohne Fehlerbehandlung implementiert (Werkstücke können sortiert werden)
Praktikum 6	FSMs sind vollständig modelliert. Die Anlage ist vollständig implementiert. Abgabe des finalen Requirement Design Dokument.
Praktikum 7	Alle nicht realisierten Funktionalitäten sind dokumentiert und begründet. Gesamtanlage ist bereit für die Abnahmetests durch den Kunden.

	Fehlerzustände sind dokumentiert. “Lessons Learned” ausgefüllt. Abgabe von Dokumenten, Planung, Code und Protokollen
--	--

Zur Visualisierung des Projektplans wird die [Jira Roadmap](#) verwendet. Das Projekt ist in verschiedene Abschnitte aufgeteilt, die hier auf der oberen Ebene mittels sogenannter “Epics” dargestellt werden (vgl. Gantt-Chart). Der Name jedes Epics ist als vorweggenommener Endzustand formuliert, damit auf den ersten Blick klar ist, was das Ziel ist. Jedes Epic wird auf User Stories und Aufgaben herunter gebrochen, die erledigt werden müssen für die Erreichung des (Teil-)Ziels. Ziel ist es eine Granularität zu schaffen, damit Aufgaben möglichst unabhängig voneinander bearbeitet und somit gut auf die Teammitglieder verteilt werden können.

2.3 Risiken

Risikobeschreibung	Hypothetisch / bekannt	Wahrscheinlichkeit	Maßnahme
Teammitglied bricht das Praktikum ab	bekannt	gering	Absprache mit Kunde über wegfallende Requirements
Teammitglied ist krank / nicht verfügbar	bekannt	normal	Gute Dokumentation, Verteilung der Aufgaben an andere Teammitglieder
Kein Zugang zum Labor	bekannt	normal	Nutzung der Simulation zum Testen der Software
Verzug durch technische Schwierigkeiten	bekannt	normal	Technische Beratung anfragen bei Profs.

2.4 Qualitätssicherung

Um die Qualität der umgesetzten Features sicherzustellen, werden Unit- und Modultests mit der GoogleTest Suite erstellt. Vor dem Abschluss von Feature-Branches müssen alle Tests bestanden werden.

Mit den Abnahmetests wird die korrekte Funktion des Gesamtsystems aus Kundensicht getestet.

3 Problemanalyse

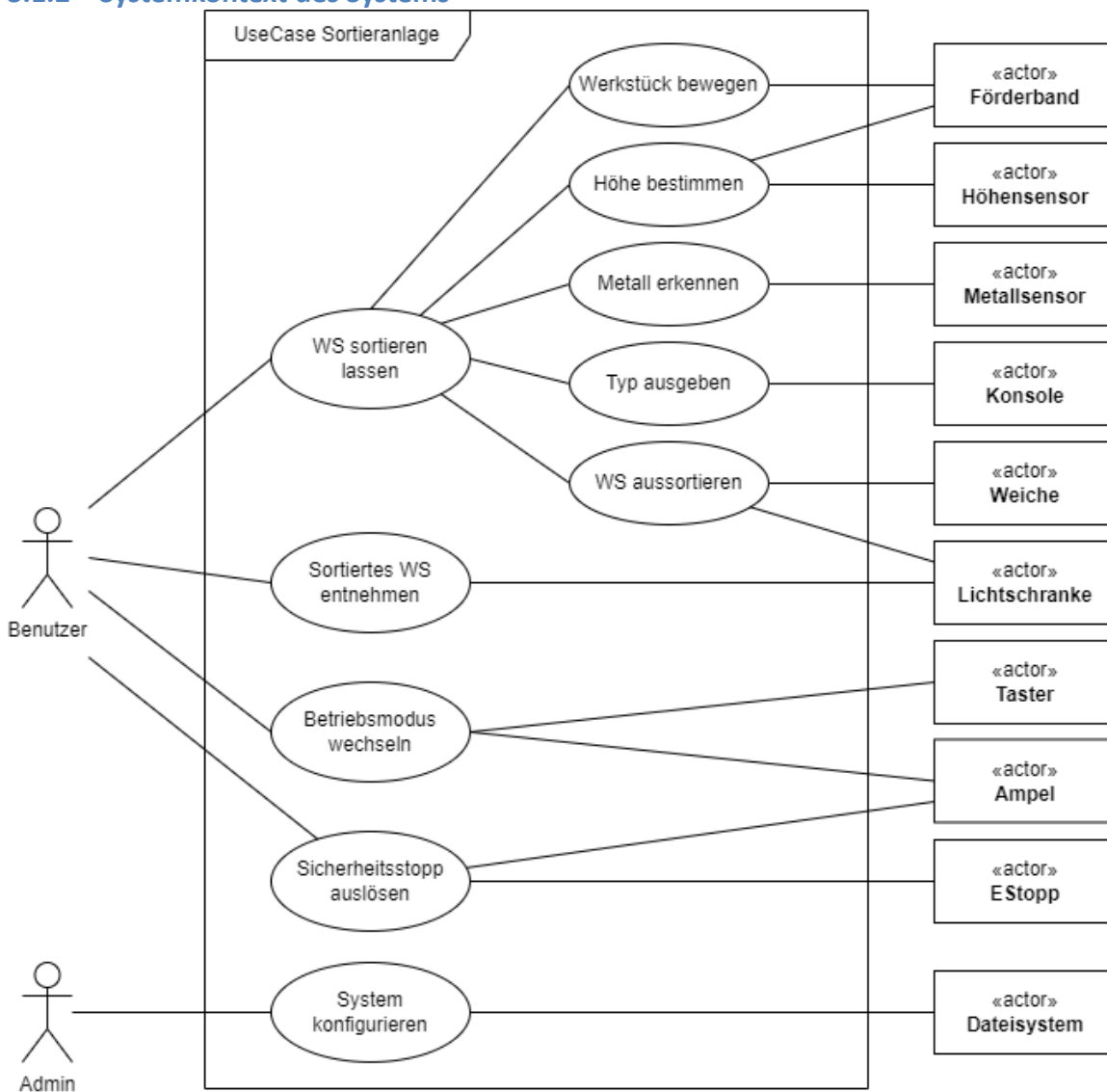
3.1 Analyse des Kundenwunsches

Der Kunde stellt ein fertiges System zur Verfügung, das so programmiert werden soll, dass aufgelegte Werkstücke am Ende eines FBM in vorgegebener Reihenfolge ankommen sollen. Zur Ermittlung der Werkstücktypen können deren Eigenschaften durch an der Anlage montierte Sensoren bestimmt werden. Werkstücke, die nicht in die Reihenfolge passen, sollen auf Rutschen aussortiert werden.

3.1.1 Stakeholder

Stakeholder	Interessen
Kunde	Fehlerfreie und richtig sortierte Bausteine.
Entwickler	Fehlerfreie und termingerechte Implementierung der Software
Anwender	Benutzerfreundliche Sortieranlage, welche vollautomatisch nach auflegen eines Werkstückes läuft
Tester	Testen / Wartung der Software

3.1.2 Systemkontext des Systems



3.1.3 Anforderungen

Lfd. Nr. / ID	Beschreibung	Fußnote
ANF01	Auf der Anlage sollen die Werkstücke in folgender vorgegebener Reihenfolge sortiert werden: <Type A> → <Type B> → <Type C>	12
ANF02	Die Sortier-Reihenfolge soll aus allen nicht-binären Werkstücken über eine auf dem System abgelegten Datei konfigurierbar sein. Die Konfigurationsdatei soll benutzerfreundlich gestaltet sein.	15, 43, 44
ANF03	Flache Werkstücke werden von FBM1 erkannt und aussortiert, sofern sie nicht der Konfiguration entsprechen oder die Rutsche an FBM1 voll ist.	17
ANF04	Werkstücke, die nicht der vorgegebenen Reihenfolge entsprechen, werden spätestens an FBM2 aussortiert.	18
ANF05	Auf dem FBM1 können sich mehrere Werkstücke befinden.	22
ANF06	Auf dem FBM2 darf sich maximal 1 Werkstück befinden. Die Übergabe von FBM1 an FBM2 erfolgt daher auch vereinzelt.	23
ANF07	Auf der Anlage (beide FBM) sollen die Werkstücke langsam durch die Höhenmessung transportiert werden	25
ANF08	Es darf kein Werkstück von der Anlage fallen.	26
ANF09	Sind beide Rutschen voll, läuft der Sortierbetrieb so lange weiter, bis eine Aussortierung eines Werkstückes nicht mehr erfolgen kann.	28
ANF10	Ist die Rutsche auf FBM1 voll, so soll die Aussortierung über FBM2 erfolgen	38
ANF11	Ist die Rutsche auf FBM2 voll, so soll die Aussortierung über FBM1 erfolgen.	39
ANF12	Wenn sich auf FBM1 kein Werkstück befindet, soll FBM1 anhalten.	37
ANF13	Wenn sich auf FBM2 kein Werkstück befindet, soll FBM2 anhalten.	37

ANF14	<p>Wenn ein Werkstück das Ende von FBM 2 erreicht, werden folgende Daten auf der Konsole ausgegeben:</p> <ul style="list-style-type: none"> • Werkstück-ID • Werkstück-Typ • Mittlerer Höhenmesswert aus der Mitte des Werkstücks von FBM1 in Millimeter • Mittlerer Höhenmesswert aus der Mitte des Werkstücks von FBM2 in Millimeter • Überschlagen (ja/nein) 	29-34
ANF15	Eine volle Rutsche ist an der entsprechenden Anlage zu signalisieren.	27
ANF16	An einem FBM können unterschiedliche Varianten der Sortiermechanik eingebaut sein (Auswerfer oder Weiche), die beide vom System unterstützt werden müssen	45-55
ANF17	Die Lichtschranke an der Höhenmessung darf nicht im Production-Mode verwendet werden.	58
ANF18	<p>Auf die Anlage werden mehrere Typen von WS unterschiedlich auf das Band gelegt, die alle erkannt werden müssen:</p> <ul style="list-style-type: none"> • Flache WS • Hohe WS mit BuM • Hohe WS mit BoM • Hohe WS ohne Bohrung <p>Diese haben unterschiedliche Farben (rot, schwarz, weiß) und Gewicht (Metall eingearbeitet oder ohne Metall).</p> <p>Alle anderen Werkstücke werden als "Unbekannt" eingeordnet.</p>	2-11
ANF19	Die sortierten WS am Ende von FBM2 müssen so bereitgestellt werden, dass ein Pick-and-Place Roboter sie entnehmen kann.	14
ANF20	Wenn die Lichtschranke am Anfang von FBM1 frei ist, können neue WS hier eingelegt werden.	19-21
ANF21	Eine Änderung der Aussortierung bedingt durch eine volle Rutsche (siehe ANF10 und ANF11) ist dem Bediener zu signalisieren.	40

