Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Franziska Ehlers

Handling von optischen Fehlteilen einer flexiblen Fertigung nach dem Industrie 4.0 Leitbild

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Franziska Ehlers

Handling von optischen Fehlteilen einer flexiblen Fertigung nach dem Industrie 4.0 Leitbild

Masterthesis eingereicht im Rahmen der Masterprüfung
im Masterstudiengang Automatisierung
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. -Ing. Florian Wenck
Zweitgutachter : Prof. Dr.-Ing. Jochen Maaß

Abgegeben am 21. August 2018

Franziska Ehlers

Thema der Masterthesis

Handling von optischen Fehlteilen einer flexiblen Fertigung nach dem Industrie 4.0 Leitbild

Stichworte

Ereignisdiskrete Systeme (DES), Supervisory Control Theory (SCT), Steuerungssynthese, SPS, Automaten, Generatoren, DESTool, Modellfabrik, Assistenzsystem, Industrie 4.0, Handarbeitsplatz, Codelesegerät

Kurzzusammenfassung

Diese Arbeit befasst sich mit dem Handling von optischen Fehlteilen einer flexiblen Fertigung unter Berücksichtigung des Industrie 4.0 Leitbilds. Es wird ein optisches Identifikationssystem in die bestehende Modellfabrik integriert, welches eine Klassifikation der optischen Fehlteile ermöglicht. Für die Reparatur der Fehlteile wird ein Handarbeitsplatz mit Assistenzsystem eingerichtet. Die Steuerung für das ereignisdiskrete Handling von optischen Fehlteilen wird nach der Supervisory Control Theory synthetisiert. Abschließend wird anhand verschiedener Testszenarien geprüft, ob die Steuerungsaufgabe erfüllt wird.

Franziska Ehlers

Title of the paper

Handling of optical defective parts of a flexible manufacturing according to the industry 4.0 basic principle

Keywords

Discrete event systems, Supervisory Control Theory (SCT), Control synthesis, PLC, Automata, DESTool, model factory, assistance system, Industry 4.0, manual workstation, Code reader system

Abstract

This thesis deals with the handling of optical defective parts of a flexible manufacturing under consideration of the industry 4.0 basic principle. A visual identification system which classifies parts that are optically defective is integrated into the existing model factory. For the repair of these defective parts, a manual workstation with an assistance system is set up. The control for the discrete event handling of optical defective parts is synthesized according to the Supervisory Control Theory. Finally, various test scenarios are used to check whether the control task is fulfilled.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Symbol- und Abkürzungsverzeichnis

δ ..... Zustandsübergangsfunktion
$\overline{L}$ ..... Präfix-Hülle der Sprache L
Σ ..... Ereignisalphabet
σ ..... Ereignis
Σ^* ..... Kleene-Hülle
Σ_c ..... Menge der steuerbaren Ereignisse
Σ_{ce} ..... Menge der gemeinsamen Ereignisse
Σ_{pe} ..... Menge der privaten Ereignisse
Σ_{uc} ..... Menge der nicht steuerbaren Ereignisse
ε ..... Leerer String
A ..... Adjazenzmatrix
$Ac(G)$ ..... Erreichbarer Teil des Generators G
$cat(s,t)$ ..... Konkatenation der Strings s und t
$CoAc(G)$ ..... Ko-erreichbarer Teil des Generators G
G ..... Generator
K ..... Formale Spezifikation
$K^{\downarrow c}$ ..... Infimale präfix-geschlossene steuerbare Obersprache
$K^{\uparrow c}$ ..... Supremale steuerbare Teilsprache
L ..... Formale Sprache
$L(G)$ ..... Reguläre Sprache von G
$L_m(G)$ ..... Markierendes Verhalten von G
P ..... Natürliche Projektion
P^{-1} ..... Inverse natürliche Projektion
R ..... Akzeptor
S ..... Supervisor/Steuerung
s ..... String
$S(s)$ ..... Steuereingriff
S/G ..... Gesteuertes System

$Trim(G)$ ..... Erreichbarer und ko-erreichbarer Teil von G
X ..... Zustandsmenge
x ..... Zustand
x_0 ..... Anfangszustand
X_m ..... Menge der markierten Zustände
x_m ..... Markierter Zustand
A ..... Ausgang der SPS
AWL ..... Anwendungsliste
BSCP-NB ..... Basic Supervisory Control Problem-Nonblocking
c ..... Controllable
DES ..... Discrete event system
DFA ..... Deterministic Finit-state Automata
DI ..... Digital Input
DO ..... Digital Output
FUP ..... Funktionsplan
HAW ..... Hochschule für Angewandte Wissenschaften
KOP ..... Kontaktplan
MSCP ..... Modular Supervisory Control Problem
PLC ..... Programmable logic controller
RFID ..... Radio-frequency identification
SCL ..... Structured Control Language
SCT ..... Supervisory Control Theory
SPC ..... Strict Product Composition
SPS ..... Speicherprogrammierbare Steuerung
ST10 ..... Station 10
ST20 ..... Station 20
ST30 ..... Station 30
ST40 ..... Station 40
ST50 ..... Station 50
ST60 ..... Station 60
SYPC ..... Synchronous Product Composition
TIA ..... Totally Integrated Automation
uc ..... Uncontrollable
VR ..... Virtual Reality

1. Einführung

1.1. Einführung und Motivation

Ein Teilgebiet der Automatisierungstechnik stellt die Steuerungstechnik dar. Diese umfasst den Entwurf und die Realisierung von Steuerungen. In der Praxis werden diese üblicherweise intuitiv entworfen. Dies kann vor allem bei größeren industriellen Anlagen zu Unübersichtlichkeit und Fehlern führen. Der Test der entworfenen Steuerung erfolgt meistens erst in der Inbetriebnahmephase auf der zu steuernden Anlage. Die Überprüfung der Korrektheit erfolgt dabei empirisch durch verschiedene Testszenarien. Eine Korrektur von Fehlfunktionen ist häufig mit großem Aufwand verbunden.

Dies kann mit Hilfe von synthetisierten Steuerungen umgangen werden. Die modellbasierte Steuerungssynthese basiert auf formalen Beschreibungen des ungesteuerten Verhaltens eines Systems und auf formalen Spezifikationen, welche das geforderte Systemverhalten beschreiben. Die Steuerungssynthese liefert im Gegensatz zum intuitiven Entwurf ein beweisbar korrektes Ergebnis. Synthetisierte Steuerungen haben in den letzten Jahren in der Praxis zunehmend an Bedeutung gewonnen. In dieser Arbeit wird eine solche Steuerungssynthese für das ereignisdiskrete Handling von optischen Fehlteilen einer flexiblen Montageanlage durchgeführt.

Die Umsetzung innovativer Industrie 4.0 Konzepte ist derzeit bei vielen Unternehmen im Gespräch. Vor allem im Hinblick auf die Rolle des Menschen in einer Fabrik ist davon auszugehen, dass durch den technischen Fortschritt sich dessen Aufgaben- und Anforderungsspektrum verändert. Dabei wird er ein erweitertes Aufgaben- und Verantwortungsspektrum übernehmen. Daher gewinnt die Umsetzung von Unterstützungssystemen zunehmend an Bedeutung.

1.2. Zielsetzung

Im Fachbereich Automatisierungstechnik der Hochschule für angewandte Wissenschaften Hamburg (HAW) wurde im Jahr 2016 eine Modellfabrik installiert. Diese stammt von der

Firma Köster Systemtechnik GmbH und bildet einen vollautomatisierten Montage- und Prüfprozess einer realen Fabrik nach. In dieser Modellfabrik werden als Produkt Relaiskarten im Gehäuse mit Deckel hergestellt. In dieser Masterarbeit ist für diese Anlage die Handhabung von optischen Fehlteilen nach dem Industrie 4.0 Leitbild zu realisieren. Die Aufgabenstellung beinhaltet die Integration des vorhandenen Codelesegeräts der Firma Siemens, sowie den Aufbau eines Handarbeitsplatzes und den Entwurf einer Steuerung für das Handling von optischen Fehlteilen.

Abbildung 1.1.: Gesamtansicht der Modellfabrik

Mit Hilfe des Codelesegeräts sollen Werkstücke, welche optisch nicht in Ordnung sind, erkannt und aussortiert werden. Dabei gilt es verschiedene optische Fehler zu unterscheiden. Die erkannten Fehlteile sollen für die weitere Bearbeitung aus dem Prozess hinausgeführt werden. Für diese Bearbeitung wird ein Handarbeitsplatz mit Assistenz, in Anlehnung an Industrie 4.0 Arbeitsplätze, eingerichtet. An diesem erfolgt die Reparatur der optischen Fehlteile durch einen Montagemitarbeiter.

Für das Aussortieren der Fehlteile soll eine modellbasierte Steuerung entworfen werden. Die Steuerungssynthese wird nach der Supervisory Control Theory durchgeführt. Dafür sind geeignete Ansätze, Beschreibungsformen und Entwicklungstools auszuwählen. Mit Hilfe eines automatischen Codegenerators erfolgt die Realisierung der modellbasierten Steuerung. Der lauffähige Programmcode soll auf einer vorhandenen Siemens Steuerung realisiert und in das vorhandene Steuerungsprogramm integriert werden. Im Anschluss an die Implementierung soll die Steuerung getestet werden.

1.3. Gliederung

Diese Arbeit umfasst acht Kapitel. Nach der Einleitung in diesem Kapitel werden für das weitere Verständnis die für diese Arbeit erforderlichen Grundlagen in Kapitel 2 eingeführt. Dazu gehören unter anderem die Grundlagen zu ereignisdiskreten Systemen, der Supervisory Control Theory, der verwendeten Software, dem Codelesegerät und Assistenzsystemen nach dem Industrie 4.0 Leitbild. In Kapitel 3 wird die Modellfabrik vorgestellt. Die Konzeption des Handarbeitsplatzes, sowie die Auswahl der Beschreibungsform, des Entwurfsansatzes und des Entwicklungstools erfolgt in Kapitel 4. Kapitel 5 beinhaltet die Modellbildung des ungesteuerten Verhaltens der Strecke und der Spezifikationen. Zusätzlich wird in diesem Kapitel die Steuerungssynthese durchgeführt. Die Realisierung und Implementierung werden in Kapitel 6 beschrieben. In Kapitel 7 wird die Funktion der entworfenen Steuerung mit Hilfe verschiedener Testszenarien überprüft. Abschließend wird die Arbeit in Kapitel 8 zusammengefasst und es wird ein Ausblick gegeben.

2. Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen dieser Masterthesis behandelt. Abschnitt 2.1 befasst sich mit ereignisdiskreten Systemen und deren Beschreibungsformen, sowie der Supervisory Control Theory. Die für den Steuerungsentwurf verwendete Software wird in Abschnitt 2.2 vorgestellt. Abschnitt 2.3 beschreibt das zum Einsatz gebrachte optische Identifkationssystem und seine Funktionen. Im abschließenden Abschnitt 2.4 werden die Anforderungen an einen Handarbeitsplatz nach dem Industrie 4.0 Leitbild erläutert.

2.1. Ereignisdiskrete Systeme

In diesem Unterkapitel werden die Grundlagen, Begriffe und Konzepte von ereignisdiskreten Systemen (Engl. Discrete-Event System, DES) und der Supervisory Control Theory (SCT) eingeführt.

2.1.1. Begriffe und Definitionen

Allgemein beschreibt ein System einen Zusammenschluss von miteinander verknüpften Elementen, die gemeinsam eine Aufgabe erfüllen. Diese werden auf Grund ihrer unterschiedlichen Eigenschaften in Klassen unterteilt. Im Rahmen dieser Arbeit wird nur die Klasse der ereignisdiskreten Systeme betrachtet. Ein DES ist nach [24, S.16] wie folgt definiert: *„Ein ereignisdiskretes System ist ein dynamisches System mit einem diskreten Zustandsraum, dessen Zustände sich spontan durch das asynchrone Auftreten von Ereignissen über die Zeit ändern. Ein Ereignis ist eine Erscheinung oder eine Aktion ohne zeitliche Dauer“*.

Ein DES wird als dynamisch bezeichnet, da die Wirkung eines auftretenden Ereignisses vom aktuellen Zustand des Systems abhängt. Die Zustandsübergänge eines DES liegen den auftretenden Ereignissen zu Grunde, daher sind DES ereignis- und nicht zeitgesteuert. Viele Systeme sind nicht von Natur aus ereignisdiskret, sie können jedoch häufig durch Abstraktion als solche betrachtet werden [10].

Ereignisdiskrete Systeme werden meist durch Modelle dargestellt. Dabei unterscheidet man zwischen logischen, zeitbewerteten, stochastischen und zeitbewerteten stochastischen Modellen. Logische Modelle beschränken sich darauf das Verhalten des DES durch eine vom System generierte Ereignisfolge zu beschreiben. Dabei ist nur die Reihenfolge der Ereignisse relevant, die Zeitpunkte des Auftretens der Ereignisse werden nicht berücksichtigt. Werden die Zeitpunkte der auftretenden Ereignisse mit in die Beschreibung einbezogen, spricht man von zeitbewerteten DES. Stochastische DES beziehen, ausgehend von den logischen DES, die Auftrittswahrscheinlichkeit bestimmter Ereignisse mit ein. Durch Kombination der zeitbewerteten und der stochastischen Darstellung erhält man ein zeitbewertetes stochastisches DES.

In dieser Arbeit werden nur logische ereignisdiskrete Systeme betrachtet.

2.1.2. Sprachen

Für die Beschreibung von ereignisdiskreten Systemen werden reguläre Sprachen angewendet. Diese sind Teil der formalen Sprachen. Es werden zunächst einige Grundlagen der Sprachentheorie eingeführt. In der nachfolgenden Auflistung werden die grundlegenden Begriffe, Definitionen und Operationen aufgeführt. Diese sind aus [6] und [29] entnommen.

- Ein *Symbol* σ ist ein Zeichen und kann ein Ereignis darstellen
- Ein *Alphabet* Σ beschreibt eine endliche nichtleere Menge von paarweise verschiedenen Symbolen
- Ein *String s* ist eine endliche Sequenz von Symbolen $\sigma_1\sigma_2\sigma_3 \ldots \sigma_k$
- ε ist eine Sequenz ohne Symbole mit $\varepsilon \notin \Sigma$, $\varepsilon \neq \varnothing$
- Σ^+ beschreibt die Menge aller möglichen Strings bzw. Ereignisfolgen aus Σ, ausgenommen ε
- Die *Kleene-Hülle* Σ^* von Σ ist die Menge aller möglicher Ereignisfolgen inklusive ε und wird definiert als $\Sigma^* = \{\varepsilon\} \cup \Sigma^+$
- Die Konkatenation *cat* ist definiert als $cat(\varepsilon, s) = cat(s, \varepsilon) = s, s \in \Sigma^*$ und $cat(s, t) = st$, mit $s, t \in \Sigma^+$
- Ein *Präfix* von s ist $\overline{s} = t$, wenn $s = cat(t, \sigma) = t\sigma$ mit $t \in \Sigma^*$ und $\sigma \in \Sigma$
- Ein *Suffix* eines Strings *s* ist *t* ab σ, wenn $s = u\sigma t$ mit $u \in \Sigma^*$, $\sigma \in \Sigma$, $t \in \Sigma^*$
- $|s|$ beschreibt die Länge eines Strings und ist definiert als $|\varepsilon| = 0$, $|s| = k$ für $s = \sigma_1 \ldots \sigma_k \in \Sigma^+$

- t ist *Substring* von s für $s = utv$ mit $u, t, v \in \Sigma^*$

Eine formale Sprache L über einem Alphabet Σ ist definiert als eine beliebige Teilmenge $L \in \Sigma^*$. Als reguläre Sprachen werden formale Sprachen bezeichnet, die von endlichen Automaten erkannt werden. Da es sich bei Sprachen um Mengen handelt, können die grundlegenden Operationen der Mengenlehre, wie Vereinigung, Schnitt, Differenz, Komplement und symmetrische Differenz, auf Sprachen angewendet werden. Weitere Operationen auf Sprachen sind:

- *Konkatenation*: Die Konkatenation zweier Sprachen L_1, $L_2 \subseteq \Sigma^*$ enthält die zusammengesetzten Strings aus L_1 und L_2 und ist definiert als:

$$L_1 L_2 = \{s \in \Sigma^* | (s = s_1 s_2) \wedge (s_1 \in L_1) \wedge (s_2 \in L_2)\} \tag{2.1}$$

- *Präfix-Hülle*: In der Präfix-Hülle $\overline{L}$ einer Sprache $L \subseteq \Sigma^*$ sind alle Präfixe aller Strings $s \in L$ enthalten. Sie ist definiert als:

$$\overline{L} = \{s \in \Sigma^* | \exists t \in \Sigma^* (st \in L)\} \tag{2.2}$$

  Eine Sprache ist präfix-geschlossen, wenn $L = \overline{L}$ gilt.

- *Kleene-Hülle*: Die Kleene-Hülle L^* enthält alle Strings, die durch Konkatenation einer endlichen Anzahl von Strings der Sprache L entstehen. Außerdem enthält sie den leeren String ε. Für $L \subseteq \Sigma^*$ ist L^* definiert als:

$$L^* = \{\varepsilon \cup L \cup LL \cup LLL \cup \ldots\} \tag{2.3}$$

Unter den Operationen Vereinigung, Schnitt, Komplement, Konkatenation, Bildung der Präfix- und der Kleene-Hülle sind die regulären Sprachen abgeschlossen.

2.1.3. Automaten als Beschreibungsform

Eine Möglichkeit zur Beschreibung von ereignisdiskreten Systemen sind endliche deterministische Automaten (Engl. Deterministic Finit-state Automata, DFA). Mit Hilfe von DFAs kann die Systemstruktur eines logischen DES modelliert werden. Der Begriff des Generators wurde von Ramadge und Wonham im Rahmen ihrer Arbeit zur SCT [26, 28] eingeführt. Im Gegensatz zur ursprünglichen Definition von DFAs als passive Erkenner (Akzeptoren) von Sprachen wird ein Generator als aktive Komponente interpretiert, welche spontan Ereignisse erzeugt, die zu Zustandsübergängen führen können. Ein DFA hingegen überprüft, ob eine gegebene Ereignisfolge zur Sprache des Automaten gehört und deswegen akzeptiert oder nicht akzeptiert wird.

Ein Generator wird durch das Fünftupel [29]

$$G = (X, \Sigma, \delta, x_0, X_m) \tag{2.4}$$

beschrieben. Dabei stellt die Menge X die endliche Zustandsmenge des Generators dar. Die Menge der möglichen Ereignisse ist in Σ angegeben. Mit Hilfe der Zustandsübergangsfunktion δ wird das dynamische Verhalten des Generators beschrieben. Diese ist definiert als

$$\delta : X \times \Sigma \to X. \tag{2.5}$$

Die Zustandsübergangsfunktion δ weist jedem Zustand $x \in X$ einen Folgezustand $x' \in X$ zu, der durch das Auftreten eines Ereignisses $\sigma \in \Sigma$ erreicht wird. Es gilt

$$x' = \delta(x, \sigma), x' \in X. \tag{2.6}$$

Existiert eine Zustandsübergangsfunktion für das Ereignis σ im aktuellen Zustand, wird die Schreibweise $\delta(x, \sigma)!$ verwendet. $\neg\delta(x, \sigma)!$ gibt an, dass die Zustandsübergangsfunktion im aktuellen Zustand nicht definiert ist. Für die Erweiterung der Zustandsübergangsfunktion auf Strings gilt für $s \in \Sigma^*$ und $\sigma \in \Sigma$

$$\delta(x, \varepsilon) = x, \tag{2.7}$$

$$\delta(x, s\sigma) = \delta(\delta(x, s), \sigma), \tag{2.8}$$

unter der Voraussetzung, dass $x' := \delta(x, s)!$ und $\delta(x', \sigma)!$.

Das erzeugte Ereignis σ muss nicht zwingend zu einem Zustandsübergang führen, dies wird als Schlinge bezeichnet. Der Generator würde dann nach dem Auftreten des Ereignisses im aktuellen Zustand verweilen. Bei Generatoren handelt es sich um deterministische Automaten, da jedem Zustand x genau ein Folgezustand x' zugeordnet ist.

Der Anfangszustand x_0 gibt an, in welchem Zustand der Generator sich anfangs befindet. X_m gibt die Menge der markierten Zustände an. Dies sind Zustände, die zum Ziel der Anwendung führen. Dabei gilt $X_m \subseteq X$.

In Abbildung 2.1 ist die grafische Darstellung eines Generators zu sehen. Die Zustände werden durch Kreise repräsentiert. Zeigt ein Pfeil auf einen Zustand, ist dies der Anfangszustand. Die markierten Zustände werden durch einen doppelten Kreis hervorgehoben. Zustandsübergänge werden durch die gerichteten Kanten und deren Beschriftung dargestellt.

Neben der grafischen Darstellung des Generators gibt es auch die Möglichkeit, ihn als *Adjazenzmatrix* darzustellen [13]. In der Adjazenzmatrix A werden alle Zustandsübergänge

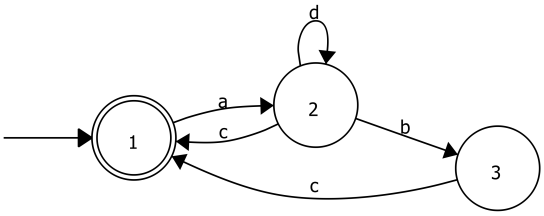


Abbildung 2.1.: Generator mit $X = \{1, 2, 3\}$, $\Sigma = \{a, b, c, d\}$, $x_0 = 1$ und $X_m = \{1\}$

angezeigt. Sie enthält $N \times N$ Elemente, wobei N der Anzahl der Zustände entspricht. Ist eine Übergangsfunktion δ von j nach i definiert, so ist das Element a_{ij} gleich dem Ereignis σ. Ist sie nicht definiert, ist das Element gleich 0. Für den Generator aus Abbildung 2.1 ergibt sich folgende Adjazenzmatrix

$$A = \begin{pmatrix} 0 & c & c \\ a & d & 0 \\ 0 & b & 0 \end{pmatrix}. \tag{2.9}$$

Die reguläre Sprache eines Generators $L(G)$ beinhaltet alle Strings die G aus dem Anfangszustand x_0 generieren kann. Diese beschreibt das ungesteuerte Verhalten des Generators und ist nach [24] definiert als

$$L(G) = \{s \in \Sigma^* | \delta(x_0, s)!\}. \tag{2.10}$$

Das ungesteuerte Verhalten $L(G)$ ist präfix-geschlossen $L(G) = \overline{L(G)}$. Dies gilt, da ein String, der einen Pfad von x_0 zu einem anderen Zustand repräsentiert, nur dann existieren kann, wenn auch alle seine Präfixe existieren.

Die reguläre Sprache $L_m(G) \subseteq L(G)$, welche alle Strings $s \in L(G)$ enthält, die zu einem markierten Zustand $x \in X_m$ führen, wird als markierendes Verhalten von G bezeichnet. Dies ist nach [24] definiert als

$$L_m(G) = \{s \in L(G) | \delta(x_0, s) \in X_m\}. \tag{2.11}$$

Tritt der Fall ein, dass $X = X_m$ ist, gilt $L(G) = L_m(G)$. Daraus folgt, dass das markierende Verhalten in diesem Fall auch präfix-geschlossen ist.

Eine weitere mögliche Beschreibungsform von ereignisdiskreten Systemen sind Petrinetze. Diese werden in dieser Thesis jedoch nicht weiter behandelt.

2.1.4. Komposition

Eine wichtige Operation für Automaten ist die Komposition. Diese dient dem Zusammenschalten von Komponentenmodellen. Die Komposition ermöglicht es die Komponenten einer komplexen Anlage einzeln zu modellieren und diese zu einem Gesamtmodell zusammenzuführen. Dies wird auch als kompositionale Modellbildung bezeichnet. Eine weitere Anwendung ist das Zusammenschalten von einem Systemmodell mit dem Modell einer Steuerung. Durch die Komposition wird die strukturierte Beschreibung eines Systems in ein strukturloses Gesamtmodell überführt. In diesem Abschnitt werden zwei Kompositionsoperatoren vorgestellt, die im weiteren Verlauf dieser Thesis verwendet werden. Die hier verwendeten Definitionen stammen aus [24] und [13].

Zunächst werden die Alphabete Σ_1 und Σ_2 zweier Generatoren in gemeinsame Ereignisse (engl. common events, ce) und private Ereignisse (engl. private events, pe) unterteilt

$$\Sigma_{ce} = \Sigma_1 \cap \Sigma_2 \tag{2.12}$$

$$\Sigma_{pe} = (\Sigma_2 - \Sigma_1) \cup (\Sigma_1 - \Sigma_2). \tag{2.13}$$

Produktkomposition

Die strenge Synchronisation (SPC), auch Produktkomposition genannt, synchronisiert zwei gegebene Komponenten G_1 und G_2 auf ihre gemeinsamen Ereignisse. Das Auftreten von privaten Ereignissen ist bei dieser Art der Komposition ausgeschlossen. Für die strenge Synchronisation zweier Komponenten G_1 und G_2 mit $x_1 \in X_1, x_2 \in X_2$ und $\sigma \in \Sigma_{ce}$ gilt:

$$G = G_1 ||_{SPC} G_2 = (X_1 \times X_2, \Sigma_{ce}, \delta, (x_{01}, x_{02}), X_{m1} \times X_{m2}) \text{ mit} \tag{2.14}$$

$$\delta((x_1, x_2), \sigma) = \begin{cases} \delta_1(x_1, \sigma) \times \delta_2(x_2, \sigma) & \text{falls } \delta_1(x_1, \sigma)! \wedge \delta_2(x_2, \sigma)! \\ \text{nicht definiert} & \text{sonst} \end{cases} \tag{2.15}$$

Im komponierten System G sind nur die Zustandsübergänge enthalten, die beide Komponenten gemeinsam ausführen. Es kommt dadurch zu einem totalen Gleichschritt der Komponenten. Der SPC-Operator wird in der Automatisierungstechnik zum Beispiel für das Zusammenschalten von Strecke und Steuerung angewendet. Die reguläre Sprache des Kompositionsgenerators enthält nur die gemeinsamen Strings, die von den beiden Komponenten erzeugt werden

$$L(G) = L(G_1 \times G_2) = L(G_1) \cap L(G_2). \tag{2.16}$$

Dies gilt ebenfalls für das markierende Verhalten $L_m(G)$.

