Requirements and Design Documentation

Software Engineering 2

Department Informatik - HAW Hamburg

SE2P – Praktikum – SS13

Bani, Helmi, 1783382, [Helmi.Bani@haw-hamburg.de](mailto:Helmi.Bani@haw-hamburg.de)

Taieb, Edris,

Feutat Tamou, steve herve, 1888921, steveherve.feutattamou@haw-hamburg.de

Änderungshistorie:

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Datum** | **Anmerkungen** |
| 001 | Steve Feutat | 04.04.2013 | Meilestone 1 |
| 002 | Steve Feutat | 14.04.2013 | Meilstone 2 |
| 00.3 | Helmi Bani | 07.05.2013 | Meilstone 3 |

Inhalt

Motivation 3

Randbedingungen 3

Entwicklungsumgebung 3

Werkzeuge 3

Sprachen 3

Requirements und Use Cases 3

Anforderungen 3

Use-Case-Diagramm 3

Design 4

System Architektur 4

Datenmodell 4

Verhaltensmodell 4

Implementierung 4

Algorithmen 4

Patterns 5

Mapping Rules 5

Testen 5

Unit Test/Komponenten Test 5

Integration Test/System Test 5

Regressionstest 5

Abnahmetest 5

Testplan 5

Testprotokolle und Auswertungen 6

Projektplan 6

Verantwortlichkeiten 6

PSP und Zeitplan 6

Lessons Learned 6

Glossar 7

Abkürzungen 7

Anhänge 7

1. Motivation

In diesem Projekt soll die Software-Architektur für eine Werkstück-Sortieranlage konzipiert und modelliert werden. Die Anlage besteht aus zwei Festo Förderband und jede Transferstrecke besitzt eine Einheit zur Höhenmessung, zur Detektion von Metall, sowie zum Auswurf von Werkstücken.

Jede Festo Transferstrecke wird von einem GEME Rechner gesteuert. Dafür sollen die digitalen Ein- und Ausgänge verwendet werden. Außerdem findet über die serielle Schnittstelle der GEME Rechner eine Kommunikation jener statt, um sich über die aktuelle Verfügbarkeit des anderen Transfermoduls zu erkundigen

**1.1. Skateholder**

* Kunde :

Prof Dr. Pareigis

* Projektausführung:

Helmi Bani

Taieb Edris

Steve Feutat

1. Randbedingungen
   1. Entwicklungsumgebung

* QNX Virtual Machine
* Förderbandsimulation SEAP 1.1
* Betriebssystem: QNX
* Hardware: Werkstück-Sortieranlage
  1. Werkzeuge
* Github(als Fileserver)
* Projektverwaltung: Redmine
* Erstellung von Diagrammen: Visual Paradigm
* Microsoft Project
  1. Sprachen
* Programmiersprache: C++
* Bibliotheken: ioaccess

1. Requirements und Use Cases
   1. Anforderungen

* **Anwendungsfall: Werkstück einlegen Band 1**

Werkstücke werden am Anfang von Band 1 eingelegt und vom System registriert.

Aktoren**:** Kunde oder Benutzer

Vorbedingungen**:** der Anfang von Band 1 ist frei

* **Beschreibung des Ablaufs**
* ein Werkstück wird am Anfang der Band1 eingelegt
* das System prüft ob Anfang von Band1 frei ist
* wenn ja , prüft das System ob das Werkstück schon registriert ist
* wenn nein, bekommt das Werkstück ein ID und die Höhenmessung wird durchgeführt.(was wenn h<Flachstück Höhe?).
* das System prüft ob das Werkstück zu flach ist
* wenn nicht ,prüfe ob Bohrung nach oben zeigt
* wenn ja wird das Werkstück weiter an band 2 gereicht
* **Alternativer Ablauf**
* Band 1 ist noch nicht frei Bandstopp,Fehlermeldung
* das Stück hat schon ein ID,prüfe ob Typ vorhanden ist .wenn ja sortieren,
* wenn nicht prüfe ob Höhe 1 vorhanden ist,wenn zu flach sortieren
* wenn nicht weiter zu band 2 reichen.
* Werkstück mit Typ=flach sortieren
* wenn nein Band 1 anhalten und Werkstück umdrehen und Band wieder Anmachen

