Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Annika Schakowsky

Steuerungssynthese zur Optimierung eines ereignisdiskreten Demontageprozesses mit einem 6-Achsen-Knickarmroboter

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

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

Annika Schakowsky

Steuerungssynthese zur Optimierung eines ereignisdiskreten Demontageprozesses mit einem 6-Achsen-Knickarmroboter

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. Annabella Rauscher-Scheibe

Abgegeben am 19. Dezember 2017

Annika Schakowsky

Thema der Masterthesis

Steuerungssynthese zur Optimierung eines ereignisdiskreten Demontageprozesses mit einem 6-Achsen-Knickarmroboter

Stichworte

Steuerungssynthese, Supervisory Control Theory, SCT, ereignisdiskrete Systeme, Automaten, Modellfabrik, DESTool, Steuerungsentwurf, Demontageprozess

Kurzzusammenfassung

Diese Thesis befasst sich mit der Steuerungssynthese eines ereignisdiskreten Demontageprozesses innerhalb einer Modellfabrik. Zuerst wird die bestehende Steuerung der Modellfabrik optimiert und erweitert. Anschließend folgt der Steuerungsentwurf auf Basis der Supervisory Control Theory. Zur Realisierung des Demontageprozesses wird ein 6-Achsen-Knickarmroboter verwendet. Es werden mehrere Steuerungsvarianten entworfen und realisiert, die miteinander verglichen werden.

Annika Schakowsky

Title of the paper

Control synthesis to optimize a discrete event dismantling process with a 6-axis articulated robot

Keywords

Control synthesis, Supervisory Control Theory, SCT, discrete event systems, automata, model factory, DESTool, control design, dismantling process

Abstract

This thesis deals with the control synthesis of a discrete event dismantling process within a model factory. First the existing control of the model factory is optimized and extended. Then it is followed by the control design based on the Supervisory Control Theory. A 6-axis articulated robot is used for realization of the dismantling process. Various control options are designed and realized, which are compared with each other.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Symbol- und Abkürzungsverzeichnis

Symbole

$\in$	Element von
$\notin$	Kein Element von
$\cup$	Vereinigung zweier Mengen
$\cap$	Schnitt zweier Mengen
$\setminus$	Differenz zweier Mengen
$\subset$ / $\subseteq$	Teilmenge
A^C	Komplement von A
$\times$	Kreuzprodukt zweier Mengen
Σ	Endliches Alphabet
$\varnothing$	Leere Menge
ε	Leeres Wort
Σ^*	Menge aller Worte über Σ
Σ^+	Menge aller nichtleeren Worte über Σ
$\exists$	Existenzquantor („es gibt mindestens ein“)
L	Formale Sprache
$L(A)$	Generierte Sprache von A
$L_m(A)$	Markierende Sprache von A
L^*	Kleene-Abschluss der Sprache L
$\overline{L}$	Präfix-Hülle der Sprache L
δ	Zustandsübergangsfunktion / erweiterte Zustandsübergangsfunktion
$\hat{\delta}$	Erweiterte Zustandsübergangsfunktion

$\delta(q, a)!$ Existenz einer Kante im Zustand q für das Symbol a

$\neg\delta(q, a)!$ Nichtexistenz einer Kante im Zustand q für das Symbol a

q_0 Startzustand

F Menge akzeptierter Zustände

Σ_c Steuerbare Ereignisse

Σ_{uc} Nicht steuerbare Ereignisse

Σ_o Beobachtbare Ereignisse

Σ_{uo} Nicht beobachtbare Ereignisse

Σ_{ce} Gemeinsame Ereignisse

Σ_{pe} Private Ereignisse

$||_{SPC}$ Strenge Synchronisation

$||_{SYPC}$ Synchrones Produkt

$K^{\uparrow C}$ Supremale steuerbare Teilsprache

Abkürzungen

AG Aktiengesellschaft

AS-i Aktor-Sensor-Interface

AWL Anweisungsliste

BSCP-NB Basic Supervisory Control Problem - Nichtblockierend

DEA Deterministischer endlicher Automat

DES Discrete event systems, ereignisdiskretes System

DESTool Analyse- und Synthesetool für ereignisdiskrete Systeme

DIN Deutsches Institut für Normung

FBS Funktionsbausteinsprache

GDPT Globales dezentrales Problem mit Toleranz

GDPZT Globales dezentrales Problem ohne Toleranz

GmbH Gesellschaft mit begrenzter Haftung

HAW Hochschule für angewandte Wissenschaften

IEC International Electrotechnical Commission

IO Eingang/Ausgang (engl.: Input/Output)

KOP Kontaktplan

LDPT Lokales dezentrales Problem mit Toleranz

LDPZT Lokales dezentrales Problem ohne Toleranz

MSCP Modular Supervisory Control Problem

PLC Programmable Logic Controller

Profinet Process Field Network

RFID Radio-Frequency Identification

SCL Structured Control Language

SCT Supervisory Control Theory

SPC Strenge Synchronisation

SPS Speicherprogrammierbare Steuerung

ST Strukturierter Text

SYPC Synchrones Produkt

TCT Synthesetool für ereignisdiskrete Systeme

TIA Totally Integrated Automation

WinCC Windows Control Center

1. Einführung

Unter der Automatisierungstechnik wird verstanden, Systeme so zu steuern, dass diese selbstständig ihre Funktion erfüllen. Typische Ziele einer Automatisierung sind beispielsweise, den wirtschaftlichen, zuverlässigen und sicheren Betrieb einer Anlage oder eines Produktes sicherzustellen. Durch Überwachen und Melden werden physikalische Größen oder Prozesszustände fortlaufend auf Abweichungen überwacht. Das Anzeigen und Bedienen ermöglicht einer Person, durch Beobachten und gezielte Eingaben Einfluss auf den Prozess zu nehmen. Die Automatisierungstechnik kann in zwei Teilbereiche unterschieden werden: die Regelungstechnik und die Steuerungstechnik. Das Ziel einer Regelung ist es, Prozessgrößen trotz Störungen konstant zu halten beziehungsweise an variable Führungsgrößen anzupassen. Bei einer Steuerung werden die Prozessgrößen durch von außen vorgegebene Führungsgrößen gezielt beeinflusst, um einen vorgegebenen Ablauf des Prozesses zu gewährleisten. Zum Entwurf einer Steuerung wird in der Praxis oft der direkte Entwurf eingesetzt, bei der die Steuerung intuitiv entwickelt wird, die Umsetzung aber sehr fehleranfällig ist. Aus diesem Grund gibt es einen weiteren Entwurf mittels Synthese, bei der bereits vor der Realisierung Fehler ermittelt werden können. Ein Ansatz, der auf den Entwurf mittels Synthese aufbaut, ist die Supervisory Control Theory (SCT). [Litz (2013), S. 6f; Lunze (2012a), S. 1f; Zander (2015), S. 4]