Parallele Komposition

Das synchrone Produkt (SYPC), auch parallele Komposition genannt, synchronisiert zwei gegebene Komponenten ebenfalls auf die gemeinsamen Ereignisse. Im Gegensatz zum SPC-Operator wird das Auftreten von privaten Ereignissen nicht verboten. Für die parallele Komposition zweier Komponenten G_1 und G_2 mit $x_1 \in X_1$, $x_2 \in X_2$ und $\sigma \in \Sigma_{ce}$ gilt:

$$G = G_1||_{SYPC}G_2 = (X_1 \times X_2, \Sigma_{ce}, \delta, (x_{01}, x_{02}), X_{m1} \times X_{m2}) \text{ mit} \tag{2.17}$$

$$\delta((x_1, x_2), \sigma) = \begin{cases} \delta_1(x_1, \sigma) \times \delta_2(x_2, \sigma) & \text{falls } \delta_1(x_1, \sigma)! \wedge \delta_2(x_2, \sigma)! \\ \delta_1(x_1, \sigma) \times x_2 & \text{falls } \delta_1(x_1, \sigma)! \wedge \neg\delta_2(x_2, \sigma)! \wedge \sigma \notin \Sigma_{ce} \\ x_1 \times \delta_2(x_2, \sigma) & \text{falls } \neg\delta_1(x_{!}, \sigma)! \wedge \delta_2(x_2, \sigma)! \wedge \sigma \notin \Sigma_{ce} \\ \text{nicht definiert} & \text{sonst} \end{cases} \tag{2.18}$$

Der Operator führt zu einem partiellen Gleichschritt der Komponenten. Dies findet zum Beispiel in der Ressourcenzuteilung Anwendung. Besitzen die Komponenten keine gemeinsamen Ereignisse, bewegen sie sich völlig unabhängig voneinander. Dieser Fall wird auch als Shuffle bezeichnet. Sind die Ereignisalphabete der Komponenten identisch, hat die SYPC-Operation dieselbe Wirkung wie die SPC-Operation. Dies wird auch als Meet bezeichnet. Die reguläre Sprache des Kompositionsgenerators ist definiert als

$$L(G) = L(G_1||_{SYPC}G_2) = P^{-1}_{\Sigma_1}(L(G_1)) \cap P^{-1}_{\Sigma_2}(L(G_2)). \tag{2.19}$$

Die verwendete Operation P^{-1}_{Σ} steht für die inverse natürliche Projektion. Die natürliche Projektion $P : \Sigma^* \rightarrow \Sigma^*_1$ entfernt aus einem String $s \in \Sigma^*$ alle Ereignisse $\sigma \in \Sigma - \Sigma_1$. Für die natürliche Projektion gilt

$$P(\varepsilon) = \varepsilon, \tag{2.20}$$

$$P(\sigma) = \begin{cases} \sigma \text{ falls } \sigma \in \Sigma_1 \\ \varepsilon \text{ sonst} \end{cases} \tag{2.21}$$

$$P(s\sigma) = P(s)P(\sigma) \text{ für } s \in \Sigma^*, \sigma \in \Sigma. \tag{2.22}$$

Die inverse natürliche Projektion $P^{-1} : \Sigma^*_1 \rightarrow 2^{\Sigma^*}$ ist folgendermaßen definiert

$$P^{-1}(t) = \{s \in \Sigma^* | P(s) = t\}. \tag{2.23}$$

Das bedeutet im Bezug auf einen Generator, dass für jedes Ereignis $\sigma \in \Sigma - \Sigma_1$ eine Schlinge an jeden Zustand angefügt wird.

2.1.5. Analyse ereignisdiskreter Systeme

Ein Vorteil der Modellierung von ereignisdiskreten Systemen mit Hilfe von Automaten und formalen Sprachen ist deren Analysierbarkeit. Unter Einsatz der in diesem Abschnitt vorgestellten Analysemethoden lassen sich unterschiedliche Fragestellungen beantworten.

Erreichbarkeits- und Ko-Erreichbarkeitsanalyse

Mit Hilfe der Erreichbarkeits- und Ko-Erreichbarkeitsanalyse wird geprüft, welche Zustände eines Generators vom Anfangszustand aus erreichbar sind und ob das System von einem beliebigen Zustand aus einen markierten Zustand erreichen kann. Existiert ein String $s \in \Sigma^*$, der dazu führt, dass ein Zustand $x \in X$ vom Anfangszustand x_0 erreicht werden kann, dann ist dieser Zustand erreichbar. Für diesen Zustand x gilt die Zustandsübergangsfunktion $\delta(x_0, s) = x$. Auf Grund von Fehlern in der Modellierung oder durch Komposition, können nicht erreichbare Zustände in einem Modell vorhanden sein. Diese können durch Anwendung der Ac-Operation entfernt werden. Die Ac-Operation ist nach [6] wie folgt definiert

$$Ac(G) = (X_{ac}, \Sigma, \delta_1 ac, x_0, X_{ac,m}) \text{ mit} \tag{2.24}$$

$$X_{ac} = \{x \in X | (\exists s \in \Sigma^*) \delta(x_0, s) = x\}, \tag{2.25}$$

$$x_{ac,m} = X_m \cap X_{ac}, \tag{2.26}$$

$$\delta_{ac} = \delta|_{X_{ac} \times \Sigma \to X_{ac}}. \tag{2.27}$$

Die Notation $\delta_{ac} = \delta|_{X_{ac} \times \Sigma \to X_{ac}}$ steht für die Einschränkung von δ auf die erreichbaren Zustände. Ein Generator ist also genau dann erreichbar, wenn $G = Ac(G)$. Die Ac-Operation hat keinen Einfluss auf $L(G)$ und $L_m(G)$ und reduziert den Generator auf seinen erreichbaren Teil.

Die Analyse der Ko-Erreichbarkeit durchsucht einen Generator G auf Zustände, für die ein String existiert, der zu einem markierten Zustand führt. Existiert ein String $s \in \Sigma^*$, sodass $\delta(x, s) \in X_m$, dann ist der Zustand $x \in X$ ko-erreichbar. Die nicht ko-erreichbaren Zustände können mit Hilfe der CoAc-Operation entfernt werden. Der Generator wird dadurch auf seinen ko-erreichbaren Teil reduziert. Dies hat Auswirkungen auf die Sprache $L(G)$, da erreichbare Zustände entfernt werden können. Das markierende Verhalten $L_m(G)$ wird jedoch nicht verändert.

Die CoAc-Operation ist nach [6] wie folgt definiert

$$CoAc(G) = (X_{coac}, \Sigma, \delta_{coac}, x_{0,coac}, X_m) \text{ mit} \tag{2.28}$$

$$X_{coac} = \{x \in X | (\exists s \in \Sigma^*) \delta(x_0, s) \in X_m, \tag{2.29}$$

$$x_{0,coac} = \begin{cases} x_0 & \text{falls } x_0 \in X_{coac}, \\ \text{nicht definiert} & \text{sonst}, \end{cases} \tag{2.30}$$

$$\delta_{coac} = \delta|_{X_{coac} \times \Sigma \to X_{coac}}. \tag{2.31}$$

Auch hier steht die Notation $\delta_{coac} = \delta|_{X_{coac} \times \Sigma \to X_{coac}}$ für die Einschränkung von δ auf die ko-erreichbaren Zustände. Ein Generator G ist also genau dann ko-erreichbar, wenn $G = CoAc(G)$. Für einen ko-erreichbaren Generator gilt folgende sprachentheoretische Bedingung

$$L(G) = \overline{L_m(G)}. \tag{2.32}$$

Das bedeutet, es können nur Strings generiert werden, die zu einem markierten Zustand führen.

Nacheinander ausgeführt bilden diese beiden Operationen die Trim-Operation [6]. Der entstandene Generator enthält weder nicht-erreichbare (überflüssige) noch nicht ko-erreichbare Zustände. Die Trim-Operation wird wie folgt definiert

$$Trim(G) = CoAc[Ac(G)] = Ac[CoAc(G)]. \tag{2.33}$$

Ein Generator G ist genau dann trim, wenn $G = Trim(G)$.

Blockierung, Livelock und Deadlock

Die Analyse eines Generators auf Blockierung, Deadlock und Livelock überprüft, ob es Situationen gibt, in denen ein markierter Zustand nicht mehr erreicht werden kann. Enthält ein Generator G einen erreichbaren Zustand $x \notin X_m$ mit $\neg\delta(x, \sigma)!, \forall\sigma \in \Sigma$, dann spricht man von einem Deadlock. Das bedeutet, dass, wenn dieser Zustand erreicht wird, der Generator keine Ereignisse mehr generieren und somit auch kein markierter Zustand erreicht werden kann. In diesem Fall gilt nach [24]

$$\overline{L_m(G)} \subset L(G). \tag{2.34}$$

Erreicht ein Generator eine Zustandsmenge $\tilde{X} \subseteq X$ mit $x \notin X_m, \forall x \in \tilde{X}$, von der aus kein markierter Zustand mehr erreicht werden kann, spricht man von einem Livelock. Der Generator kann nur noch innerhalb dieser Zustandsmenge schalten. Es gilt ebenfalls die Inklusion aus Gleichung (2.34).

Ein Generator wird als blockierend bezeichnet, wenn er Deadlocks und/oder Livelocks enthält. Das bedeutet, dass die Inklusion aus Gleichung (2.34) gilt. Folglich ist ein Generator nichtblockierend, wenn

$$\overline{L_m(G)} = L(G) \tag{2.35}$$

gilt.

2.1.6. Supervisory Control Theory

Die Supervisory Control Theory (SCT) besteht aus einer Zusammenstellung von Methoden zur formalen Steuerungssynthese für ereignisdiskrete Systeme. Sie umfasst sowohl Methoden für logische als auch für zeitbewertete DES. In dieser Arbeit werden nur die Methoden für logische DES betrachtet. Erstmals wurde die SCT in der Dissertation von P.J. Ramadge [19] formalisiert. Aus dieser sind zahlreiche Publikationen zu diesem Thema hevorgegangen [20, 28, 25]. In der Industrie wird das Konzept der SCT noch selten angewendet. Dies liegt vor allem an der hohen Komplexität von Industrieanlagen, welche zu einem erhöhten Rechenaufwand führt. Außerdem sind die für die Beschreibung von ereignisdiskreten Systemen verwendeten Generatoren und Sprachen in der Industrie eher unbekannt. Aus diesen Gründen wurden die Methoden der SCT überwiegend an kleinen theoretischen Beispielen untersucht. Die in diesem Abschnitt beschriebenen Methoden der SCT beziehen sich auf Generatoren und reguläre Sprachen. Es können jedoch auch Petrinetze für die Anwendung der SCT als Beschreibungsform verwendet werden.

Für die Anwendung der SCT wird das System in die zu steuernde Strecke G und die Spezifikation K aufgeteilt. Diese werden getrennt voneinander mit Hilfe von Generatoren modelliert. Die Strecke steht dabei für das ungesteuerte Verhalten des Systems. Mit der Spezifikation wird das gewünschte Verhalten des Systems beschrieben. Aus den Modellen der Strecke und der Spezifikation wird die Steuerung S, auch Supervisor genannt, synthetisiert. Das Soll-Verhalten des Systems wird durch die Beeinflussung der Steuerstrecke durch den Supervisor erzielt. Dieser erlaubt und verbietet in einem gewissen Rahmen Ereignisse. Es sind jedoch nicht alle Ereignisse beeinflussbar. Zu diesen gehören zum Beispiel Sensorsignale. Aus diesem Grund wird das Ereignisalphabet des Systems in die beiden Teilalphabete der steuerbaren Σ_c (engl. controllable) und der nicht steuerbaren (engl. uncontrollable) Ereignisse Σ_{uc} partitioniert [29].

$$\Sigma = \Sigma_c \cup \Sigma_{uc} \text{ mit } \Sigma_c \cap \Sigma_{uc} = \varnothing \tag{2.36}$$

Der Supervisor kann auf die steuerbaren Ereignisse, zum Beispiel das Ansteuern von Aktoren, zugreifen und diese verhindern. Die nicht steuerbaren Ereignisse können nicht vom Supervisor beeinflusst werden, es können nur unerwünschte Zustandsübergänge durch das

Erlauben und Verbieten der steuerbaren Ereignisse verhindert werden. In Abbildung 2.2 ist das Blockschaltbild des geschlossenen Steuerkreises S/G für den monolithischen Ansatz dargestellt. Der monolithische Ansatz wird angewendet, wenn für das Modell der Strecke nur ein einziger Supervisor entworfen wird.

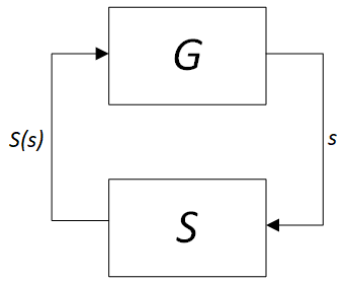


Abbildung 2.2.: Steuerkreis S/G

Der Supervisor befindet sich in der Rückführung des Steuerkreises. Die von der Strecke generierte Ereignisfolge s wird an den Supervisor zurückgeführt. Dieser überwacht die Ereignisfolge und deaktiviert die unerwünschten steuerbaren Ereignisse. Die Ereignismenge $S(s)$ enthält nur die erlaubten Folgeereignisse der rückgeführten Strings und die nicht steuerbaren Ereignisse. Die deaktivierten Ereignisse dürfen in dieser Menge nicht enthalten sein. Das geschlossene Verhalten des Systems $L(S/G) \subseteq L(G)$ von S/G mit $s \in \Sigma^*$ und $\sigma \in \Sigma$ ist nach [29] wie folgt definiert:

1. $\varepsilon \in L(S/G)$,
2. $(s \in L(S/G) \wedge \sigma \in S(s) \wedge s\sigma \in L(G)) \rightarrow s\sigma \in L(S/G)$,
3. keine anderen Strings gehören zu $L(S/G)$.

Laut der Definition ist der leere String ε immer in $L(S/G)$ enthalten und ein String s des gesteuerten Systems darf nur dann durch ein Folgeereignis σ erweitert werden, wenn dieses sowohl in S erlaubt als auch in G möglich ist. Die im Steuerkreis erlaubten Strings aus $L_m(G)$ bilden das markierende Verhalten des geschlossenen Steuerkreises

$$L_m(S/G) = L(S/G) \cap L_m(G). \tag{2.37}$$

Der Steuerkreis S/G wird als nichtblockierend bezeichnet, wenn gilt

$$\overline{L_m(S/G)} = L(S/G). \tag{2.38}$$

Die Sprache $L(S/G)$ besteht somit nur aus den Präfixen des markierenden Verhaltens. Dies stellt sicher, dass jede generierte Ereignisfolge sich immer auf einem Pfad zu einem markierten Zustand befindet.

Die Spezifikationen, welche das Verhalten des gesteuerten Systems festlegen, werden in dieser Arbeit ebenfalls formal mit Generatoren beschrieben. Diese müssen auf Steuerbarkeit bezüglich der Strecke G überprüft werden. Eine Spezifikation $K \subseteq \Sigma^*$ ist steuerbar bezüglich G, wenn folgende Bedingung erfüllt wird

$$\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}. \tag{2.39}$$

Diese Bedingung fordert, dass kein String $s \in \overline{K}$ durch das Auftreten eines nicht steuerbaren Ereignisses $\overline{K}$ verlassen darf. Ist die Spezifikation K nicht steuerbar, wird eine möglichst gering von K abweichende Spezifikation gesucht, die steuerbar bezüglich G ist. Diese wird als supremale steuerbare Teilsprache $K^{\uparrow C}$ (etwas mehr einschränkend als K) oder infimale präfix-geschlossene steuerbare Obersprache $K^{\downarrow C}$ (etwas mehr erlaubend als K) bezeichnet. In [6], [24], [29] und [7] sind die Definitionen und Algorithmen zur Berechnung dieser Teilsprachen zu finden.

Die Untersuchung der Steuerbarkeit gibt an, ob für eine gegebene Spezifikation K und eine Strecke G ein Supervisor S existiert. Es wird jedoch keine Aussage darüber getroffen, ob der Supervisor nichtblockierend bezüglich G ist. Für eine steuerbare Spezifikation K existiert genau dann ein nichtblockierender Supervisor S bezüglich G, sodass $L_m(S/G) = K \wedge L(S/G) = \overline{K}$, wenn

$$K = \overline{K} \cap L_m(G). \tag{2.40}$$

Diese Eigenschaft wird als $L_m(G)$-Abgeschlossenheit bezeichnet.

Die verschiedenen Standardprobleme der SCT und deren Lösungen sind in [6, 24, 29] zusammengefasst. Diese definieren Ansätze zur Erstellung eines Supervisors. Für diese Arbeit ist nur das Basic Supervisory Control Problem-Nonblocking Case (BSCP-NB) relevant. Bei Problemen dieser Art werden Systeme mit nicht steuerbaren Ereignissen behandelt. Das BSCP-NB ist nach [24] wie folgt definiert:

Gegeben ist ein ereignisdiskretes System G mit einem Ereignisalphabet Σ mit $\Sigma_{uc} \subseteq \Sigma$ und eine $L_m(G)$-abgeschlossene Spezifikation $K \subseteq L_m(G)$. Bestimme einen nichtblockierenden Supervisor S, sodass gilt:

1. $L_m(S/G) \subseteq K$
2. $L_m(S/G)$ „maximal “ ist, also für jeden anderen nichtblockierenden Supervisor S' mit $L_m(S'/G) \subseteq K$ gilt:
$$L_m(S'/G) \subseteq L_m(S/G) \tag{2.41}$$

Die gesuchte Steuerung muss für die Lösung des Problems so gewählt werden, dass

$$L(S/G) = \overline{K^{\uparrow C}}. \tag{2.42}$$

Die daraus resultierende Steuerung kann als Liste ihrer Steuereingriffe oder als Akzeptor R, welcher die Sprache $\overline{K^{\uparrow C}}$ markiert, realisiert werden. Für die Darstellung als Liste sind die Steuereingriffe $S(s)$ nach [24] formal definiert durch

$$S(s) = [\Sigma_{uc} \cap \{\sigma \in \Sigma | \delta(\delta(x_0, s), \sigma)!\}] \cup \{\sigma \in \Sigma_c | s\sigma \in \overline{K^{\uparrow C}}\}. \tag{2.43}$$

Für die Repräsentation des Supervisors als Akzeptor wird $R = (Y, \Sigma, \zeta, y_0, Y_m)$ so konstruiert, dass $Y = Y_m$, Σ gleich dem Ereignisalphabet der Strecke G, $R = Trim(R)$ und ζ, sodass $L_m(R) = L(R) := \overline{K^{\uparrow C}}$ ist. Mit Hilfe der strengen Synchronisation (SPC) lässt sich das Verhalten des geschlossenen Steuerkreises wie folgt berechnen:

$$L(G||_{SPC}R) = L(G) \cap L(R) = L(G) \cap \overline{K^{\uparrow C}} = L(S/G) \tag{2.44}$$

$$L_m(G||_{SPC}R) = L_m(G) \cap L_m(R) = \overline{K^{\uparrow C}} \cap L_m(G) = L(S/G) \cap L_m(G) = L_m(S/G) \tag{2.45}$$

2.1.7. Strukturelle Entwurfsansätze

Der in Abschnitt 2.1.6 vorgestellte Ansatz zur Steuerungssynthese entwirft auf Basis eines unstrukturierten Gesamtmodells der Strecke einen Supervisor. Dieser übernimmt die gesamte Steuerungsaufgabe. Die große Modellkomplexität, Steuerungsstruktur und algorithmische Komplexität sprechen in der Realität häufig gegen diesen Ansatz. Aus diesen Gründen wird der monolithische Ansatz durch Modularisierung und Hierarchisierung strukturell erweitert. In diesem Abschnitt werden drei strukturelle Ansätze der Supervisory Control Theory vorgestellt.

Modularer Entwurfsansatz

Der modulare Entwurfsansatz hat eine Aufteilung der globalen Steuerungsaufgabe auf mehrere Supervisor zum Ziel. Diese bilden zusammen den modularen Supervisor S_{mod}. Für diesen Ansatz wird vorausgesetzt, dass alle Streckenereignisse für sämtliche Supervisor global zur Verfügung stehen. In Abbildung 2.3 ist die Architektur des Ansatzes beispielhaft für zwei Supervisor dargestellt.

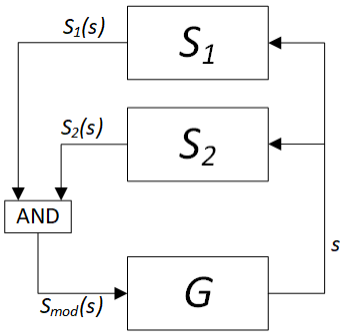


Abbildung 2.3.: Steuerkreis des modularen Ansatzes

Den Supervisorn S_1 und S_2 werden alle Ereignisse des Systems zugeführt. Der von G generierte String s wird von ihnen überwacht, dabei verbieten sie die nach ihrer Steuerungsaufgabe unerwünschten Folgeereignisse duch die Steuereingriffe $S_1(s)$ und $S_2(s)$. Über die Konjunktion der Steuereingriffe wird festgelegt, ob ein Folgeereignis ausgeführt oder verboten wird. Somit wird ein Ereignis σ nur dann ausgeführt, wenn S_1 und S_2 es erlauben ($\sigma \in S_1(s) \wedge \sigma \in S_2(s)$). Der allgemeine modulare Supervisor $S_{mod} : L(G) \rightarrow 2^{\Sigma}$ für n zulässige Supervisor ist formal nach [24] wie folgt definiert:

$$S_{mod}(s) = S_1(s) \cap S_2(s) \cap \ldots \cap S_n(s) \tag{2.46}$$

Aus Gleichung (2.46) geht hervor, dass sich die Steuereingriffe des modularen Supervisors aus der Schnittmenge der Steuereingriffe der individuellen Supervisor zusammensetzen. Daraus folgt nach [6] und [24] für das Verhalten des geschlossenen Steuerkreises S_{mod}/G:

$$L(S_{mod}/G) = L(S_1/G) \cap L(S_2/G) \cap \ldots \cap L(S_n/G) \tag{2.47}$$

$$L_m(S_{mod}/G) = L_m(S_1/G) \cap L_m(S_2/G) \cap \ldots L_m(S_n/G) \tag{2.48}$$

Für die modulare Lösung des Entwurfsproblems aus Abschnitt 2.1.6 gelten dieselben Güteanforderungen wie für die monolithische Lösung. Es wird gefordert, dass die Gesamtspezifikation K in Form von präfix-geschlossenen Teilspezifikationen vorliegt oder sich in diese Form zerlegen lässt, da nur bei präfix-geschlossenen Sprachen die Steuerbarkeit unter Schnittmengenbildung erhalten bleibt. Für zwei Teilspezifikationen K_1 und K_2, welche steuerbar bezüglich G und präfix-geschlossen sind, gilt nach [24]

$$(K_1 \cap K_2)^{\uparrow C} = K_1^{\uparrow C} \cap K_2^{\uparrow C}. \tag{2.49}$$

Das Modular Supervisor Control Problem (MSCP) gibt an, dass aus der Spezifikation $K = K_1 \cap K_2 \cap \ldots \cap K_n$ mit $K_i = \overline{K_i}, i = 1, \ldots, n$ ein modularer Supervisor bestimmt werden soll, sodass gilt:

$$L(S_{mod}/G) = K^{\uparrow C} \tag{2.50}$$

Wegen der präfix-geschlossenen Teilspezifikationen können die einzelnen Supervisor S_i zur Lösung des Problems unabhängig voneinander berechnet werden, sodass nach [24]

$$L(S_i/G) = K^{\uparrow C} \text{ für } i = 1, \ldots, n \tag{2.51}$$

gilt. Im Anschluss ist der modulare Supervisor nach Gleichung (2.46) zu berechnen.

Die Lösung aus Gleichung (2.51) sagt nichts über die Eigenschaft des Nichtblockierens aus, da diese unter Schnittmengenbildung nicht erhalten bleibt. Daher muss ein nichtblockierender modularer Supervisor nach [27] die folgende Bedingung erfüllen:

$$\overline{L_m(S_1/G) \cap L_m(S_2/g) \cap \ldots \cap L_m(S_n/G)} = L_m(S_1/G) \cap L_m(S_2/G) \cap \ldots \cap L_m(S_n/G) \tag{2.52}$$

Der Vorteil des modularen Ansatzes besteht darin, dass die Zustandsräume der zu entwerfenden Supervisor generell kleiner als bei einem monolithischen Supervisor sind. Dadurch ist die Steuerung flexibler bei Änderungen. Jedoch wird weiterhin ein unstrukturiertes Gesamtmodell der Strecke benötigt. Dies kann mit Hilfe des lokal-modularen Ansatzes umgangen werden.

Lokal-modularer Entwurfsansatz

Der lokal-modulare Ansatz wurde als Erweiterung des modularen Ansatzes von de Queiroz und Cury eingeführt, mit dem Ziel, sowohl die Steuerungsaufgabe als auch das System zu modularisieren. Die nachfolgenden Definitionen, Begriffe und Zusammenhänge für den lokal-modularen Ansatz wurden aus [17], [16] und [24] entnommen.

Für den lokal-modularen Ansatz wird kein unstrukturiertes Gesamtmodell der Strecke benötigt. Es werden nur diejenigen Komponenten zu einem lokalen System zusammengefasst, die zur Einhaltung einer Spezifikation notwendig sind. Dies ist vor allem bei Systemen mit stark nebenläufigen Komponenten von Vorteil. Ein Beispiel für eine Anwendung des lokal-modularen Ansatzes ist in [18] aufgeführt. Abbildung 2.4 zeigt die Architektur des lokal-modularen Ansatzes beispielhaft für zwei Supervisor. Die aus den Spezifikationen und den dazugehörigen lokalen Systemen berechneten Supervisor steuern jeweils nur einen Teil des Gesamtsystems.

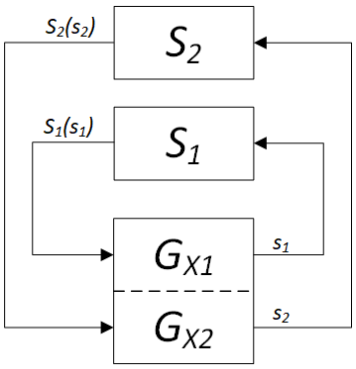


Abbildung 2.4.: Steuerkreis des lokal-modularen Ansatzes

Für die Anwendung dieses Ansatzes wird zunächst das Kompositionssystem aus n' Komponenten G'_j aufgestellt. Im weiteren Verlauf dieses Abschnitts wird auf Grund der Übersichtlichkeit auf den Index SYPC verzichtet. Wenn nicht anders angegeben, steht das Symbol $||$ für die parallele Komposition. Aus der Komposition der Teilmodelle ergeben sich das Gesamtmodell $G = ||_{i=1}^{n'} G'_i$ und das Alphabet der Strecke $\Sigma = \bigcup_{i=1}^{n'} \Sigma_i^{n'}$. Sind die Komponenten des Kompositionssystems asynchron zueinander, wird dies als Produktsystem bezeichnet [21]. Das Produktsystem mit der größtmöglichen Anzahl an asynchronen Komponenten ist das feinste Produktsystem. Die Bildung des feinsten Produktsystems hat zur Folge, dass die Strecke asynchronisiert wird. Eine Synchronisation von Komponenten erfolgt durch gemeinsame Ereignisse der Komponenten oder durch eine Spezifikation, die die Synchronisation zweier Komponenten fordert. Die durch eine Spezifikation zwanghaft synchronisierten Komponenten werden zu einem lokalen System zusammengefasst. Diese sind definiert als:

Sei G ein Produktsystem mit n Komponenten G_i und seien m gegebene Spezifikationen K_{Xj} definiert über $\Sigma_{Xj} \subseteq \Sigma$. Die lokalen Systeme G_{Xj}, welche die von einer Spezifikation K_{Xj} zwanghaft synchronisierten Komponenten enthalten, sind definiert als

$$G_{Xj} = ||_{i \in N_{Xj}} G_i \text{ mit } N_{Xj} = \{k \in \{1, \ldots, n\} | \Sigma_k \cap \Sigma_{Xj} \neq \varnothing\}. \tag{2.53}$$

Komponenten, die durch keine Spezifikation eingeschränkt werden, sind keinem lokalen System zugeordnet und werden als inhärent asynchrone Komponenten bezeichnet. Diese haben keinen Einfluss auf das Gesamtsystem und können deshalb vernachlässigt werden. Daraus folgt das eingeschränkte System:

$$G_e = ||_{j=1}^{m} G_{Xj} \text{ und } \Sigma_e = \bigcup_{i=1}^{m} \Sigma_{Xj} \tag{2.54}$$

Die Spezifikationen können anschließend an die lokalen Systeme, das eingeschränkte System und das Gesamtsystem angepasst werden:

$$E_{Xj} = K_{Xj} || L_m(G_{Xj}) \tag{2.55}$$

$$E_{Xje} = K_{Xj} || L_m(G_e) \tag{2.56}$$

$$E_X = K_{Xj} || L_m(G) \tag{2.57}$$

Das folgende Beispiel aus [24] soll zum besseren Verständnis der Anpassung des Systemmodells und der Spezifikationen dienen. Es ist ein Kompositionssystem G aus fünf Komponenten $G_i' = (X_i', \Sigma_i', \delta_i', x_{i,0}', X_{i,m}')$ mit $i = 1, \ldots, m$ gegeben. Des Weiteren sind $K_{X1} \subset \Sigma_{X1}^*$ und $K_{X2} \subset \Sigma_{X2}^*$ zwei gegebene Spezifikationen. In Abbildung 2.5 sind die Relationen zwischen den Ereignisalphabeten dargestellt.