ANF22	Es soll ein möglichst hoher Durchsatz an Werkstücken erreicht werden	42
ANF23	Der Betrieb der Anlage soll jederzeit sicher sein und darf keine Gefährdung des Bedieners hervorrufen	59
ANF24	<p>Folgende Fehlerzustände sollen erfasst und dem Bediener signalisiert werden:</p> <ul style="list-style-type: none"> • Beide Rutschen voll und ein notwendiges Aussortieren ist nicht mehr möglich <p>Die weiteren Fehlerfälle werden nach Absprache nicht umgesetzt</p>	62
ANF25	Durch Drücken des "Start"-Tasters (< 2sec) wechselt die Anlage in den Betriebszustand, durch langes (>= 2sec) Drücken in den Service-Mode.	70-71
ANF26	Durch Drücken des "Stop"-Tasters (< 2sec) wechselt die Anlage in den Ruhezustand (Wenn keine Fehler oder Warnungen vorliegen).	72-73
ANF27	Durch Drücken eines "E-Stopp" Schalters, steht die ganze Anlage (beide FBM!) still. Sind beide E-Stopp Schalter herausgezogen, bleibt die Anlage weiterhin so lange stehen, bis der Reset-Taster gedrückt wurde.	75-77
ANF28	Dem Benutzer sollen sinnvolle Hinweise zur Bedienung der Anlage angezeigt werden durch Nutzung der LEDs an den Tastern oder anderer Anzeigeelemente.	78
ANF29	Die grüne Ampel soll im Production-mode dauerhaft leuchten. Im Service-Mode soll sie grün blinken.	80-81
ANF30	Die gelbe Ampel soll bei anliegenden Warnungen blinken.	82
ANF31	<p>Die rote Ampel soll anliegende Fehler wie folgt anzeigen:</p> <ul style="list-style-type: none"> • Anstehend unquittiert: Schnelles Blinken (1 Hz) • Anstehend quittiert: Dauerhaftes Leuchten • Gegangen unquittiert: Langsames Blinken (0,5 Hz) 	95-99

ANF32	<p>Mit anliegenden Fehlern soll wie folgt umgegangen werden:</p> <ul style="list-style-type: none"> • Neu aufgetreten: Anstehend unquittiert • Fehler von Bediener durch Drücken der “Reset”-Taster quittiert: Anstehend quittiert • Anstehend quittiert und Fehler behoben: Wechsel in den Zustand “OK / Kein Fehler” • Automatisch behobene Fehler: Wechsel in den Zustand “Gegangen unquittiert”. Dieser wird durch quittieren verlassen 	83-93
-------	---	-------

3.1.4 Use Cases / User Stories

ID	UC-01
Titel	E-Stopp Funktion auslösen
Beschreibung	Das Use Case beschreibt die Funktionsweise des E-stopp Schalters.
Akteure	Anlage (Lamp, E-Stopp, ButtonReset), Benutzer.
Vorbedingung	<ul style="list-style-type: none"> • Die Anlage ist im Production-Mode.
Hauptszenario	<ol style="list-style-type: none"> 1. Ein E-Stopp Schalter an FBM1 oder FBM2 wird gedrückt. 2. Die Anlage stoppt. 3. Der E-Stopp Schalter wird herausgezogen. 4. Vorderband wird ausgeräumt. 5. Reset-Button an FBM1 und FBM2 wird betätigt.
Nachbedingung	Die Anlage ist im Production-Mode.

ID	UC-02
Titel	Geflippte Werkstücke erkennen
Beschreibung	Das Use Case beschreibt die Funktion der Anlage, wenn ein Werkstück bei Übergabe zwischen beiden FBM geflippt wird.
Akteure	FBM2 (LB1, LB2, LB3, LB4, Motor, Weiche), Console.
Vorbedingung	<ul style="list-style-type: none"> • Anlage ist im Production-Mode. • Am Ende des FBM2 soll ein Werkstück mit Bohrung kommen. • Werksück mit BoM wird am Anfang des FBM1, so dass die Bohrung nach unten zeigt. • Das Werkstück wird bei Übergabe von FBM 1 auf FBM 2 geflippt. • Das Werkstück muss sortiert werden.

Hauptszenario	<ol style="list-style-type: none"> 1. FBM2 erkennt, dass das Werkstück geflippt wurde 2. Das Werkstück wird nicht aussortiert 3. Das Werkstück wird zum Ende des FBM2 transportiert. 4. Die Daten des Werkstücks werden auf der Konsole mit „Überschlagen“ zusätzlich angezeigt.
Nachbedingung	<ul style="list-style-type: none"> • Anlage ist weiter im Production-Mode. • Kein Werkstück befindet sich auf FBM2

ID	UC-03
Titel	Rutsche 1 voll erkennen.
Beschreibung	Das Use Case beschreibt die Funktion der Anlage, wenn die Rutsche auf FBM 1 voll ist.
Akteure	Anlage (LB1, LB2, LB3, LB4, Motor, MD, HM, Weiche)
Vorbedingung	<ul style="list-style-type: none"> • Anlage ist im Production-Mode. • Rutsche 1 ist voll. • Rutsche 2 ist leer. • Ein flaches Werkstück befindet sich am MD des FBM1.
Hauptszenario	<ol style="list-style-type: none"> 1. System erkennt, dass das Werkstück aussortiert werden muss. 2. Das Werkstück wird zum FBM 2 transportiert. 3. Das Werkstück wird in Rutsche auf FBM 2 aussortiert.
Nachbedingung	Das Werkstück liegt in der Rutsche von FBM2

ID	UC-04
Titel	Rutsche 2 voll erkennen
Beschreibung	Das Use Case beschreibt die Funktion der Anlage, wenn die Rutsche auf FBM 2 voll ist.
Akteure	Anlage (LB1, LB2, LB3, LB4, Motor, HM, MD, Weiche)
Vorbedingung	<ul style="list-style-type: none"> • Anlage ist im Production-Mode. • Rutsche 2 ist voll. • Rutsche 1 ist leer. • Das (nicht flaches) Werkstück befindet sich am MD des FBM1 und passt nicht in die Sortierreihenfolge.
Hauptszenario	<ol style="list-style-type: none"> 1. System erkennt, dass das Werkstück aussortiert werden muss. 2. Das Werkstück wird in Rutsche auf FBM 1 aussortiert
Nachbedingung	Das Werkstück liegt in der Rutsche von FBM1

ID	UC-05
Titel	Beide Rutschen voll erkennen
Beschreibung	Das Use Case beschreibt die Fehlerbehandlung, wenn beide Rutschen voll sind und ein Werkstück aussortiert wird.
Akteure	Anlage (LB1, LB2, LB3, LB4, Motor, Weiche, ButtonReset, ButtonStart, Lamp).
Vorbedingung	<ul style="list-style-type: none"> • Anlage ist im Production-Mode. • Beide Rutschen sind voll. • Ein Werkstück muss aussortiert werden

Hauptszenario	<ol style="list-style-type: none"> 1. Grüne Lampe aus. 2. Die Anlage stoppt. 3. Rote Lampe blinkt (1Hz). 4. Benutzer drückt Button-Reset(<= 2 Sek.). 5. Rote Lampe leuchtet dauerhaft. 6. Benutzer behebt den Fehler. 7. Benutzer drückt Button-Start (<=2 Sek.). 8. Rote Lampe aus. 9. Grüne Lampe an.
Nachbedingung	Anlage ist im Production-Mode

ID	UC-06
Titel	Sortiervorgang starten
Beschreibung	Das Use Case beschreibt den Sortiervorgang der Anlage.
Akteure	Anlage (LB1, LB2, LB3, LB4, Motor, Weiche, HM, MD), Benutzer
Vorbedingung	<ul style="list-style-type: none"> • Anlage ist im Production-Mode. • Beide Rutsche sind leer. • Die von Anlage angenommene Reihenfolge ist: • WS-F → WS-BOM → WS-BM. • Reihenfolge der Werkstücke auf Anlage: <ul style="list-style-type: none"> ○ WS-BOM → WS-F → WS-F → WS-BOM → Werkstück BuM → WS-BOM. • Die Werkstücke werden nacheinander am Anfang des FBM1 gelegt.
Hauptszenario	<ol style="list-style-type: none"> 1. WS-BOM wird in Rutsche auf FBM 1 aussortiert. 2. WS-F wird zum Ende transportiert. 3. WS-F wird in Rutsche auf FBM 1 aussortiert. 4. WS-BOM wird zum Ende transportiert. 5. WS-BM wird zum Ende transportiert. 6. WS-BOM wird in Rutsche auf FBM 1 aussortiert.
Nachbedingung	<ul style="list-style-type: none"> • 2 Werkstück BoM sind in Rutsche auf FBM 1. • 1 WS-F ist in Rutsche auf FBM 2. • 1* WS-F → 1* WS-BOM → 1* WS-BM wurden zum Ende des FBM 2 transportiert

ID	UC-07
Titel	<i>Sortiervorgang starten für ein Werkstück</i>
Beschreibung	<i>Das Use Case beschreibt den Sortiervorgang der Anlage für ein Werkstück.</i>
Akteure	<i>Anlage (LB1, LB2, LB3, LB4, Motor, Weiche, HM, MD), Benutzer</i>
Vorbedingung	<ul style="list-style-type: none"> • <i>Anlage ist im Production-Mode.</i> • <i>Beide Rutsche sind leer.</i> • <i>Ein Werkstück wird am Anfang des FBM1 gelegt oder von FBM1 an FBM2 übergeben.</i>