1.1. Zielsetzung der Thesis

In dieser Thesis werden mehrere übergeordnete Steuerungen für die Modellfabrik, die sich an der Hochschule für angewandte Wissenschaften Hamburg (HAW) befindet, entworfen. Die Modellfabrik bildet einen realen Automatisierungsprozess ab, in dem Relais für Windkraftanlagen im Gehäuse mit Verschluss hergestellt werden. Zunächst soll die bestehende Steuerung der Modellfabrik optimiert und um neue Funktionen erweitert werden. Die Optimierung soll erfolgen, damit der Betrieb der Modellfabrik zuverlässiger, sicherer und schneller wird. In der Erweiterung soll mit Hilfe eines modellbasierten Ansatzes ein Verfahren entwickelt werden, welches elektrisch fehlerhafte Werkstücke demontiert und intakte Komponenten wieder dem Produktionsprozess hinzufügt. Dazu sollen verschiedene Steuerungsvarianten entwickelt werden, bei denen ein 6-Achsen-Knickarmroboter und eine Linearachse an der Rückführung beteiligt sind. Ziel ist es, den Ausschuss von intakten Komponenten

zu reduzieren. Der modellbasierte Ansatz soll verwendet werden, da er gegenüber anderen Ansätzen, wie zum Beispiel der direkten Umsetzung, Vorteile bietet. Beispielsweise werden Fehler bereits vor der Inbetriebnahme gefunden. Die Modellfabrik soll mit einer Beschreibungsform, wie zum Beispiel Automaten oder Petrinetze, modelliert und die Steuerungsvarianten mit Hilfe eines Synthesetools entworfen werden. Die Realisierung erfolgt mit einem automatischen Codegenerator. Anschließend sollen die Steuerungsvarianten anhand verschiedener Kriterien miteinander verglichen werden, um die beste Variante zu finden.

1.2. Gliederung der Thesis

Diese Thesis umfasst acht Kapitel. Nach der Einführung in diesem Kapitel werden in Kapitel 2 Grundlagen zu verschiedenen Themen dieser Thesis eingeführt. Es wird unter anderem auf Automaten und den modellbasierten Steuerungsentwurf mit Hilfe der Supervisory Control Theory (SCT) eingegangen. Kapitel 3 beschreibt die Funktion der Modellfabrik und die vorgenommenen Optimierungen. Die Konzeption, in der verschiedene Entwicklungstools, Beschreibungsformen und Steuerungsansätze miteinander verglichen und bewertet werden, erfolgt in Kapitel 4. In Kapitel 5 werden die Streckenmodelle, die für die Modellfabrik erstellt werden, sowie die Spezifikationen an die Steuerung beschrieben. Zusätzlich erfolgt hier der Entwurf der Steuerungen. Die entworfenen Steuerungen werden in Kapitel 6 in die bestehende Steuerung der Modellfabrik integriert. Zuvor wird die bestehende untergeordnete Steuerung um einige Funktionen erweitert. In Kapitel 7 werden die entwickelten Steuerungsvarianten anhand verschiedener Kriterien miteinander verglichen und anhand dessen eine Empfehlung, welche Variante am besten geeignet ist, gegeben. Eine Zusammenfassung dieser Thesis und ein Ausblick auf mögliche Erweiterungen erfolgen in Kapitel 8.

2. Grundlagen

In diesem Kapitel werden Grundlagen für alle Begriffe, die in dieser Thesis verwendet werden, eingeführt. Abschnitt 2.1 umfasst einige mengentheoretische Grundlagen, die unter anderem im Abschnitt 2.2 zur Einführung der formalen und regulären Sprachen verwendet werden. Abschnitt 2.3 beschreibt die Eigenschaften ereignisdiskreter Systeme. In Abschnitt 2.4 werden Grundlagen zur Graphentheorie eingeführt, die im Abschnitt 2.5 zur Beschreibung von Automaten verwendet werden. Abschnitt 2.6 beschreibt die Modellierung ereignisdiskreter Systeme. Anschließend folgen in Abschnitt 2.7 und 2.8 Grundlagen zum Steuerungsentwurf und der Supervisory Control Theory (SCT). Am Ende des Kapitels wird in Abschnitt 2.9 die verwendete Software zum Steuerungsentwurf vorgestellt. Die verwendete Literatur wird am Ende eines Abschnittes angegeben und gilt für den gesamten Abschnitt beziehungsweise bis zur letzten Literaturangabe innerhalb eines Abschnittes.

2.1. Mengentheoretische Grundlagen

Eine Menge ist die Zusammenfassung unterscheidbarer Objekte. Mit den Elementen einer Menge werden die zu der Menge gehörenden Objekte bezeichnet. $x \in A$ bedeutet, dass ein Element x Teil einer Menge A ist, für $x \notin A$ ist x nicht Teil der Menge A. Für zwei Mengen A und B gelten folgende Definitionen:

Vereinigung

$$A \cup B = \{x \quad | x \in A \text{ oder } x \in B\}, \tag{2.1}$$

Schnitt

$$A \cap B = \{x \quad | x \in A \text{ und } x \in B\}, \tag{2.2}$$

Differenz

$$A \backslash B = \{x \quad | x \in A \text{ und } x \notin B\}, \tag{2.3}$$

Teilmenge

$$A \subseteq B = \{x \quad | x \in A \Rightarrow x \in B\}. \tag{2.4}$$

A heißt Teilmenge von B, wenn alle Elemente aus A auch Elemente von B sind. Der Operator $\subset$ wird verwendet, wenn die Gleichheit der Mengen A und B explizit ausgeschlossen ist, der Operator $\subseteq$ erlaubt auch die Gleichheit der beiden Mengen.

Potenzmenge

$$2^B = \{A \quad | A \subseteq B\}. \tag{2.5}$$

Die Potenzmenge 2^B ist die Menge aller Teilmengen von B.

Komplement

$$A^C = \{x \in \Omega \quad | x \notin A\} \tag{2.6}$$

mit $\Omega =$ Grundmenge bzw. Menge aller betrachteten Objekte.