Nachbedingung**:** Werkstücke mit Typ zu flach sortiert

Ausnahmeablauf**:** 3a-Metallstück wurde falsch eingelegt: Bandstopp,Fehlermeldung

Ende:IDS und Höhe1 vorhanden für alle Werkstücke am Ende von Band 1.

* **Anwendungsfall: Werkstück umdrehen auf band 2**
* Werkstücke mit Metalleinsatz werden am Anfang von Band 2 umgedreht wenn die Bohrung nach oben zeigt.

Aktoren**:** Kunde oder Benutzer

Vorbedingungen**:** Metalleinsatz wurde erkannt, Bohrung zeigt nach oben und ID und Höhe 1 vorhanden es gibt nicht mehr als ein Werkstück auf Band 2

* **Beschreibung des Ablaufs**
* ein Werkstück wird am Ende der Band1 weitergereicht.
* das System prüft am Anfang von Band 2 ob ein Metalleinsatz vorhanden ist
* wenn ja , prüft das System ob die Bohrung nach unten zeigt
* wenn ja, bekommt das Werkstück der Typ mit Bohrung und Metalleinsatz und die Höhenmessung wird durchgeführt und das Stück sortiert.
* **Alternativer Ablauf**
* Typ ist Werkstück mit Bohrung ohne Metalleinsatz. die Höhenmessung wird durchgeführt und das Werkstück sortiert

Nachbedingung**:** Höhe 2 ist bei den Werkstücken gemessen.

Ausnahmeablauf**:** Metallstück wurde nicht von Band 1 weitergereicht also kein ID oder Höhe 1 vorhanden: Bandstopp,Fehlermeldung

Ende**:** alle Werkstücke sind sortiert

* **Anwendungsfall Anlage stoppen**

um Werkstücke umzudrehen wird Band2 oder band1 angehalten

Aktoren**:** Kunde oder Benutzer

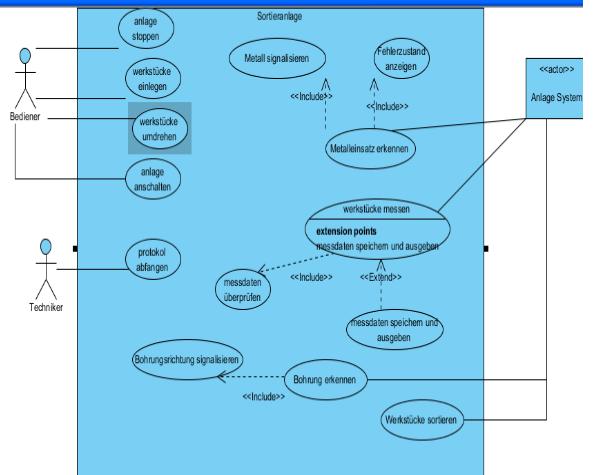
Vorbedingungen**:** Metalleinsatz wurde erkannt ,Bohrung zeigt nach oben .Am ende von Band 1 muss die Bohrung nach oben zeigen, die Anlage läuft.

* **Beschreibung des Ablaufs**
* die stopptaste für Band 1 oder Band 2 wird gedrückt
* das System unterbricht alle Aktivitäten auf dem Band.
* die gelbe leuchte blinkt
* **Ausnahmeablauf**
* Metallstück wurde nicht von Band 1 weitergereicht also kein Id oder Höhe 1 vorhanden :Bandstopp,Fehlermeldung
* **Alternativer Ablauf**
* Typ ist Werkstück mit Bohrung ohne Metalleinsatz. die Höhenmessung wird durchgeführt und das Werkstück sortiert

Nachbedingung**:** Höhe 2 ist bei den Werkstücke gemessen.