Hauptszenario	<ol style="list-style-type: none"> 1. <i>Das Werkstück unterbricht LB1</i> 2. <i>Das Werkstück wird zur HD transportiert</i> 3. <i>Das Werkstück wird zur MD transportiert</i> 4. <i>Das Werkstück wird erkannt, ob es aussortiert werden muss</i> 5. <i>Das Werkstück unterbricht LB2</i> 6. <i>Das Werkstück muss nicht aussortiert werden</i> 7. <i>Das Werkstück unterbricht LB3</i>
Nachbedingung	<ol style="list-style-type: none"> 6a) <i>Das Werkstück muss aussortiert werden</i> 7a) <i>Das Werkstück wird von Weiche aussortiert</i> 8a) <i>Das Werkstück unterbricht LB4</i>

ID	UC-08
Titel	
Beschreibung	
Akteure	
Vorbedingung	
Hauptszenario	
Nachbedingung	

3.1.5 Absprachen

Nachfolgend sind die Absprachen ausgeführt, die mit dem Kunden getroffen wurden bzgl. der Anforderungen:

ID	Absprache
ASP-01	Wenn auf FB1 ein WS als "ungültig" erkannt wird (passt nicht zu der gewünschten Sortier-Reihenfolge), aber auf FB2 als "gültig" oder umgekehrt, soll immer aufgrund des an FBM2 erkannten WS-Typs entschieden werden, ob das WS aussortiert oder durchgelassen wird.
ASP-02	Vom Kunden wurde folgender Vorschlag gemacht: "Wenn ein WS as FBM2 aussortiert werden muss, die Rutsche jedoch voll ist: Können wir nicht zurück fahren und an FB1 aussortieren, wenn frei?" Dieses Feature wird nach Absprache nicht umgesetzt aufgrund hohem Programmieraufwand und kein Produktivitätsgewinn. In diesem Fall erfolgt eine Fehlermeldung und Bandstopp.
ASP-03	Der Höhenmesswert an FBM2 soll als "maximale Höhe in Millimeter" ausgegeben werden. Höhenangaben sollen generell immer in Millimeter erfolgen.
ASP-04	Werkstücke sollen bevorzugt an FBM2 aussortiert werden, da sich sonst alle schnell in der Rutsche stauen. Ausnahme: Aussortierung an FBM1 ist fest vorgegeben (flache WS).
ASP-05	Der Anfang von FBM1, an dem neue Werkstücke aufgelegt werden, ist definiert durch den Bereich in dem die erste Lichtschranke unterbrochen wird.
ASP-06	Die Lichtschranke der Höhenmessung wird zum Abnahmetest abgeschaltet. In unserem Prototypen ist diese noch vorhanden, im Produktivsystem wird die Hardware jedoch eingespart, da der Systemarchitekt der Meinung ist, das bekommen die Softwerker auch ohne hin.
ASP-07	Eine volle Rutsche wird am jeweiligen FBM als Warnung signalisiert (gelbe Ampel blinkt).

ASP-08	Da die Bewertung, ob ein WS der gewünschten Reihenfolge entspricht, auf Grundlage von FBM2 erfolgt (siehe ASP-01), wird dieser WS-Typ auch als "finaler Typ" ausgegeben. Zusätzlich wird angegeben, dass sich das WS überschlagen hat.
ASP-09	Wenn eine Rutsche voll ist und die Aussortierung deshalb am jeweils anderen FBM erfolgt, wird diese Situation neben dem Blinken der gelben Ampel an der vollen Rutsche (Warnung) zusätzlich über das Leuchten von Q1 (Aussortierung an FBM1) bzw. Q2 (Aussortierung an FBM2) signalisiert.

3.2 Hardware: Analyse der technischen Gegebenheiten

3.2.1 Technischer Aufbau und Hardwarekomponenten

Die Festo-Transfersysteme verfügen über unterschiedliche Sensoren und Aktoren, mit dessen Hilfe unsere Sortieranlage realisiert wird.

Sensorik:

Für die Registrierung der Position von WS werden Lichtschranken verwendet. Damit Entscheidungen zur Sortierung anhand der Höhe von WS getroffen werden können, werden Höhsensoren verwendet. Um das Metall in WS zu erkennen wird ein Metallsensor verwendet. Zudem befindet sich an einem Transfersystem ein Bedienfeld mit den Tasten Start, Stop und Reset. Rechts daneben befindet sich der E-Stopp Schalter. Es handelt sich um einen üblichen E-Stopp-Schalter, der bei Betätigung einklinkt. Durch Herausziehen wird er wieder in seine ursprüngliche Position versetzt. Hierbei handelt es sich nicht um einen klassischen E-Stopp Schalter, der die Anlage stromlos setzt. Der sichere Betrieb nach Betätigung des Schalters muss deshalb über die Steuerungssoftware realisiert werden.

Aktorik:

Die Förderbandmodule werden durch einen eigenen Steuerungscomputer gesteuert mit einem Beaglebone Black. Die beiden Computer sind über Ethernet gekoppelt.

Das FB lässt sich durch entsprechende Ansteuerung des Motors in langsamer und schneller Geschwindigkeit und dazu nach links oder rechts bewegen.

Zur Aussortierung von WS in eine Rutsche existieren zwei Varianten des Hardwareaufbaus. Die eine Variante verfügt über eine Weiche, die zweite Variante über einen Kicker. Die Weiche ist im stromlosen Zustand geschlossen. Beim Öffnen fließt Strom durch die Magnetspule, die die Weiche betätigt. Der Kicker lässt im stromlosen Zustand WS passieren. Fließt Strom durch den Kicker, so fährt dieser aus und drückt ein Werkstück aktiv in die Rutsche. Im ausgefahrenen Zustand können keine WS passieren und werden auch nicht in die Rutsche befördert.

Um den Benutzer über den Zustand des Systems zu informieren, verfügt ein Transfersystem über eine Konsolenausgabe (mit verbundener Momentics IDE über "stdout") und für die zusätzliche Visualisierung über eine Ampel. Die Ampel kann die Farben grün, orange und rot darstellen.

Darüber hinaus verfügt das Bedienfeld über Status-LED's (jeweils an den Tastern Start und Reset sowie Q1 und Q2).

Das FB hat eine Länge von 700 mm mit einer Tiefe von 70 mm. Der Abstand zu den äußeren Förderbandbegrenzungen beträgt 5 mm. Somit beträgt die effektive nutzbare Gesamthöhe des FB 80 mm. Die am Transfersystem angebrachte Rutsche ist 180 mm lang und 70 mm breit. Sie weist eine Verengung von 30 mm an der zulaufenden Seite vom FB auf.

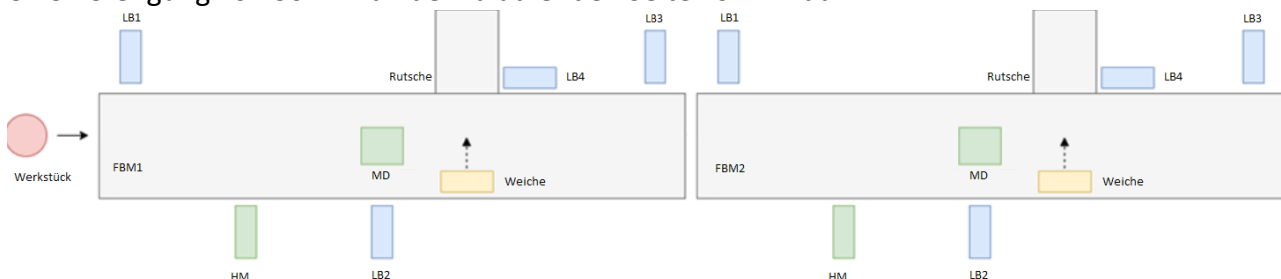
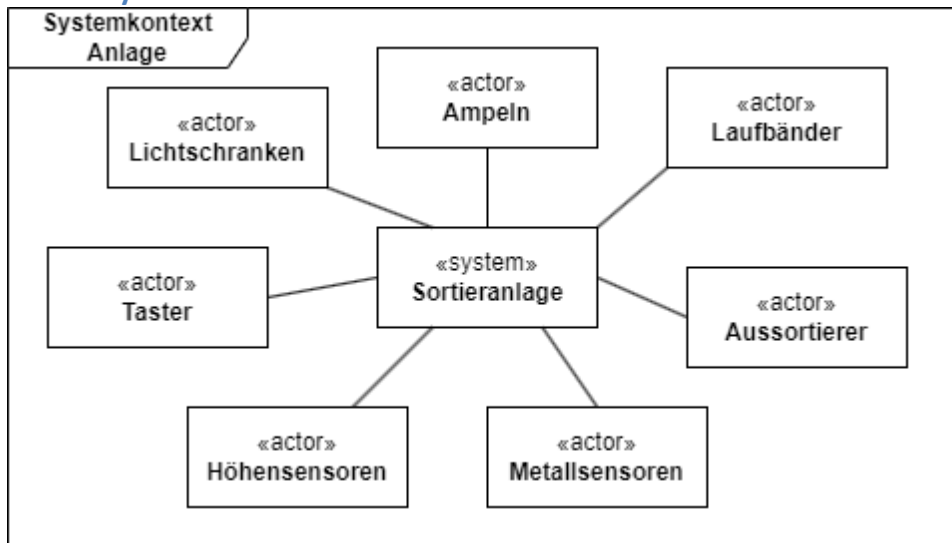


Abbildung 1: Hardware Darstellung der Sortieranlage

HW_REQ_002	Binär codierte WS haben die gleiche Höhe wie hohe WS. Diese müssen voneinander unterschieden werden können.
HW_REQ_003	Die Lichtschranke, die erkennt, wenn sich ein WS unter dem Höhenmesser befindet, steht im Produktivbetrieb nicht zur Verfügung. Die Erkennung der Präsenz von Werkstücken unter der Höhenmessung muss deshalb anders erfolgen.