Kreuzprodukt

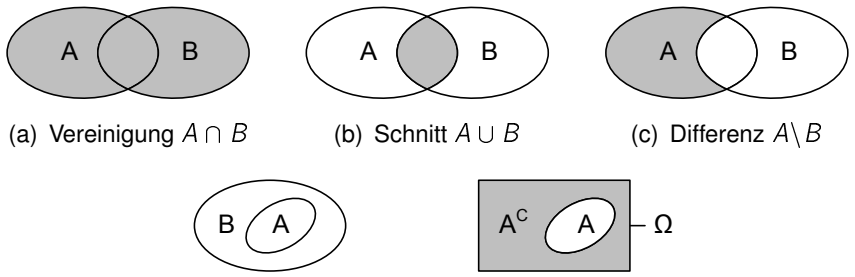
$$A \times B = \{(a, b) \quad | a \in A \text{ und } b \in B\}. \tag{2.7}$$

Beim Kreuzprodukt darf die Reihenfolge der Komponenten a und b eines Elements (a, b) nicht vertauscht werden.

Die Venn-Diagramme aus Abbildung 2.1 zeigen die graphische Darstellung der Operationen aus Definition 2.1 bis 2.4 und 2.6. [Cramer und Nešlehová (2015), S. 2, S. 48, S. 60; Lunze (2012b), S. 612; Modler und Kreh (2014), S. 13f, S. 20f]

2.2. Formale und reguläre Sprachen

Zur Begriffsdefinition der regulären Sprachen werden zunächst einige Grundbegriffe und Operationen nach Tabelle 2.1 eingeführt.



(a) Vereinigung $A \cap B$ (b) Schnitt $A \cup B$ (c) Differenz $A \backslash B$

(d) Teilmenge $A \subseteq B$ (e) Komplement A^C

Abbildung 2.1.: Mengenoperationen nach [Cramer und Nešlehová (2015), S. 48; Modler und Kreh (2014), S. 20f]

Abkürzung	**Bedeutung**
$a, b, c, ...$	Symbole
$\Sigma = \{a, b, c, ...\}$	Endliches Alphabet
$w = a_0 ... a_{n-1}$ mit $a_i \in \Sigma$ für $i = 0, ..., n-1$	Folge oder endliches Wort über Σ
$\lvert w \rvert$	Länge von w
$w(i)$	i-tes Symbol a_i von w
p mit $w = ps$	Präfix von w
s mit $w = ps$	Suffix von w
wv	Konkatenation zweiter Worte w und v
$\varnothing$	Leere Menge
ε	Leeres Wort oder Folge der Länge null
Σ^*	Menge aller Worte über Σ
Σ^+	Menge aller nichtleeren Worte über Σ

Tabelle 2.1.: Grundbegriffe und Operationen

Die Konkatenation der Worte $w = w_1 w_2 ... w_m$ und $v = v_1 v_2 ... v_n$ hängt die Folge v hinten an die Folge w an, sodass $wv = w_1 w_2 ... w_m v_1 v_2 ... v_n$ entsteht. Das leere Wort ε ist das neutrale Element bezüglich der Konkatenation von Worten, sodass gilt:

$$w\varepsilon = \varepsilon w = w \quad \text{für alle } w \in \Sigma^*. \tag{2.8}$$

Eine (formale) Sprache L ist eine Menge von Worten, also jede Teilmenge $L \subseteq \Sigma^*$. Aus diesem Grund können mengentheoretische Operationen wie Vereinigung, Schnitt, Komplement

und Differenz auf Sprachen angewendet werden. Für Sprachen gelten folgende Begriffe nach Tabelle 2.2.

Abkürzung	Bedeutung
L^C	Komplement der Sprache L
L_1L_2	Konkatenation der Sprachen L_1 und L_2
L^*	Kleene-Abschluss der Sprache L
$\overline{L}$	Präfix-Hülle der Sprache L

Tabelle 2.2.: Operationen auf Sprachen

Das Komplement L^C der Sprache L enthält alle Elemente aus Σ^*, die nicht Teil der Sprache L sind. Die Konkatenation L_1L_2 zweier Sprachen L_1 und L_2 besteht aus allen Worten, die sich durch die Konkatenation eines Wortes aus L_1 mit einem Wort aus L_2 ergeben. Der Kleene-Abschluss der Sprache L ist die Vereinigung aller Konkatenationen der Sprache L mit sich selbst. Die Präfix-Hülle der Sprache L ist definiert durch:

$$\overline{L} = \{p \in \Sigma^* \mid \exists s \in \Sigma^*(ps \in L)\}. \tag{2.9}$$

$\exists s \in \Sigma^*$ bedeutet, dass es mindestens ein $s \in \Sigma^*$ gibt, für das die Bedingung in der runden Klammer zutrifft. Gleichung (2.9) besagt, dass die Präfix-Hülle $\overline{L}$ aus sämtlichen Präfixen aller Worte $p \in L$ besteht. Gilt $L = \overline{L}$, heißt die Sprache L präfix-abgeschlossen.

Die Menge aller regulären Sprachen über Σ^* wird induktiv definiert durch:

1. Die leere Menge $\varnothing$ und die Menge $\{\varepsilon\}$, die nur aus dem leeren Wort besteht, sind reguläre Sprachen.
2. Die Mengen $\{a\}$ für alle $a \in \Sigma$ sind reguläre Sprachen.
3. Sind R_1 und R_2 reguläre Sprachen über Σ, dann sind auch die Vereinigung $R_1 \cup R_2$ und die Konkatenation R_1R_2 reguläre Sprachen über Σ.
4. Ist R eine reguläre Sprache über Σ, dann ist auch der Kleene-Abschluss R^* eine reguläre Sprache über Σ.