Ende: alle Werkstücke sind sortiert

* 1. Use-Case-Diagramm



1. Design

* Software Archtitektur :

Die Software erfasst die Zustände des Systems über Sensoren und aktualisiert seine inneren Variablen, das sogenannte Abbild.

Das Abbild wird zur Ermittlung von Steuerungswerten ausgewertet.

Anhand der ermittelten Werte können die Aktoren angesprochen und gesteuert werden.

Die Hardware Abstraktion Schicht, kurz HAL, dient zur Kapselung der Hardware.

Die HAL stellt Funktionen bereit, mit denen die Hardware angesprochen werden kann.

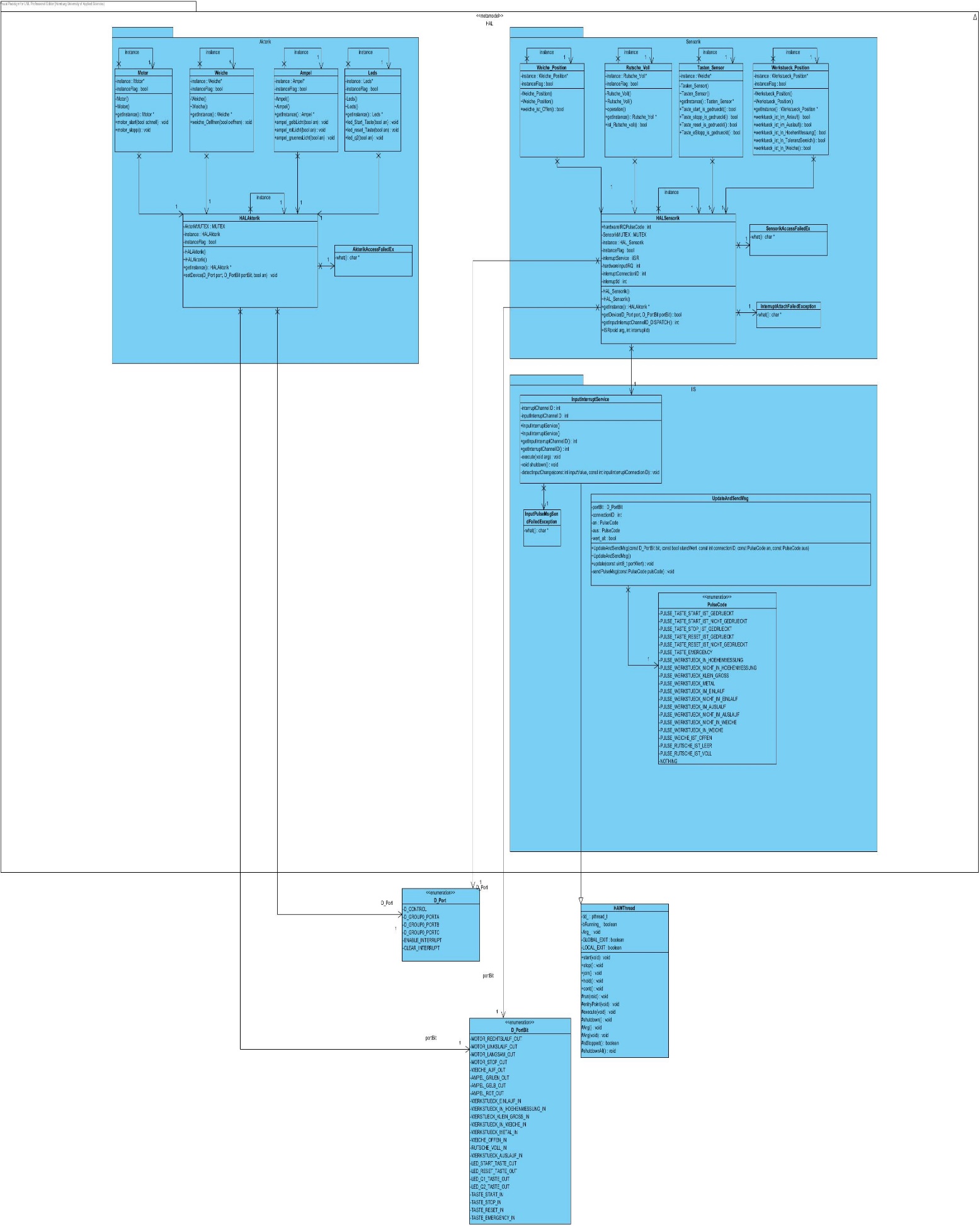
Die  Aktorik wird direkt vom Automaten angesprochen. Die Sensorik spricht die