3.3 Softwareebene

3.3.1 Systemkontext der Software



3.3.2 Resultierende Anforderungen an die Software

Lfd. Nr. / ID	Beschreibung
REQS_001	Beim Auflegen eines WS am Anfang von FBM1 wird der Motor aktiviert und das FB bewegt sich.
REQS_002	Wenn ein aufgelegtes WS den Bereich der Höhenmessung erreicht, bewegt sich der Motor am jeweiligen FBM in langsamer Geschwindigkeit. Beim Verlassen des Bereichs bewegt sich der Motor wieder in schneller Geschwindigkeit.
REQS_003	<p>Ein flaches WS, das nicht der gewünschten Reihenfolge entspricht, wird an FBM1 aussortiert. Bei voller Rutsche wird das WS durchgelassen und an FBM2 aussortiert.</p> <p>Ein nicht-flaches WS, das nicht der gewünschten Reihenfolge entspricht, wird an FBM2 übergeben und dort aussortiert. Bei voller Rutsche an FBM2 erfolgt die Aussortierung an FBM1.</p> <p>Bei gewünschter Aussortierung und falls beide Rutschen voll sind, erfolgt ein Bandstopp und Fehlermeldung an beiden Anlagen.</p>
REQS_004	Bei voller Rutsche erfolgt eine Warnmeldung. Hierbei blinkt die gelbe Ampel am betroffenen FBM und es wird eine Warnung auf der Konsole ausgegeben mit der Aufforderung, die Rutsche freizumachen.
REQS_005	Bei der Übergabe eines WS von FBM1 an FBM2 (LS am Anfang von FBM2 unterbrochen) wird der Motor an FBM2 aktiviert und das FB bewegt sich.
REQS_006	<p>Wenn ein WS das Ende von FBM2 erreicht, stoppt der Motor. Der Motor kann erst wieder anlaufen, wenn die LB wieder frei ist.</p> <p>Der Motor stoppt auch, wenn ein WS aussortiert wurde.</p>

REQS_007	Beim Drücken des E-Stopp muss ein sofortiger Bandstopp beider FBM erfolgen.
REQS_008	<p>Im Ruhezustand wechselt die Anlage durch das kurze Drücken des Start-Tasters vom Ruhezustand in den Betriebszustand.</p> <p>Im Ruhezustand wechselt die Anlage durch das lange Drücken des Start-Tasters vom Ruhezustand in den Service-Mode.</p>
REQS_009	Im Betriebszustand wechselt die Anlage durch das Drücken des Stopptasters in den Ruhezustand. Dieser Wechsel ist nur möglich, wenn keine Fehler oder Warnungen vorliegen.
REQS_010	<p>Im Betriebszustand wechselt die Anlage beim Auftreten eines Fehlers, der sich nicht von selbst beheben lässt, in den Fehlermodus, der sich wie folgt verhält:</p> <ol style="list-style-type: none"> 1. Rote Ampel blinkt schnell (1 Hz) 2. Drücken des Reset-Tasters: Rote Ampel leuchtet dauerhaft 3. Drücken des Start-Tasters: Wechsel in den Betriebszustand <p>Im Betriebszustand wechselt die Anlage beim Auftreten eines Fehlers, der sich von selbst behebt, in den Fehlermodus. Wenn er sich danach von selbst behebt, verhält sich die Anlage wie folgt:</p> <ol style="list-style-type: none"> 1. Rote Ampel blinkt langsam (0,5 Hz) 2. Drücken des Start-Tasters: Wechsel in den Betriebszustand
REQS_011	<p>Das System muss folgende Konfiguration ermöglichen, die beim Programmstart aus einer Konfigurationsdatei eingelesen wird:</p> <ul style="list-style-type: none"> • Modus: Master oder Slave • Sortiermechanik: Weiche oder Auswerfer • Gewünschte Sortierreihenfolge durch Einlesen einer Konfigurationsdatei auf dem Master-System <p>Wird eine ungültige oder fehlende Konfiguration festgestellt, erfolgt eine Fehlermeldung und ein Wechsel in den Betriebszustand ist nicht möglich.</p>

REQS_012	<p>Wenn ein WS an der Weiche von FBM1 angekommen ist, werden folgende Informationen auf der Konsole ausgegeben:</p> <ul style="list-style-type: none"> • WS-ID • WS-Typ • Mittlere Höhe in Millimetern an FBM1 <p>Wenn ein WS an der Weiche von FBM2 angekommen ist, werden folgende Informationen auf der Konsole ausgegeben:</p> <ul style="list-style-type: none"> • WS-ID • WS-Typ • Mittlere Höhe in Millimetern an FBM1 • Maximale Höhe in Millimetern an FBM2 • WS hat sich überschlagen: ja/nein <p>An beiden FBM erfolgt zusätzlich die Ausgabe, ob das WS gültig ist (der gewünschten Reihenfolge entspricht) oder eine Aussortierung erfolgt.</p>
----------	--

3.3.3 Nachrichten und Signale

HAL

Folgende Events werden jeweils von HAL FBM1 (Master) und FBM2 (Slave) an Logic (Master) über Eventmanager gesendet. Die X steht für die Nummer der FBM, kann somit 1 oder 2 sein.

Name	Beschreibung	
ESTOP_X_PRESSED	E-Stopp an FBM1/2 wurde gedrückt	
ESTOP_X_RELEASED	E-Stopp an FBM1/2 wurde gelöst	
START_X_SHORT	An einer der beiden Anlagen wurde die Start-Taste gedrückt (≤ 2 sec)	
START_X_LONG	An einer der beiden Anlagen wurde die Start-Taste gedrückt (> 2 sec)	
STOP_X_SHORT	An einer der beiden Anlagen wurde die Stop-Taste gedrückt (≤ 2 sec)	
RESET_X_SHORT	An einer der beiden Anlagen wurde die Reset -Taste gedrückt (≤ 2 sec)	
RESET_X_LONG	An einer der beiden Anlagen wurde die Reset-Taste gedrückt (> 2 sec)	
LB1_X_BLOCKED	LB Start FBM1 unterbrochen	

LB1_X_UNBLOCKED	LB Start FBM1 wieder frei	
LB2_X_BLOCKED	LB an der Weiche unterbrochen	
LB2_X_UNBLOCKED	LB an der Weiche wieder frei	
LB3_X_BLOCKED	LB Ende FBM1 unterbrochen	
LB3_X_UNBLOCKED	LB Ende FBM1 wieder frei	
LB4_X_BLOCKED	LB oben an der Rutsche unterbrochen	
LB4_X_UNBLOCKED	LB oben an der Rutsche wieder frei	
MD_X_PAYLOAD	<i>Payload= ist es ein Metal (True oder False)</i>	
HM_X_PAYLOAD	<i>Payload= die Height des WS in mm</i>	

Folgende Events werden jeweils von Logic (Master) an HAL FBM1 und FBM2 über Eventmanager gesendet.

Name	Beschreibung	
MOTOR_X_FAST	Startet den Motor mit Normaler Geschwindigkeit.	
MOTOR_X_SLOW	Startet den Motor mit Langsamer Geschwindigkeit.	
MOTOR_X_STOP	Hält den Motor an.	
LAMP_X_G_ON	Schaltet die Grüne Lampe an.	
LAMP_X_GRN_OFF	Schaltet die Grüne Lampe aus.	
LAMP_X_RED_ON	Schaltet die rote Lampe an.	
LAMP_X_RED_OFF	Schaltet die rote Lampe aus.	
LAMP_X_GELB_ON	Schaltet die gelbe Lampe an.	
LAMP_X_GELB_OFF	Schaltet die gelbe Lampe aus.	
ESTOP_X_PRESSED	E-Stopp an FBM1/2 wurde gedrückt	
ESTOP_X_RELEASED	E-Stopp an FBM1/2 wurde gelöst	
SORT_X_OUT	Weiche sortiert aus den WS	
LAMP_X_RED_1hz	Rote Lampe blinkt (1 hz)	
LAMP_X_RED_0_5hz	Rote Lampe blinkt (0.5 hz)	
QLED1_X_ON	Schaltet die Q1 Led an.	
QLED1_X_OFF	Schaltet die Q1 Led aus.	
QLED2_X_ON	Schaltet die Q2 Led an.	
QLED2_X_OFF	Schaltet die Q2 Led aus.	

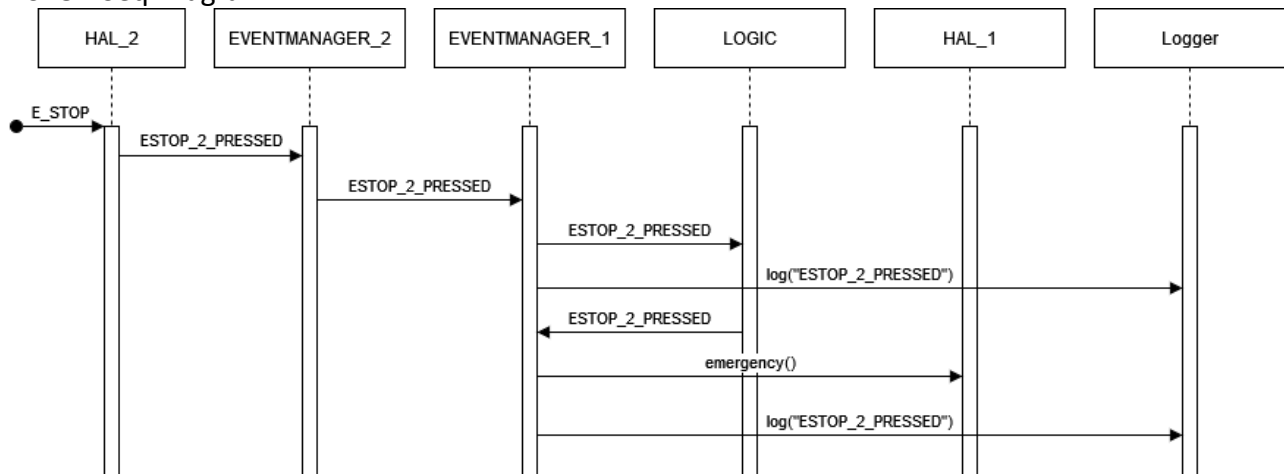
Watchdog

Der Watchdog läuft auf FBM1 und FBM2 und sendet all seine Events an den eigenen EventManager.