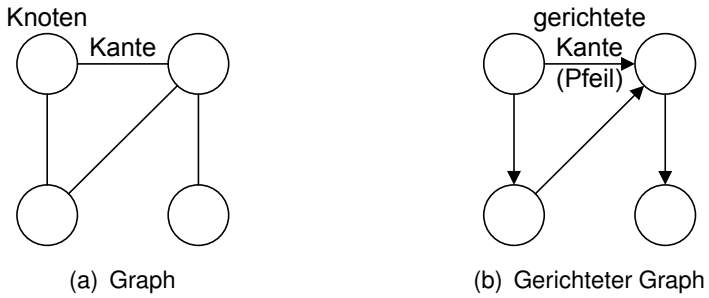
[Güting und Erwig (1999), S. 21; Hofmann und Lange (2011), S. 3; Wenck (2006), S. 27; Wilhelm u. a. (2012), S. 14ff]

2.3. Ereignisdiskrete Systeme

In der Automatisierungstechnik versteht man unter einem System meist die Umformung zeitabhängiger Eingangssignale $\mathbf{x}(t) = x_1(t), ..., x_k(t)$ zu zeitabhängigen Ausgangssignalen $\mathbf{y}(t) = y_1(t), ..., y_n(t)$. Zu jedem Zeitpunkt t_0 liefert der Zustand $\mathbf{z}(t) = z_1(t), ..., z_i(t)$ des Systems die benötigten Informationen, um bei bekanntem Eingang $\mathbf{x}(t)$ für $t \geq t_0$ den Ausgang $\mathbf{y}(t)$ eindeutig bestimmen zu können.
Signale in einem System können diskret oder kontinuierlich sein. Abbildung 2.3 auf Seite 20 zeigt einen Trajektorienvergleich zwischen kontinuierlichen und diskreten Systemen. Bei ereignisdiskreten Systemen (engl.: discrete event systems, DES) sind die Eingangssignale diskrete Ereignisse, die nicht gleichzeitig auftreten dürfen. Gegebenenfalls muss dies durch geeignete Maßnahmen, wie zum Beispiel einer Priorisierung, sichergestellt werden. Die Zustände des Systems ändern sich abhängig von den Ereignissen am Eingang zu unvorhersehbaren und zufälligen Zeitpunkten. Der Übergang zwischen zwei Zuständen ist in der Realität meist kontinuierlich (Abbildung 2.3 (b)). In einer vereinfachten Modellvorstellung wird der Übergang auf ein diskretes Ereignis reduziert (Abbildung 2.3 (d)). [León und Kiencke (2013), S. 1, 4, 6ff]

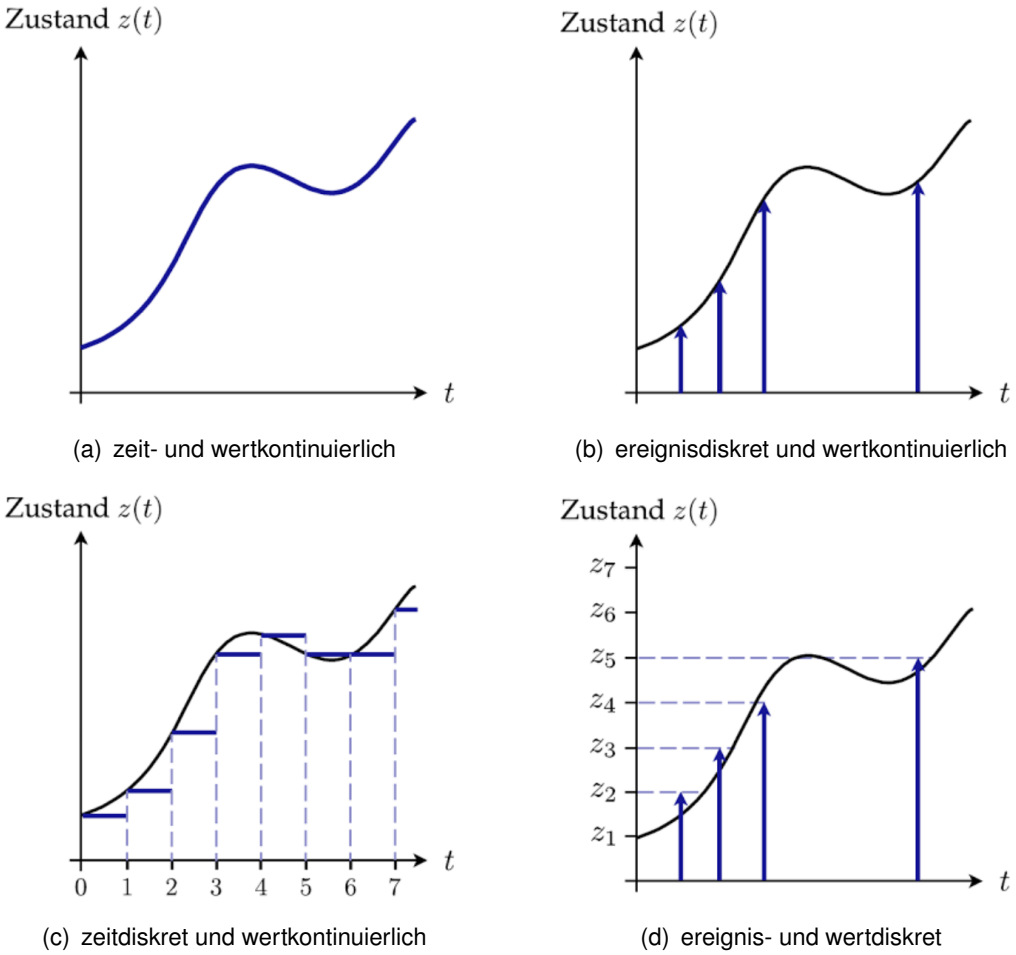
2.4. Graphentheorie

Komplexe Systeme, deren einzelne Elemente in logischer Beziehung zueinander stehen, können mit Hilfe der Graphentheorie beschrieben werden. Die Elemente des Systems werden durch Knoten dargestellt. Eine Verbindung zwischen zwei Knoten stellt die Verbindung der Elemente dar und wird als Kante bezeichnet. Ein Graph besteht aus einer Menge von Knoten und Kanten (Abbildung 2.2 (a)). In einem gerichteten Graphen sind alle Kanten gerichtet (Abbildung 2.2 (b)). Eine gerichtete Kante wird als Pfeil bezeichnet.