Automaten über eine selbst definierte ISRs mittels Pulse Messages an.

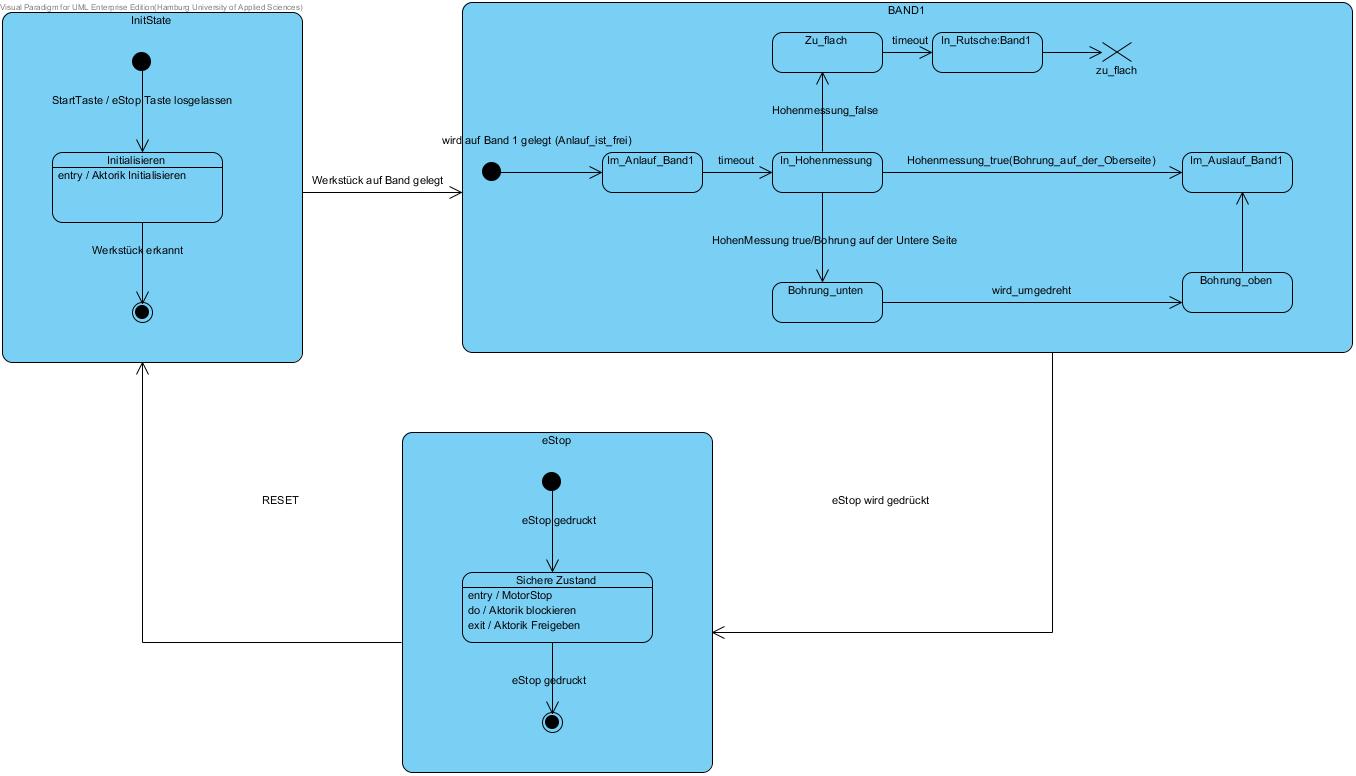
* 1. System Architektur



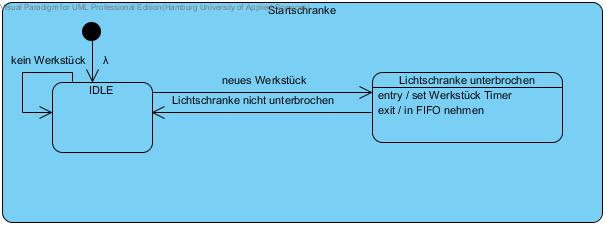
* 1. Datenmodell



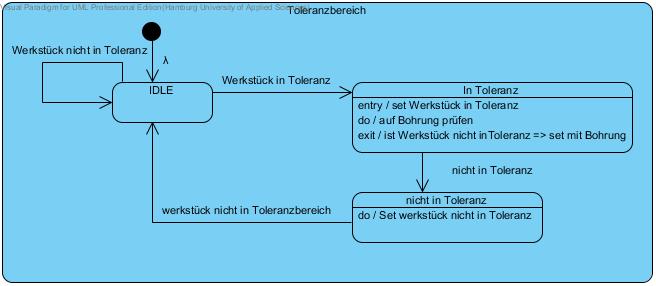
* 1. Verhaltensmodell



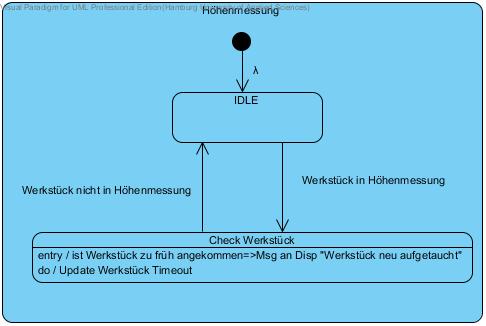
**4.3.1** **Startschranke**



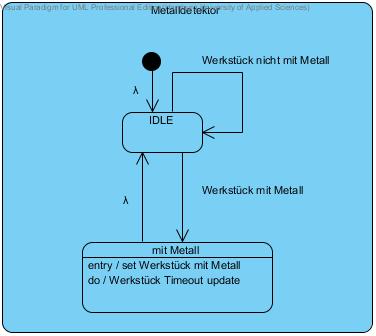
**4.3.2 Toleranzbereich**



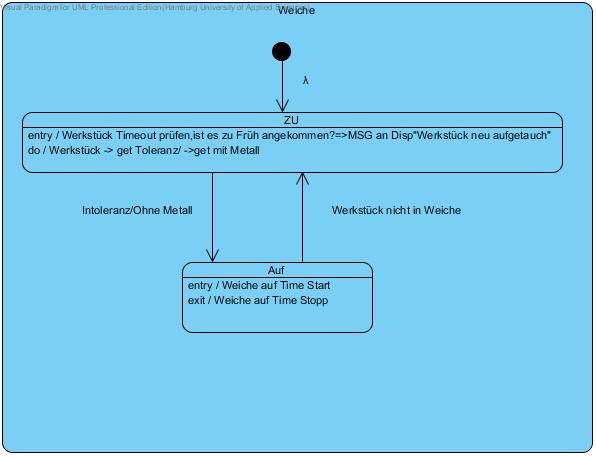
**4.3.3 Höhenmessung**



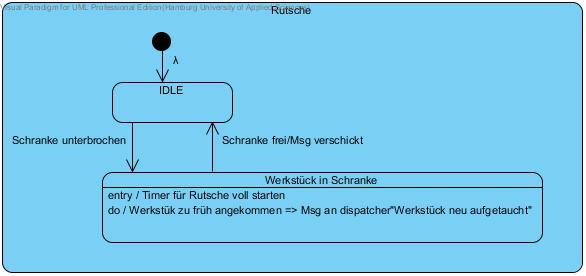
**4.3.4 Metalldetektor**



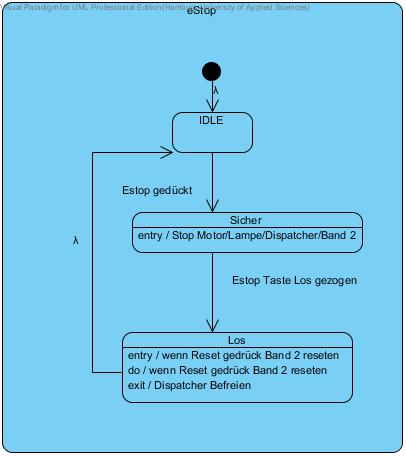
**4.3.5 Weiche**



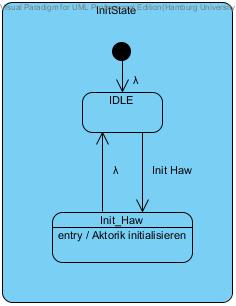
**4.3.6 Rutsche**



**4.3.7 eStop**



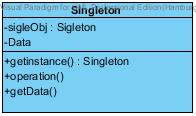
**4.3.9 InitState**



1. Implementierung
   1. Algorithmen

Wichtige Algorithmen, die Sie hier benutzt haben.

* 1. Patterns
* **Singelton Pattern**