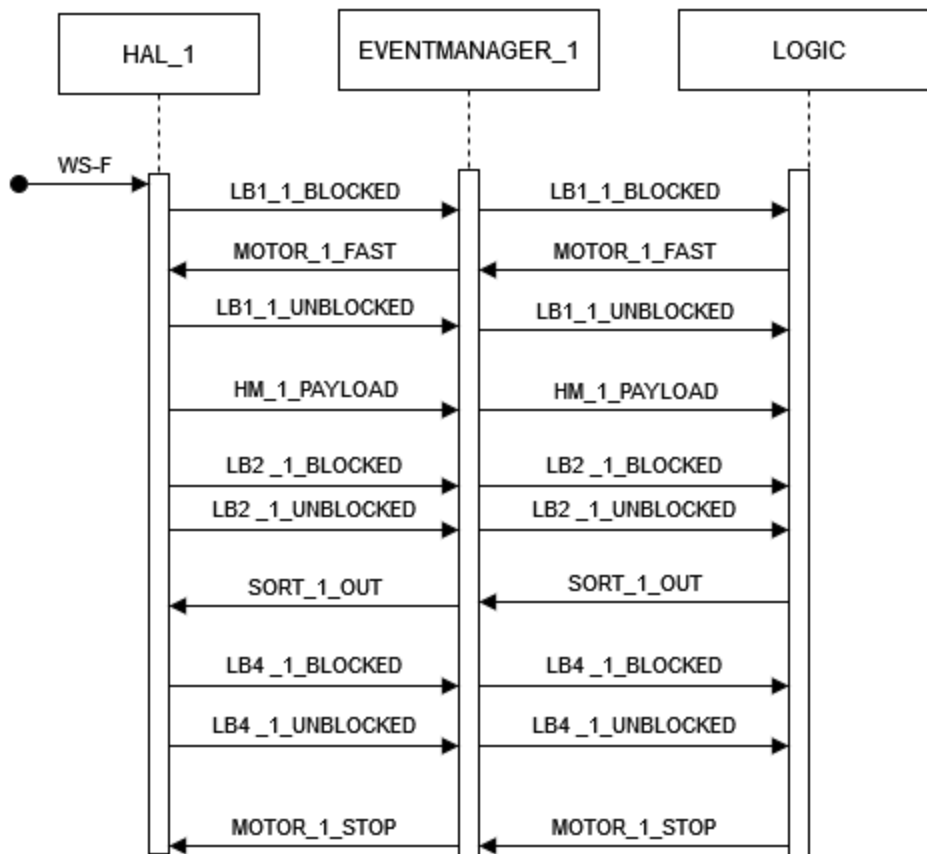
Name	Beschreibung
WD_CONN_ESTABLISHED	Nach einer Verbindungsaufforderung (z. B. nach dem Programmstart) hat die Partneranlage innerhalb der maximal erlaubten Zeit geantwortet
WD_CONN_LOST	Die Verbindung zur Partneranlage wurde unterbrochen
WD_CONN_REESTABLISHED	Nach einem vorigen ConnectionLost Event wurde die Verbindung wiederhergestellt
WD_HEARTBEAT_FBM1	Heartbeat Message von FBM1 an FBM2
WD_HEARTBEAT_FBM2	Heartbeat Message von FBM2 an FBM1