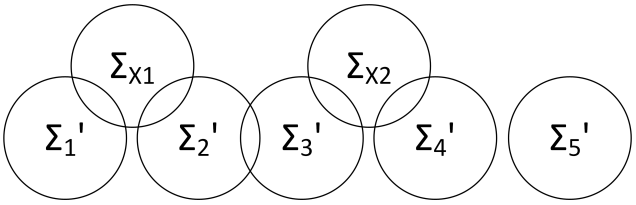


Abbildung 2.5.: Venn-Diagramm der Ereignisalphabete, Quelle: [24, S.158]

Das feinste Produktsystem ergibt sich aus dem Venn-Diagramm aus Abbildung 2.5: $G_1 = G_1'$, $G_2 = G_2' || G_3'$, $G_3 = G_4'$ und $G_4 = G_5'$. Die lokalen Systeme ergeben sich aus der zwanghaften Synchronisation durch die Spezifikationen: $G_{X1} = G_1 || G_2$ und $G_{X2} = G_2 || G_3$. Die Komponente G_4 ist keinem lokalen System zugeordnet. Daraus folgt für das eingeschränkte System $G_e = G_1 || G_2 || G_3$. Die Spezifikationen werden durch $E_{X1} = K_{X1} || G_{X1}$ und $E_{X2} = K_{X2} || G_{X2}$ angepasst.

Damit sich die Spezifikationen durch ihre Steuereingriffe nicht gegenseitig blockieren, müssen sie lokal-modular sein. Die Eigenschaft der lokalen Modularität ist nach [24] folgendermaßen definiert:

I sei eine beliebige Indexmenge und $L_i \subseteq \Sigma_i^*, i \in I$, dann ist die Menge $\{L_i, i \in I\}$ lokal-modular, wenn

$$||_{i \in I} \overline{L_i} = \overline{||_{i \in I} L_i}. \tag{2.58}$$

Für zwei lokale Supervisor gilt nach [16] der folgende Zusammenhang:

Es seien ein Kompositionssystem G mit n' Komponenten und zwei lokale Spezifikationen K_{X1} und K_{X2} gegeben. $supC(E_{X1}, G_{X1})$ sei die supremale steuerbare Teilsprache von

E_{X1} bezüglich G_{X1} und $supC(E_{X2}, G_{X2})$ entsprechend. Sind die supremalen steuerbaren Teilsprachen lokal-modular, dann gilt

$$supC(E_{X1e} \cap E_{X2e}, G_E) = supC(E_{X1}, G_{X1}) || supC(E_{X2}, G_{X2}). \tag{2.59}$$

Aus Gleichung (2.59) geht hervor, dass es ausreichend ist, für lokal-modulare Spezifikationen, die Supervisor mittels $L(S_1/G_{X1}) = supC(E_{X1}, G_{X_1})$ und $L(S_2/G_{X2}) = supC(E_{X2}, G_{X_2})$ zu bestimmen.

Dezentraler Entwurfsansatz

Der von Wonham und Lin eingeführte dezentrale Ansatz [11, 12] stellt eine Erweiterung des modularen Ansatzes auf partielle Beobachtbarkeit dar. In der Realität besitzen viele verteilte Systeme Einschränkungen bezüglich der Verfügbarkeit von Ereignissen. Sensorsignale können zum Beispiel durch räumliche Trennung nicht an jeder Steuerung verfügbar sein. Folglich steht den einzelnen Supervisorn nur ein Teil der Streckenereignisse zur Verfügung. In Abbildung 2.6 ist die Architektur des dezentralen Entwurfsansatzes dargestellt.

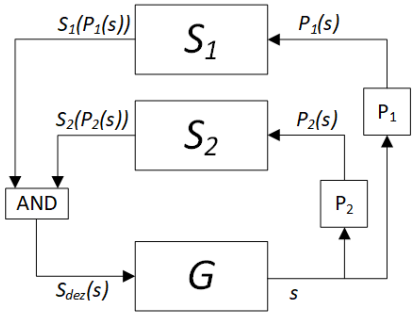


Abbildung 2.6.: Steuerkreis des dezentralen Ansatzes

Auf den dezentralen Entwurfsansatz wird in dieser Arbeit nicht näher eingegangen. Details zu diesem Ansatz sind in [11], [12] und [24] zu finden.

2.2. Software zum Steuerungsentwurf

In diesem Abschnitt werden die verschiedenen Software- und Entwicklungstools, welche verwendet werden, kurz vorgestellt. Das zur Steuerungssynthese verwendete Programm DES-Tool ist in Abschnitt 2.2.1 beschrieben. Abschnitt 2.2.2 befasst sich mit dem Codegenerator ACArrow. Das Siemens TIA Portal wird in Abschnitt 2.2.3 vorgestellt.

2.2.1. DESTool

Für die Analyse und Steuerungssynthese von Automaten wird in dieser Arbeit das Programm DESTool verwendet. Dieses ist frei erhältlich und wurde an der Universität Erlangen-Nürnberg entwickelt. DESTool ist mit einer grafischen Benutzeroberfläche ausgestattet, welche es ermöglicht Automaten per „Drag and Drop“ zu erstellen. Das Programm bietet eine große Anzahl an Funktionen. Auf Grund des großen Umfangs des Programms werden hier nur die für diese Arbeit relevanten Funktionen vorgestellt. Weitere Informationen zu DESTool und seinen Funktionen sind auf der Internetseite [14] des Programms zu finden.

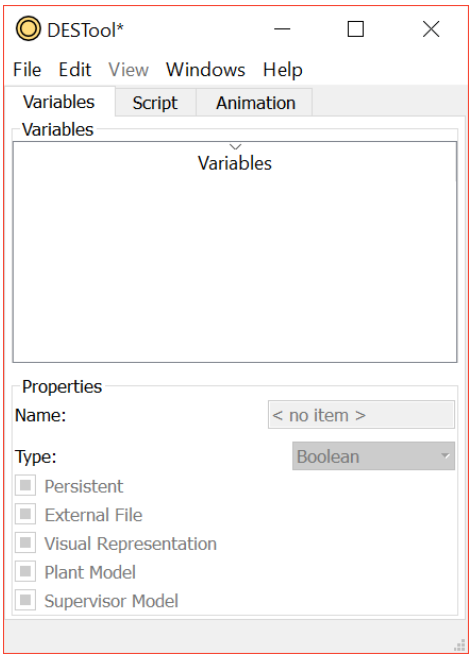


Abbildung 2.7.: Hauptmenü DESTool

In Abbildung 2.7 ist das Hauptmenü von DESTool dargestellt. Dieses ist in drei Reiter unterteilt: „Variables“, „Script“ und „Animation“. Im Reiter „Variables“ können per Rechtsklick neue Variablen eingefügt werden. Dafür stehen verschiedene Variablentypen zur Verfügung. Im Rahmen dieser Arbeit werden nur Variablen der Typen „System“, „Alphabet“ und „Boolean“ verwendet. Variablen vom Typ „System“ werden für Generatoren angelegt. Der Variablentyp „Alphabet“ wird für das Alphabet eines Generators genutzt. Dies kann zum Beispiel für das Einfügen von Schlingen verwendet werden. Ergebnisse von Operationen, wie beispielsweise die Überprüfung von Steuerbarkeit, werden in Variablen des Typs „Boolean“ gespeichert. Die angelegten Variablen werden im Fenster angezeigt. Über den Reiter „Script“ gelangt man in das Untermenü für die Operationen. Hier können die verschiedenen Funktionen für die Analyse der Modelle und die Steuerungssynthese eingefügt werden. Die in dieser Arbeit

verwendeten Funktionen sind in Tabelle 2.1 aufgeführt. Eine Simulation der Modelle ist im Reiter „Animation“ möglich.

Tabelle 2.1.: Verwendete DESTool Funktionen

Funktion	Beschreibung
IsNonblocking	Prüft, ob zwei Systeme nichtblockierend sind
IsControllable	Prüft, ob eine Spezifikation K steuerbar bzgl. G ist
Parallel	Führt die SYPC Komposition aus
Product	Führt die SPC Komposition aus
SubConNB	Berechnet die nichtblockierende supremale steuerbare Teilsprache
SubReduce	Berechnet den reduzierten Supervisor

2.2.2. Codegenerator

Wie in Abschnitt 2.2.1 beschrieben sind Supervisormodelle S das Ergebnis der SCT. Damit diese von der SPS interpretiert werden können, müssen sie in Programmcode umgewandelt werden. Dafür wird in dieser Thesis der von Nadine Gohert im Rahmen ihrer Masterthesis [8] entwickelte Codegenerator ACArrow verwendet. Dieser wandelt die Petrinetze eines MATLAB Protokolls oder die Automaten eines DESTool Projekts in Programmcode um, der von einer SPS interpretiert werden kann. Der Codegenerator stellt somit die Verknüpfung zwischen dem Entwicklungstool und dem Steuergerät dar.

Für die Bedienung von ACArrow steht eine grafische Benutzeroberfläche zur Verfügung. In Abbildung 2.8 ist das Hauptmenü des Codegenerators zu sehen. Im oberen Bereich des Fensters befinden sich zwei Reiter, über die ausgewählt werden kann, ob die Codegenerierung für Automaten oder Petrinetze durchgeführt werden soll.

Das Hauptmenü für die Codegenerierung von Automaten ist in fünf Reiter unterteilt: „Codegenerator“, „Database“, „Option“, „Choice“ und „Help“.

Im Menü „Codegenerator“ kann über die Schaltfläche „Browse and Upload“ ein DESTool Projekt ausgewählt und importiert werden. Wenn sich mehrere steuerbare Ereignisse an einem Zustand der Modelle befinden, legt der Codegenerator eine automatische priorisierende Auswahl fest. Um dies zu umgehen, kann im Reiter „Choice“ eingestellt werden, dass die Auswahl zufällig stattfinden soll.

Die Startadressen der Merker können im Reiter „Option“ bestimmt werden. Des Weiteren ist hier eine Definition der Anfangsbuchstaben der verschiedenen Arten von Ereignissen (Ausgang, Eingang, steuerbar oder nicht steuerbar) möglich. Diese müssen mit den in DESTool verwendeten Bezeichnungen übereinstimmen.

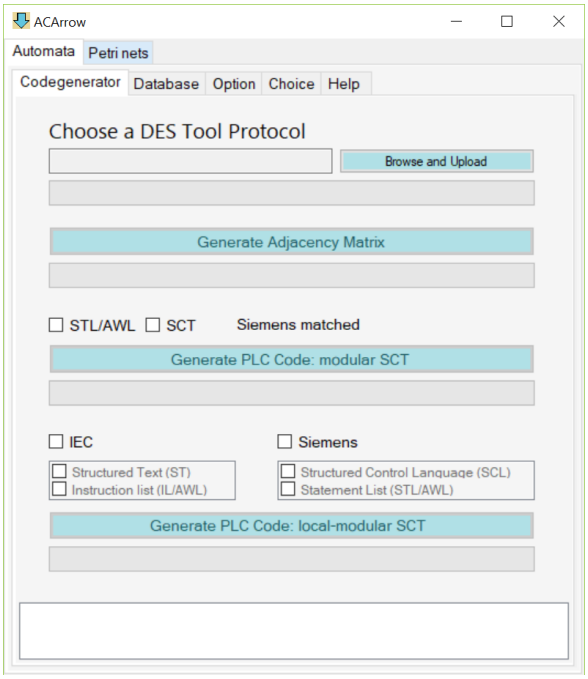
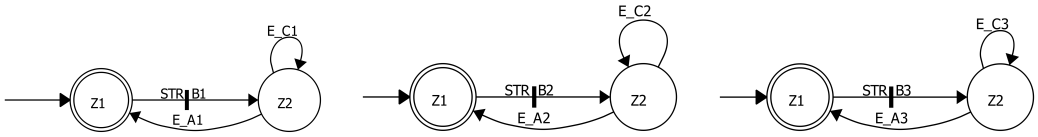


Abbildung 2.8.: Hauptmenü ACArrow

Wurden die entsprechenden Einstellungen vorgenommen, kann im Reiter „Codegenerator“ die Codegenerierung des importierten Projekts gestartet werden. Dafür muss eine Auswahl der Programmiersprache erfolgen. Außerdem wird zwischen dem modularen und dem lokal-modularen Entwurfsansatz unterschieden. Soll die Codegenerierung für den modularen Ansatz erfolgen, können die Programmiersprachen AWL oder ST/SCL ausgewählt werden. Für den lokal-modularen Ansatz muss zusätzlich eine Auswahl zwischen IEC und Siemens Syntax getroffen werden. Über die Schaltfläche „Generate PLC Code: modular SCT“ bzw. „Generate PLC Code: local-modular SCT“ wird die Codegenerierung für den jeweiligen Ansatz gestartet. Im Rahmen dieser Thesis wird ausschließlich Programmcode in der Programmiersprache SCL mit Siemens Syntax generiert und verwendet.

Die Funktionsweise des Codegenerators soll anhand eines Beispiels für eine Steuerung nach dem lokal-modularen Ansatz erklärt werden. Abbildung 2.9 zeigt die Komponenten einer Fertigungslinie. Diese besteht aus drei Maschinen und zwei Puffern, die jeweils ein Werkstück aufnehmen können. In Abbildung 2.10 sind die Modelle der Maschinen als Generatoren dargestellt. Die steuerbaren Ereignisse wurden mit der Abkürzung STR und die nicht steuerbaren Ereignisse mit E gekennzeichnet.

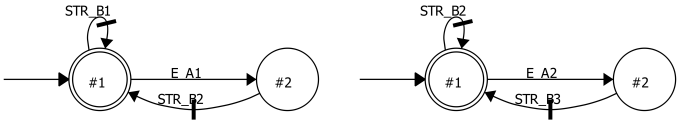
Abbildung 2.9.: Fertigungslinie



(a) G_1 (b) G_2 (c) G_3

Abbildung 2.10.: Modelle der Streckenkomponenten

Abbildung 2.11 zeigt die beiden Supervisor, die sich für die lokalen Systeme $G_{X1} = G_1||G_2$ und $G_{X2} = G_2||G3$ ergeben.



(a) S_1 (b) S_2

Abbildung 2.11.: Lokal-modulare Supervisor

Das angelegte DESTool Projekt wird in den Codegenerator ACArrow importiert und der Quellcode für den lokal-modularen Ansatz in Siemens SCL generiert. Dabei werden mehrere Quellcodedateien erzeugt, welche nachfolgend in Auszügen dargestellt werden.

Bei der lokal-modularen Codegenerierung werden 12 SCL Quelltextdateien erzeugt. In der Funktion `Main_SCT` werden die einzelnen Unterfunktionen aufgerufen. Die Funktion `Init_SCT` initialisiert die ersten Zustände der Generatoren. Das Einlesen der Eingänge und Schreiben der Ausgänge erfolgt in den Funktionen `Read_Input_SCT` und `Write_Output_SCT`. In den Funktionen `C_Event_Plant_SCT`, `C_Event_Super_SCT`, `UC_Event_Plant_SCT` und `UC_Super_Plant_SCT` existiert für jede abgehende Kante eines Zustands, eine bedingte Anweisung. Diese wird ausgeführt, wenn der aktuelle Zustand aktiv und das Ereignis an der abgehenden Kante möglich ist. Wird diese Bedingung erfüllt, wird der aktuelle Zustand zurückgesetzt und der Nachfolgezustand aktiv. Die Abfrage der Bedingungen ist in vier Funktionen unterteilt, da zwischen steuerbaren und nicht steuerbaren Ereignissen des Supervisors bzw. der Steuerstrecke unterschieden wird. In

der Funktion `C_Events_Enable_SCT` erfolgt die Zuweisung der steuerbaren Ereignisse in Abhängigkeit von den aktiven Zuständen. Die Funktionen `Choice_SCT`, `Load_2_Memory_UC_Events_SCT` und `Load_2_Memory_C_Events_SCT` sind nur für die Art der Codegenerierung relevant, deshalb wird hier nicht weiter darauf eingegangen. In [8] können weitere Informationen zum Codegenerator ACArrow nachgelesen werden.

```
//Main_SCT
IF NOT "InitBit_SCT" THEN
"Init_SCT"();
"InitBit_SCT":=1;
ELSE
"Read_Input_SCT"();
"Load_2_Memory_UC_Events_SCT"();
"UC_Events_Plant_SCT"();
"UC_Events_Super_SCT"();
"C_Events_Enable_SCT"();
"Choice_SCT"();
"C_Events_Plant_SCT"();
///////Space for Interfaces
"C_Events_Super_SCT"();
"Write_Output_SCT"();
END_IF;

//Init_SCT
"G_Z1":=1;
⋮
"S2_Z1":=1;

//C_Events_Plant_SCT
IF "G_Z1" AND "STR_B1_0" THEN
"G_Z2":=1;
"G_Z1":=0;
"STR_B1":=1;
END_IF;
⋮
IF "G_Z3" AND "STR_B2_0" THEN
"G_Z4":=1;
"G_Z3":=0;
"STR_B2":=1;
END_IF;
```

```
//Read_Input_SCT
"R_TRIG_DB_0"(CLK:="E_A1",
Q=>"E_A1_0");
⋮
"R_TRIG_DB_5"(CLK:="E_C3",
Q=>"E_C3_0");

IF NOT "InitInput_SCT" THEN
"InitInput_SCT":=1;
"E_A1_0":=0;
⋮
"E_C3_0":=0;
END_IF;

//C_Events_Enable_SCT
"STR_B1_0":= "S1_Z1";
⋮
"STR_B3_0":= "S2_Z2";

//UC_Events_Plant_SCT
IF "G_Z2" AND "E_A1_1" THEN
"G_Z1":=1;
"G_Z2":=0;
"E_A1_1":=0;
END_IF;
⋮
IF "G_Z4" AND "E_A3_1" THEN
⋮
END_IF;
```

```
//Load_2_Memory_C_Events_SCT
"STR_B1_1":="STR_B1";
⋮
"STR_B3_1":="STR_B3";

//C_Events_Super_SCT
IF "S1_Z2" AND "STR_B2_1" THEN
"S1_Z1":=1;
"S1_Z2":=0;
"STR_B2_1":=0;
END_IF;
⋮
"STR_B1":=0;
"STR_B2":=0;
"STR_B3":=0;

//Load_2_Memory_UC_Events_SCT
"E_A1_1":="E_A1_0";
⋮
"E_C3_1":="E_C3_0";
```

```
//UC_Events_Super_SCT
Load_2_Memory_UC_Events_SCT();
IF "S1_Z1" AND "E_A1_1" THEN
"S1_Z2":=1;
"S1_Z1":=0;
"E_A1_1":=0;
END_IF;

Load_2_Memory_UC_Events_SCT();
IF "S2_Z1" AND "E_A2_1" THEN
"S2_Z2":=1;
"S2_Z1":=0;
"E_A2_1":=0;
END_IF;
```

2.2.3. TIA Portal

Das Totally Integrated Automation Portal (TIA Portal) der Firma Siemens ist ein Programm zur Verwaltung, Konfiguration, Programmierung und Visualisierung von Prozessen. Mit Hilfe des TIA-Portals wird die Steuerung in der realen Hardware implementiert. Für die SPS-Programmierung können die folgenden Programmiersprachen verwendet werden:

- Kontaktplan (KOP)
- Anwendungsliste (AWL)
- Funktionsplan (FUP)
- Strukturierte Sprache (S7-SCL)
- Ablaufsprache (S7-Graph)

Mit Hilfe der zur Verfügung gestellten S7-PLCSIM, kann das Verhalten der Steuerung simuliert werden. Eine Anleitung für das Arbeiten mit dem TIA Portal zeigt [1].

2.3. Optisches Identifikationssystem

Das optische Codelesegerät Simatic MV440 von Siemens ist für die Erkennung von 1D- und 2D-Codes, sowie Klarschrift und Objekten ausgelegt. Das Lesegerät ist über Ethernet mit einem Computer verbunden. Mit Hilfe eines Internet-Browsers kann der Codeleser über die Web Server Bedienoberfläche programmiert werden. In diesem Abschnitt wird nur die Erkennung von Objekten beschrieben. Die weiteren Funktionen können in [2] nachgelesen werden.

Abbildung 2.12.: Startseite der Bedienoberfläche

Die Bedienoberfläche des Lesegeräts wird über die IP-Adresse des Geräts aufgerufen. Auf der Startseite (siehe Abbildung 2.12) muss zum Programmieren des Geräts der Menüpunkt „Einrichten“ ausgewählt werden. Das Hauptmenü zum Einrichten der MV440 ist in Abbildung 2.13 zu sehen. Auf der linken Seite des Fensters können die einzelnen Aufgaben in Form von Schaltflächen aufgerufen werden. Das Menü ist in folgende Aufgaben unterteilt: „Einrichten“, „Verbindungen“, „Programme“, „Auswerten“, „Optionen“, „Info“, „Verwalten“ und „Stop“.

Die Einstellung des Bildes wird im Menüpunkt „Einrichten“ vorgenommen. Dafür wird eine kleine Anleitung in einem Textfeld im oberen Bereich angezeigt. Im Untermenü „Verbindungen“ können die Kommunikationsschnittstellen, die Integration des Lesegeräts, sowie die Zuordnung der digitalen Ein- und Ausgänge eingestellt werden. Die eigentliche Programmierung der MV440 wird im Menüpunkt „Programme“ durchgeführt. Auf diesen Punkt wird später näher eingegangen. Die Aufgabe „Auswerten“ startet ein vom Benutzer ausgewähltes Programm und zeigt das Ergebnis der Auswertung an. Weitere Einstellungen, wie zum Beispiel Benutzerberechtigungen oder Beleuchtung, können unter „Optionen“ vorgenommen

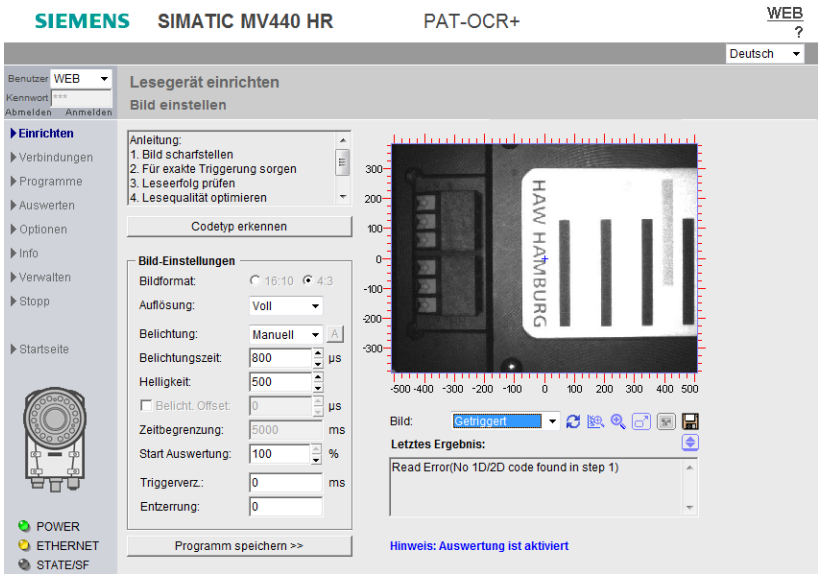


Abbildung 2.13.: Hauptmenü zum Einrichten der MV440

werden. Die Geräte- und Diagnosedaten sind unter „Info“ zu finden. Im Untermenü „Verwalten“ können die notwendigen Firmwareupdates, ein Rücksetzen/Wiederherstellen der Gerätekonfiguration durchgeführt, sowie Lizenzen eingelesen werden.

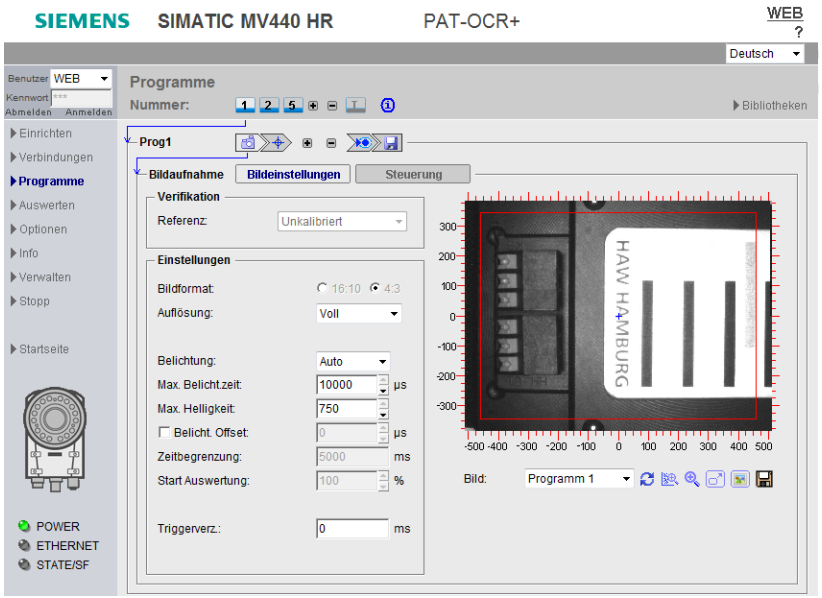


Abbildung 2.14.: Menü für die Programmierung des Codelesegeräts

In Abbildung 2.14 ist das Menü für die Erstellung von Programmen dargestellt. Ein Pro-

gramm für die Erkennung und Klassifikation von Objekten besteht aus mehreren Schritten: Bildaufnahme, Objekterkennung und Ergebnis. Im Bildaufnahmeschritt müssen zunächst die erforderlichen Bildeinstellungen vorgenommen werden. Dazu gehören beispielsweise das Bildformat, die Auflösung, die Belichtungszeit und die Helligkeit. Zusätzlich muss unter dem Reiter „Steuerung“ die Art der Triggerung für die Bildaufnahme ausgewählt werden. Für den Objekterkennungsschritt muss zunächst eine Modellbibliothek erstellt werden. In dieser können mehrere Modelle hinterlegt werden. Jedes einzelne Modell kann als Klasse definiert werden, so dass nicht nur eine Objekterkennung, sondern auch eine Klassifikation durchgeführt werden kann. Für die Erstellung eines Modells muss ein Bild des jeweiligen Objektes, das gefunden bzw. klassifiziert werden soll, in die Modellbibliothek geladen werden. Im Objekterkennungsschritt kann anschließend ausgewählt werden, mit welcher Modellbibliothek gearbeitet und welche Modelle erkannt werden sollen. Zusätzlich muss die Genauigkeit, mit der ein erkanntes Objekt einer Modellklasse zugeordnet werden soll, eingestellt werden. Hierbei ist darauf zu achten, dass das Programm bei einer zu hohen Genauigkeit länger für die Auswertung benötigt. Außerdem muss ausgewählt werden, wie das Ergebnis ausgegeben werden soll. Es können zum Beispiel die zugehörige Klasse oder die Position des gefundenen Objekts ausgegeben werden. Abschließend wird im Ergebnisschritt die Bildung des Gesamtergebnisses festgelegt und das Programm gespeichert.

2.4. Handarbeitsplatz nach dem Industrie 4.0 Leitbild

Der Begriff Industrie 4.0 beschreibt die Verknüpfung der Produktion mit moderner Kommunikationstechnik. Dabei wird das Ziel verfolgt, die Produktionsleistung durch die effiziente Zusammenarbeit der Komponenten eines Produktionsprozesses zu verbessern. In diesem Zusammenhang ist es wichtig, den Menschen bei seinen Aufgaben zu unterstützen. Daher werden zunehmend Assistenzsysteme eingesetzt.

Diese sollen dafür sorgen, dass die Arbeitsabläufe schneller, effizienter, fehlersicherer und robuster ablaufen. In Form von Pick-by-Light bzw. Put-to-Light Systemen sind diese schon oft vertreten. Ein Handarbeitsplatz nach dem Industrie 4.0 Leitbild sollte daher mit solch einem Assistenzsystem ausgestattet sein.