(a) Graph (b) Gerichteter Graph

Abbildung 2.2.: Verschiedene Arten von Graphen nach [León und Kiencke (2013), S. 33]



(a) zeit- und wertkontinuierlich

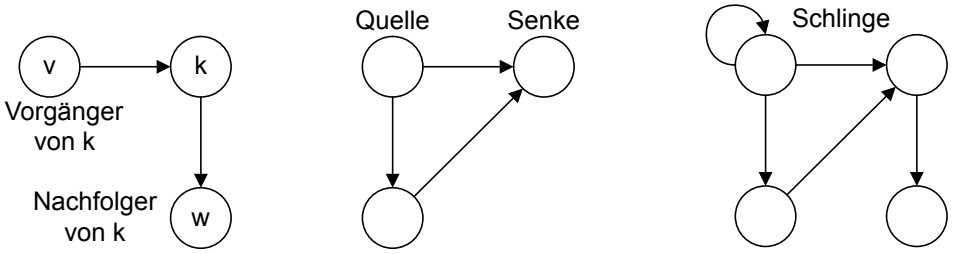
(b) ereignisdiskret und wertkontinuierlich

(c) zeitdiskret und wertkontinuierlich

(d) ereignis- und wertdiskret

Abbildung 2.3.: Trajektorienvergleich zwischen kontinuierlichen und diskreten Systemen nach [León und Kiencke (2013), S. 7]

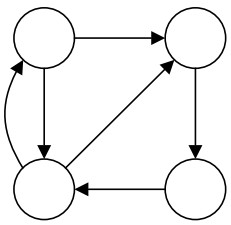
Ist einer Kante ein Kantengewicht, wie zum Beispiel einem Namen oder einem Label zugeordnet, wird die Kante gewichtet genannt. Pfeile, die sowohl denselben Anfangs- als auch denselben Endknoten haben, werden als parallele Pfeile bezeichnet. Ist ein Pfeil in einem Graphen mit mehreren Kantengewichten gewichtet, stellt dies parallele Pfeile mit unterschiedlichen Kantengewichten dar. In einem Graphen mit den Knoten v, k und w, in dem es eine gerichtete Kante von v nach k und von k nach w gibt, heißt v Vorgänger von k und w Nachfolger von k (Abbildung 2.4 (a)). Für einen Knoten kann es mehrere Vorgänger und Nachfolger geben. Die Vereinigungsmenge der Vorgänger und Nachfolger von k wird als Nachbarn bezeichnet. Ein Knoten ohne Vorgänger wird Quelle und ein Knoten ohne Nachfolger Senke genannt (Abbildung 2.4 (b)). Eine Schlinge (engl. selfloop) ist ein Pfeil, dessen Anfangs- und Endknoten identisch sind (Abbildung 2.4 (d)). Ein Graph heißt endlich, wenn sowohl die Menge der Kanten als auch die Menge der Knoten endlich sind.



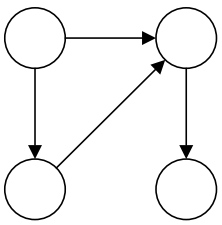
(a) Vorgänger und Nachfolger (b) Quelle und Senke (c) Schlinge

Abbildung 2.4.: Bezeichnungen bei Graphen nach [León und Kiencke (2013), S. 34]

Eine Folge von Kanten, bei denen der Endknoten jeder Kante mit dem Anfangsknoten der nachfolgenden Kante übereinstimmt, wird als Pfad bezeichnet. Die Länge eines Pfades ist die Anzahl der Kanten, aus denen ein Pfad besteht. In einem einfachen Pfad kommt jeder Knoten nur einmal vor. Einfache Pfade werden in der Literatur auch mit dem Begriff Kantenfolge bezeichnet.
Ein Knoten v heißt von einem Knoten w erreichbar, wenn ein Pfad vom Knoten w zum Knoten v existiert. Ein endlicher gerichteter Graph heißt stark zusammenhängend, wenn für je zwei Knoten v und w sowohl v von w als auch w von v erreichbar sind (Abbildung 2.5 (a)). Ist nur v von w oder w von v erreichbar, wird der Graph als schwach zusammenhängend bezeichnet (Abbildung 2.5 (b)). [León und Kiencke (2013), S. 33ff; Lunze (2012b), S. 614f, 618]



(a) Stark zusammenhängender Graph



(b) Schwach zusammenhängender Graph

Abbildung 2.5.: Stark und schwach zusammenhängender Graph nach [León und Kiencke (2013), S. 37]

2.5. Automaten

Eine Modellform für ereignisdiskrete Systeme sind deterministische endliche Automaten (DEA). Sie bestehen aus einer endlichen Menge von Zuständen Q, einer endlichen Menge von Eingabesymbolen Σ, einer Übergangsfunktion δ, einem Startzustand q_0 und einer Menge finaler oder akzeptierter Zustände F. Dabei müssen der Startzustand q_0 ein Element aus Q und die Menge akzeptierter Zustände F eine Teilmenge von Q sein. Mit Hilfe der Übergangsfunktion δ wird der Folgezustand des DEA berechnet. Der Funktion wird ein Zustand aus Q und ein Eingabesymbol aus Σ übergeben. Sie hat als Rückgabewert einen Zustand aus Q. Wenn zum Beispiel q_0 der Startzustand, q_1 ein Zustand aus Q und a ein Eingabesymbol aus Σ ist, geht der Automat nach Auftreten von a vom Zustand q_0 in den Zustand $q_1 = \delta(q_0, a)$ über.

Ein DEA wird häufig in der „Tupel“-Notation angeben, wobei A für den Namen des DEA steht:

$$A = (Q, \Sigma, \delta, q_0, F). \tag{2.10}$$

Deterministische Automaten können von einem Zustand in genau einen Folgezustand übergehen. Im Gegensatz dazu können sich nichtdeterministische Automaten gleichzeitig in mehreren Zuständen befinden. Unter einem endlichen Automaten versteht man, dass der Automat eine endliche Menge von Zuständen besitzt. [Hopcroft u. a. (2011), S. 25, 71f]

In den folgenden Abschnitten werden grundlegende Eigenschaften von Automaten eingeführt.

2.5.1. Automatengraph

Ein Automat kann über einen gerichteten Graphen dargestellt werden. Dieser Graph wird als Automatengraph bezeichnet. Jeder Zustand aus Q wird durch einen Knoten dargestellt. Ein Pfeil zwischen zwei Zuständen q_1 und q_2 aus Q ist immer dann vorhanden, wenn die Zustandsübergangsfunktion δ dem Zustand q_1 den Folgezustand q_2 zuordnet. Der Pfeil wird mit dem entsprechenden Eingabesymbol a aus Σ, welches den Übergang auslöst, beschriftet. Auf den Startzustand q_0 zeigt ein Pfeil, der in keinem Knoten beginnt. Akzeptierte Zustände aus F werden mit einem doppelten Kreis dargestellt. Abbildung 2.6 zeigt einen Automatengraph mit $Q = \{0, 1, 2\}$, $\Sigma = \{a, b, c, d\}$, $q_0 = 0$, $F = \{0\}$, $\delta(0, a) = 1$, $\delta(1, b) = 0$, $\delta(1, c) = 2$ und $\delta(2, d) = 0$. [Hopcroft u. a. (2011), S. 74f; Lunze (2012b), S. 60f]