3.3.4 Sequenz Diagramm (Events)

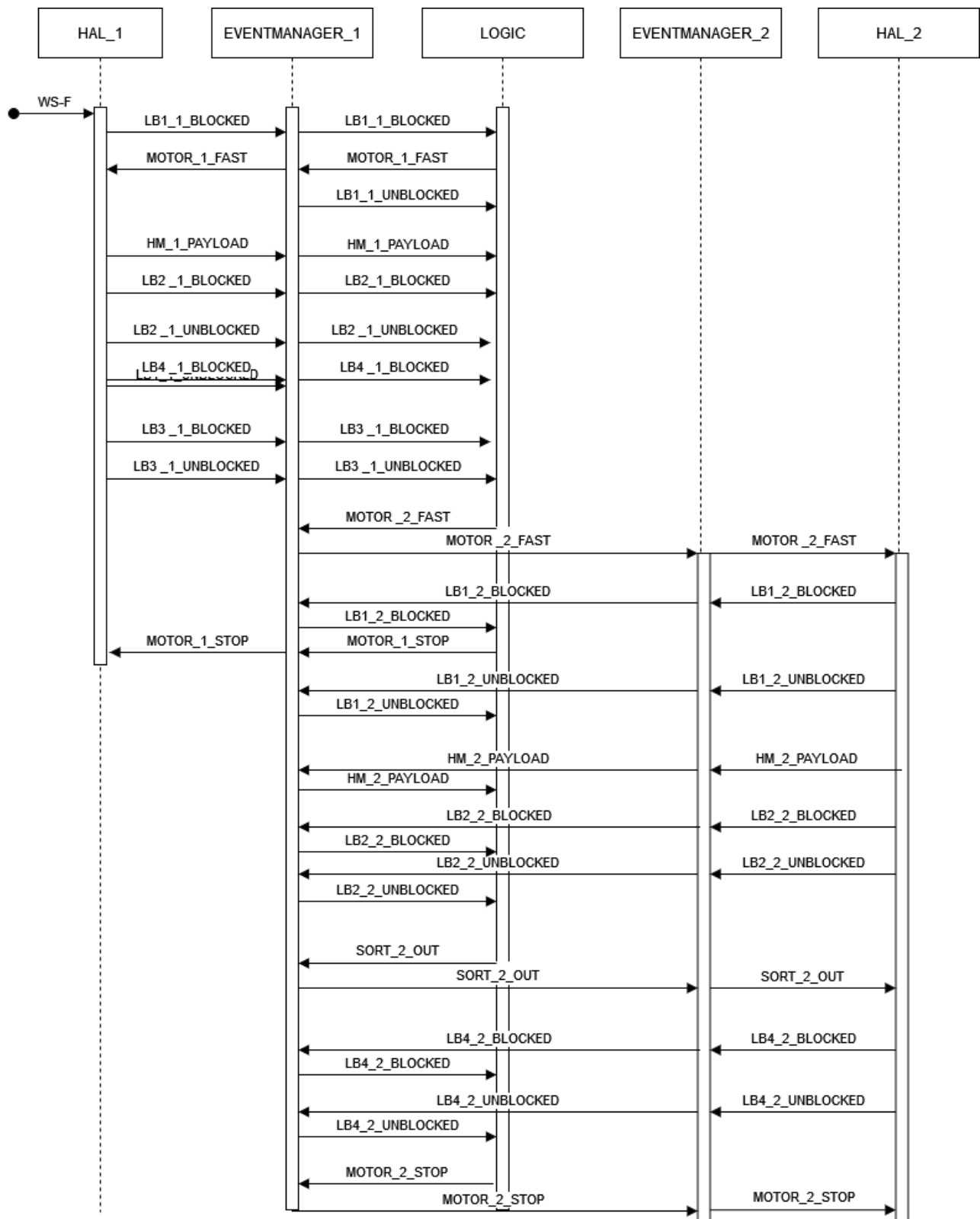
E-STOP Seq-Diagramm



Sortieren eine WS-F in FBM1



Sortierung WS-F RUTSCHE 1 VOLL

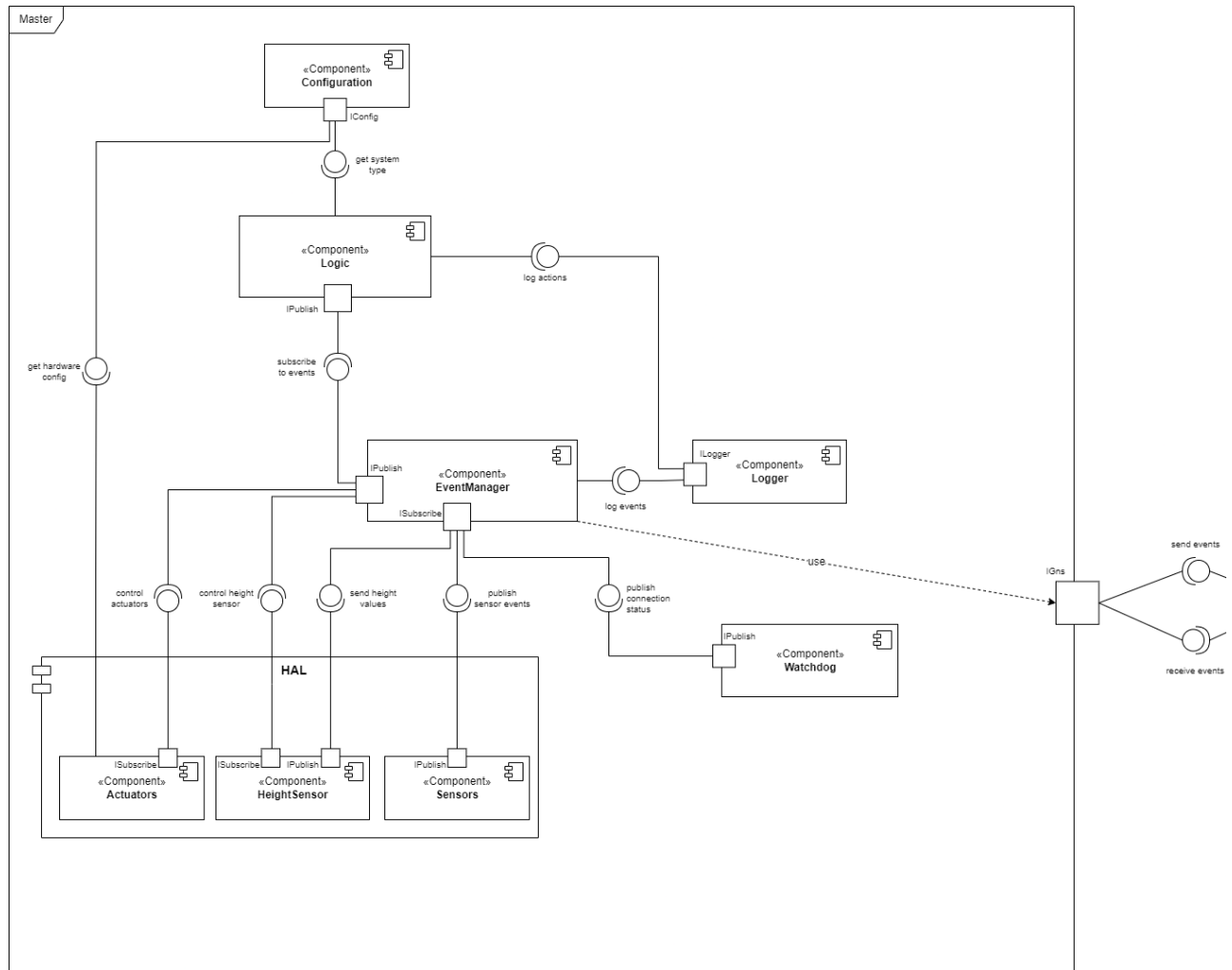


Watchdog Seq-Diagram

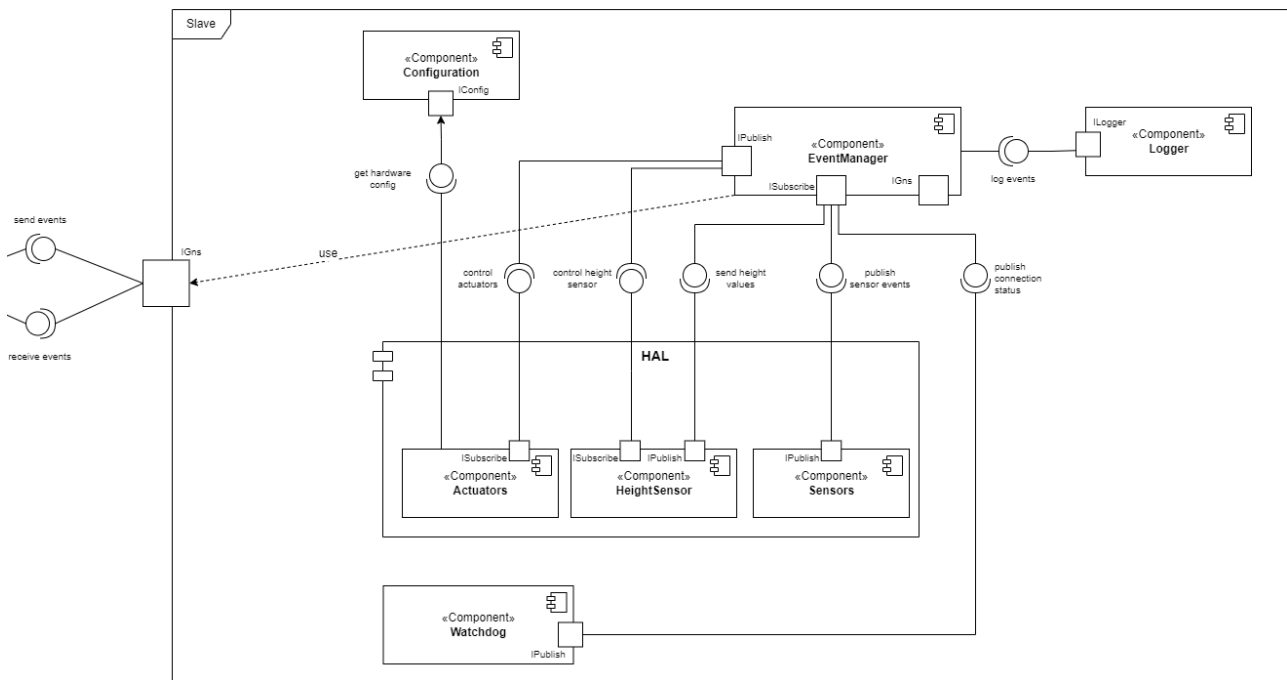
4 Software-Design

4.1 Software Architektur

Master



Slave



4.1.1 Beschreibung der Komponenten

Der **EventManager** (EVM) ist die zentrale Schnittstelle, die alle im System auftretenden Events empfängt und an interessierte Komponenten weiterleitet. Dem EventManager angeschlossene Komponenten können sich für bestimmte Events anmelden ("subscribe"). Wenn diese auftreten, benachrichtigt der EventManager die Komponente darüber über den Aufruf einer Funktion ("publish").

Nachfolgend ist die Funktionsweise der einzelnen Komponenten kurz beschrieben:

Configuration

Beinhaltet die Konfiguration des Systems, auf dem die aktuelle Software läuft:

- Master oder Slave
- Auswerfer/Weiche
- WS-Reihenfolge (wird bei Programmstart eingelesen)
- Kalibrierwerte (wird in einer Datei gespeichert nach der Kalibrierung im Service-Mode)

EventManager

Empfängt Events von anderen Komponenten (Publishers) und leitet sie an Interessenten (Subscribers) weiter (an andere Anlage per GNS).

Watchdog

Überwacht Verfügbarkeit der Partneranlage, meldet Fehler und wenn ein voriger Verbindungsabbruch wieder behoben ist.

Logic

Beinhaltet die FSM's (Finite State Machines) zur Steuerung der Sortieranlage sowie Informationen über die auf der Anlage befindlichen Werkstücke. Dies wird über eine Klasse "WorkpieceManager" realisiert.

HAL

- Sensorik meldet Sensor-Events (z. B. LB unterbrochen/wieder frei, WS in Höhenmessung, Metall detektiert) an den EventManager
- Sensorik beinhaltet auch den Höhenmesser (ADC), der in einem eigenen Thread läuft und auf Anfrage eine kontinuierliche Höhenmessung startet bzw. stoppt. Die Messergebnisse werden nach der Messung als Event an den EVM geschickt (Messwerte in mm)
- Aktorik lässt sich ansteuern per Methodenaufruf (explizit oder HAL-Events beim EVM)

Logger

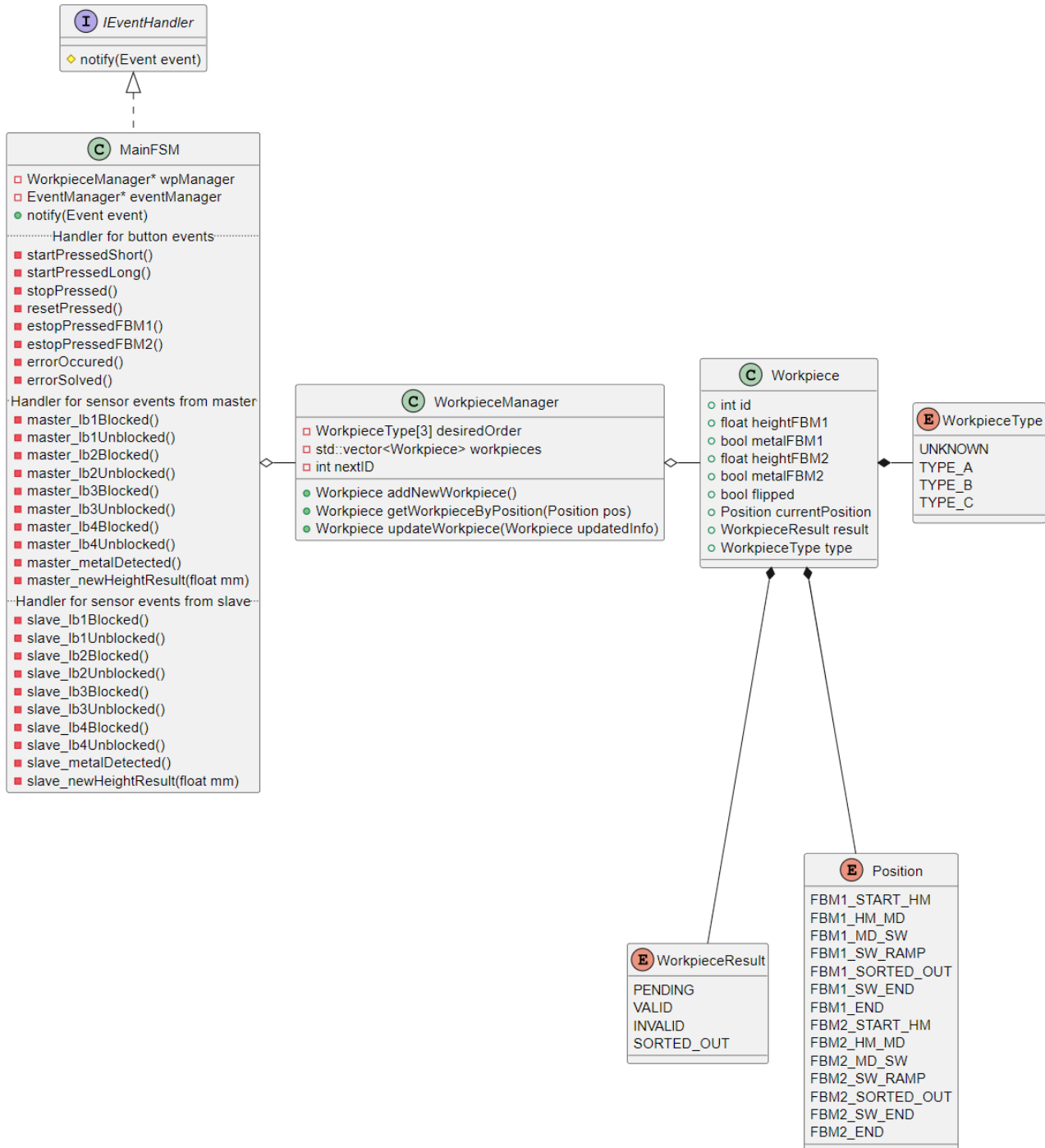
- Loggt Infos, Warnungen, Fehler und Debug-Meldungen auf die Konsole
- Alle wichtigen Events werden auf Master geloggt, was zum Debuggen auch ausreichend ist
- Ist auf alle benötigten Events subscribed, um diese zu loggen
- Logic schickt bei Statewechsel changeEvents, welche ebenfalls geloggt werden

4.2 Software Struktur

4.2.1 Logic mit WorkpieceManager

Die Logic besteht aus mehreren FSM's, von denen nachfolgend die MainFSM dargestellt ist. Diese empfängt Events, wenn Taster gedrückt wurden und steuert den aktuellen Betriebsmodus.

Zusätzlich werden auch Events der Sensoren verarbeitet, die Informationen der auf dem Band befindlichen Werkstücke aktualisiert und ggf. Aktionen ausgelöst (z. B. Schließen der Weiche, um Werkstück auszusortieren). Dafür wird die Klasse WorkpieceManager benutzt, der alle Werkstücke, deren aktuelle Position und Daten in einer internen Liste verwaltet.