**Grundidee:**

Sicherstellen, dass zu einer Klasse genau angelegt wird.

**Singleton:**

\*definiert eine statische Methode getinstance(), mit der ein Klient auf das einzige Objekt zu dieser Klasse zugreifen kann. Und ist für die Erzeugung des einzigen Objects zu sich selbst zuständig.

**Anwendbarkeit:**

Wenn genau ein Objekt zu einer Klasse geben muss, auf das Die Klienten über eine festgelegte Methode (getinstance()) zugreifen sollen.

**Mögliche Realisierung:**

**Klasse: Motor**

*Motor\* Motor::instance;*

*Motor\* Motor::getInstance(){*

*if(!instanceFlag){ //Existiert bereits eine instance?*

*instance = new Motor(); //nicht-> erzeuge eine neue instance*

*instanceFlag = true;*

*return instance;*

*}else{*

*return instance ; //sonst Aktuelle instance zurück geben*

*}*

*}*

Aufruf:

*Motor \*motor\_Hal = Motor::getInstance();*

**Vorteile:**

Zugriffskontrolle kann realisiert werden.

Das Singleton kann durch Unterklassenbildung spezialisiert werden.

Die Einzelinstanz muss nur erzeugt werden, wenn sie benötigt wird.

* **State Pattern**

**Grundidee:**

Einem Objekt ermöglichen, sein Verhalten zu ändern, wenn ein interner Zustand sich ändert, so als ob das Objekt seine Klasse gewechselt hätte.

**Teilnehmer:**

Context:

Definiert die Schnittstelle, die für Klienten von Interesse ist.

enthält ein Objekt zu einer State-Unterklasse die den aktuellen zustand definiert .

State:

Definiert die Schnittstelle zur Kapselung des Verhaltens eines Automat.

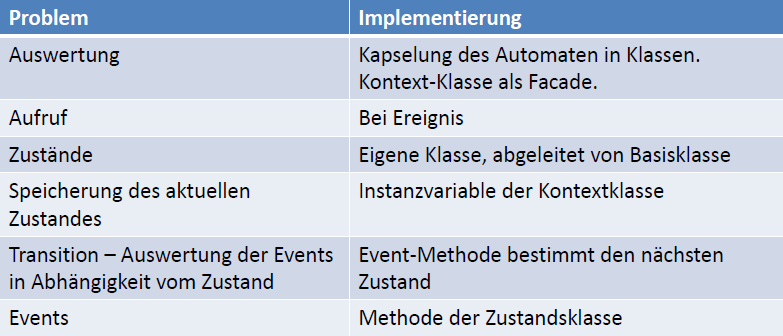
StateA, StateB :

je Unterklasse implementiert ein Verhalten das einem Zustand des Context-Objekts zugeteilt ist

**Anwendbarkeit:**

Dieses Pattern kommt bei der Implementierung der Statemachines zum Einsatz und ist ein Software-Verhaltens-Pattern.