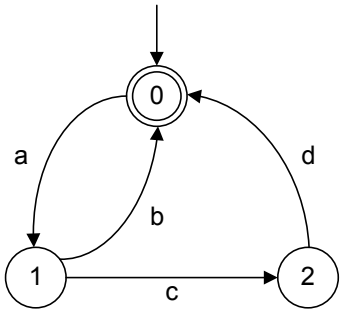
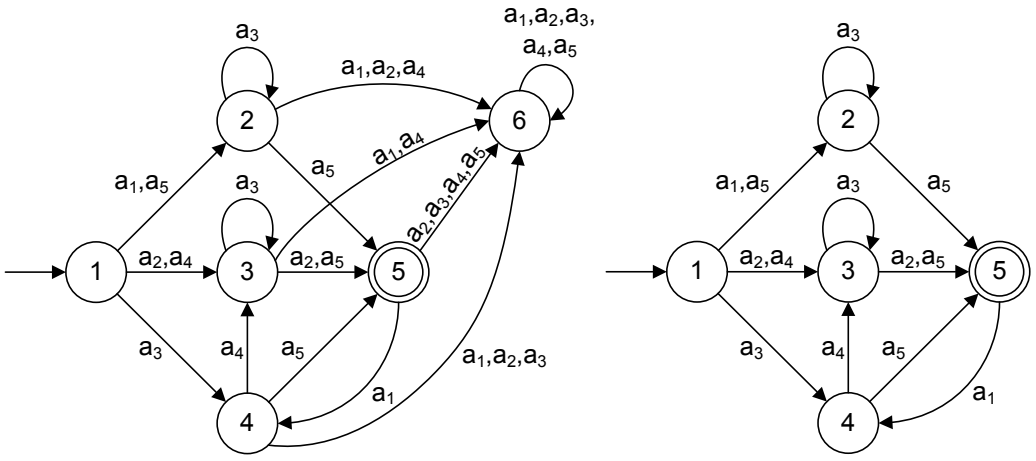


Abbildung 2.6.: Beispiel für einen Automatengraph

2.5.2. Totale und partielle Zustandsübergangsfunktion

Die Zustandsübergangsfunktion eines Automaten kann total oder partiell definiert sein. Bei totaler Definition ist die Zustandsübergangsfunktion in jedem Zustand für jedes Eingabesymbol definiert (siehe Abbildung 2.7 (a)). Bei partieller Definition kann es Zustände geben, in denen nicht alle Eingabesymbole auftreten können (siehe Abbildung 2.7 (b)). Die Bezeichnung $\delta(q, a)!$ bei einem Automaten mit partiell definierter Zustandsübergangsfunktion bedeutet, dass im Zustand q beim Auftreten des Ereignisses a eine abgehende Kante existiert. Die Nichtexistenz dieser Kante wird mit $\neg\delta(q, a)!$ ausgedrückt. Automaten mit partiell definierter Zustandsübergangsfunktion lassen sich jedoch in die totale Form umwandeln. Die Vorgehensweise wird in [Lunze (2012b), S. 75] beschrieben. [Lunze (2012b), S. 75]



(a) Total definiert (b) Partiell definiert

Abbildung 2.7.: Automat mit total und partiell definierter Zustandsübergangsfunktion nach [Lunze (2012b), S. 71, 75]

2.5.3. Sprache des Automaten

Ein DEA kann eine Folge von Eingabesymbolen $w = a_1 a_2 \ldots a_n$ akzeptieren oder zurückweisen. Befindet sich der DEA in seinem Startzustand q_0, können die Folgezustände $q_1, q_2, \ldots, q_n$, in die der Automat nach Auftreten der einzelnen Eingabesymbole wechselt, über die Übergangsfunktion berechnet werden. Für alle Folgezustände q_i mit $i = 1, 2, \ldots, n$ gilt $q_i = \delta(q_{i-1}, a_i)$. Wenn q_n ein Element von F ist, wird die Eingabefolge w akzeptiert, in jedem anderen Fall wird sie zurückgewiesen.

Um die Sprache eines DEA zu definieren, wird die erweiterte Übergangsfunktion $\hat{\delta}$ verwendet. Sie gibt an, in welchem Zustand sich der Automat ausgehend von einem beliebigen Startzustand nach Abarbeitung einer beliebigen Folge von Eingabesymbolen befindet. Der erweiterten Übergangsfunktion werden ein beliebiger Startzustand q sowie die Folge von Eingabesymbolen w übergeben und sie gibt den Zustand, in dem sich der Automat nach Abarbeitung der Eingabefolge befindet, zurück. Die erweiterte Übergangsfunktion $\hat{\delta}$ wird induktiv über die Länge der Folge von Eingabesymbolen definiert:

$$\hat{\delta}(q, \varepsilon) = q, \tag{2.11}$$

$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a) = \delta(p, a). \tag{2.12}$$

Gleichung (2.11) bedeutet, dass der Automat bei keiner Eingabe im Zustand q verbleibt. Gleichung (2.12) zerlegt die Eingabefolge w in eine Form $x\,a$. Dabei ist a das letzte Eingabesymbol von w und x das Präfix der Eingabefolge w. Beispielweise kann $w = 1101$ zerlegt

werden in $x = 110$ und $a = 1$. Zur Berechnung von $\hat{\delta}(q, w)$ wird zunächst der Zustand p, in dem sich der Automat nach Abarbeitung der Eingabefolge x befindet, berechnet. Anschließend wird über die Übergangsfunktion δ der Zustand berechnet, in dem sich der Automat ausgehen vom Zustand p nach Auftreten von a befindet.

Die Sprache eines DEA $A = \{Q, \Sigma, \delta, q_0, F\}$ wird als $L(A)$ bezeichnet und ist definiert durch:

$$L(A) = \{w \quad | \hat{\delta}(q_0, w) \in F\}. \tag{2.13}$$

Das bedeutet, die Sprache $L(A)$ ist die Menge aller Worte w aus Σ^*, die den Automaten vom Startzustand q_0 in einen akzeptierten Zustand F führt. Die Sprache eines DEA ist immer eine reguläre Sprache. In der Literatur wird die erweiterte Übergangsfunktion $\hat{\delta}$ auch als δ bezeichnet. Daher wird im Folgenden sowohl die Übergangsfunktion als auch die erweiterte Übergangsfunktion mit δ bezeichnet. [Hopcroft u. a. (2011), S. 72, 75f, 79]

2.5.4. Automaten als Akzeptoren und Generatoren

Automaten können als Akzeptoren (Erkenner) und Generatoren der Sprache L eingesetzt werden. Ein Akzeptor ist ein passives System, das nach Abschnitt 2.5.3 entscheidet, ob eine Folge von Eingabesymbolen w zu einer vorgegebenen Sprache L gehört. Ein Akzeptor hat immer eine total definierte Zustandsübergangsfunktion.