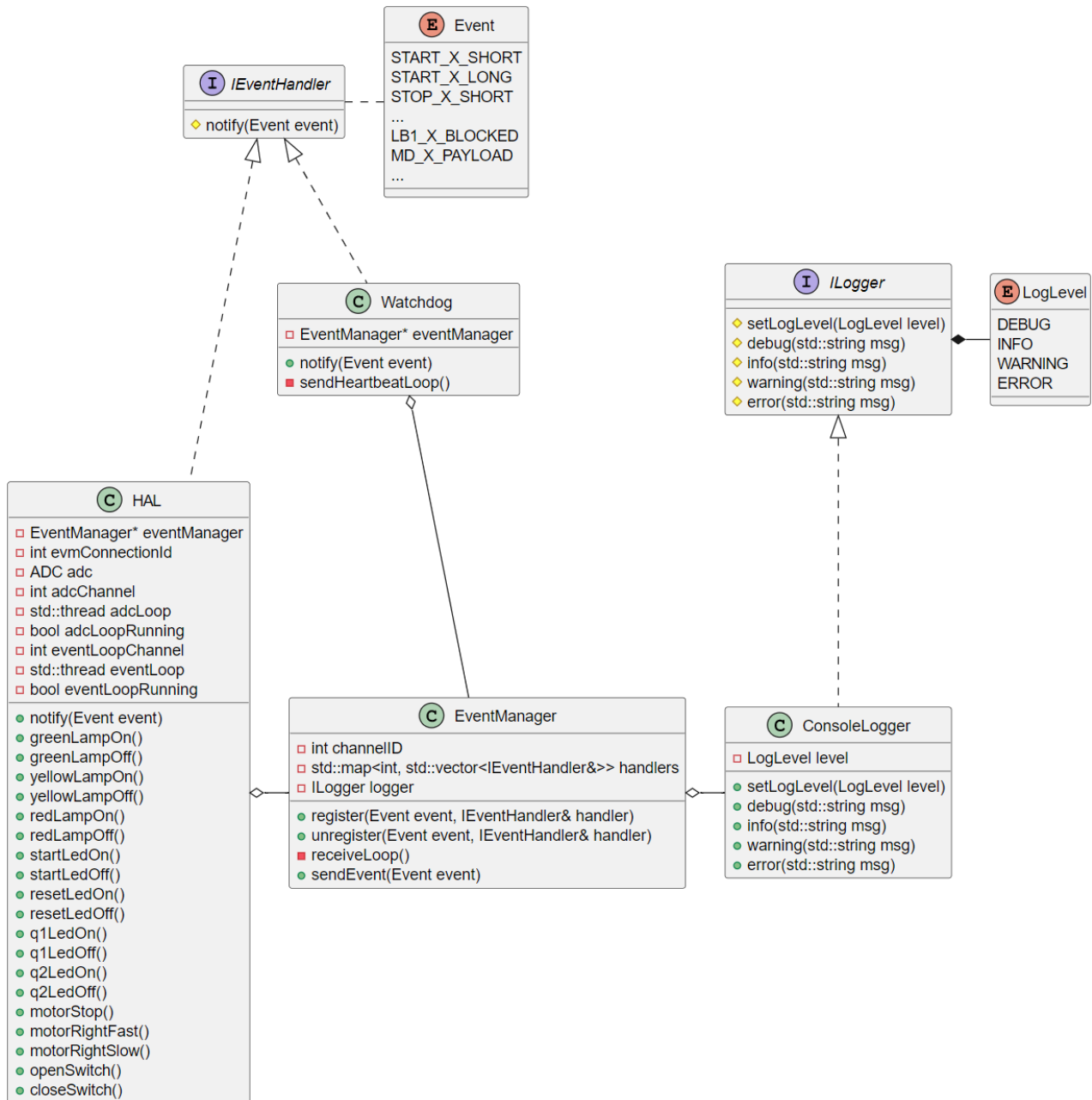


4.2.2 HAL, EventManager, Watchdog und Logger

Nachfolgend ist ein Klassendiagramm der HAL, des EventManagers und Loggers aufgeführt. Der EventManager agiert auch mit anderen Komponenten, allerdings eignet sich die HAL sehr gut, um das Konzept zum Behandeln von Events anhand des Reactor Patterns darzustellen.

Es existiert eine abstrakte Klasse ("Interface") **IEventHandler**, die lediglich über eine Methode `notify` verfügt. Jede Klasse, die dieses Interface nun implementiert, kann sich beim EventManager über die Methode `register` anmelden. Beim Auftreten dieses Events ruft der EventManager die Callback-Methode `notify` auf. Das weitere Verarbeiten des Events geschieht nun in der Klasse des spezifischen EventHandlers (Beispiel: HAL Aktorik).

Auf dem Klassendiagramm ist auch der Logger aufgeführt, welcher auch über ein Interface implementiert wird. Dies lässt Raum dafür, den *ConsoleLogger*, der Meldungen auf die Konsole loggt, einfach auszutauschen, z. B. über einen Logger, der die Log-Meldungen per MQTT sendet oder in eine Datei schreibt.




4.2.3 Configuration

Die Configuration Komponente ist als Singleton implementiert, es existiert damit nur ein Objekt davon zur Laufzeit. Zum Programmstart wird die aktuelle Config aus der Datei eingelesen und intern gespeichert. Komponenten, die die aktuelle Konfiguration benötigen können diese über eine Referenz auf das Objekt einlesen.

Die Konfiguration enthält folgende Informationen:

- `isMaster` – `true`: Configuration läuft auf dem Master-System, `false`: Slave
- `pusher` – `true`: Zum Aussortieren wird der Auswerfer verwendet, `false`: Weiche
- `configuredWorkpieceOrder`: Die gewünschte Reihenfolge der Sortierung
- `calOffset`: Das kalibrierte Offset (Laufband-Höhe)

- `adcIncPerMillimeter`: Die kalibrierte Höhe eines flachen Werkstücks (wird zur zukünftigen Berechnung benötigt)

 Configuration
<ul style="list-style-type: none"> □ <code>Configuration*</code> instance □ <code>bool</code> <code>isMaster</code> □ <code>bool</code> <code>pusher</code> □ <code>WorkpieceType[3]</code> <code>configuredWorkpieceOrder</code> □ <code>int</code> <code>calOffset</code> □ <code>int</code> <code>adcIncPerMillimeter</code>
<ul style="list-style-type: none"> ● <code>Configuration getInstance()</code> ● <code>readConfig(std::string configFilePath)</code> ● <code>bool systemIsMaster()</code> ● <code>bool pusherUsed()</code> ● <code>WorkpieceType[] getDesiredOrder()</code> ● <code>saveCalibration(int offset, int flatHeight)</code> ● <code>float convertAdcToMillimeter(int adcValue)</code>

4.3 Verhaltensmodellierung

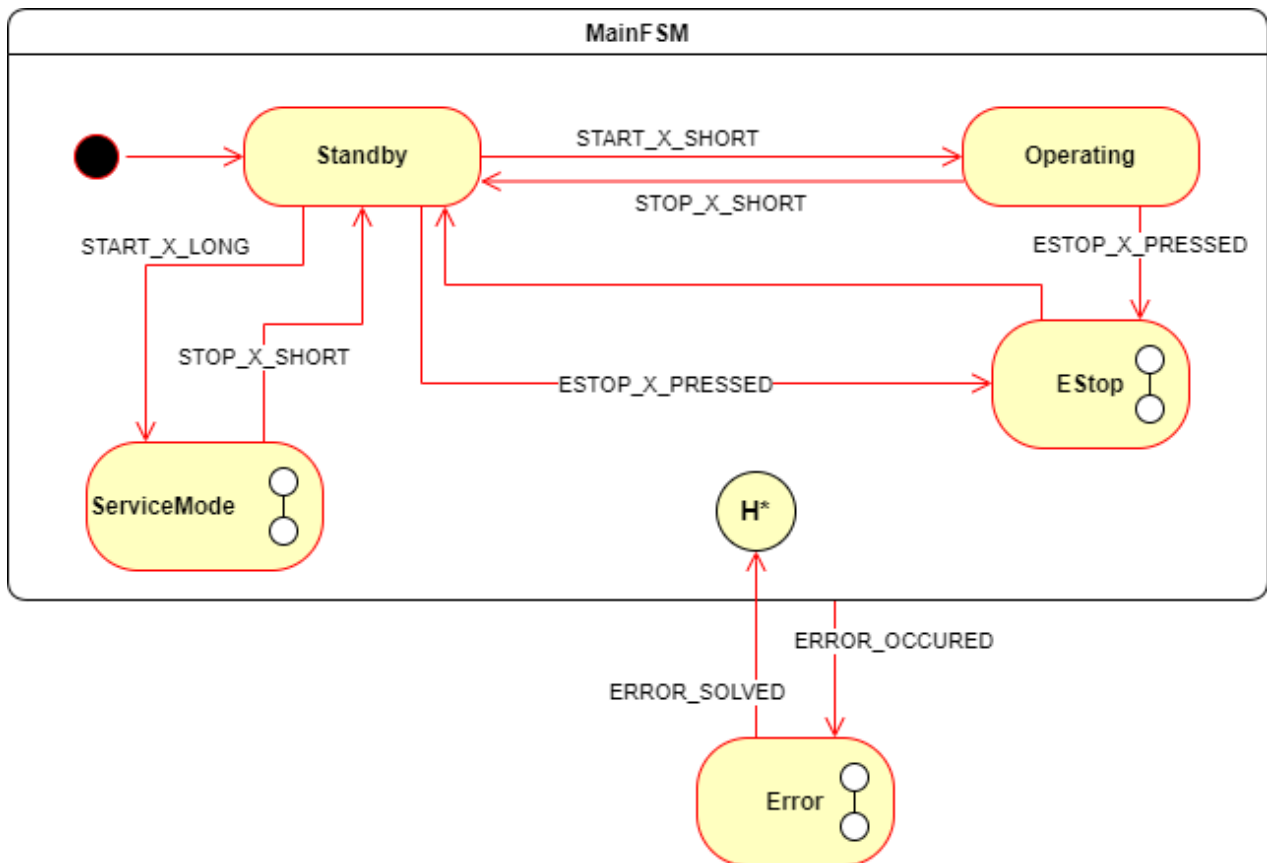
4.3.1 MainFSM

Zur Steuerung des aktuellen Betriebsmodus verwenden wir eine MainFSM.