Im Allgemeinen sollte die Assistenz über ein mobiles Endgerät, wie zum Beispiel Tablets, Smart Watches oder Virtual Reality Brillen, erfolgen. Mit diesen wird der Arbeiter nun mit geeigneten Anleitungen, Videos oder Zeichnungen, Schritt für Schritt durch den Arbeitsprozess geführt [3]. Daher sollte das Assistenzsystem in Echtzeit laufen. Die einzelnen Montageschritte sollten dabei mit Hilfe von Kameraüberwachung oder Datenhandschuhen überwacht werden, damit ein Fehlverhalten des Arbeiters erkannt wird und das Assistenzsystem darauf reagieren kann.

Ein Beispiel für die Realisierung solch eines Assistenzsystems ist die Montagestation aus [4]. Der Hardwareaufbau dieser Montagestation besteht aus einer Microsoft Kinect, einem Tablet PC, einem Bildschirm und einem RFID Lesegerät. Über die RFID Tags, welche sich am Produkt befinden, kann mit Hilfe des RFID Lesegeräts der Produktstatus in Erfahrung gebracht werden. Die Kinect und der Tablet PC verfolgen mittels Kamera den Arbeitsplatz. Wenn ein Werkstück an der Montagestation bearbeitet werden soll, wird eine virtuelle Anleitung aufgerufen. Auf dem Bildschirm vor dem Arbeiter wird nun das Kamerabild des Tablets, welches mit entsprechenden Anweisungen angereichert wurde, angezeigt. Die einzelnen Montageschritte werden während des Arbeitsprozesses mittels Objekt- und Handerkennung verfolgt. Dadurch kann das Assistenzsystem auf Fehler in der Bearbeitung reagieren. Der Aufbau der Montagestation und die Anzeige der virtuellen Anweisungen sind in Abbildung 2.15 zu sehen.

Abbildung 2.15.: Aufbau der Montagestation (links) und Anzeige der virtuellen Anweisung (rechts), Quelle: [4, S.536]

3. Modellfabrik

In diesem Kapitel wird die Montageanlage aus dem Labor für Automatisierungstechnik der HAW Hamburg näher beschrieben. Zunächst wird in Abschnitt 3.1 der Montageprozess dargestellt. Abschnitt 3.2 befasst sich mit den für diese Arbeit relevanten Komponenten der Modellfabrik.

3.1. Prozessbeschreibung

Die Montageanlage bildet einen komplexen vollautomatischen Montage- und Prüfprozess mit in der Industrie verwendeten Komponenten nach. Diese ist modular aufgebaut und besteht aus sechs autarken Teilbereichen, welche einzeln nutzbar sind. Es werden verschiedene Relais (ein- oder zweipolig) im Gehäuse mit Deckel montiert. Das Endprodukt besteht dementsprechend aus drei zusammengefügten Komponenten (Unterteil, Relaiskarte und Deckel mit HAW-Logo). In Abbildung 3.1 ist der schematische Aufbau der Gesamtanlage dargestellt.

In der Lagerstation (Station 20) befinden sich die Gehäuseunterteile. Diese werden mit Hilfe eines pneumatischen Achshandlings den beiden parallelen Montagestrecken zugeführt. An der nachfolgenden Montagestation (Station 30) werden die Gehäuseunterteile mit Relaiskarten bestückt. Ein 6 Achsroboter der Firma Denso entnimmt einem produktionsnahem Zuführlager die Relaiskarten und setzt diese lagerichtig in die Gehäuseunterteile ein. Nach der Montage werden die Werkstücke weitergeleitet und über ein pneumatisches Handling auf einem Werkstückträger des Transfersystems abgelegt. Die Werkstückträger sind mit RFID-Tags ausgestattet, welche Informationen über das Werkstück enthalten. Bei der Beladung mit einem Werkstück aus der Montagestation wird der Relaiskartentyp an den Leitstand (Station 10) gesendet und über das RFID-System der entsprechende RFID-Tag beschrieben. Über eine Pufferstrecke wird das Werkstück den nachfolgenden Stationen zugeführt.

An der elektrischen Prüfstation (Station 40) wird der RFID-Tag ausgelesen und die entsprechende elektrische Prüfung durchgeführt. Das Prüfergebnis wird im Anschluss auf den RFID-Tag geschrieben. Das geprüfte Werkstück wird über die Pufferstrecke der nachfolgenden Verdeckelungsstation (Station 50) zugeführt. Zunächst wird an dieser Station der Wert des RFID-Tags ausgelesen. Die Deckelmontage erfolgt nur bei bestandener elektrischer Prüfung. Defekte Werkstücke werden nicht verdeckelt.

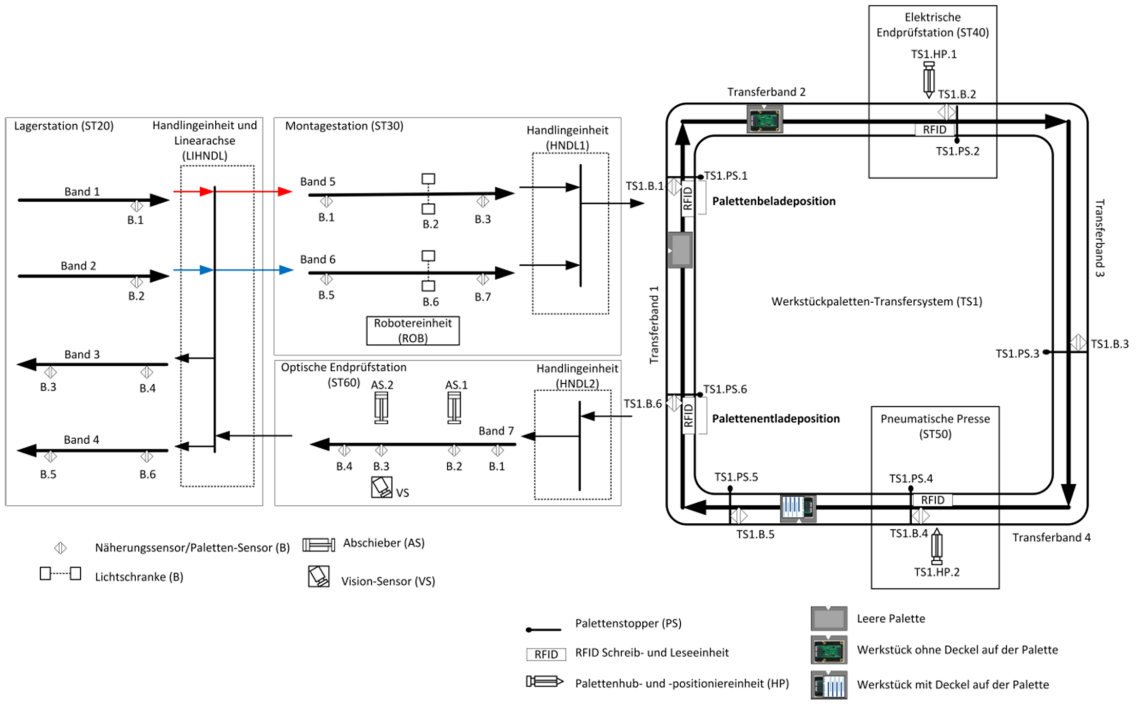


Abbildung 3.1.: Schematischer Aufbau der Modellfabrik, Quelle: [5, S.45]

Anschließend wird das Werkstück an die optische Endprüfstation (Station 60) weitergeleitet. Der Fertigungszustand wird erneut aus dem RFID-Tag gelesen und das Werkstück mittels eines pneumatischem Handlings dem Transfersystem entnommen. Über eine Pufferstrecke wird der leere Werkstückträger zur erneuten Beladung bereitgestellt. Hat das entnommene Werkstück die elektrische Prüfung bestanden, wird es an den Vision Sensor der Firma Siemens weitergeleitet. Die optische Endkontrolle mit Hilfe des Vision Sensors ist derzeit noch nicht implementiert. Zum jetzigen Zeitpunkt erfolgt die optische Endkontrolle mit Hilfe zweier Taster, mit denen der Bediener auswählt, ob das Werkstück optisch in Ordnung ist. Besteht das Werkstück die optische Prüfung, wird es über das pneumatische Achshandling dem Fertigprodukt-Lager (Station 20) zugeführt, ansonsten wird das Werkstück über einen pneumatischen Schieber aussortiert. Des Weiteren werden Werkstücke, welche die elektrische Prüfung nicht bestanden haben, an dieser Station durch den Roboter demontiert und die Unterteile in den Prozess wiedereingelastet.

Die einzelnen Stationen werden jeweils mit einer eigenen speicherprogrammierbaren Steuerung der Firma Siemens gesteuert. Diese sind an ein Profinet-IO-System angeschlossen. Die Kommunikation zwischen den Stationen erfolgt über den Leitstand (Station 10) mit Hilfe von Transferbereichen (E/A-Bereiche). Eine direkte Kommunikation zwischen den Stationen 20 bis 60 findet nicht statt.

3.2. Beschreibung der Komponenten

Nachfolgend werden die für diese Arbeit relevanten Arbeitsstationen detailliert beschrieben. Dabei wird auf deren Komponenten hinsichtlich Funktion und Aufbau eingegangen.

Station 20: Lagerstation

Die Lagerstation besteht aus vier Transportbändern, einer Linearachse und einer Handlingeinheit. Der Aufbau der Station ist in Abbildung 3.2 dargestellt. Zu ihren Aufgaben gehört die Zuführung von Gehäuseunterteilen in den Montageprozess, sowie die Lagerung der fertigen Werkstücke.

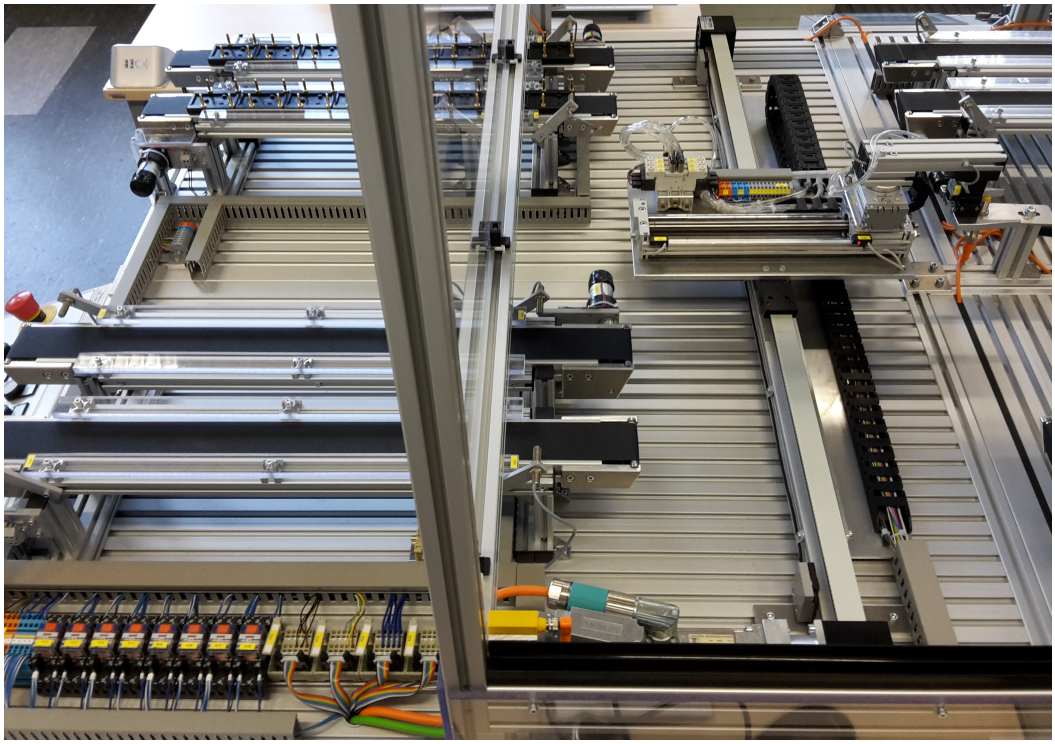


Abbildung 3.2.: Station 20

Die Zufuhr der Gehäuseunterteile erfolgt über die oberen beiden Transportbänder (Band 1 und 2), welche als Eingangslager dienen. Auf Band 1 werden die Gehäuseunterteile für einpolige Werkstücke und auf Band 2 für zweipolige Werkstücke gelagert. Wird von dem Achshandling ein Gehäuseunterteil von einem der beiden Bänder entnommen, läuft das jeweilige Band, bei Rechtsdrehung des Bandmotors, nach rechts bis der jeweilige Näherungssensor (B.1 oder B.2) ausgelöst wird.

Die unteren beiden Bänder (Band 3 und 4) dienen als Ausgangslager. Hier werden die fertigen Werkstücke gelagert. Es wird dabei nicht zwischen einpoligen und zweipoligen Werkstücken unterschieden. Zunächst wird Band 3 gefüllt, bis sowohl der Näherungssensor am Bandende (B.3) als auch der am Bandanfang (B.4) ausgelöst werden. Danach wird mit der Befüllung von Band 4 begonnen. Sobald beide Bänder vollständig belegt sind, werden keine Werkstücke mehr von Station 60 abgeholt. Detektiert einer der Näherungssensoren am Bandanfang (B.4 bzw. B.6) ein Werkstück, dreht der entsprechende Bandmotor für eine kurze Zeit nach links und transportiert das Werkstück ein Stück in Richtung Bandende, um den Anfangsbereich für das nächste Werkstück frei zu machen.

Mit Hilfe der pneumatischen Dreiachs-Handlingeinheit wird Station 30 mit Gehäuseunterteilen aus dem Eingangslager beliefert. Außerdem transportiert die Handlingeinheit die fertigen Werkstücke von Station 60 zum Ausgangslager. Die Handlingeinheit ist mit einem pneumatischen Werkstückgreifer ausgestattet. Über eine Linearachse wird sie zu den einzelnen Bändern gefahren. Diese wird durch einen Servomotor angetrieben und ist mit einem Absolutwertgeber ausgestattet. Die Regelung des Servomotors erfolgt durch den Umrichter Sinamics S120 der Firma Siemens. Über eine Profinet-Schnittstelle kommuniziert der Umrichter mit der übergeordneten SPS von Station 20.

Die Handlingeinheit und Linearachse können folgende Aufträge ausführen:

- Gehäuseunterteil von Band 1 (Station 20) zu Band 5 (Station 30) bringen
- Gehäuseunterteil von Band 2 (Station 20) zu Band 6 (Station 30) bringen
- Fertiges Werkstück von Band 7 (Station 60) zu Band 3 (Station 20) bringen
- Fertiges Werkstück von Band 7 (Station 60) zu Band 4 (Station 20) bringen
- Gehäuseunterteil nach Demontage eines defekten Werkstücks von Band 7 (Station 60) zu Band 5 (Station 30) bringen
- Gehäuseunterteil nach Demontage eines defekten Werkstücks von Band 7 (Station 60) zu Band 6 (Station 30) bringen

Station 60: Optische Endkontrolle

Station 60 besteht aus einem Transportband, einem Vision Sensor, einer pneumatischen Handlingeinheit und zwei pneumatischen Abschiebern. Der Aufbau der Station ist in Abbildung 3.3 dargestellt. An dieser Station findet die optische Endkontrolle der verdeckelten Werkstücke, sowie die Demontage der defekten, nicht verdeckelten Werkstücke statt. Die optische Endkontrolle der Werkstücke mit Hilfe des Vision Sensors ist derzeit noch nicht implementiert. Die Auswahl, ob ein Werkstück optisch in Ordnung ist oder nicht, wird zurzeit manuell über zwei Taster getroffen.

Über die pneumatische Handlingeinheit werden die Werkstücke den ankommenden Werkstückträgern entnommen und dem Transportband (Band 7) zugeführt. Sie besteht aus einer vertikalen Achse, einer Drehachse und einem Greifer. Mit Hilfe der vertikalen Achse lässt sich die Handlingeinheit heben und senken. Band 7 ist mit vier Näherungssensoren für die Lageerkennung der Werkstücke ausgestattet. Diese befinden sich am Bandanfang (B.1) und -ende (B.4), sowie an den beiden Abschiebern (B.2 und B.3).

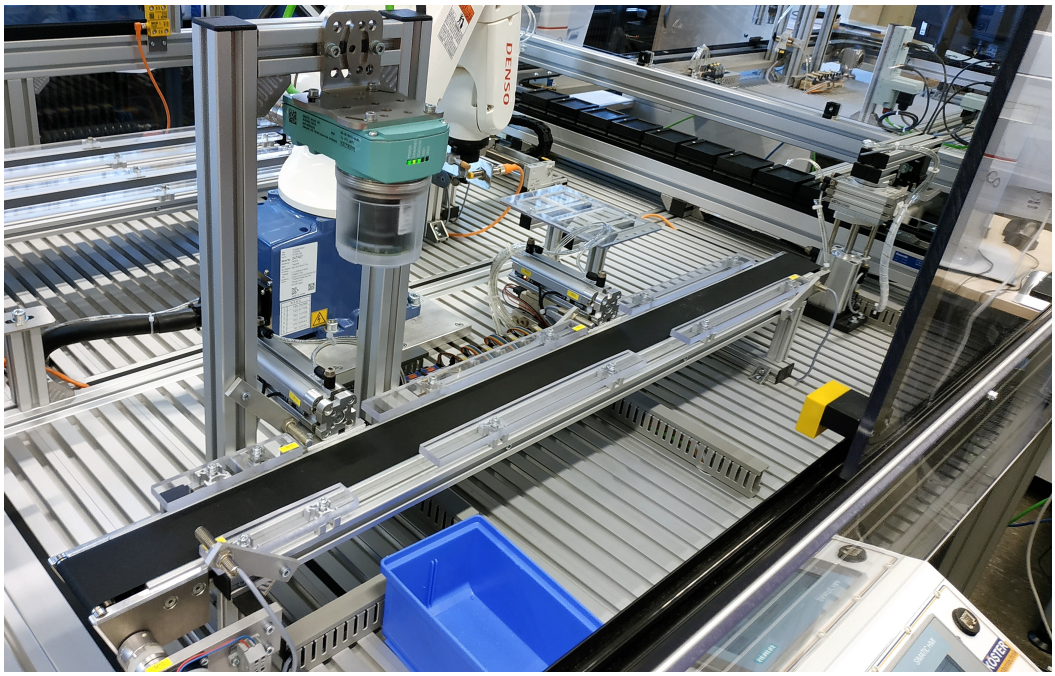


Abbildung 3.3.: Station 60

Trifft ein Werkstückträger an Station 60 ein, wird zunächst dessen RFID-Tag ausgelesen. Anschließend wird es mit Hilfe der Handlingeinheit auf das Transportband gelegt. Handelt es sich dabei um ein defektes Werkstück, wird dieses bis zum ersten pneumatischen Abschieber transportiert. Das defekte Werkstück wird entweder durch den Abschieber aussortiert oder durch den Roboter demontiert und das Gehäuseunterteil wieder in den Montageprozess eingelastet. Für die Demontage und Wiedereinlastung kann zwischen vier Steuerungsvarianten ausgewählt werden:

1. Ist der Produktionsauftrag noch nicht abgeschlossen, entfernt der Roboter die defekte Platine und bringt das Gehäuseunterteil zum entsprechenden Montageband an Station 30. Wird kein weiteres Werkstück produziert, wird das Fehlteil abgeschoben.
2. An Station 60 befindet sich ein Lager für Gehäuseunterteile in dem vier Unterteile eingelagert werden können. Ist in diesem Lager ein Platz frei, entfernt der Roboter

die defekte Platine und lagert das Gehäuseunterteil ein. Ist das Lager voll, wird das Fehlteil abgeschoben. Wird ein neues Werkstück produziert werden zunächst die Gehäuseunterteile aus dem Lager verwendet.

3. Ist der Produktionsauftrag noch nicht abgeschlossen, entfernt der Roboter die defekte Platine. Das Gehäuseunterteil wird zum Bandende von Station 60 gefahren und von der Linearachse zum entsprechenden Montageband transportiert. Wird kein weiteres Werkstück produziert, wird das Fehlteil abgeschoben.

4. Diese Steuerungsvariante ist eine Kombination aus Variante eins und zwei. Ist der Produktionsauftrag noch nicht abgeschlossen, wird das Unterteil, nach dem Entfernen der Platine, von dem Roboter direkt zum entsprechenden Montageband gebracht. Ist dies nicht der Fall, wird das Gehäuseunterteil eingelagert.

Werkstücke, die die elektrische Prüfung bestanden haben und verdeckelt sind, laufen bis vor den Sensor am zweiten Abschieber (B.3), der sich unter dem Vision Sensor befindet. Die optische Endkontrolle wird vom Bediener über zwei Taster am Bedienpult ausgeführt. Besteht ein Werkstück die optische Endkontrolle nicht, wird es von dem pneumatischen Abschieber unter dem Vision Sensor abgeschoben. Werkstücke, die die optische Endkontrolle bestehen, werden zum Bandende transportiert. Der verwendete Vision Sensor Simatic MV440 der Firma Siemens ist in Abschnitt 2.3 beschrieben.

4. Konzeption

In diesem Kapitel werden verschiedene Möglichkeiten zur Realisierung der Aufgabenstellung vorgestellt und miteinander verglichen. Die Auswahl eines geeigneten Konzepts für den Aufbau des Handarbeitsplatzes wird in Abschnitt 4.1 durchgeführt. In Abschnitt 4.2 wird die Auswahl der Beschreibungsform, des Entwicklungstools und des Entwurfsansatzes begründet.

4.1. Aufbau und Lage des Handarbeitsplatzes

Für die Bearbeitung von optischen Fehlteilen soll ein Handarbeitsplatz nach dem Industrie 4.0 Leitbild konzipiert werden. Die optischen Fehlteile können nicht von dem vorhandenen Roboter bearbeitet werden, da der Saugaufsatz des Roboters die Gehäusedeckel nicht demontieren kann. Daher soll die Bearbeitung durch einen Montagemitarbeiter durchgeführt werden. Hierfür müssen die Werkstücke, die von dem Vision Sensor als optisch nicht in Ordnung klassifiziert wurden, aus dem Prozess hinausgeführt werden. Auf Grund der Lage der Stationen und des vorhandenen Platzes wird der Handarbeitsplatz an Station 20 zwischen dem Eingangslager und dem Fertigteillager errichtet. Es steht eine Fläche von 390 $\times$ 690 mm (Breite $\times$ Länge) zur Verfügung. Durch die Wahl von Station 20 als Standort, können die Fehlteile mit Hilfe des pneumatischen Achshandlings von Station 60 direkt zum Handarbeitsplatz transportiert werden.

Mit Hilfe des Vision Sensors können verschiedene Fehlertypen unterschieden und erkannt werden. Durch die Beschaffenheit des Deckels besteht die Möglichkeit, dass dieser nicht in der richtigen Ausrichtung montiert wird. Dies führt dazu, dass das Logo, welches sich auf dem Deckel befindet, ebenfalls falsch ausgerichtet ist. Ein weiterer möglicher Fehler, der auftreten kann, ist das Fehlen des Logos auf dem Deckel. Insgesamt wird zwischen den folgenden drei Fehlertypen unterschieden:

- Fehler A: Das Logo auf dem Deckel fehlt.
- Fehler B: Der Deckel und das Logo sind um 180° verdreht.
- Fehler C: Der Deckel und das Logo sind um +/-90° verdreht.

In Abbildung 4.1 ist eine Übersicht der Fehlertypen dargestellt. Für die Bearbeitung der Fehlteile gilt es, diese Fehlertypen zu unterscheiden und zu sortieren.

Abbildung 4.1.: Übersicht der Fehlertypen (Fehler A, B, C von links nach rechts)

Der Handarbeitsplatz muss so konzipiert werden, dass eine Sortierung der Fehlertypen möglich ist. Des Weiteren soll der Handarbeitsplatz dem Industrie 4.0 Leitbild entsprechen, daher wird dieser mit einem Werkerassistenzsystem ausgestattet. Der entwickelte Handarbeitsplatz soll die folgenden Anforderungen erfüllen:

1. Sortierung der Fehlteile entsprechend der drei Fehlerkategorien.
2. Der Handarbeitsplatz darf die vorhandene Fläche von 390×690 mm nicht überschreiten.
3. Die Kosten müssen innerhalb eines Budgets von 5500 Euro liegen.
4. Eine Blockierung des Montageprozesses ist zu vermeiden, daher ist ein Überlaufschutz einzuplanen.
5. Assistenz des Montagearbeiters durch ein Werkerassistenzsystem, welches mit einer grafischen Benutzeroberfläche für eine einfache Bedienung ausgestattet ist.

In diesem Abschnitt werden drei verschieden Möglichkeiten für den grundlegenden Aufbau vorgestellt, hinsichtlich der genannten Anforderungen bewertet und miteinander verglichen. Bei der Bewertung werden nur die Kriterien 1–4 berücksichtigt. Die Realisierung der Anforderung 5 hängt nicht von dem Aufbau des Handarbeitsplatzes ab. Daher wird die Konzeption des Assistenzsystems separat betrachtet.

4.1.1. Variante 1: Werkstückrutschen

Eine mögliche Variante für den Aufbau des Handarbeitsplatzes ist eine Werkstückzuführung über Rutschen. Dabei werden die optischen Fehlteile mit Hilfe des pneumatischen Achs-

handlings an Station 60 abgeholt und auf einer Werkstückrutsche abgelegt. Über die Rutsche werden die Werkstücke aus dem Prozess und dem Schutzkäfig herausgeführt, sodass diese für den Montagemitarbeiter erreichbar sind. Bei dieser Variante ist zu beachten, dass für jede Fehlerkategorie eine eigene Werkstückrutsche benötigt wird, damit eine Sortierung der Fehlteile möglich ist. Der Sortierprozess wird somit von dem pneumatischen Achshandling übernommen. Wird nur eine einzelne Werkstückrutsche verwendet, ist eine Sortierung nicht möglich, da die Werkstücke nicht in unterschiedliche Zwischenlager geschleust werden können. Eine Skizze des Aufbaus ist in Abbildung 4.2 dargestellt.

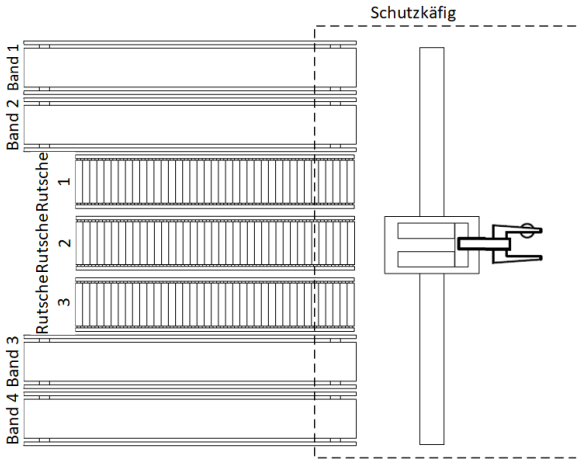


Abbildung 4.2.: Skizze des schematischen Aufbaus von Variante 1

Das Problem bei dieser Variante liegt im Platzbedarf des Aufbaus. Die produzierten Werkstücke sind 50 mm breit. Rutschen, welche für diese Werkstückbreite ausgelegt sind besitzen je nach Verarbeitung und Werkstückführung eine Breite von 60–100 mm. Daraus resultiert bei drei Rutschen eine Breite von 180–300 mm. Hinzu kommt, dass sich am Anfang der Rutschen Sensoren befinden müssen. Diese werden für die Rückmeldung an das pneumatische Achshandling, dass das Ziel erreicht wurde, benötigt. Auch dient es als Meldung an das Assistenzsystem, dass ein Fehlteil am Handarbeitsplatz angekommen ist. Diese Sensoren und deren Verdrahtung müssen ebenfalls in der Gesamtbreite des Aufbaus einberechnet werden. Somit ist die in den Anforderungen vorgegebene Breite von 390 mm schnell ausgelastet.