Das Entwurfsmuster State wird verwendet wenn, das Verhalten eines Objekts von einem Zustand abhängt, und es zur Laufzeit in Abhängigkeit von einem Zustand ändern muss.

* Wo können Probleme auftreten und wie sind sie durch das State-Pattern implementiert?
* Quelle: Emil SE2 Vorlesung:

Es wird verwendet, wenn es für die Routine einer Statemachine, unterschiedliches Verhalten geben kann. Statemachines können so „sauberer“ von einem State in den anderen wechseln, ohne dabei auf große monolithisch bedingte Statements zurückgreifen zu müssen.

Auszug aus dem Code:

*#ifndef ASBD1\_CONTEXT\_H\_*

*#define ASBD1\_CONTEXT\_H\_*

*#include"Asbd1\_IDLE.h"*

*#include"Asbd1\_IDLE.cpp"*

*#include"Asbd1\_WkAuslauf.h"*

*#include"Asbd1\_WkAuslauf.cpp"*

*class Asbd1\_Context {*

*friend class Asbd1\_Zustand;*

*public:*

*Asbd1\_Context();*

*void getCurrentState();*

*void start();*

*void reset();*

*private:*

*Asbd1\_Zustand\* zustand;*

*};*

*#endif /\* ASBD1\_CONTEXT\_H\_ \*/*

Beispiel für einen Statewechsel:

*void Asbd1\_WkAuslauf::z\_Reset(){new (this) Asbd1\_IDLE;}*

* 1. Mapping Rules

Wichtige Mapping Rules, die Sie benutzt haben, z.B. um aus Ihrem Design entsprechenden Code zu erstellen.

1. Testen

Machen Sie sich Gedanken über Unit-Test, Komponententest, Integrationtest, Systemtest, Regressionstest und Abnahmetest.

* 1. Unit Test/Komponenten Test

|  |  |  |  |
| --- | --- | --- | --- |
| Testfall | Beschreibung | Vorbedingung | Erfolgsfall |
| 1. | Ersten Werkstuck einlegen | Band gestoppt | Band startet |
| 2. | Werkstück in der Mitte einlegen | Band stoppt | Band Startet nicht |
| 3. | Werkstück in der Mitte einlegen | Band läuft | Band Stop  Leuuchte: Rot  Fehlerzustand Aktualisiert |
| 4. | Werkstück einlegen | Band läuft  Rutsche Voll | Band Stop  Leuchte Rot  Fehlerzustand Aktualisiert |
| 5. | Werkstück aus dem Band entfernen | Band läuft | Nach einer bestimmte Zeit Band Stop  Fehler Aktualisiert |
| 6. | Werkstück zu Flach | Band läuft  Rutsche nicht voll | Werkstück wird aussortiert |
| 7. | Werkstück Bohrung nach oben und Metalleinsatz einlegen | Band läuft  Rutsche nicht voll | Band läuft weiter |
| 8. | Werkstuck Bohrung nach oben und Metall Einsatzeinlegen | Band lauft  Rutsche nicht voll  Band2 frei ist | Weiche von Band Öffnen |
| 9. | Werkstuck Bohrung nach oben ohne Metalleinsatz einlegen | Band1 lauft  Rutsche nicht voll  Band2 frei | Weiche von Band1 öffnen Band startet Weiche von Band2 öffnen |
| 10. | Werkstuck Bohrung nach unten und Metalleinsatz einlegen | Band1 lauft  Rutsche nicht voll  Band2 frei  Band1 wird angehalten  Bediener dreht den Werkstuck um wenn es am Ende vom Band ist | Weiche von Band1 öffnen  Band2 startet  Leuchte : gelb  Werkstuck wird aussortiert wenn er am Ende |
| 11. | Werkstuck Bohrung nach unten und Metalleinsatz einlegen | Band1 lauft  Rutsche nicht voll  Band2 frei  Werkstuck wird am Messung Ende von Band1 nicht umgedreht | Weiche von Band1 öffnen  Band2 startet  Fehler signalisieren nach |