Diese reagiert im Wesentlichen auf die Tastendrucke (Start, Stop, E-Stop) und wechselt die Betriebsmodi entsprechend.

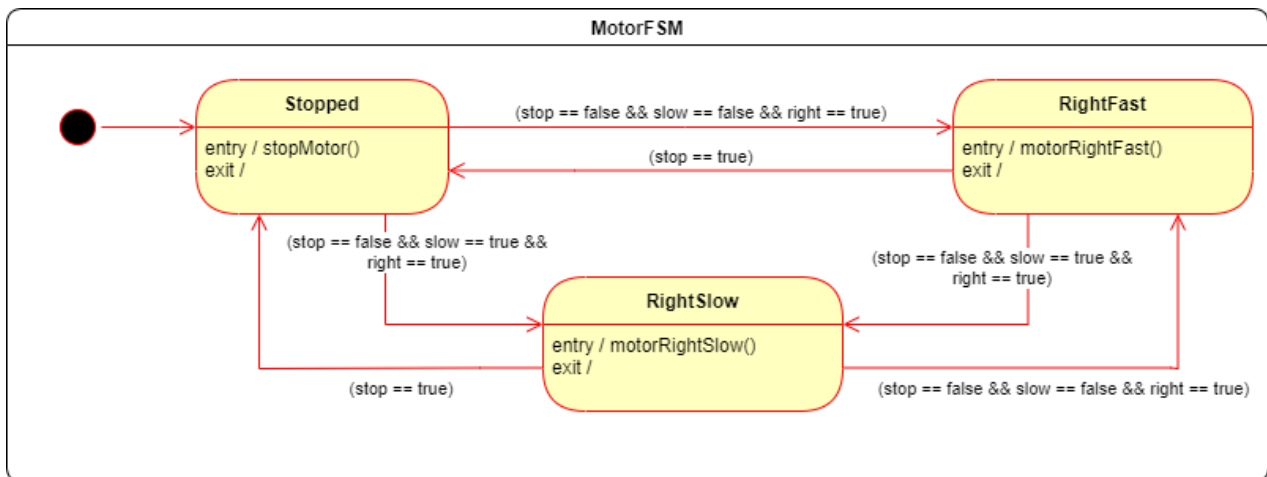
Im Betriebszustand ("Operating") wird auch auf Events von den Sensoren reagiert. Diese müssen so verarbeitet werden, dass die Informationen zu den auf den Bändern befindlichen Werkstücken aktualisiert werden. Dafür verwenden wir einen internen "WorkpieceManager", der die aktuellen Informationen und Positionen der Werkstücke kennt.

- Standby: Ruhezustand – alles aus
- Operating: Betriebszustand - Grüne Lampe leuchtet, bereit zum Einlegen neuer Werkstücke. E
- ServiceMode: Hier werden Selbsttests und Kalibrierungen durchgeführt. Da dies ein spezieller Modus ist, der nur von befähigten Personen durchgeführt werden kann, behandeln wir nicht das Drücken des E-Stopps
- EStop: alles aus – die Sub-Statemachine ist dafür zuständig, dass dieser Modus wieder verlassen werden kann, wenn die Anlage wieder sicher ist
- Error: Fehlerzustand – die Sub-Statemachine ist dafür zuständig, dass der Fehler gemäß Anforderung quittiert und behoben wird



4.3.2 MotorFSM

Da es möglich ist, dass mehrere Komponenten zur gleichen Zeit Anfragen zur Änderung der Geschwindigkeit stellen, ist es notwendig diese zu priorisieren. Dafür verwenden wir eine MotorFSM, von der jeweils für FBM1 und FBM2 eine Instanz auf dem Master erstellt wird. Beim Wechsel in einen neuen State verschickt die MotorFSM Events an den EventManager, der diese dann an die Aktorik (bzw. an den EventManager des Slaves) weiterleitet.



5 Implementierung

Anmerkung: Nur wichtige Implementierungsdetails sollen hier erklärt werden. Code-Beispiele (snippets) können hier aufgelistet werden, um der Erklärung zu dienen. Welche Patterns haben Sie für Ihre Implementierung benutzt.

Anmerkung: Bitte KEINE ganzen Programme hierhin kopieren!

6 Qualitätssicherung

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

6.1 Teststrategie

Definieren Sie Zeitpunkte für die jeweiligen Teststufen in Ihrer Projektplanung. Dazu können Sie die Meilensteine zu Hilfe nehmen. Überlegen Sie, wie die Test-Architektur der jeweiligen Teststufen aussehen. Verwenden Sie Testmethoden wie z.B. Grenzwertanalyse, 100% Zustandsabdeckung, 100% Transitionsüberdeckung, Tiefensuche, Breitensuche, etc. Versuchen Sie, so gut wie möglich, Ihre Tests zu automatisieren.

Phase 1: Modultest

Während der ersten Entwicklungsphase werden die Teilkomponenten (Module) des Systems unabhängig voneinander entwickelt. Deren Funktionalität wird mithilfe automatisierter Unit-Tests überprüft. Bevor ein Modul fertig gestellt werden kann, muss es als „Black Box“ getestet werden, d. h. die Schnittstellen, die ein Modul für andere Komponenten zur Verfügung stellt, werden getestet anhand der Design-Spezifikation. Dabei ist es ggf. erforderlich, dass das Verhalten anderer Komponenten simuliert wird (bspw. das Versenden von Events und Vergleich der Reaktion des Moduls darauf).

Phase 2: Integrationstest

Sobald alle Einzelmodule entwickelt sind, wird deren Zusammenwirken miteinander getestet. Dabei muss das Verhalten des Gesamt-Systems simuliert werden. Unterstützung dabei kann die zur Verfügung gestellte Simulationssoftware bieten, mit der Abläufe an der realen Anlage simuliert und reproduzierbar wiederholt werden können.

Phase 3: Systemtest

Mit dem Systemtest wird die fertige Software auf das Zielsystem aufgespielt und die Funktionalität des Systems als Ganzes getestet. Die Einhaltung der Kundenanforderungen, Sicherheitsfunktionen sowie Qualitätsanforderungen werden getestet. Abweichungen sind festzuhalten und zu beseitigen, bevor die Abnahme durch den Kunden erfolgen kann.

Phase 4: Abnahmetests

Nach erfolgreichem Bestehen des Systemtests und Beseitigen letzter Bugs, erfolgt die Abnahme durch den Kunden durch Abnahmetests, bei denen die Einhaltung der Kundenanforderungen überprüft werden. Die Testfälle sind unter **6.2 Testszenarien/Abnahmetest** definiert. Während der Abnahme werden die Testergebnisse protokolliert

6.2 Testszenarios/Abnahmetest

Test ID	Tx	Getestete Anforderungen	Axx	Priorität	<i>normal</i>
Beschreibung					
Vorbedingung					
Input Daten					
Ablaufbeschreibung		Aktion	Erwartung	Erfüllt	

Test ID	T1	Getestete Anforderungen	ANF23, ANF27	Priorität	<i>normal</i>
Beschreibung	Funktionsweise des E-Stopp				
Vorbedingung	Anlage befindet sich im Betriebszustand und beide Bänder bewegen sich. Beide E-Stopp sind herausgezogen.				
Input Daten	E-Stopp, Reset, Start-Taster				
Ablaufbeschreibung		Aktion	Erwartung	Erfüllt	
		E-Stopp FBM1 drücken	Beide FBM stehen still. Die Ampel geht aus		
		Reset und Start-Taster drücken	Die Anlage bleibt im Fehlerzustand		
		E-Stopp FBM1 herausziehen, Bänder freiräumen, Reset und Start-Taster drücken	Die Anlage wechselt in den Betriebszustand		
		E-Stopp FBM2 drücken	Beide FBM stehen still. Die Ampel geht aus		
		Reset und Start-Taster drücken	Die Anlage bleibt im Fehlerzustand		
		E-Stopp FBM2 herausziehen, Bänder freiräumen, Reset und Start-Taster drücken	Die Anlage wechselt in den Betriebszustand		

Test ID	T2	Getestete Anforderungen	ANF25, ANF26	Priorität	<i>normal</i>
Beschreibung	Wechsel der Betriebsmodi				
Vorbedingung	Anlage befindet sich im Ruhezustand				
Input Daten	Start- und Stopp-Taster				
Ablaufbeschreibung		Aktion	Erwartung	Erfüllt	
		Kurzes Drücken des Start-Tasters	Anlage wechselt in den Betriebszustand		
		Drücken des Stopp-Tasters	Anlage wechselt in den Ruhezustand		
		Langes Drücken des Start-Tasters	Anlage wechselt in den Service-Mode		
		Drücken des Stopp-Tasters	Anlage wechselt in den Ruhezustand		

6.3 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 waren. 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

7 Technische Schulden

Führen Sie bekannte technische Mängel Ihres Produktes auf. Zeigen Sie dem Kunden, dass Ihnen bewusst ist, was noch an offenen Punkten existieren. Damit zeigen Sie unter anderem, dass sie den Überblick über Ihr Projekt haben. Optional: Wieviel Zeit würde eine mögliche Beseitigung von einzelnen Baustellen umfassen?

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 muss und 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

Abkürzung	Bedeutung
Anlage	FBM1 und FBM2
FBM1	Erstes Förderband Modul (vorderes)
FBM2	Zweites Förderband Modul (hinteres)
EVM	EventManager

9.2 Abkürzungen

Abkürzung	Bedeutung
HAL	Hardware Abstraktion Layer.
FBM	Förderband Modul (Eine gesamte Anlage mit Beaglebone und Hardware)
FB	Förderband eines FBM
Production-Mode	Die Anlage sortiert die Werkstücke.
WS	Werkstück, der auf die Anlage zur Sortierung gelegt wird
LB	Light-Barrier (Lichtschranke)
LB1	Light-Barrier Start
LB2	Light-Barrier Weiche
LB3	Light-Barrier Ende
LB4	Light-Barrier Rutsche
BuM	Bohrung und Metal
BoM	Bohrung ohne Metal
MD	Metal Detector
HM	Höhen Messsensor

9.3 **Abbildungsverzeichnis**

Optional: Ein gutes Dokument beinhaltet auch ein Abbildungsverzeichnis. Wir in unserem Praktikumsumfeld benötigen es nicht zwingend.