Ein Generator der Sprache L ist ein aktives System, das die zu einer Sprache L gehörende Folge von Eingabesymbolen $w = a_1 a_2 a_3 \ldots a_n$ erzeugt. Die Folge von Eingabesymbolen wird spontan und ohne erkennbaren Auslöser generiert. Eine Generierung führt nicht zwangsläufig zu einem Zustandsübergang sondern kann auch eine Schlinge sein. Ein Generator kann sowohl eine partiell als auch eine total definierte Zustandsübergangsfunktion haben. [Lunze (2012b), S. 84f; Wenck (2006), S. 28]

Die Sprache eines Generators wird unterschieden in generierte und markierende Sprache. Die generierte Sprache $L(G)$ eines Generators $G = (Q, \Sigma, \delta, q_0, F)$ ist definiert durch:

$$L(G) = \{w \in \Sigma^* \quad | \delta(q_0, w)!\}. \tag{2.14}$$

Gleichung (2.14) besagt, dass es alle Worte w entlang eines gerichteten Pfades im Automatengraf enthält, die vom Anfangszustand q_0 aus generiert werden können. Die Worte w entstehen aus der Konkatenation der Symbole entlang dieses Pfades. $L(G)$ ist durch die Definition präfix-abgeschlossen, da der Pfad nur existiert, wenn auch alle Präfixe existieren. Die

markierende Sprache $L_m(G)$ ist eine Teilmenge der generierten Sprache $L_m(G) \subseteq L(G)$ und ist definiert durch:

$$L_m(G) = \{w \in L(G) \quad | \delta(q_0, w) \in F\}. \tag{2.15}$$

Es fasst alle Worte w aus $L(G)$ zusammen, die zu einem markierten Zustand führen. [Cassandras und Lafortune (2008), S. 62f]

2.5.5. Erreichbarkeit und Blockierung

Ein endlicher Automat kann Zustände enthalten, die mit keiner Folge von Eingabesymbolen zu erreichen sind. Diese Zustände tragen nicht zur Sprache des Automaten bei. Ebenso kann es Zustände geben, von denen aus kein akzeptierter Zustand erreicht werden kann. Ein Zustand eines Automaten heißt erreichbar, wenn es eine Folge von Eingabesymbolen w gibt, mit der ein Zustand q vom Startzustand q_0 aus erreicht werden kann. Gibt es eine Folge von Eingabesymbolen v, mit dem vom Zustand p ein akzeptierter Zustand erreicht werden kann, wird der Zustand p co-erreichbar genannt. Ein Zustand, der sowohl erreichbar als auch co-erreichbar ist, wird trim genannt. Abbildung 2.8 zeigt einen Automatengraphen, in dem die Zustände eins bis sechs erreichbar, eins bis vier sowie sieben und acht co-erreichbar und die Zustände eins bis vier trim sind. [Erk und Priese (2008), S. 71]

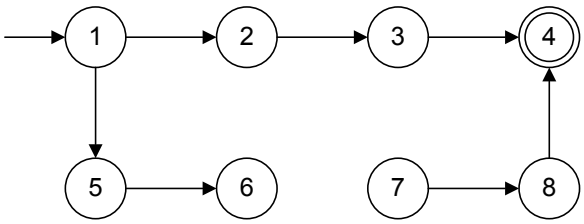


Abbildung 2.8.: Automatengraph mit erreichbaren und co-erreichbaren Zuständen nach [Erk und Priese (2008), S. 71]

Eine weitere Analysemöglichkeit von Automatengraphen ist die Untersuchung, ob der Graph blockierend ist. Bei einer Blockierung kann aus dem aktuellen Zustand kein akzeptierter Zustand mehr erreicht werden. Enthält der Graph einen Deadlock oder Livelock, kann es zu einer Blockade führen.
Ein Deadlock kann auftreten, wenn es einen erreichbaren Zustand p aus Q gibt, der kein akzeptierter Zustand aus F ist und von dem es keine abgehenden Kanten gibt. In dem Automatengraphen aus Abbildung 2.9 ist der Zustand 7 ein Deadlock. Der Zustand 13 ist kein Deadlock, da er vom Startzustand aus nicht erreichbar ist.

Ein Livelock kann auftreten, wenn es eine Menge von Zuständen $\widetilde{Q}$ aus Q gibt, von denen keiner ein akzeptierter Zustand aus F ist, ein Zustand q aus $\widetilde{Q}$ vom Startzustand q_0 durch eine Folge von Eingabesymbolen erreicht werden kann und alle abgehenden Kanten der Zustände aus $\widetilde{Q}$ zu einem Zustand der Menge $\widetilde{Q}$ führen. Die Zustandsmenge $\widetilde{Q} = \{3, 4, 5, 6\}$ aus Abbildung 2.9 ist ein Livelock. Die Zustandsmenge $\widetilde{P} = \{8, 9, 10, 11\}$ ist kein Livelock, da der Zustand 11 ein akzeptierter Zustand ist.

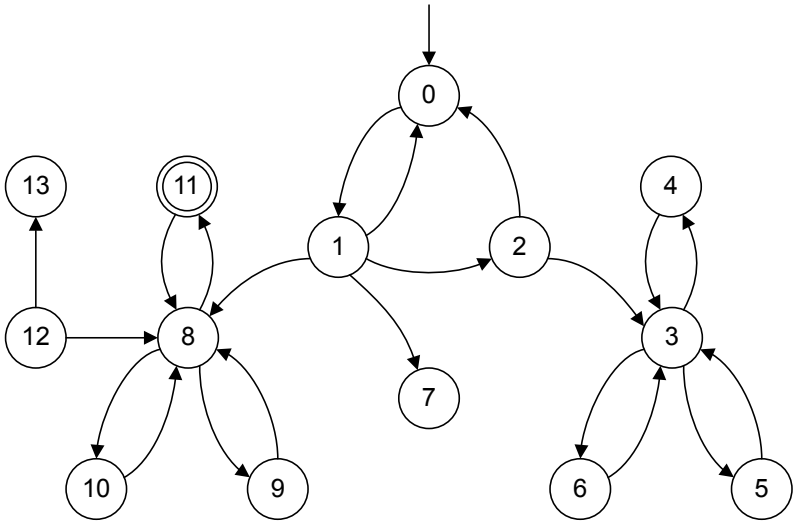


Abbildung 2.9.: Automatengraph mit blockierenden Zuständen nach [Wenck (2006), S. 49]