* 1. Integration Test/System Test

Test Szenarien mit beiden Laufbändern

* 1. Regressionstest

Die Anlage wird in verschiedenen Anwendungsbereich eingesetzt und deswegen brauchen wir ein Verfahren zum Testen der Hauptfunktionen. Damit neue Anforderungen zur Steuerung der Anlage angepasst werden können und mit minimalen Fehler risiko implementiert werden können.

Anhand der unten angegebenen Grafik haben wir dieses Testverfahren überlegt und wir werden versuchen es im System zu integrieren soweit es uns die Möglichkeit es uns erlaubt.

Die Durchsetzung des Tests wird nach jedem Meilenstein Ausgeführt, und gleich danach wird einen Test für den nächsten Meilenstein vorbereitet.

Die Wichtigste Aktivität ist diesem Modell ist die Testfallermittlung, mit denen der Test durchgeführt werden soll.

Es wir einen Umfang der Prüfung nach jeden Meilenstein bestimmt. In der Regel wird die Konzentration auf die Funktionen im Code und gleichzeitig die Beobachtung auf der Anlage.

Die Auswertung wird regelmäßig dokumentiert und falls Fehlersituation auftauchen wird dieser natürlich behoben und in dem nächsten Meilenstein berücksichtigt.

Die Funktionale Spezifikation ist eine Möglichkeit parallel zum Programm zur Ermittlung von Ziele die das Testergebnis bestimmen und dient als Hilfestellung für die Auswertung von Den Tests.

Szenarien:

* Bei Übertragung des Codes an das Laufband, dürfen keine Fehler auftreten.
* Ampelsteuerung soll zu jedem Zeitpunkt möglich sein.
* Events der Sensorik werden korrekt zugeordnet.
* Stop- / Starttaster funktionieren korrekt
* E-Stopp funktioniert korrekt
* Ansteuerung der Hardware funktioniert wie implementiert
* Kommunikation der GEME-Rechner funktioniert fehlerfrei

Hardware (Laufband) ist korrekt aufgebaut und ausgerichtet

* 1. Abnahmetest

Leiten Sie die Abnahmebedingungen aus den Kunden-Anforderungen her.

Geben Sie an, welche Anforderungen erfolgreich und eventuell nicht erfolgreich implementiert sind.

* 1. Testplan

Der Testplan wurde aus übersichtsgründen nicht in dieses Dokument integriert. Es liegt als Anhang bei

* 1. 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

1. Projektplan
   1. Verantwortlichkeiten

Verantwortliche innerhalb des Projekts (Projektleiter, Tester, Implementierer, etc.) benennen.

* 1. PSP und Zeitplan

Projektstrukturplan, Ressourcenplan, Zeitplan, Abhängigkeiten von Arbeitspaketen,

Eventueller Zeitverzug, etc.

1. Lessons Learned

Was lief gut, was lief schlecht in diesem Projekt (technisch und organisatorisch)?

Was haben Sie gelernt?

Weitere Anregungen und Erkenntnisse durch das Projekt.

Glossar

Eindeutige Begriffserklärungen

Abkürzungen

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

Anhänge

Auflistung aller Artefakten dieses Projekts

1. Alle Modell-Dateien (Visual Paradigm, Petri-Netze etc.)
2. Source Code und Code Dokumentationen (z.B. Doxygen)
3. Test Protokolle
4. Meeting Protokolle
5. Projektplan
6. etc.