Der Vorteil dieser Variante besteht darin, dass es ein sehr kostengünstiger Ansatz ist. Eine Rutschenzuführung von Festo Didactic kann bereits für 230 Euro und ein geeigneter Näherungssensor für 100 Euro erworben werden. Hinzu kämen die Kosten für die Steuerungshardware und die für die Visualisierung benötigten Komponenten, welche in jedem der drei Ansätze benötigt werden. Die Gesamtkosten lägen somit innerhalb des Budgets.

Die Realisierung des Überlaufschutzes ist in dieser Variante nur an einer anderen Station

möglich, da an Station 20 nicht genügend Platz für eine weitere Werkstückrutsche vorhanden ist. Diese wird jedoch benötigt, wenn der Überlauschutz direkt am Handarbeitsplatz realisiert werden soll.

4.1.2. Variante 2: Drehteller

Eine weitere Möglichkeit zur Zuführung der Fehlteile ist die Verwendung eines Drehtellers. Bei dieser Variante werden die optischen Fehlteile von dem pneumatischen Achshandling auf einem Drehteller, welcher mit einem Näherungssensor ausgestattet ist, abgelegt. Der Montagemitarbeiter hat die Möglichkeit diesen zu drehen und das Werkstück außerhalb des Schutzkäfigs zu entnehmen. Abbildung 4.3 zeigt den schematischen Entwurf des Aufbaus.

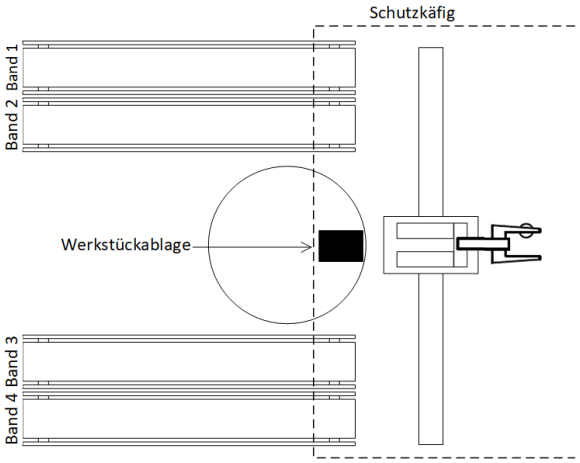


Abbildung 4.3.: Skizze des schematischen Aufbaus von Variante 2

Wird ein Werkstück auf dem Drehteller abgelegt, löst der Näherungssensor aus. Dieses Ereignis meldet dem Montagemitarbeiter, dass ein Werkstück zur Reparatur vorliegt. Nun kann der Drehteller so gedreht werden, dass sich das Fehlteil außerhalb des Schutzkäfigs befindet. Der Vorteil dieser Variante ist, dass sie besonders kostengünstig und platzsparend ist. Der Drehteller kann selbst hergestellt werden, sodass nur der Näherungssensor, die Steuerungshardware und die Visualisierungskomponenten angeschafft werden müssten. Alternativ kann auch ein elektrisch angetriebener Drehteller erworben werden.

Der Nachteil bei dieser Variante ist, dass keine Sortierung nach Fehlertypen erfolgt. Es befindet sich immer nur ein Fehlteil zurzeit auf dem Drehteller. Ein weiteres Problem liegt darin, dass ein Montagestau auftreten kann, wenn das Fehlteil nicht sofort bearbeitet wird. Dies verlangsamt den Montageprozess. Wie in Variante eins ist die Realisierung eines Überlaufschutzes nur an einer anderen Station möglich.

4.1.3. Variante 3: Transportband

Eine weitere Variante für den Handarbeitsplatz ist ein Aufbau aus einem Transportband und drei pneumatischen Abschiebern. Hier werden die optischen Fehlteile mit Hilfe des Transportbands aus dem Prozess herausgeführt. Die Sortierung erfolgt mittels der Abschieber. Diese schleusen die Werkstücke je nach Fehlerkategorie vom Band aus. Der schematische Aufbau ist in Abbildung 4.4 dargestellt.

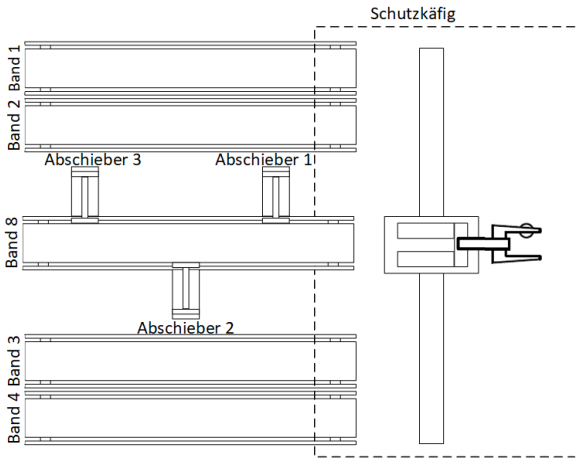


Abbildung 4.4.: Skizze des schematischen Aufbaus von Variante 3

Befindet sich ein Werkstück, welches optisch nicht in Ordnung ist, am Bandende von Station 60 wird dieses von dem pneumatischen Achshandling abgeholt und zum Transportband des Handarbeitsplatzes transportiert. Das Band fährt das Werkstück bis zu dem entsprechenden Abschieber der jeweiligen Fehlerkategorie. Befindet sich das Werkstück vor diesem Abschieber, wird dieser ausgefahren und das Fehlteil vom Band ausgeschleust. Die Positionsbestimmung der Werkstücke auf dem Transportband erfolgt mit Näherungssensoren am Bandanfang und an den Abschiebern.

Für diese Variante werden ausschließlich Komponenten verwendet, die bereits an den Stationen 20 und 60 verbaut sind, sodass deren Maße und Preise bekannt sind. Der Aufbau in dieser Variante benötigt eine Fläche von 320×690 mm und erfüllt somit Anforderung zwei. Die Kosten dieser Variante belaufen sich auf etwa 2500 Euro und liegen somit im vorgegebenen Budget.

Der Hauptvorteil dieses Aufbaus ist, dass die Sortierung der Fehlteile am Handarbeitsplatz erfolgt und nicht durch die Linearachse durchgeführt wird. Dadurch wird diese entlastet und es müssen weniger neue Positionen eingelernt werden. Ein weiterer Vorteil dieser Variante

besteht darin, dass die Realisierung eines Überlaufschutzes sowohl direkt am Handarbeitsplatz, sowie an einer anderen Station möglich ist. Wird sich für eine Realisierung am Handarbeitsplatz entschieden, können die Fehlteile, sobald die Puffer voll sind bis zum Bandende gefahren werden.

Für diese Variante muss die SPS an Station 20 erweitert werden. Es stehen nicht genügend freie digitale Ein- und Ausgänge für die Ansteuerung der Hardware zur Verfügung. Dies erhöht die Gesamtkosten, dieses Aufbaus und ist somit ein Nachteil. Die Gesamtkosten liegen jedoch weiterhin innerhalb des vorgegebenen Budgets.

4.1.4. Vergleich der Varianten

Für die Entscheidung, welche der drei Varianten für den Handarbeitsplatz angewendet wird, werden sie miteinander verglichen. Dafür werden die Anforderungen anhand ihrer Relevanz gewichtet, um ein aussagekräftiges Ergebnis zu erhalten. Gewichtet werden die Anforderungen mit dem Faktor 5 „sehr wichtig“ bis 1 „unwichtig“. Als sehr wichtig werden die Anforderungen zwei und drei eingestuft, da nur eine begrenzte Fläche und ein begrenztes Budget zur Verfügung stehen. Diese dürfen auf keinen Fall überschritten werden. Die Anforderungen eins und vier werden mit dem Faktor 3 gewichtet. Die Sortierung der Fehlteile nach Fehlerkategorien wird zwar gefordert, hat aber keine Auswirkung auf die Grundfunktion des Handarbeitsplatzes und ist somit nicht so wichtig wie die Einhaltung des Budgets und der Fläche. Für die Realisierung des Überlaufschutzes verhält es sich ebenso.

Die Varianten werden nun hinsichtlich der gegebenen Anforderungen bewertet. Die Bewertung erfolgt von 5 (++) „trifft zu“ bis 1 (- -) „trifft nicht zu“. Eine Sortierung der Fehlteile ist nur in den Varianten eins und drei möglich, daher werden diese mit 5 (++) bewertet. Variante zwei erfüllt diese Anforderung nicht und erhält daher eine Bewertung von 1 (- -). Die zur Verfügung stehende Fläche wird von allen drei Varianten eingehalten, jedoch benötigt der Aufbau in Variante zwei weniger Platz als die anderen beiden und wird daher am besten bewertet. Die Einhaltung des vorgegebenen Budgets wird ebenfalls von allen Varianten erfüllt. Der Aufbau aus Variante drei ist der teuerste und wird daher nur mit 4 (+) bewertet. Ein Überlaufschutz ist für alle Varianten realisierbar. Auf Grund dessen, dass in Variante drei der Überlaufschutz sowohl an einer anderen Station als auch am Handarbeitsplatz realisiert werden kann, erhält diese eine bessere Bewertung. In Tabelle 4.1 ist die Bewertung der Varianten zusammengefasst.

Tabelle 4.1.: Vergleich der Varianten

Anforderung	Gewichtung	Variante 1	Variante 2	Variante 3
Sortierung nach Fehlerkategorien	3	5 (++)	1 (- -)	5 (++)
Benötigte Fläche	5	3 (○)	5 (++)	4 (+)
Budget wird eingehalten	5	5 (++)	5 (++)	4 (+)
Realisierung eines Überlaufschutzes	3	4 (+)	4 (+)	5 (++)
Gesamt		67	65	70

Aus Tabelle 4.1 geht hervor, dass Variante drei die gegebenen Anforderungen am besten erfüllt. Daher wird der Handarbeitsplatz nach diesem Konzept realisiert. Der zu realisierende Handarbeitsplatz wird aus einem Transportband mit drei Abschiebern, welche mit Endlagenschaltern ausgestattet sind, und Näherungssensoren an den Abschiebern, sowie am Bandanfang und Bandende, bestehen.

4.1.5. Konzept des Assistenzsystems

In Abschnitt 2.4 wurden die Grundlagen zu Assistenzsystemen nach dem Industrie 4.0 Leitbild vorgestellt. Für die Realisierung eines solchen Systems gibt es, je nach Anforderungen, verschiedene Möglichkeiten. In dieser Arbeit werden folgende Anforderungen an das Assistenzsystem gestellt:

1. Das Assistenzsystem soll dem Mitarbeiter eine Anleitung für die Reparatur der Fehlteile grafisch zur Verfügung stellen.
2. Die Assistenz soll über ein mobiles Endgerät erfolgen, welches von jedem Mitarbeiter komfortabel bedient werden kann und ihn bei der Montage nicht behindert.
3. Ein Fehlgriff des Arbeiters soll vermieden werden.

Für die Umsetzung der ersten beiden Anforderungen eignen sich beispielsweise Virtual Reality Brillen (VR Brille) oder Tablets. Bei beiden handelt es sich um optische Assistenzsysteme. Mit Hilfe einer VR Brille kann die Anleitung für die Reparatur in das natürliche Sichtfeld integriert werden. Der Vorteil dabei besteht darin, dass die Nutzung des Geräts nicht ortsgebunden ist und der Arbeiter beide Hände frei hat. Der Nachteil bei diesen Geräten liegt in dem hohen Lernaufwand. Für jede neue Anwendung muss das Gerät neu eingelernt werden. Außerdem ist eine VR Brille im Vergleich zu einem Tablet sehr kostenintensiv.

Die Umsetzung des Assistenzsystems mit einem Tablet ermöglicht eine einfache Bedienung per Touch. Über eine grafische Benutzeroberfläche kann der Arbeiter auswählen, welche Anleitung er benötigt und den Vorgang quittieren. Der Nachteil dieser Variante besteht darin,

dass der Arbeitsprozess für die Bedienung des Tablets unterbrochen werden muss. Außerdem muss das Tablet fest montiert werden, damit der Mitarbeiter beide Hände für die Reparatur frei hat. Dabei ist darauf zu achten, dass es trotz fester Montage bewegt werden kann (zum Beispiel mit einem höhenverstellbaren Monitorarm). Im Vergleich zur VR Brille ist dies eine kostengünstigere Variante.

Auf Grund der in der Modellfabrik verwendeten Steuerungshardware fällt die Entscheidung auf ein Touchpanel der Firma Siemens zur Umsetzung des Assistenzsystems. Dies ermöglicht die Bedienung über eine grafische Benutzeroberfläche und die Integration in die vorhandene Hardware ohne großen Aufwand. Für eine komfortable Bedienung wird das Touchpanel an einem Monitorarm montiert, welcher höhenverstellbar ist. Somit werden die Anforderungen eins und zwei erfüllt.

Für die Umsetzung der dritten Anforderung gibt es verschiedene Möglichkeiten. In den meisten bestehenden Assistenzsystemen ist eine Kameraüberwachung des Arbeitsplatzes integriert, um einen Fehlgriff des Montagearbeiters zu vermeiden. Dies ist jedoch mit einem hohen bildverarbeitungstechnischen Aufwand verbunden. Auf Grund der geringen Größe des Handarbeitsplatzes und den eng zusammenstehenden Komponenten gestaltet sich die Überwachung mit Hilfe von Bildverarbeitung als schwierig. Außerdem kann dies zu Ungenauigkeiten führen.

Eine weitere Möglichkeit ist die Montage von Lichtschranken an den einzelnen Zwischenlagern. Nimmt der Mitarbeiter ein Fehlteil aus der entsprechenden Kiste, wird die Lichtschranke ausgelöst. Wird nach der Auswahl des zu bearbeitenden Fehlers die falsche Lichtschranke ausgelöst, kann das System dem Mitarbeiter mitteilen, dass die aufgerufene Anleitung nicht zu dem entnommenen Fehlteil gehört und somit eine falsche Bearbeitung verhindern. Der Nachteil dieser Variante liegt in der Ungenauigkeit der Überwachung. Bei der Entnahme eines Fehlteils kann an der Lichtschranke vorbeigegriffen werden. Außerdem können die Fehlteile, sobald sie vom Band geschoben werden, in der entsprechenden Kiste übereinander liegen und somit die Lichtschranke auslösen.

Eine Alternative, die keine direkte Überwachung ermöglicht, ist die Nutzung des Pick-by-Light Verfahrens. Hier werden die Kisten mit einer Meldeleuchte ausgestattet, die dem Mitarbeiter signalisiert, dass etwas entnommen werden soll. Nach Auswahl des zu bearbeitenden Fehlers leuchtet die Meldeleuchte an der entsprechenden Kiste. Mit dieser Variante wird ein Fehlgriff jedoch nicht verhindert. Die Realisierung gestaltet sich als einfach und kostengünstig.

Im Vergleich dieser drei Varianten hinsichtlich Kosten, Nutzen und Aufwand, ist ersichtlich, dass die Realisierung eines Pick-by-Light Systems für diese Anwendung ausreichend ist. Daher wird diese Variante in dieser Arbeit realisiert.

4.2. Auswahl der Beschreibungsform, des Entwicklungstools und des Steuerungsansatzes

In diesem Abschnitt wird die Auswahl der Beschreibungsform, des Entwicklungstools und des Steuerungsansatzes begründet.

4.2.1. Auswahl der Beschreibungsform

Für die Modellierung und Analyse von ereignisdiskreten Systemen eignen sich besonders netzartige Beschreibungsformen. Dazu gehören zum Beispiel die in Abschnitt 2.1.3 vorgestellten Generatoren. Eine weitere mögliche Beschreibungsform sind Petrinetze. Diese wurden 1962 von Carl Adam Petri in seiner Dissertation [15] vorgestellt. Petrinetze generieren ebenfalls Sprachen und können als ereignisdiskrete Beschreibungsform genutzt werden. Die Entscheidung, welche der beiden Beschreibungsformen am besten für die jeweilige Anwendung geeignet ist, hängt von verschiedenen Faktoren ab.

Petrinetze stellen eine Erweiterung von Automaten dar, mit deren Hilfe sich parallele Prozesse besser darstellen lassen. Sie finden vor allem Anwendung in der prozessorientierten Modellbildung. Der Vorteil besteht darin, dass nebenläufige Teilprozesse und deren Kopplungen im Petrinetz durch parallele Pfade strukturell sichtbar dargestellt werden. Generatoren hingegen eignen sich für die Darstellung von parallelen Teilprozessen nicht so gut, da die Modelle äußerst komplex werden. Für die Darstellung von sequentiellen Prozessen sind beide Beschreibungsformen geeignet und unterscheiden sich nicht. In Bezug auf die Modellfabrik ist die Modellierung von parallelen Prozessen weniger relevant, da keine bis wenige Prozesse parallel ausgeführt werden.

Abgesehen von den anlagenspezifischen Anforderungen müssen für eine Entscheidung weitere Faktoren berücksichtigt werden. Dazu gehören die formale Analysierbarkeit der Modelle, sowie die Existenz von Tools zur Steuerungssynthese nach SCT. Für Generatoren gibt es eine Vielzahl an Tools, in denen die Steuerungssynthese implementiert ist. Diese Tools sind häufig mit einem grafischen Editor ausgestattet, welcher die Erstellung von Generatoren erleichtert. Für die Steuerungssynthese nach SCT auf Basis von Petrinetzen ist nur ein Tool bekannt. Dies ist die Toolbox SPNBOX [9] für MathWorks MATLAB. Die Erstellung von Petrinetzen mit SPNBOX erfolgt ohne grafischen Editor.

Des Weiteren gestaltet sich die formale Analysierbarkeit von Generatoren deutlich leichter als bei Petrinetzen. In einem Generator kann nicht mehr als ein Zustand gleichzeitig aktiv sein. In Petrinetzen hingegen können mehrere Stellen mit einer Marke belegt sein. Dies führt dazu, dass der aktuelle Zustand des Gesamtsystems aus mehreren Teilzuständen besteht,

die sich unabhängig voneinander verändern können. Die Analyse von Petrinetzen bezüglich ihrer Erreichbarkeit setzt die Berechnung eines Erreichbarkeitsgraphen voraus. Für die Berechnung des Erreichbarkeitsgraphen kehrt man auf die Beschreibungsebene von Generatoren zurück, daher ist diese meistens nicht effizient durchführbar. Eine automatische Codegenerierung ist für beide Beschreibungsformen mit Hilfe des Codegenerators ACArrow möglich.

Nach Abwägung der genannten Vor- und Nachteile steht die Entscheidung für Generatoren als Beschreibungsform für die Modellfabrik und Spezifikationen fest.

4.2.2. Auswahl des Entwicklungstools

Für die erfolgreiche Anwendung der SCT sind Entwicklungswerkzeuge zur Modellierung, Analyse und Steuerungssynthese essentiell. Zur ereignisdiskreten Steuerungssynthese nach SCT existieren zahlreiche akademische Tools, welche auf Generatoren und formalen Sprachen basieren. Zu den bekanntesten frei erhältlichen Tools gehören TCT, Supremica und DESTool mit libFaudes.

Das Entwicklungstool TCT wurde an der Universität Toronto in der Arbeitsgruppe von W.M. Wonham entwickelt. Es verfügt über eine umfangreiche Sammlung an effizienten Analyse- und Synthesefunktionen. Ein einfacher Einstieg in das Tool wird durch die zahlreichen Publikationen und Dokumentationen, wie zum Beispiel [7], [29] und [23], in denen TCT Anwendung findet, ermöglicht. Das Tool besitzt eine textuelle Oberfläche, die Bedienung erfolgt über die Eingabe von Befehlen in einem Konsolenfenster. Die fehlende grafische Benutzeroberfläche wirkt sich nachteilig auf die Übersichtlichkeit und Bedienbarkeit aus.

Das Tool Supremica bietet im Gegensatz zu TCT eine grafische Bedienoberfläche und besitzt einen Codegenerator. Dieser ermöglicht eine automatische Codegenerierung in verschiedenen Programmiersprachen, wie zum Beispiel ANSI-C. Supremica besitzt eine Vielzahl an Analyse- und Synthesefunktionen und wird ständig weiter entwickelt. Die Bedienung dieses Tools ist im Gegensatz zu DESTool weniger komfortabel.

Das Entwicklungstool DESTool wurde in Abschnitt 2.2.1 vorgestellt. Es besitzt eine grafische Bedienoberfläche und ist leicht zu bedienen. DESTool überzeugt durch eine große Anzahl an Analyse- und Synthesefunktionen, eine gute Dokumentation und Übersichtlichkeit. Mit Hilfe der eingebundenen Simulationsfunktion kann das Verhalten der gesteuerten Strecke simuliert und analysiert werden. Auf Grund dieser Vorzüge wird in dieser Arbeit DESTool zur Modellbildung, Analyse und Steuerungssynthese verwendet.

4.2.3. Auswahl des Steuerungsansatzes

Für die Steuerungssynthese stehen der monolithische, der modulare, der lokal-modulare und der dezentrale Ansatz zur Verfügung. Diese wurden in Abschnitt 2.1.7 erläutert. Die vier Ansätze werden nachfolgend hinsichtlich verschiedener Streckeneigenschaften und ihrer algorithmischen Komplexität miteinander verglichen.

Für den monolithischen Ansatz wird ein unstrukturiertes Gesamtmodell der Strecke, sowie eine Gesamtspezifikation für die globale Steuerungsaufgabe benötigt. Dies führt zu einer hohen Modellkomplexität. Daher ist dieser Ansatz für komplexe Systeme nicht flexibel. Bei dem modularen Ansatz wird keine Gesamtspezifikation benötigt. Die globale Steuerungsaufgabe wird in mehrere modulare Spezifikationen aufgeteilt. Der modulare Supervisor weist dadurch im Allgemeinen eine geringere Modellkomplexität als der monolithische auf. Für den modularen Ansatz wird jedoch weiterhin ein unstrukturiertes Gesamtmodell der Strecke zur Überprüfung der Steuerbarkeit und des Nichtblockierens benötigt. Dieser Ansatz eignet sich besonders, wenn alle Ereignisse global zur Verfügung stehen und die Spezifikationen partitionierbar sind.

Der lokal-modulare Ansatz benötigt im Gegensatz zum modularen kein unstrukturiertes Gesamtmodell der Strecke. Durch die Modularisierung der Strecke und der Spezifikationen sinkt die algorithmische Komplexität bei der Supervisorsynthese und der Modellanalyse. Dieser Ansatz eignet sich besonders, wenn die Strecke viele asynchrone Komponenten enthält. Eine Erweiterung des modularen Ansatzes ist der dezentrale Ansatz. Dieser eignet sich besonders für verteilte Steuerungen, bei denen nicht alle Streckenereignisse global für jede Steuerung zur Verfügung stehen.

Der monolithische und der dezentrale Ansatz sind für die Steuerungssynthese bei dieser Anwendung weniger geeignet. Durch das Zusammenwirken vieler Komponenten entsteht bei dem monolithischen Ansatz im Vergleich zu den anderen Ansätzen eine große Modellkomplexität. Dies führt zu einem erhöhten Zeit- und Speicheraufwand bei der Synthese. In der Modellfabrik sind die einzelnen Stationen und somit auch deren Steuerungshardware miteinander vernetzt. Dadurch stehen alle Streckenereignisse global für jeden Supervisor zur Verfügung. Daher kann der dezentrale Ansatz in diesem Fall keine geringere algorithmische Komplexität als der modulare Ansatz erzielen.

Es eignen sich für diese Anwendung sowohl der modulare als auch der lokal-modulare Ansatz. Für den modularen Ansatz müssten jedoch im Vergleich zum lokal-modularen Ansatz alle Streckenereignisse zur Verfügung gestellt werden. Dies führt zu einem vermeidbaren hohen Datenaustausch. Außerdem wird für den modularen Ansatz weiterhin ein unstrukturiertes Gesamtmodell der Strecke benötigt. Dies ist beim lokal-modularen Ansatz nicht der Fall. Daher wird für die in dieser Arbeit betrachtete Problemstellung der lokal-modulare Ansatz

verwendet. Betrachtet man die in Abbildung 5.9 dargestellten Relationen zwischen den Ereignisalphabeten der Strecke und der Spezifikationen ist ersichtlich, dass keine Spezifikation alle Streckenkomponenten einschränkt. Daher muss als lokales System nicht das Kompositionsmodell berechnet werden. Somit weist dieser Ansatz eine geringere Modellkomplexität auf als der modulare Ansatz.

5. Modellbildung und Steuerungsentwurf

In diesem Kapitel wird die Modellbildung der Streckenkomponenten und Spezifikation, sowie die Steuerungssynthese durchgeführt. Zu Beginn des Kapitels werden Annahmen zur Modellbildung und zum Steuerungsentwurf getroffen. Abschnitt 5.2 beschreibt die Modellbildung der verwendeten Streckenkomponenten und Abschnitt 5.3 die der Spezifikationen. Abschließend erfolgt die Synthese der Supervisor.

5.1. Annahmen zur Modellbildung und zum Steuerungsentwurf

Für die Modellbildung und den Steuerungsentwurf werden die folgenden Annahmen getroffen, um die Modellkomplexität zu reduzieren:

- Es wird davon ausgegangen, dass alle Hardwarekomponenten korrekt arbeiten. Es wird nur der fehlerfreie Betrieb berücksichtigt.
- Bei der Modellierung wird nur das logische Verhalten der Komponenten berücksichtigt. Die Zeitpunkte, zu denen die Ereignisse auftreten, werden nicht einbezogen.
- Ein manuelles Auslösen von Sensoren wird ausgeschlossen. Ausschließlich Werkstücke können die Sensoren auslösen.
- Die Inbetriebnahme der Modellfabrik erfolgt gemäß der Anleitung im Anhang. Dabei wird an Station 20 die Steuerungsvariante 4 ausgewählt.
- Nach dem Einschalten der Anlage befinden sich sämtliche Komponenten in ihrem definierten Initialzustand.
- Es befinden sich zu jeder Zeit ausreichend Unterteile auf den Zufuhrbändern, sowie ausreichend Platinen und Deckel in den dafür vorgesehenen Magazinen.
- Ein manuelles Eingreifen in den Prozess ist nur zur Bearbeitung von optischen Fehlteilen am Handarbeitsplatz und zur Leerung des Fertigteillagers gestattet.

- Die Türen des Schutzkäfigs dürfen während der Produktion nicht geöffnet werden.

5.2. Modellbildung der Streckenkomponenten

Für die in Kapitel 3 vorgestellte Modellfabrik soll eine ereignisdiskrete Lösung für das Handling von optischen Fehlteilen entworfen und implementiert werden. Dazu gehört die Erkennung von optischen Fehlteilen, der Transport zum Handarbeitsplatz, sowie die Steuerung des Handarbeitsplatzes. Für die Steuerung der grundlegenden Funktionen der Modellfabrik besteht bereits eine untergeordnete Steuerung, daher müssen nur die Komponenten modelliert werden, die für die Lösung der Steuerungsaufgabe relevant sind. Für die spätere Codegenerierung werden nicht steuerbare Ereignisse mit „E“ und steuerbare Ereignisse mit „STR“ gekennzeichnet.

5.2.1. Modellbildung der optischen Fehlererkennung G_1

Die Erkennung von optischen Fehlteilen erfolgt über den Vision Sensor. Dieser wird durch eine unterlagerte Steuerung gesteuert und wird daher nicht modelliert. Für die zu entwerfende Steuerung wird jedoch die Information benötigt, ob ein Werkstück optisch in Ordnung oder fehlerhaft ist. Modelliert man diese Eigenschaft des Werkstücks, erhält man einen Generator der aus zwei Zuständen besteht. Das generische Modell des Werkstücks ist in Abbildung 5.1 als Generator dargestellt.

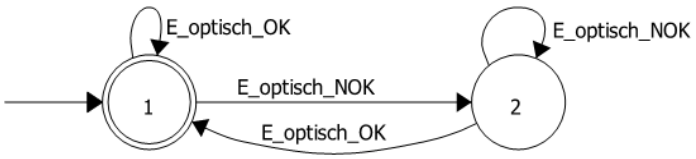


Abbildung 5.1.: Generisches Modell eines Werkstücks

Der Initialzustand des Generators gibt an, dass kein optisches Fehlteil detektiert wurde. Registriert der Vision Sensor ein optisches Fehlteil, wechselt der Generator in den zweiten Zustand. Dieser repräsentiert ein Werkstück, das optisch nicht in Ordnung ist. Sobald ein optisch fehlerfreies Werkstück detektiert wird, wechselt der Generator zurück in seinen Initialzustand. In Tabelle 5.1 sind die Bedeutungen der Ereignisse beschrieben.

Tabelle 5.1.: Ereignisdefinitionen für das Ergebnis der optischen Endkontrolle

Ereignis	Beschreibung
E_optisch_OK	Werkstück hat die optische Endkontrolle bestanden
E_optisch_NOK	Werkstück hat die optische Endkontrolle nicht bestanden

5.2.2. Modellbildung der Handlingeinheit und Linearachse G_2

Das pneumatische Achshandling von Station 20 führt den Transport der Werkstücke von Station 20 zu Station 30 und Station 60 zu Station 20 durch. Die Steuerung erfolgt durch eine unterlagerte Linearachsensteuerung und eine Schrittkette. Die Schrittkette besteht aus einer Aneinanderreihung von folgenden Anweisungen:

- Linearachse zum Band 1, 2, 3, 4, 5, 6 oder 7, zum Handarbeitsplatz oder zur Grundstellung fahren
- vertikale Achse zu Station 20 oder Station 30/60 fahren
- Greifer zu Station 20 oder Station 30/60 drehen
- Greifer öffnen oder schließen

Die Handlingeinheit mit Linearachse kann fünf verschiedene Aufträge ausführen. Für jeden Auftrag existiert eine eigene Schrittkette. Eine Rückmeldung, dass ein Auftrag abgeschlossen ist, existiert nicht. Daher wird das ungesteuerte Verhalten des pneumatischen Achshandlings durch einen Zustand dargestellt. Abbildung 5.2 stellt das ungesteuerte Verhalten als Generator dar.

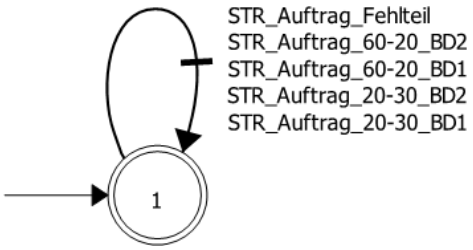


Abbildung 5.2.: Generisches Modell der Linearachse

Im ungesteuerten Fall kann jeder Auftrag ausgeführt werden. Durch die unterlagerte Steuerung ist festgelegt, dass ein neuer Auftrag erst ausgeführt werden kann, wenn das pneumatische Achshandling sich wieder in seiner Grundstellung befindet und die aktuelle Schrittkette beendet ist. Dies ist durch eine gegenseitige Verriegelung der Schrittketten sichergestellt.

Durch die steuerbaren Ereignisse werden die jeweiligen Schrittketten aktiviert. In Tabelle 5.2 sind die Bedeutungen der Ereignisse zusammengefasst.

Tabelle 5.2.: Ereignisdefinitionen für die Handlingeinheit und Linearachse an Station 20

Ereignis	**Beschreibung**
STR_Auftrag_20-30_BD1	Schrittkette für den Transport eines Werkstücks von Band 1 (ST20) zu Band 5 (ST30) starten
STR_Auftrag_20-30_BD2	Schrittkette für den Transport eines Werkstücks von Band 2 (ST20) zu Band 6 (ST30) starten
STR_Auftrag_60-20_BD1	Schrittkette für den Transport eines Werkstücks von Band 7 (ST60) zu Band 3 (ST20) starten
STR_Auftrag_60-20_BD2	Schrittkette für den Transport eines Werkstücks von Band 7 (ST60) zu Band 4 (ST20) starten
STR_Auftrag_Fehlteil	Schrittkette für den Transport eines Werkstücks von Band 7 (ST60) zum Handarbeitsplatz starten

5.2.3. Modellbildung eines abholbereiten optischen Fehlteils G_3

Für den Transport eines optischen Fehlteils zum Handarbeitsplatz benötigt Station 20 die Information, ob sich das Werkstück am Bandende von Station 60 befindet. Ob sich das Fehlteil in Position befindet, wird mit Hilfe des Näherungssensors am Bandende ermittelt. Diese Information wird von Station 60 über Station 10 an Station 20 gesendet. Die Positionsbestimmung kann durch einen Generator mit zwei Zuständen beschrieben werden:

1. Initialzustand: Am Bandende befindet sich kein optisches Fehlteil
2. Am Bandende befindet sich ein optisches Fehlteil

Abbildung 5.3 bildet die Positionsermittlung als Generator nach.

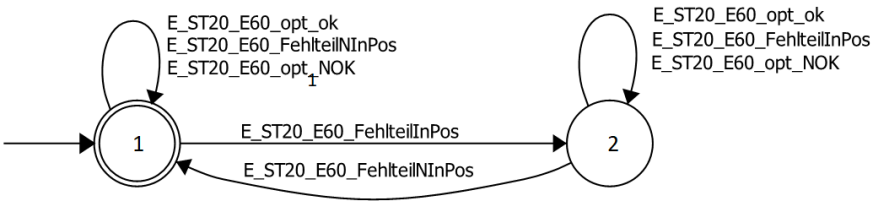


Abbildung 5.3.: Generisches Modell eines abholbereiten optischen Fehlteils

Wird der Näherungssensor am Bandende durch ein optisches Fehlteil ausgelöst, wechselt der Generator vom Initialzustand in den zweiten Zustand. Solange sich der Generator in diesem Zustand befindet, liegt das Fehlteil abholbereit am Bandende. Sobald es abgeholt wurde und der Sensor nicht mehr ausgelöst ist, wechselt der Generator zurück in den Initialzustand. Zusätzlich wird für die spätere Steuerung an Station 20 die Information benötigt, ob ein Werkstück optisch in Ordnung oder fehlerhaft ist. Diese Ereignisse werden durch Schlingen an den Zuständen dargestellt, da sie keinen Zustandswechsel erzeugen. Die Bedeutungen der Ereignisse sind in Tabelle 5.3 zusammengefasst.

Tabelle 5.3.: Ereignisdefinitionen für abholbereite optische Fehlteile

Ereignis	**Beschreibung**
E_ST20_E60_opt_ok	Werkstück ist optisch in Ordnung
E_ST20_E60_opt_NOK	Werkstück ist optisch nicht in Ordnung
E_ST20_E60_FehlteilInPos	Optisches Fehlteil befindet sich vor dem Näherungsensor am Bandende von Station 60
E_ST20_E60_FehlteilNInPos	Optisches Fehlteil befindet sich nicht vor dem Näherungsensor am Bandende von Station 60

5.2.4. Modellbildung der Abschieber G_4, G_5 und G_6

Die Sortierung der optischen Fehlteile am Handarbeitsplatz wird mit Hilfe von pneumatischen Abschiebern durchgeführt. Je nach Fehlerkategorie werden die Fehlteile von den Abschiebern vom Band ausgeschleust. Es befinden sich drei Instanzen dieser Komponente (AS1, AS2 und AS3) am Handarbeitsplatz an Station 20. Die Modellierung wird für den Abschieber AS1 beschrieben. Die Modellierung von AS2 und AS3 erfolgt analog. Das Verhalten des Abschiebers lässt sich in drei Zuständen zusammenfassen:

1. Initialzustand: Abschieber ist eingefahren

2. Abschieber wird ausgefahren
3. Abschieber ist ausgefahren

Abbildung 5.4 stellt das ungesteuerte Verhalten des Abschiebers AS1 als Generator dar.

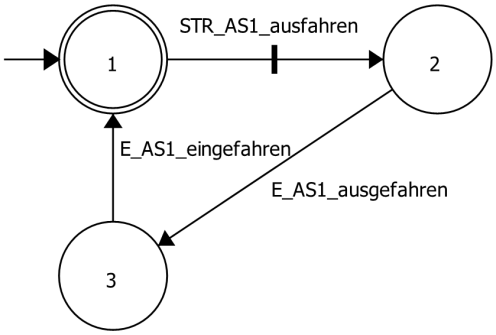


Abbildung 5.4.: Generisches Modell des Abschiebers AS1

Durch das Setzen des steuerbaren Ereignisses *STR_AS1_ausfahren*, wechselt der Generator vom Initialzustand in den zweiten Zustand. Befindet sich der Generator in diesem Zustand, wird eine unterlagerte Steuerung aufgerufen, welche das Ausgangssignal für das Ausfahren des Abschiebers setzt. Ist der Abschieber vollständig ausgefahren, wird der entsprechende Endlagenschalter am Abschieber ausgelöst und der Generator wechselt in den dritten Zustand. In diesem Zustand wird das Ausgangssignal des Abschiebers zurückgesetzt. Da die Abschieber über monostabile Ventile angesteuert werden, wird der Abschieber automatisch eingefahren, sobald das Ausgangssignal zurückgesetzt wird. Der Zustandswechsel vom dritten Zustand zurück zum Initialzustand erfolgt, wenn der Abschieber vollständig eingefahren ist und der entsprechende Endlagenschalter ausgelöst wird. Die Bedeutungen der Ereignisse sind in Tabelle 5.4 beschrieben.

Tabelle 5.4.: Ereignisdefinitionen für den Abschieber AS1

Ereignis	**Beschreibung**
STR_AS1_ausfahren	Abschieber ausfahren
E_AS1_ausgefahren	Abschieber ist ausgefahren
E_AS1_eingefahren	Abschieber ist eingefahren

5.3. Modellbildung der Spezifikationen

Für die in Kapitel 3 beschriebene Modellfabrik soll eine Steuerung für das Handling von optischen Fehlteilen entwickelt werden. Die Steuerungsaufgabe beinhaltet die Erkennung

und den Transport von optischen Fehlteilen zum Handarbeitsplatz, sowie die Sortierung der Fehlteile mit Hilfe der drei Abschieber am Handarbeitsplatz.

Die optische Endkontrolle mit Hilfe des Vision Sensors erfolgt in einer unterlagerten Steuerung. Wird ein optisches Fehlteil detektiert, so soll dieses mittels dem pneumatischen Achshandling von Station 20 zum Handarbeitsplatz transportiert werden. An diesem sollen die Fehlteile anhand von drei definierten Fehlerkategorien sortiert werden. Die Sortierung erfolgt mit Hilfe der drei Abschieber. Diese schleusen je nach Fehlerkategorie das Werkstück vom Band in die entsprechende Kiste.

Zusätzlich zu der Sortierung soll ein Puffermanagement für die einzelnen Kisten entwickelt werden. Sobald die jeweilige Pufferkapazität ausgelastet ist, gilt es zu vermeiden, dass der Montageprozess blockiert wird. Daher ist eine Lösung für einen Pufferüberlauf zu entwickeln.

Die Steuerungsaufgabe wird mit Hilfe der in diesem Abschnitt modellierten Spezifikationen gelöst. Diese werden als Generatoren modelliert und verbieten die, im ungesteuerten Verhalten enthaltenen, unerwünschten Ereignisse. Dabei dürfen jedoch nur steuerbare Ereignisse verboten werden, nicht steuerbare Ereignisse müssen jederzeit möglich sein. Aus diesen Spezifikationen werden die resultierenden Supervisor erstellt. Aus Gründen der Übersichtlichkeit wurden in den nachfolgenden Abbildungen die Schlingen der nicht steuerbaren Ereignisse an den Zuständen nicht dargestellt.

5.3.1. Spezifikation K_1 Linearachse

Die Spezifikation K_1 ist für die Koordinierung der Aufträge der Linearachse von Station 20 und somit für den Transport der Fehlteile von Station 60 zum Handarbeitsplatz verantwortlich. Das Modell der Spezifikation besteht aus drei Zuständen und ist in Abbildung 5.5 dargestellt.

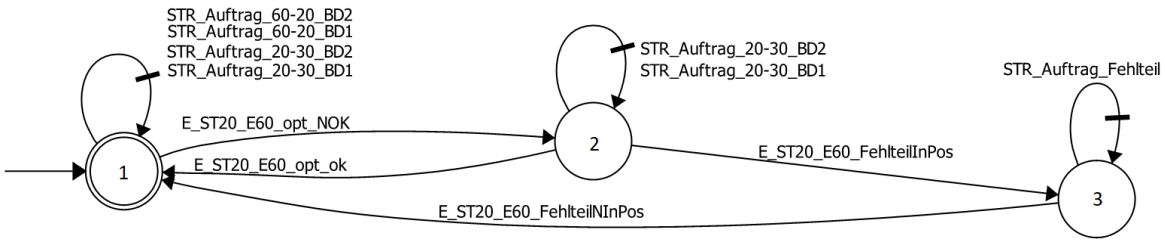


Abbildung 5.5.: Spezifikation K_1

Im Initialzustand wurde kein optisches Fehlteil detektiert und es können bis auf den Transport eines Fehlteils zum Handarbeitsplatz alle Aufträge von der Linearachse ausgeführt werden.

Wird ein Werkstück von dem Vision Sensor als optisch nicht in Ordnung erkannt, wechselt der Generator in den zweiten Zustand. Die Linearachse kann nun nur noch Unterteile von Station 20 zu den Montagebändern von Station 30 transportieren. Der Transport von Werkstücken zum Fertigteillager wird verboten. Durch das Auslösen des Näherungssensors am Bandende wechselt der Generator in den dritten Zustand. Das Fehlteil befindet sich nun abholbereit am Bandende von Station 60. Um einen Montagestau zu vermeiden, kann lediglich der Transport des Fehlteils zum Handarbeitsplatz durchgeführt werden. Wurde das Fehlteil von der Linearachse abgeholt, erfolgt ein Zustandswechsel von Zustand drei zum Initialzustand. Zusätzlich ist ein Zustandswechsel von Zustand zwei zum Initialzustand möglich. Dieser erfolgt durch das Detektieren eines Werkstücks, das optisch in Ordnung ist. Dies ist notwendig, da bei einem Pufferüberlauf ein optisches Fehlteil nicht zum Bandende geführt wird.

5.3.2. Spezifikation K_2 Abschieber

Die Sortierung der optischen Fehlteile erfolgt am Handarbeitsplatz mit Hilfe von drei pneumatischen Abschiebern. Im ungesteuerten Verhalten können die drei Abschieber unabhängig voneinander aus- und eingefahren werden. Dies soll mit der Spezifikation K_2 vermieden werden. Sie ist dafür verantwortlich, dass nur ein Abschieber zur Zeit ausgefahren wird. In Abbildung 5.6 ist das Modell der Spezifikation, welches aus zehn Zuständen besteht dargestellt.

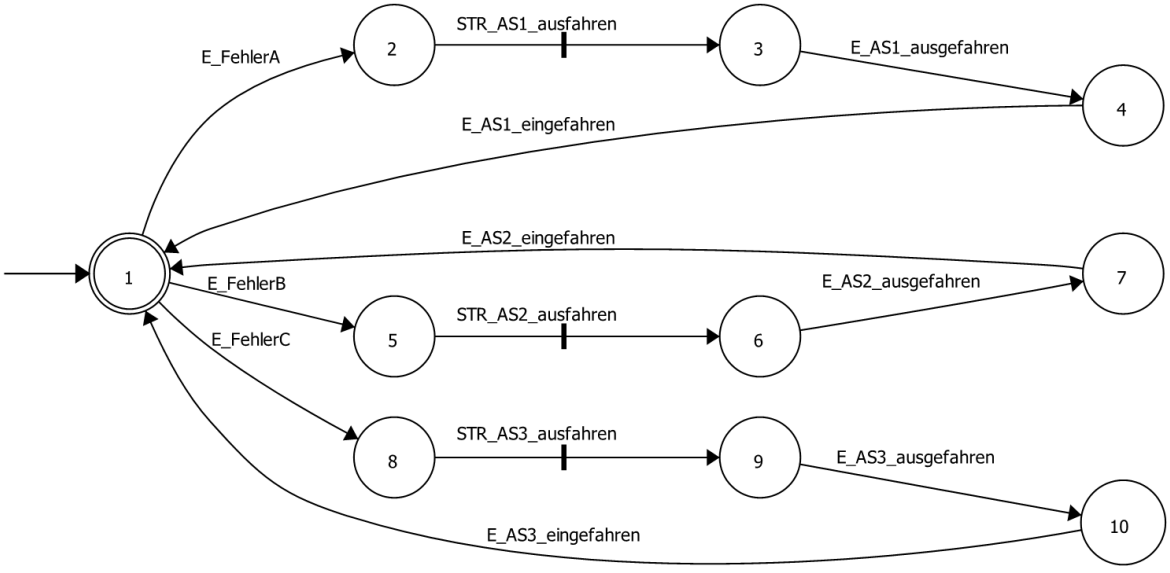


Abbildung 5.6.: Spezifikation K_2

Der Vision Sensor unterscheidet bei der optischen Endkontrolle zwischen drei verschiedenen Fehlerkategorien (A, B und C). Das Ergebnis der Prüfung wird über Station 10 an Station 20 gesendet. Je nachdem welcher Fehlerkategorie ein Werkstück angehört, wird der entsprechende Abschieber am Handarbeitsplatz ausgefahren. Im Initialzustand befinden sich alle drei Abschieber in ihrer Grundstellung. Befindet sich ein Fehlteil vor dem entsprechenden Abschieber, erfolgt ein Zustandswechsel. Ist es ein Fehlteil der Kategorie A und es befindet sich vor dem ersten Abschieber, wechselt der Generator in den zweiten Zustand. Durch das Setzen des steuerbaren Ereignisses *STR_AS1_ausfahren* erfolgt ein Zustandswechsel von Zustand 2 in Zustand 3. Die unterlagerte Steuerung für den Abschieber wird gestartet und das Ausgangssignal gesetzt. Sobald der Abschieber ganz ausgefahren ist und das Fehlteil ausgeschleust wurde, wechselt der Generator in den vierten Zustand. Das Ausgangssignal wird zurückgesetzt und der Abschieber fährt zurück in seine Grundstellung. Nach Erreichen der Grundstellung befindet sich der Generator wieder im Initialzustand. Während des Ausschleusungsprozesses können die anderen Abschieber nicht ausgefahren werden. Das Ausfahren der Abschieber AS2 und AS3 erfolgt analog.

5.3.3. Spezifikation K_3, K_4 und K_5 Puffermanagement am Handarbeitsplatz

Die optischen Fehlteile werden am Handarbeitsplatz mit Hilfe der pneumatischen Abschieber in unterschiedlichen Kisten sortiert. Diese können nur eine bestimmte Menge an Fehlteilen aufnehmen. Daher ist es notwendig ein Puffermanagement zu entwickeln. Dies wird für jede der drei Kisten durchgeführt. Die Kapazität der einzelnen Kisten beträgt zwei Werkstücke. Diese darf nicht überschritten werden. Für das Puffermanagement sind die Spezifikationen K_3, K_4 und K_5 verantwortlich. In Abbildung 5.7 ist die Spezifikation K_3 als Generator mit drei Zuständen dargestellt.

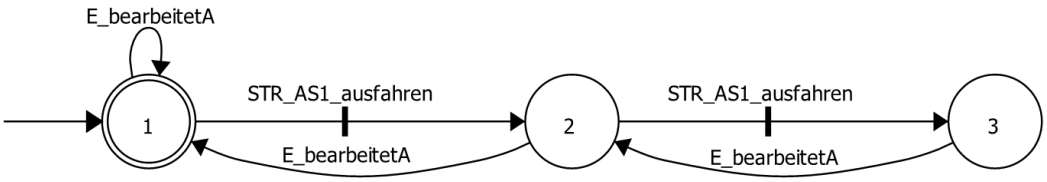


Abbildung 5.7.: Spezifikation K_3

Die Zustände der Spezifikation repräsentieren in aufsteigender Reihenfolge die Belegung des Puffers für Fehlteile der Kategorie A von leer bis voll. Im Initialzustand befindet sich kein Fehlteil im Puffer und der Abschieber AS1 kann ausgefahren werden. Dadurch wechselt der Generator in den zweiten Zustand. Wird ein weiteres Werkstück in die Kiste sortiert, erfolgt

ein Zustandswechsel von Zustand zwei in Zustand drei. Der Puffer ist nun voll belegt und der Abschieber kann nicht mehr ausgefahren werden. Damit neue Fehlteile der Kategorie A ausgeschleust werden können, muss der Puffer zunächst geleert werden. Dies erfolgt durch die Reparatur der optischen Fehlteile durch einen Montagemitarbeiter. Nach abgeschlossener Reparatur quittiert dieser den Fehler und der Generator wechselt vom aktuellen Zustand in den vorherigen. Anschließend kann erneut ein Fehlteil im Puffer abgelegt werden. Die Spezifikationen K_4 und K_5 werden analog modelliert.

5.3.4. Spezifikation K_6 Erkennung optischer Fehlteile und Pufferüberlauf

Die Puffer am Handarbeitsplatz können jeweils nur zwei Fehlteile aufnehmen. Sind diese vollständig belegt können keine weiteren optischen Fehlteile am Handarbeitsplatz aussortiert werden. Dies hat zur Folge, dass der Montageprozess blockiert wird, da keine weiteren Fehlteile von Station 60 zum Handarbeitsplatz transportiert werden können. Diese Situation gilt es zu vermeiden. Aus diesem Grund soll eine Lösung für den Pufferüberlauf entwickelt werden. Ist ein Puffer voll belegt, wird diese Information von Station 20 über Station 10 an Station 60 weitergeleitet. Es besteht die Möglichkeit optische Fehlteile mit Hilfe des pneumatischen Abschiebers unter der Kamera vom Band auszuschleusen, sobald der entsprechende Puffer am Handarbeitsplatz voll ist. Dies wird mit Hilfe der Spezifikation K_6 realisiert. Diese ist als Generator mit 3 Zuständen in Abbildung 5.8 dargestellt.

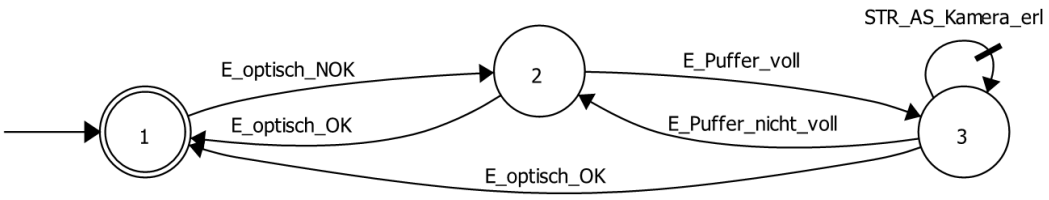


Abbildung 5.8.: Spezifikation K_6

Der Initialisierungszustand beschreibt die Situation, dass kein optisches Fehlteil detektiert wurde. Besteht ein Werkstück die optische Endkontrolle nicht, wechselt der Generator in den zweiten Zustand. Es befindet sich ein optisches Fehlteil unter der Kamera. Ist der entsprechende Puffer voll, erfolgt ein Zustandswechsel vom zweiten Zustand in den dritten Zustand. In diesem Zustand wird das steuerbare Ereignis *STR_AS_Kamera_erl* generiert. Dieses startet die unterlagerte Steuerung für den Abschieber unter der Kamera. Die unterlagerte Steuerung setzt das Ausgangssignal zum Ausfahren des Abschiebers und das Fehlteil wird vom Band ausgeschleust. Der Generator verbleibt in diesem Zustand, bis der entsprechende

Puffer nicht mehr voll ist oder ein Werkstück die optische Endkontrolle besteht. Ist der entsprechende Puffer nicht mehr voll, wechselt der Generator zurück in den zweiten Zustand. Der Zustandswechsel von Zustand zwei bzw. Zustand drei zurück zum Initialzustand erfolgt durch die Detektion eines fehlerfreien Werkstücks.

5.4. Steuerungsentwurf

In diesem Abschnitt werden die Supervisor aus den Modellen der Strecke und den Spezifikationen nach dem lokal-modularen Entwurfsansatz berechnet. Die Synthese erfolgt mit dem Entwicklungstool DESTool. Die Generatoren der Streckenkomponenten werden durch die sechs Spezifikationen synchronisiert. In Abbildung 5.9 sind die Relationen zwischen den Ereignisalphabeten der Streckenkomponenten

$$G_i' = (X_i', \Sigma_i', \delta_i', x_{i,0}', X_{i,m}'), i = [1, \ldots, 6]$$

und der Spezifikationen $K_{Xj} \subseteq \Sigma_{Xj}^*, j = [1, \ldots, 6]$ dargestellt.

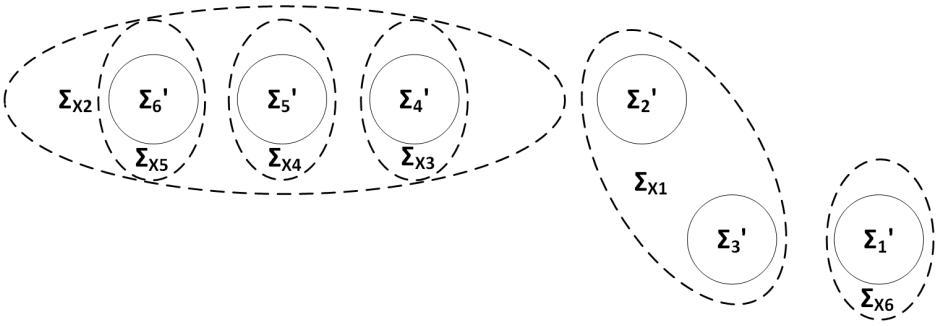


Abbildung 5.9.: Relationen zwischen den Ereignisalphabeten

Anhand der Relationen ist erkennbar, dass die Spezifikationen K_2, K_3, K_4 und K_5 auf die drei Abschieberkomponenten zugreifen. Daher werden diese zu einer Spezifikation zusammengefasst. Dafür werden die einzelnen Spezifikationen mit den fehlenden Ereignissen aus den anderen Spezifikationen erweitert. Für jedes fehlende Ereignis wird eine Schlinge an jedem Zustand angefügt. Anschließend werden die Spezifikationen mit der Produktkomposition zusammengefasst

$$K_{2,gesamt} = K_2 ||_{SPC} K_3 ||_{SPC} K_4 ||_{SPC} K_5.$$

Die ungesteuerte Strecke besteht aus sechs Komponenten, welche über disjunkte Ereignisalphabete definiert sind. Die einzelnen Komponenten laufen vollständig asynchron zueinander. Somit liegt die Strecke als Produktsystem vor. Das feinste Produktsystem ergibt sich wie folgt:

$$G_i = G_i', i = [1, \ldots, 6]$$

In DESTool werden mit Hilfe des *SYPC*-Operators die lokalen Systeme und die angepassten Spezifikationen berechnet. Die lokalen Systeme $G_{Xi}, i = [1, 2, 3]$ ergeben sich wie folgt:

$$G_{X1} = G_1 \text{ mit } \Sigma_{G_{X1}} = \Sigma_{G_1}$$

$$G_{X2} = G_2 || _{SYPC} G_3 \text{ mit } \Sigma_{G_{X2}} = \Sigma_{G_2} \cup \Sigma_{G_3}$$

$$G_{X3} = G_4 || _{SYPC} G_5 || _{SYPC} G_6 \text{ mit } \Sigma_{G_{X3}} = \Sigma_{G_4} \cup \Sigma_{G_5} \cup \Sigma_{G_6}$$

Die lokalen Systeme werden für die Anpassung der Spezifikationen mit den fehlenden Ereignissen aus den entsprechenden Spezifikationen erweitert. Dafür wird für jedes fehlende Ereignis in dem entsprechenden lokalen System eine Schlinge an jedem Zustand angefügt. Nach dieser Erweiterung können die Spezifikationen an die lokalen Systeme mit Hilfe der parallelen Komposition angepasst werden. Die angepassten Spezifikationen $E_{Xi}, i = [1, 2, 3]$ ergeben sich wie folgt:

$$E_{X1} = K_1 || _{SYPC} G_{X2}$$

$$E_{X2} = K_{2,ges} || _{SYPC} G_{X3}$$

$$E_{X3} = K_6 || _{SYPC} G_{X1}$$

Anschließend werden die angepassten Spezifikationen auf Steuerbarkeit ihrer lokalen Systeme überprüft. Die Überprüfung der Steuerbarkeit ergibt folgende Ergebnisse:

$$IsControllable(E_{X1}, G_{X2}) \checkmark$$

$$IsControllable(E_{X2}, G_{X3}) \checkmark$$

$$IsControllable(E_{X3}, G_{X1}) \checkmark$$

Die angepassten Spezifikationen E_{Xi} sind alle steuerbar bezüglich ihrer lokalen Systeme. Daher muss keine supremale steuerbare Teilsprache $SupC(E_{Xi}, G_{Xi})$ für $i = [1, 2, 3]$ berechnet werden. Im Anschluss an die Überprüfung der Steuerbarkeit werden die angepassten Spezifikationen mit Hilfe der Funktion *IsNonblocking* auf lokale Modularität geprüft. For-

mal ist die folgende Bedingung zu überprüfen:

$$||_{i\in I}\overline{E_{Xi}} = \overline{||_{i\in I}E_{Xi}} \text{ für } I = [1,2,3]$$

Die Überprüfung auf lokale Modularität ergibt folgende Ergebnisse:

$$IsNonblocking(E_{X1}, E_{X2})\checkmark$$

$$IsNonblocking(E_{X1}, E_{X3})\checkmark$$

$$IsNonblocking(E_{X2}, E_{X3})\checkmark$$

Das Ergebnis der Überprüfung der angepassten Spezifikationen zeigt, dass alle drei Spezifikationen blockierungsfrei sind. Mit Hilfe der DESTool Funktion *SupConNB* können aus den angepassten Spezifikationen und ihren lokalen Systemen die lokal modularen Supervisor berechnet werden. Die Berechnung der einzelnen Supervisor, sowie ihre Anzahl an Zuständen (n) und Transitionen (t) sind in Tabelle 5.5 dargestellt.

Tabelle 5.5.: Ergebnis der Supervisorsynthese

Supervisor	**Berechnung**	**n**	**t**
S_1	$SupConNB(E_{X1}, G_{X2}) = E_{X1}$	5	33
S_2	$SupConNB(E_{X2}, G_{X3}) = E_{X2}$	270	1836
S_3	$SupConNB(E_{X3}, G_{X1}) = E_{X3}$	3	13

Die berechneten Supervisor werden abschließend mit Hilfe der DesTool Funktion *SupReduce* nach [22] reduziert. Der Funktion werden ein lokal-modularer Supervisor und das entsprechende lokale System als Eingangsparameter übergeben. Aus diesen wird ein alternativer Supervisor mit einer reduzierten Anzahl an Zuständen und Transitionen, welcher das gesteuerte Verhalten aufrechterhält, berechnet. Die drei Supervisor wurden nacheinander mit dieser Funktion reduziert. In Tabelle 5.6 sind die reduzierten und nicht reduzierten Supervisor mit ihrer jeweiligen Anzahl an Zuständen und Transitionen zusammengefasst.

Tabelle 5.6.: Ergebnis der Supervisor-Reduzierung

Supervisor	**n**	**t**	**Reduzierter Supervisor**	**n**	**t**
S_1	5	33	$SupReduce(S_1, G_{X2})$	3	19
S_2	270	1836	$SupReduce(S_2, G_{X3})$	135	945
S_3	3	13	$SupReduce(S_3, G_{X1})$	3	13
Gesamt	278	1882	Gesamt	141	977

An den Ergebnissen aus Tabelle 5.6 ist erkennbar, dass die Gesamtmenge an Zuständen und Transitionen mit Hilfe der Supervisor-Reduzierung um etwa 49 % reduziert wurde.

6. Realisierung

Dieses Kapitel ist in drei Abschnitte unterteilt. Zunächst wird in Abschnitt 6.1 die Einbindung des Codelesegeräts in die Modellfabrik beschrieben. Abschnitt 6.2 befasst sich mit der Realisierung des Handarbeitsplatzes. Die Implementierung der Steuerung für das Handling von optischen Fehlteilen wird in Abschnitt 6.3 beschrieben.

6.1. Einbindung des Codelesegeräts

Für die Einbindung des vorhandenen Codelesegeräts in die Modellfabrik muss zunächst ein geeignetes Programm mit Hilfe der Web Server Bedienoberfläche erstellt werden. Dieses muss so entworfen werden, dass sowohl eine Erkennung von optischen Fehlteilen, als auch eine Unterscheidung zwischen verschiedenen Fehlerfällen möglich ist. Dabei soll zwischen den folgenden Fehlerkategorien unterschieden werden:

- Fehler A: Es befindet sich kein Logo auf dem Deckel.
- Fehler B: Das Logo ist um 180° verdreht.
- Fehler C: Das Logo ist um +/- 90° verdreht.

Mit Hilfe der Bedienoberfläche des Identifikationssystems können Programme zum Lesen von Klarschrift, 1D- und 2D-Codes, sowie zur Objekterkennung erstellt werden. Für diese Anwendung ist ein Programm zur Objekterkennung am besten geeignet. Dafür wird zunächst eine Modellbibliothek mit Referenzbildern für die einzelnen Fehlerfälle, sowie für Werkstücke, die optisch in Ordnung sind, erstellt. Die einzelnen Kategorien stellen dabei ein eigenes Modell dar, dem eine Klasse zugeordnet wird. Es werden insgesamt vier Modelle erstellt, je eins pro Fehlerkategorie, sowie eins für optische Gutteile. Ihnen werden die Klassen A, B, C und D zugewiesen. Dies ermöglicht eine Klassifikation der Bilder, die während der optischen Endkontrolle aufgenommen wurden. Abbildung 6.1 zeigt die Erzeugung eines Modells.

Das eigentliche Programm zur Objekterkennung besteht aus drei Schritten. Im ersten Schritt werden die erforderlichen Einstellungen zur Bildaufnahme vorgenommen. Dazu gehört die Einstellung der Auflösung, Belichtung und Art der Triggerung. Für die Auflösung und Belichtung werden die voreingestellten Werte übernommen. Die Aufnahme eines Bildes soll dann

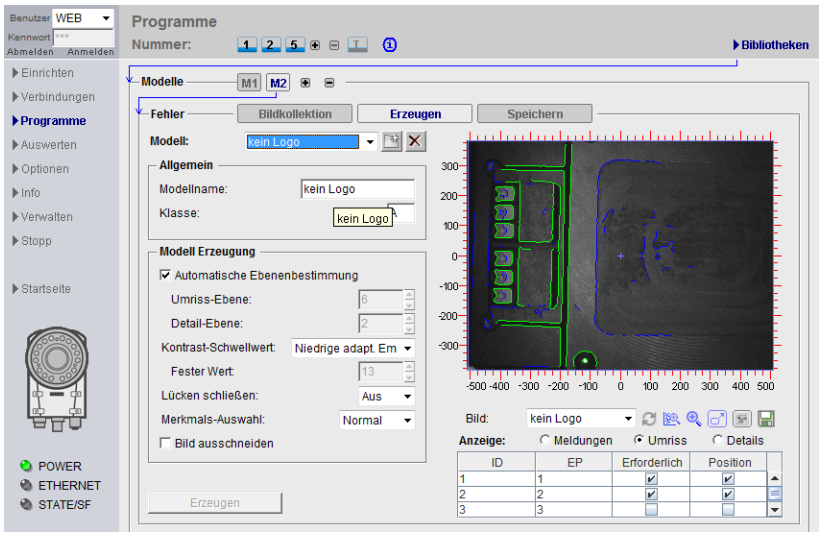


Abbildung 6.1.: Erzeugung eines Modells

erfolgen, wenn sich ein Werkstück unter dem Identifikationssystem befindet, daher wird als Art der Triggerung die Option Einzel-Trigger ausgewählt. Im Anschluss an die Bildaufnahme folgt der Objekterkennungsschritt. Hier erfolgt die Klassifikation der aufgenommenen Bilder. In diesem Schritt werden die Aufgaben der Objekterkennung konfiguriert. Zunächst werden dafür die erstellte Modellbibliothek, sowie die Modelle, die in diesem Schritt erkannt werden sollen, ausgewählt. Des Weiteren wird der Fangbereich, in dem nach Objekten gesucht werden soll, eingestellt. In Abbildung 6.2 sind die vorgenommenen Einstellungen zu sehen.

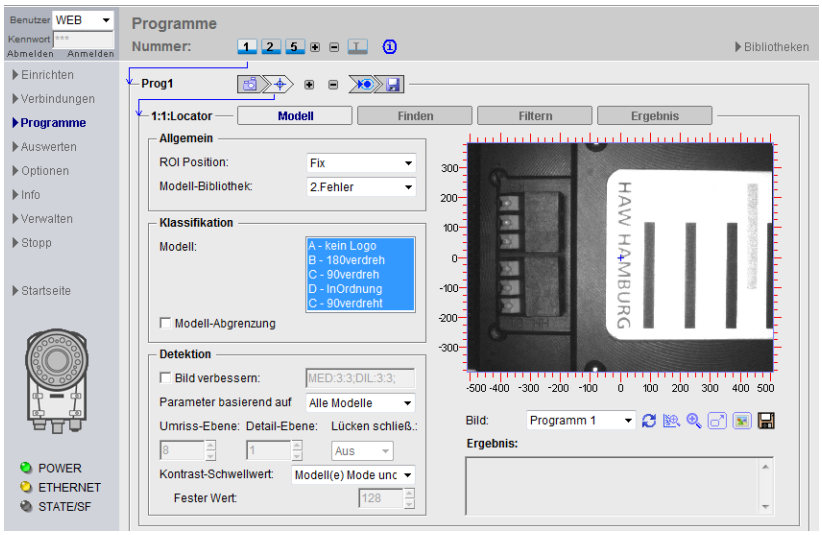


Abbildung 6.2.: Einstellungen im Objekterkennungsschritt

Zusätzlich ist die Genauigkeit der Objekterkennung festzulegen. Es wird eine minimale Übereinstimmung von 50 % eingestellt. Diese Genauigkeit ist für diese Anwendung ausreichend, da sich die einzelnen Klassen auf Grund der Beschaffenheit des Logos stark unterscheiden. Dabei spielt vor allem die Position und Ausrichtung der Schrift eine Rolle. Als Ergebnis wird in diesem Schritt die gefundene Objektklasse ausgegeben. Das Programm wird mit einem Ergebnisschritt abgeschlossen. In diesem wird festgelegt, wie das Gesamtergebnis ausgegeben werden soll. In diesem Fall wird als Gesamtergebnis die Objektklasse ausgegeben. Diese wird in einem String gespeichert.

Für die Verarbeitung des Ergebnisstrings mit einer SPS muss die Art der Verbindung festgelegt werden. In diesem Fall ist das Codelesegerät über PROFINET mit der SPS verbunden. Zusätzlich muss der in der Hardwarekonfiguration verwendete Gerätename angegeben werden. Die vorgenommenen Einstellungen sind in Abbildung 6.3 zu sehen. Die Kommunikation der SPS mit dem Codelesegerät erfolgt über den Baustein Ident-Profil. Dies ist ein vorgefertigter komplexer Baustein, der alle Befehle und Funktionen für optische Lesesysteme enthält. Vor der Parametrierung des Bausteins muss die PLC-Variable „hwConnect“ vom Datentyp „IID_HW_Connect“ angelegt werden. Diese adressiert den Kommunikationskanal des Ident-Systems. Mit Hilfe des Ident-Profils kann der Ergebnisstring ausgelesen und das Codelesegerät bei Auftreten eines Fehlers zurückgesetzt werden.

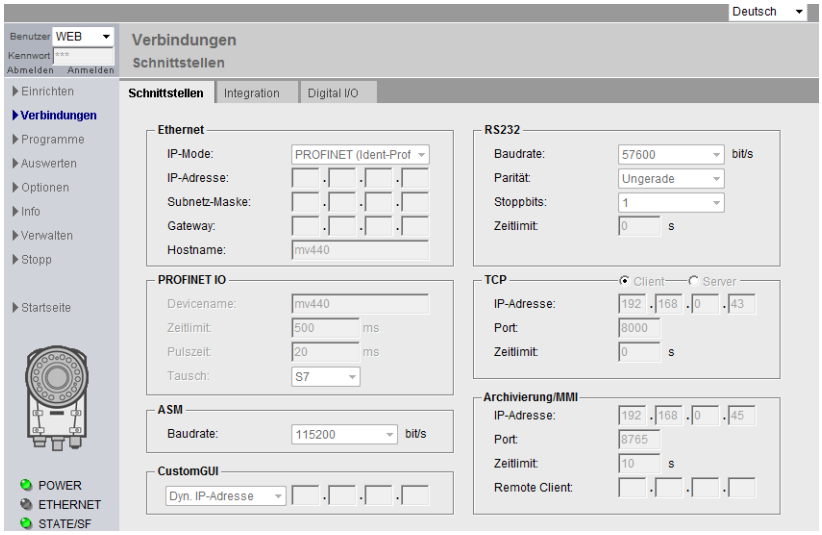


Abbildung 6.3.: Verbindungseinstellungen des Codelesegeräts

6.2. Realisierung des Handarbeitsplatzes

In diesem Abschnitt wird die Realisierung des Handarbeitsplatzes beschrieben. Diese erfolgt nach dem in Kapitel 4 ausgewählten Konzept. Der Hardwareaufbau besteht aus einem Transportband, drei pneumatischen Abschiebern mit Endlagenschalten, Einweglichtschranken vor den einzelnen Abschiebern, sowie am Bandanfang und -ende und einem Touchpanel, welches an einem Monitorarm befestigt ist.

Das verwendete Transportband stammt von der Firma Köster Systemtechnik GmbH. Dieses hat ein Maß von 680 $\times$ 100 mm. Es wird von einem Gleichstrommotor mit Rechts- und Linkslauf angetrieben. An diesem Transportband sind die drei pneumatischen Abschieber der Firma Aventics montiert. Diese haben jeweils ein Maß von 110 $\times$ 35 mm. Die Ansteuerung der Abschieber erfolgt über eine Ventilinsel mit monostabilen Ventilen. Jeder Abschieber ist mit zwei Endlagenschaltern ausgestattet, welche den ausgefahrenen bzw. den eingefahrenen Zustand signalisieren.

Für die Positionsbestimmung der Werkstücke auf dem Transportband befinden sich an den Abschiebern, sowie am Bandanfang und -ende Einweglichtschranken der Firma Sick. In der restlichen Modellfabrik werden dafür kapazitive Näherungsschalter eingesetzt. Diese sind jedoch sehr empfindlich und müssen häufig neu kalibriert werden. Daher wurden für diesen Aufbau Einweglichtschranken ausgewählt, da diese besonders präzise sind und weniger häufig nachjustiert werden müssen.

Abbildung 6.4.: Hardwareaufbau des Handarbeitsplatzes

Für die von den Abschiebern ausgeschleusten Fehlteile werden handelsübliche Lagersicht-

kästen mit den Maßen $102 \times 117 \times 73\,\text{mm}$ (Breite $\times$ Länge $\times$ Höhe) verwendet. Diese sind fest montiert. Mit Hilfe von selbstklebenden LED-Streifen wird das Pick-by-Light Prinzip realisiert. Diese werden in die Lagersichtkästen geklebt. Je nach gewählter Anleitung leuchtet der LED-Streifen in der entsprechenden Kiste grün, bis der Vorgang quittiert wird. Der Hardwareaufbau des Handarbeitsplatzes ist in Abbildung 6.4 dargestellt.

Das Assistenzsystem wird auf einem Touchpanel der Firma Siemens realisiert. Hierbei handelt es sich um das Modell TP 700 Comfort. Es hat ein Maß von $214 \times 158\,\text{mm}$. Mit Hilfe eines Gehäuses der Firma Rittal wird es an einem Monitorarm, welcher höhenverstellbar und schwenkbar ist, montiert. Dieser ist in Abbildung 6.5 dargestellt.

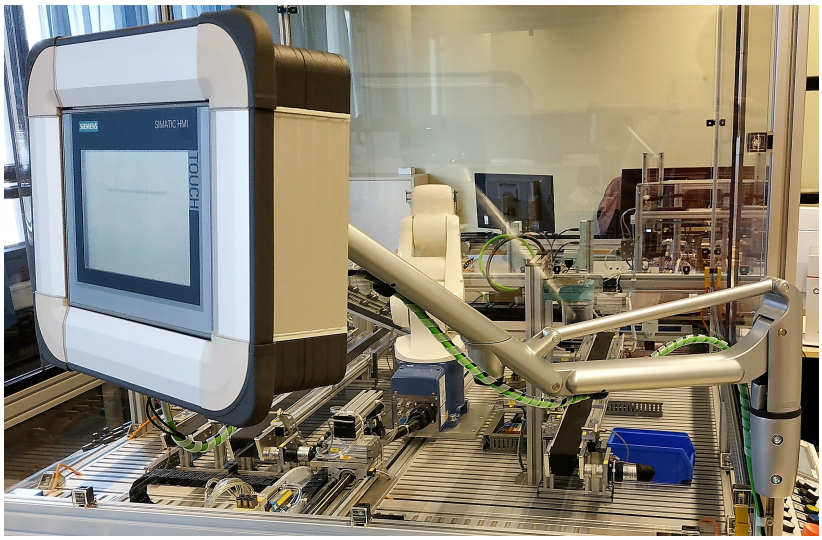


Abbildung 6.5.: Monitorarm mit Touchpanel

An der vorhandenen SPS an Station 20 stehen für die Verdrahtung der Komponenten nicht genügend freie digitale Ein- und Ausgänge (DI/DO) zur Verfügung, daher wird die SPS um eine DI- und eine DO-Baugruppe erweitert. Die einzelnen Komponenten werden sowohl mit der SPS als auch mit der Spannungsversorgung verdrahtet. Im Anschluss an die Verdrahtung erfolgt eine Überprüfung, ob diese korrekt durchgeführt wurde und die Spannung bedenkenlos eingeschaltet werden kann. Nach dem Einschalten der Spannung erfolgt ein Signalcheck zur Überprüfung, ob die Signale an den richtigen Ein-/Ausgängen der SPS ankommen. Erst nach bestandener Prüfung kann die SPS-Programmierung erfolgen.

Die Visualisierung des Assistenzsystems auf dem Touchpanel wird mit der im TIA Portal integrierten Software WinCC durchgeführt. Es werden insgesamt vier Bilder erstellt. Das Startbild „Auswahl“ beinhaltet drei Schaltflächen, über die ausgewählt werden kann, welcher Fehlertyp bearbeitet werden soll. Die Schaltflächen sind mit Steuerungsvariablen verbunden, welche angeben, ob sich ein Werkstück der jeweiligen Fehlerkategorie zur Reparatur

am Handarbeitsplatz befindet. Mit Hilfe dieser Verbindung werden die Schaltflächen so eingestellt, dass sie nur dann sichtbar sind, wenn ein entsprechendes Fehlteil zur Reparatur vorliegt. Das Bild zur Auswahl des Fehlertyps ist in Abbildung 6.6 zu sehen.

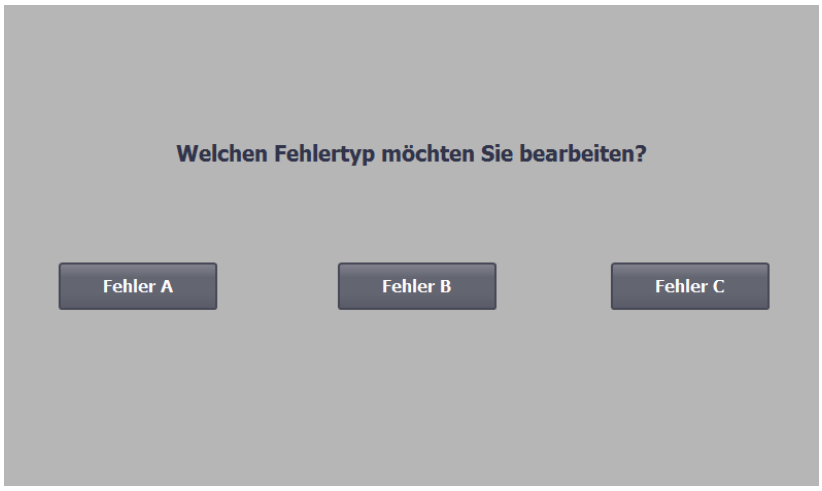


Abbildung 6.6.: Bild zur Auswahl des zu bearbeitenden Fehlertyps

Über das Betätigen der Schaltflächen gelangt man automatisch zu der entsprechenden Anleitung für den jeweiligen Fehlertyp. Außerdem wird der LED-Streifen in der entsprechenden Kiste angesteuert, sodass dieser grün leuchtet und der Mitarbeiter weiß, aus welcher Kiste das Fehlteil zu entnehmen ist. Die Bilder „Fehler A“, „Fehler B“ und „Fehler C“ enthalten die Reparaturanleitungen für die jeweilige Fehlerkategorie.

Abbildung 6.7.: Anleitung für die Reparatur eines Fehlteils der Kategorie A

Mit dieser Anleitung erhält der Arbeiter eine Schritt für Schritt Hilfestellung, um das optische Fehlteil zu reparieren. Zusätzlich befindet sich neben der Anleitung ein Foto, welches ein Werkstück zeigt, dass die optische Endkontrolle bestanden hat. Ist die Reparatur abgeschlossen, kann diese über die Schaltfläche „Reparatur abgeschlossen“ beendet werden. Hierdurch gelangt man zurück zum Startbild und der Ausgang des LED-Streifens wird zurückgesetzt. In Abbildung 6.7 ist das Bild „Fehler A“ dargestellt. Abbildung 6.8 zeigt den fertig eingerichteten Handarbeitsplatz an Station 20.

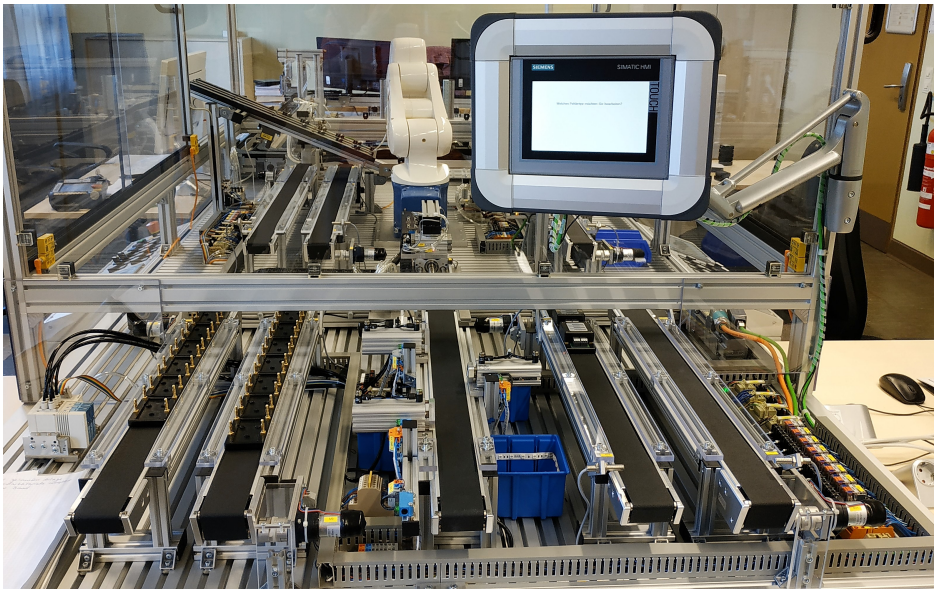


Abbildung 6.8.: Gesamtansicht des Handarbeitsplatzes

6.3. Implementierung der Steuerung

Die Implementierung der Steuerung ist in zwei Abschnitte unterteilt. In Abschnitt 6.3.1 werden die Modelle aus Kapitel 5 in Quellcode umgewandelt und in die bereits vorhandene Steuerung integriert. Abschnitt 6.3.2 beschreibt die Erweiterung der unterlagerten Steuerung.

6.3.1. Implementierung der lokal-modularen Supervisor

Für die Implementierung der lokal-modularen Supervisor auf einer SPS müssen zunächst die Modelle der reduzierten Supervisor, sowie die dazugehörigen lokalen Systeme aus dem vorherigen Kapitel in Programmcode übersetzt werden. Dafür wird der in Abschnitt 2.2.2

vorgestellte Codegenerator ACArrow verwendet. Die Codegenerierung wird für jeden reduzierten Supervisor einzeln durchgeführt. Dies ermöglicht die Implementierung auf verschiedenen Steuerungen. Insgesamt wird die Codegenerierung dreimal durchgeführt. Dabei wird im DESTool Projekt der jeweilige reduzierte Supervisor als Supervisormodell und das entsprechende lokale System als Streckenmodell gekennzeichnet. Anschließend wird die Codegenerierung für den lokal-modularen Ansatz durchgeführt. Dafür wird als Syntax Siemens und als Programmiersprache SCL ausgewählt. ACArrow erzeugt als Ausgabe jeweils 12 S7-SCL-Quelldateien. Neben diesen werden zusätzlich zwei Excel-Dateien, welche die für den Programmcode benötigten PLC-Variablen enthalten, generiert.

Die Implementierung der reduzierten Supervisor erfolgt auf den Steuerungen von Station 20 und Station 60. Auf der SPS von Station 20 werden die Supervisor S_1 und S_2 und auf der SPS von Station 60 der Supervisor S_3 implementiert. Die generierten Quelldateien werden über externe Quellen in das TIA-Portal importiert. Mit Hilfe der Funktion „Baustein aus Quelle generieren“ werden Funktionsbausteine aus den Quelldateien erzeugt. Die bei der Codegenerierung entstandenen Excel-Dateien werden in die Variablentabellen der Steuerungen importiert. Die importierten Variablen besitzen andere Namen als die Variablen der unterlagerten Steuerung. Daher muss für jeden Supervisor eine Schnittstellenfunktion erstellt werden, in der eine Zuweisung der entsprechenden Variablen erfolgt. In diesen Funktionen erfolgt außerdem der Aufruf der generierten Hauptfunktionen.

Die lokal-modularen Supervisor sind nicht für die Ausführung der erlaubten Steuerungsaktionen verantwortlich. Dies ist die Aufgabe der unterlagerten Steuerung. In dieser werden die Prozesse mit Hilfe von Schrittketten im Automatikbetrieb ausgeführt. Die implementierten Supervisor teilen der unterlagerten Steuerung lediglich über die steuerbaren Ereignisse mit, welche Steuerungsaktionen erlaubt sind.

6.3.2. Erweiterung der unterlagerten Steuerung

Neben der Implementierung der lokal-modularen Supervisor werden die unterlagerten Steuerungen in Station 20 und Station 60 erweitert. Hierzu gehören Funktionserweiterungen in der jeweiligen SPS, sowie Erweiterungen der Visualisierung.

Erweiterungen in Station 20

Für die Ansteuerung der Hardwarekomponenten des neu errichteten Handarbeitsplatzes müssen die entsprechenden Funktionen in das SPS Programm integriert werden. Hierzu gehören das Transportband und die drei pneumatischen Abschieber. Für das Transportband wird einer der vorhandenen Funktionsbausteine für die bereits vorhandenen Bänder dupliziert und angepasst. Das Transportband wird gestartet, sobald ein optisches Fehlteil am

Bandanfang abgelegt wird. Befindet sich das Werkstück vor dem entsprechenden Abschieber wird das Band gestoppt. Die Steuerung der pneumatischen Abschieber erfolgt ebenfalls über eine bereits vorhandene Funktion. Dafür wird der Funktionsbaustein von einem der beiden Abschieber an Station 60 dupliziert und angepasst. Ein Ausfahren der Abschieber ist nur möglich, wenn dies von der übergeordneten Steuerung erlaubt wird und sich ein entsprechendes Fehlteil vor dem Abschieber befindet.

Die Puffer der einzelnen Fehlerkategorien werden zusätzlich mit Zählern überwacht. Diese geben die Anzahl an freien Plätzen an. Bei Start des Automatikbetriebs werden diese mit dem Wert zwei initialisiert. Wird ein optisches Fehlteil am Handarbeitsplatz abgeschoben, wird der Wert des entsprechenden Zählers um eins verringert. Durch die Reparatur des Fehlteils wird der Wert um eins erhöht. Ist ein Puffer voll belegt wird der Ausgang des entsprechenden Zählers gesetzt und über Station 10 an Station 60 gesendet. Dieses Signal wird für den Pufferüberlauf benötigt.

Des Weiteren wird eine Schrittkette für den Transport von optischen Fehlteilen von Station 60 zum Handarbeitsplatz mit Hilfe der Linearachse erstellt. Hierfür wird eine der bereits vorhandenen Funktionen dupliziert und angepasst. In der Funktion „Auto“ werden die einzelnen Fahraufträge aufgerufen, sobald die jeweilige Startvariable gesetzt ist. Für den Aufruf der neu erstellten Schrittkette wird eine Variable eingefügt, welche gesetzt wird, sobald sich ein optisches Fehlteil am Bandende von Station 60 befindet. Die gegenseitige Verriegelung der Schrittketten, welche gewährleistet, dass nur ein Fahrauftrag zurzeit aktiv ist, muss ergänzt werden.

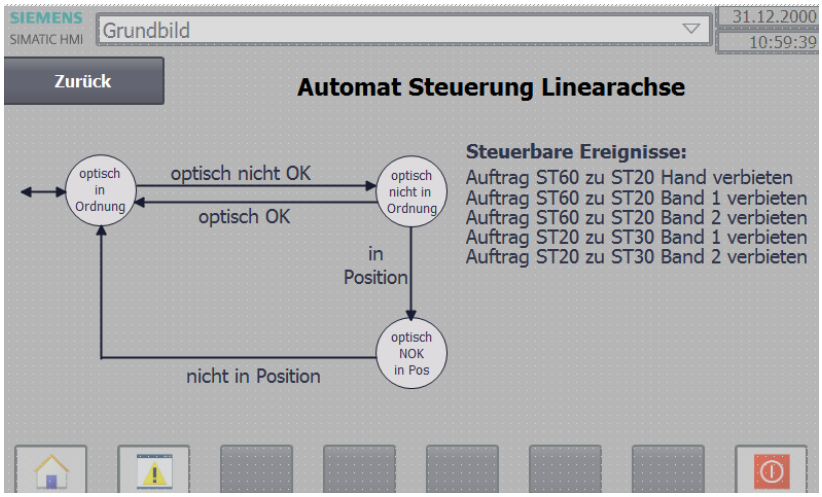


Abbildung 6.9.: Visualisierung des Supervisors S_1

Für die Visualisierung des Supervisors S_1 wird ein neues Bild für das Touchpanel an Station 20 erstellt. Dieses stellt das Modell des Supervisors dar. Im Automatikbetrieb wird der aktive

Zustand, sowie die möglichen steuerbaren und nicht steuerbaren Ereignisse grün hervorgehoben. Abbildung 6.9 zeigt das erstellte Bild.

Der Supervisor S_2, welcher für die Steuerung der Abschieber, sowie das Puffermanagement verantwortlich ist, wird nicht visualisiert. Auf Grund der großen Anzahl an Zuständen und Transitionen wäre dies sehr unübersichtlich.

Erweiterungen in Station 60

Die Kommunikation des Codelesegeräts mit der SPS an Station 60 erfolgt wie in Abschnitt 6.1 beschrieben über das Ident-Profil. Mit Hilfe dieses Bausteins wird jedoch nur der Ergebnisstring ausgelesen. Dieser enthält, abgesehen von dem Ergebnis der Klassifikation, weitere Statusmeldungen des Codelesegeräts. Für die Nutzung des Klassifikationsergebnisses als Ereignis wird der Inhalt des zweiten Bytes des Ergebnisstrings abgefragt. Dieses enthält die gefundene Modellklasse. Über bedingte Anweisungen wird je nach Inhalt des zweiten Bytes eine Boolesche Variable gesetzt, welche als Ereignis genutzt werden kann.

Das Ergebnis der optischen Endkontrolle wird zusätzlich auf dem Touchpanel an Station 60 visualisiert. Hierfür wird das bereits vorhandene Bild, welches die auf dem RFID-Tag des Werkstückträgers gespeicherten Daten anzeigt, erweitert. Es werden vier Anzeigefelder für die Modellklassen in das Bild eingefügt. Je nach Ergebnis der optischen Endkontrolle wird das entsprechende Anzeigefeld grün hinterlegt. Das Bild ist in Abbildung 6.10 dargestellt.

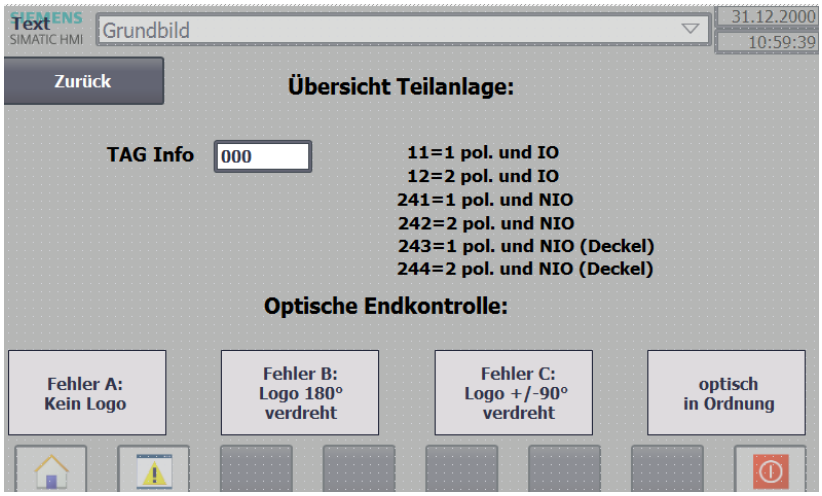


Abbildung 6.10.: Visualisierung des Ergebnisses der optischen Endkontrolle

Für die Visualisierung des Supervisors S_3 wird ein neues Bild für das Touchpanel erstellt. Dieses ist wie das Bild des Supervisors S_1 aufgebaut und in Abbildung 6.11 dargestellt.

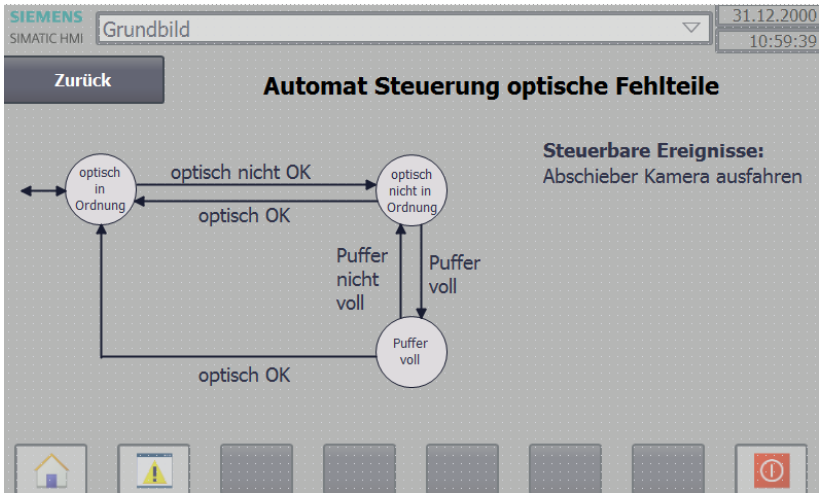


Abbildung 6.11.: Visualisierung des Supervisors S_3

7. Auswertung

In diesem Kapitel werden die implementierten Steuerungen hinsichtlich ihrer Funktion anhand verschiedener Testszenarien überprüft. Dazu gehören die korrekte Klassifikation der Fehlteile, die Einhaltung der Pufferkapazität, sowie der Pufferüberlauf. Nachfolgend werden die einzelnen Testszenarien und deren Ergebnisse beschrieben.

Testszenario 1: Korrekte Klassifikation

In dieser Messung wird geprüft, ob die im Programm des Codelesegeräts eingestellte Genauigkeit ausreichend hoch ist. Dafür wird die optische Endkontrolle für 100 Werkstücke durchgeführt. Es werden jeweils 25 Werkstücke der einzelnen Modellklassen geprüft. Die Reihenfolge ist dabei zufällig.

Tabelle 7.1.: Klassifikationsergebnisse bei einer Genauigkeit von 50 %

	Klasse A	**Klasse B**	**Klasse C**	**Klasse D**
Richtig zugeordnet	25	25	25	25
Falsch zugeordnet	0	0	0	0

Tabelle 7.1 zeigt, dass bei einer Genauigkeit von 50 % alle Werkstücke richtig klassifiziert wurden. Für ein aussagekräftiges Ergebnis, wird diese Messung zum Vergleich für eine Genauigkeit von 30 % und 70 % erneut durchgeführt.

Tabelle 7.2.: Klassifikationsergebnisse bei einer Genauigkeit von 30 %

	Klasse A	**Klasse B**	**Klasse C**	**Klasse D**
Richtig zugeordnet	25	19	23	17
Falsch zugeordnet	0	6	2	8

Tabelle 7.3.: Klassifikationsergebnisse bei einer Genauigkeit von 70 %

	Klasse A	**Klasse B**	**Klasse C**	**Klasse D**
Richtig zugeordnet	25	25	25	25
Falsch zugeordnet	0	0	0	0

Aus Tabelle 7.2 kann entnommen werden, dass bei einer Genauigkeit von 30 % insgesamt 16 Werkstücke nicht richtig klassifiziert wurden. Eine Genauigkeit von 30 % ist demnach für diese Anwendung nicht ausreichend. Wie zu erwarten werden bei einer Genauigkeit von 70 % alle Werkstücke richtig zugeordnet (siehe Tabelle 7.3). Jedoch benötigt das Codelesegerät mehr Zeit zur Auswertung des Ergebnisses. Der Vergleich der Ergebnisse zeigt, dass eine Genauigkeit von 50 % für diese Anwendung ausreichend ist, da auch bei dieser Genauigkeit alle Werkstücke korrekt zugeordnet wurden.

Testszenario 2: Einzelfehler

In diesem Test wird die Funktionalität der Steuerung im Falle eines Einzelfehlers überprüft. Es werden insgesamt drei Testläufe durchgeführt, in denen jeweils ein Fehlteil produziert wird. Es wird überprüft, ob das Werkstück korrekt klassifiziert, zum Handarbeitsplatz transportiert und richtig abgeschoben wird. Im ersten Testlauf wird ein Fehlteil der Kategorie A, im zweiten der Kategorie B und im dritten der Kategorie C produziert. In allen drei Fällen wird das Werkstück von dem Codelesegerät als Fehlteil erkannt und der richtigen Fehlerkategorie zugeordnet. Jedes Fehlteil wurde, sobald es sich am Bandende von Station 60 befindet, von der Linearachse abgeholt und zum Handarbeitsplatz transportiert. Die einzelnen Fehlteile werden vom Transportband bis zum entsprechenden Abschieber gefahren und in die richtige Kiste abgeschoben. Die Steuerungsaufgabe wird im Falle eines Einzelfehlers erfüllt.

Testszenario 3: Puffer befüllen und entleeren

Mit Hilfe dieses Testszenarios wird die korrekte Funktion der Puffersteuerung und des Assistenzsystems überprüft. Dafür werden insgesamt 3 Werkstücke derselben Fehlerkategorie produziert. Zunächst wird der Puffer bis zu seiner maximalen Kapazität von zwei Fehlteilen befüllt. Anschließend werden die Fehlteile repariert und der Kiste entnommen. Es wird geprüft, ob das dritte Werkstück nach der Entleerung in die entsprechende Kiste abgeschoben wird.

Die ersten beiden Werkstücke werden, wie im vorherigen Testszenario, von dem Codelesegerät der richtigen Fehlerkategorie zugeordnet und zum Handarbeitsplatz transportiert. Es handelt sich bei allen Werkstücken um Fehlteile der Kategorie A. Die beiden Werkstücke werden nacheinander vom ersten Abschieber am Handarbeitsplatz abgeschoben. Der Puffer der Fehlerkategorie A ist somit voll. Auf dem Touchpanel des Assistenzsystems steht nur die Reparatur eines Fehlteils dieser Kategorie zur Auswahl. Über die Betätigung der entsprechenden Schaltfläche wird die Reparaturanleitung aufgerufen. Dadurch beginnt der LED-Streifen der Kiste grün zu leuchten. Nach dem Betätigen der Schaltfläche „Reparatur abgeschlossen“ gelangt man zurück zum Startbild und der LED-Streifen hört auf zu leuchten. Dieser Vorgang wird wiederholt. Nachdem die Reparatur der beiden Fehlteile abgeschlossen ist, wird das dritte Werkstück der optischen Endkontrolle zugeführt. Dieses wird ebenfalls

der richtigen Fehlerkategorie zugeordnet, zum Handarbeitsplatz transportiert und vom ersten Abschieber ausgeschleust. Dieses Testszenario wird für die Fehlerkategorien B und C analog durchgeführt. Das Ergebnis dieses Testszenarios zeigt, dass die Puffersteuerung, sowie das Assistenzsystem korrekt funktionieren.

Testszenario 4: Pufferüberlauf

Die korrekte Funktion der Steuerung im Falle eines Pufferüberlaufs wird mit Hilfe des vierten Testszenarios geprüft. Für diesen Test werden vier Werkstücke je Fehlerkategorie produziert. Zunächst werden die drei Puffer nacheinander mit jeweils zwei Fehlteilen befüllt. Anschließend wird überprüft, ob das dritte Fehlteil der einzelnen Fehlerkategorien von dem Abschieber unter der Kamera abgeschoben wird. Die Puffer werden geleert und die verbliebenen drei Werkstücke der optischen Endkontrolle zugeführt.

Die ersten sechs Werkstücke werden von dem Codelesegerät den richtigen Fehlerkategorien zugeordnet. Nachdem der Puffer der Fehlerkategorie A vollständig befüllt ist, werden die Puffer der Kategorien B und C befüllt. Die nächsten drei Fehlteile werden nach der optischen Endkontrolle von dem Abschieber unter der Kamera abgeschoben. Nachdem die Puffer geleert wurden, werden die verbliebenen Werkstücke dem Codelesegerät zugeführt. Die einzelnen Fehlteile werden richtig zugeordnet und von der Linearachse zum Handarbeitsplatz transportiert. Die Sortierung der Fehlteile am Handarbeitsplatz wird korrekt durchgeführt. Im Falle eines Pufferüberlaufs arbeitet die Steuerung korrekt und erfüllt die Steuerungsaufgabe.

Ergebnis

Mit Hilfe der beschriebenen Testszenarien konnte die korrekte Funktion der Steuerungen und die richtige Parametrierung des Codelesegeräts sichergestellt werden. Sowohl die Spezifikationen, als auch die Streckenkomponenten aus Kapitel 5 wurden korrekt modelliert. Die geforderte Steuerungsaufgabe wird erfüllt.

8. Zusammenfassung und Ausblick

8.1. Zusammenfassung

Im Rahmen dieser Masterarbeit wurde eine Lösung für die Handhabung von optischen Fehlteilen innerhalb der Modellfabrik entwickelt. An Station 60 wurde das bereits vorhandene Codelesegerät der Firma Siemens in die Anlage integriert. Hierfür wurde ein Programm für die Erkennung und Klassifikation von Fehlteilen erstellt. Dieses kann zwischen drei verschiedenen Fehlerfällen unterscheiden. Das Ergebnis der optischen Endkontrolle wurde in das bestehende SPS-Programm integriert, sodass die manuelle Entscheidung über Taster entfällt.

Für die Reparatur der erkannten Fehlteile wurde ein Handarbeitsplatz konzipiert und realisiert. Dabei wurde das Industrie 4.0 Leitbild berücksichtigt. Der entwickelte Handarbeitsplatz ermöglicht die Sortierung der Fehlteile anhand ihrer Fehlerkategorien und ist mit einem Assistenzsystem ausgestattet. Dieses unterstützt die Mitarbeiter, durch einfache Schritt für Schritt Anleitungen, bei der Reparatur der Werkstücke. Zusätzlich wurde das Pick-by-Light Konzept angewendet, um ein falsches Verhalten des Mitarbeiters zu vermeiden.

Es wurde für das Handling der optischen Fehlteile eine ereignisdiskrete Steuerung konzipiert und synthetisiert. Die Synthese erfolgte nach den Methoden der Supervisory Control Theory von Ramadge und Wonham. Für die Modellierung der notwendigen Streckenkomponenten und der Spezifikationen wurden Generatoren als Beschreibungsform genutzt. Die Steuerungssynthese erfolgte nach dem lokal-modularen Ansatz und wurde mit Hilfe von DESTool durchgeführt. Für die Umsetzung der berechneten Supervisor in Programmcode wurde der Codegenerator ACArrow eingesetzt. Die generierten S7-SCL Quelldateien wurden in das TIA-Portal importiert und übersetzt. Die Implementierung der lokal-modularen Supervisor erfolgte auf den Steuerungen an Station 20 und Station 60. Des Weiteren wurde die unterlagerte Steuerung erweitert.

Mit Hilfe verschiedener Testszenarien wurde die lokal-modulare Steuerung hinsichtlich ihrer Funktion überprüft. Dabei wurde die korrekte Funktion der Supervisor sichergestellt. Die geforderte Steuerungsaufgabe wird erfüllt.

8.2. Ausblick

In weiterführenden Arbeiten kann die optische Endkontrolle durch die Nutzung eines anderen Vision Sensors verbessert werden. Mit Hilfe des verwendeten Codelesegeräts SIMATIC MV440 können Fehler wie zum Beispiel zerkratzte Deckel nicht erkannt werden. Es existieren zu viele Möglichkeiten, an welcher Stelle sich ein Kratzer befinden könnte, sodass kein Modell in der Modellbibliothek dafür erstellt werden kann. Mit Hilfe einer Industriekamera und geeigneten Bildverarbeitungsalgorithmen könnte die Erkennung solcher Fehler realisiert werden.

Des Weiteren kann das Assistenzsystem am Handarbeitsplatz durch eine Kameraüberwachung erweitert werden. Dies ermöglicht Fehlgriffe der Mitarbeiter gezielt zu verhindern. Zusätzlich könnten die Werkstücke mit RFID Tags ausgestattet werden auf denen die Fehlerkategorie gespeichert wird. Über ein RFID Lesegerät am Handarbeitsplatz könnte der Tag ausgelesen und die Anleitung automatisch aufgerufen werden.

Allgemein kann der Montageprozess der Modellfabrik weiter optimiert werden. Es kann untersucht werden, inwieweit die Montage der Werkstücke beschleunigt werden kann. Besonders in Bezug auf die Linearachse und die Handlingeinheiten an Station 30 und Station 60. Auch die parallele Bearbeitung mehrerer Werkstücke auf den Transportbändern an den Stationen 30 und 60 könnte in einer weiteren Arbeit untersucht und realisiert werden. Hierbei müsste die Genauigkeit erneut untersucht und angepasst werden, da durch ein erhöhtes Arbeitsaufkommen ebenfalls ein erhöhtes Risiko entsteht, dass Fehlteile produziert beziehungsweise nicht erkannt werden.

Ein weiterer Punkt ist die Optimierung der Inbetriebnahme der Modellfabrik. Der Einschaltprozess ist sehr komplex und zeitaufwändig. Eine Untersuchung, inwieweit dies automatisiert werden kann, wäre sinnvoll, um den Arbeitsprozess effizienter zu gestalten.

Literaturverzeichnis

[1] AG, Siemens: *Simatic S7-1500 Getting Started*. 2014. – URL https://www.automation.siemens.com/salesmaterial-as/interactive-manuals/getting-started_simatic-s7-1500/documents/DE/software_complete_de.pdf. – Zugriffsdatum: 05.06.2018

[2] AG, Siemens: *Simatic Ident Code-Lesesysteme Simatic MV420/Simatic MV440 Betriebsanleitung*. 2015. – URL https://cache.industry.siemens.com/dl/files/392/84553392/att_863563/v1/BA_MV420-MV440_0_de-DE.pdf. – Zugriffsdatum: 05.06.2018

[3] ANDERL, Reiner ; DUMITRESCU, Roman ; EIGNER, Martin ; GANZ, Christopher ; HUBER, Anton S. ; MICHELS, Jan S. ; SENDLER, Ulrich (Hrsg.): *Industrie 4.0 grenzenlos*. Springer Vieweg, 2016. – ISBN 978-3-662-48277-3

[4] BAUERNHANSL, Thomas (Hrsg.) ; HOMPEL, Michael ten (Hrsg.) ; VOGEL-HEUSER, Birgit (Hrsg.): *Industrie 4.0 in Produktion, Automatisierung und Logistik*. Springer Vieweg, 2014. – ISBN 978-3-658-04681-1

[5] CAMUR, Safa: *Synthese und Implementierung einer ereignisdiskreten Puffersteuerung für eine flexible Montageanlage*, Hochschule für Angewandte Wissenschaften Hamburg, Masterthesis, 2017. – URL http://edoc.sub.uni-hamburg.de/haw/volltexte/2018/4282/pdf/Masterthesis_Safa_Camur.pdf. – Zugriffsdatum: 15.07.2018

[6] CASSANDRAS, Christos G. ; LAFORTUNE, Stéphane: *Introduction to Discrete Event Sysrems*. Second Edition. Springer Science+Business Media, 2008. – ISBN 978-1-4419-4119-0

[7] FENG, Lei ; WONHAM, W. M.: TCT: A Computation Tool for Supervisory Control Synthesis. In: *Proceedings of the 8th Intertational Workshop on Discrete Event Systems* (2006), S. 388–389

[8] GOHERT, Nadine: *Automatische SPS-Codegenerierung für Syntheseverfahren der Supervisory Control Theory*, Hochschule für Angewandte Wissenschaften Hamburg, Masterthesis, 2014. – URL http://edoc.sub.uni-hamburg.de/haw/volltexte/2015/2887/pdf/MAThesis.pdf.. – Zugriffsdatum: 10.06.2018

[9] IORDACHE, Marian V. ; ANTSAKLIS, Panos J.: *SPNBOX*. 2013. – URL `http://www.letu.edu/people/marianiordache/abs/spnbox/.`. – Zugriffsdatum: 13.06.2018

[10] LEÓN, Fernando P. ; KIENCKE, Uwe: *Ereignisdiskrete Systeme Modellierung und Steuerung verteilter Systeme*. 3. Auflage. Oldenbourg Verlag München, 2013. – ISBN 978-3-486-73574-1

[11] LIN, F. ; WONHAM, W. M.: Decentralized supervisory control of discrete-event systems. In: *Information Sciences 44* (1988), Nr. 3

[12] LIN, F. ; WONHAM, W. M.: Decentralized control and coordination of discrete-event systems with partial observation. In: *IEEE Transactions on Automatic Control 35* (1990), Nr. 12, S. 1330–1337

[13] LUNZE, Jan: *Ereignisdiskrete Systeme Modellierung und Analyse dynamischer Systeme mit Automaten, Markovketten und Petrinetzen*. 2. Auflage. Oldenbourg Verlag München, 2012. – ISBN 978-3-486-71885-0

[14] MOOR, Thomas: *DESTool*. 2016. – URL `http://www.rt.eei.uni-erlangen.de/FGdes/destool/`. – Zugriffsdatum: 13.06.2018

[15] PETRI, Carl A.: Kommunikation mit Automaten / Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn. Bonn : Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, 1962 (2). – Dissertation, Schriften des IIM

[16] QUEIROZ, Max H. de ; CURY, J. E. R.: Modular Control of Composed Ssystems. In: *Proceedings of the American Control Conference* (2000), S. 4051–4055

[17] QUEIROZ, Max H. de ; CURY, J. E. R.: Modular Supervisory Control of Large Scale Discrete Event Systems. In: R.BOEL (Hrsg.) ; STREMERSCH, G. (Hrsg.): *Discrete Event Systems: Analysis and Control*. Kluwer Academic Publishers, 2000, S. 103–110

[18] QUEIROZ, Max H. de ; CURY, J. E. R.: Synthesis and Implementation of Local Modular Supervisory Control for a Manufacturing Cell. In: *Proc. of 6th International Workshop on Discrete Event Systems* (2002), S. 377–382

[19] RAMADGE, P. J.: *Supervision of discrete event processes*, Dep. of Electrical and Computer Engineering, University of Toronto, Dissertation, 1983

[20] RAMADGE, P. J.: Supervisory Control of Discrete Event Systems: A Survey and Some New Results. In: VARAIYA, P. (Hrsg.) ; KURZHANSKI, A. B. (Hrsg.): *Discrete Event Systems: Models and Applications. IIASA Conference Sopron, Hungary*. Springer Verlag, 1987, S. 69–80

[21] RAMADGE, P. J. ; WONHAM, W. M.: The Control of Discrete Event Systems. In: *Proc. IEEE, Special Issue on Discrete Event Dynamic Systems 77* (1989), Nr. 1, S. 81–98

[22] SU, R. ; WONHAM, W. M.: Supervisor Reduction for Discrete-Event Systems. 14 (2004), Nr. 1, S. 31–53

[23] UZAM, M. ; GELEN, G. ; DALCI, R.: A New Approach for the Ladder Logic Implementation of Ramadge-Wonham Supervisors. In: *XXII International Symposium on Information, Communication and Automation Technologies* (2009), S. 1–7

[24] WENCK, Florian: *Modellbildung, Analyse und Steuerungsentwurf für gekoppelte ereignisdiskrete Systeme*. Shaker Media Verlag, 2006. – ISBN 978-3-8322-5573-2

[25] WONHAM, W. M. ; RAMADGE, P. J.: On the Supremal Controllable Sublanguage of a Given Language. In: *SIAM Journal on Control and Optimization 25* (1987), Nr. 3, S. 637–659

[26] WONHAM, W. M. ; RAMADGE, P. J.: Supervisory Control of a class of discrete event processes. In: *SIAM Journal on Control and Optimization 25* (1987), Januar, Nr. 1, S. 206–230

[27] WONHAM, W. M. ; RAMADGE, P. J.: Modular Supervisory Control of Discrete-Event Systems. In: *Mathematics of Control, Signals, and Systems* (1988), Nr. 1, S. 13–30

[28] WONHAM, W. M. ; RAMADGE, P. J.: The Control of Discrete Event Systems. In: *Proc. IEEE, Special Issue on Discrete Event Dynamic Systems 77* (1989), Januar, Nr. 1, S. 81–98

[29] WONHAM, W.M.: *Supervisory Control of Discrete-Event Systems*. Dept. of Electrical and Computer Engineering, University of Toronto, Kanada. 2004

A. Anhang

Der Anhang dieser Masterarbeit befindet sich auf der beiliegenden CD, die beim Erst- und Zweitprüfer eingesehen werden kann.

A.1. Anleitung Inbetriebnahme Modellfabrik (CD)

Der Ordner „Anleitung“ beinhaltet die Anleitung für das Ein- und Ausschalten der Modellfabrik.

A.2. DESTool Projekt (CD)

Der Ordner „DESTool“ beinhaltet das angelegte DESTool Projekt mit den Modellen der Streckenkomponenten und Spezifikationen.

A.3. S7-SCL Quelldateien (CD)

Der Ordner „ACArrow“ enthält die mit Hilfe des Codegenerators ACArrow generierten Quelltextdateien.

A.4. TIA-Projekt der Modellfabrik (CD)

Der Ordner „TIA“ beinhaltet die archivierten TIA-Projekte für die Steuerung der Modellfabrik und der Visualisierung des Assistenzsystems.

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 21. August 2018	
Ort, Datum	Unterschrift