Das Blockieren und Nichtblockieren kann für Generatoren formal angegeben werden. Ein Generator heißt blockierend, wenn

$$\overline{L_m(G)} \subset L(G), \tag{2.16}$$

und nichtblockierend, wenn

$$\overline{L_m(G)} = L(G). \tag{2.17}$$

Die Definition (2.17) schließt die Existenz von Dead- und Livelocks aus, da jeder Präfix von $L_m(G)$ in $L(G)$ enthalten sein muss und keine anderen Ereignisfolgen zu $L(G)$ gehören. Die Definition zum Nichtblockieren ist später beim Steuerungsentwurf relevant. [Wenck (2006), S. 32f]

2.6. Modellierung ereignisdiskreter Systeme

Ein ereignisdiskretes System wird als Generator modelliert. Anstelle der bisher verwendeten Bezeichnungen werden systemtheoretische Begriffe verwendet. Dabei gelten nach [Lunze (2012b), S. 85] die folgenden Analogien:

Symbol – Ereignis,
Alphabet – Ereignismenge,
Folge, Wort – Ereignisfolge,
Sprache – Verhalten.

Anhand dieser Begriffe werden in Abschnitt 2.6.1 und 2.6.2 weitere Eigenschaften der Ereignisse erläutert. Anschließend werden in Abschnitt 2.6.3 zwei Kompositionsoperatoren für Generatoren eingeführt.

2.6.1. Steuerbare und beobachtbare Ereignisse

Gibt es an einem Knoten im Automatengraphen mehrere ausgehende Kanten, ist es für die Steuerung des Systems wichtig, dass durch bestimmte Eingriffe Einfluss darauf genommen werden kann, welches Ereignis als nächstes auftritt. Ein Ereignis e wird steuerbar im Zustand z genannt, wenn es möglich ist, durch eine gewählte Eingabe zu erreichen, dass der Automat, wenn er sich im Zustand z befindet, entweder als nächstes das Ereignis e ausführt und sich anschließend im Zustand $z' = \delta(z, e)$ befindet oder das Ereignis e nicht ausführt. Ein steuerbares Ereignis ist im Automatengraph mit einem kurzen orthogonalen Strich an der Kante gekennzeichnet.
Die Menge aller möglichen Ereignisse Σ wird in zwei Teilalphabete Σ_c für die steuerbaren (engl.: controllable) und Σ_{uc} für die nicht steuerbaren Ereignisse (engl.: uncontrollable) unterteilt. Für die Mengen Σ_c und Σ_{uc} gilt:

$$\Sigma = \Sigma_c \cup \Sigma_{uc} \quad \text{mit } \Sigma_c \cap \Sigma_{uc} = \varnothing. \tag{2.18}$$

Bei Generatoren kann es vorkommen, dass einige Zustandsübergänge des Systems von außen nicht sichtbar sind. Ein Ereignis, das so einen Zustandsübergang auslöst, wird nicht beobachtbar genannt. Die Menge aller möglichen Ereignisse Σ wird in zwei Teilalphabete Σ_o für die beobachtbaren (engl.: observable) und Σ_{uo} für die nicht beobachtbaren Ereignisse

(engl.: unobservable) unterteilt. Für die Mengen Σ_o und Σ_{uo} gilt:

$$\Sigma = \Sigma_o \cup \Sigma_{uo} \quad \text{mit } \Sigma_o \cap \Sigma_{uo} = \varnothing. \tag{2.19}$$

Nicht beobachtbare Ereignisse $\sigma \in \Sigma_{uo}$ können durch die natürliche Projektion aus einer Ereignisfolge w entfernt werden. Die natürliche Projektion ist eine Abbildung $P_{\Sigma_o} : \Sigma^* \to \Sigma_o^*$ mit:

$$P_{\Sigma_o}(\varepsilon) = \varepsilon, \tag{2.20}$$

$$P_{\Sigma_o}(\sigma) = \begin{cases} \sigma & \text{falls } \sigma \in \Sigma_o, \\ \varepsilon & \text{sonst,} \end{cases} \tag{2.21}$$

$$P_{\Sigma_o}(w\sigma) = P_{\Sigma_o}(w)P_{\Sigma_o}(\sigma) \quad \text{für } w \in \Sigma^*, \sigma \in \Sigma. \tag{2.22}$$

Über die inverse natürliche Projektion können nicht beobachtbare Ereignisse $\sigma \in \Sigma_{uo}$ der Ereignisfolge w wieder hinzugefügt werden. Die inverse natürliche Projektion ist eine Abbildung $P^{-1}_{\Sigma_o} : \Sigma_o^* \to 2^{\Sigma^*}$ mit:

$$P^{-1}_{\Sigma_o}(v) = \{w \in \Sigma^* \mid P_{\Sigma_o}(w) = v\}. \tag{2.23}$$

In Bezug auf Generatoren fügt die inverse natürliche Projektion an jeden Zustand Schlingen mit nicht beobachtbaren Ereignissen hinzu. [Lunze (2012a), S. 349f; Lunze (2012b), S. 228; Wenck (2006), S. 33f, 49]

2.6.2. Gemeinsame und private Ereignisse

Für zwei Generatoren $G_1 = (Q_1, \Sigma_1, \delta_1, q_{10}, F_1)$ und $G_2 = (Q_2, \Sigma_2, \delta_2, q_{20}, F_2)$ gelten für die Menge aller möglichen Ereignisse Σ, die Menge der gemeinsamen Ereignisse (engl.: common events) Σ_{ce} und die Menge der privaten Ereignisse (engl.: private events) Σ_{pe} folgende Definitionen:

$$\Sigma = \Sigma_1 \cup \Sigma_2, \tag{2.24}$$

$$\Sigma_{ce} = \Sigma_1 \cap \Sigma_2, \tag{2.25}$$

$$\Sigma_{pe} = \Sigma \backslash \Sigma_{ce} = (\Sigma_2 \backslash \Sigma_1) \cup (\Sigma_1 \backslash \Sigma_2). \tag{2.26}$$

Die Menge der gemeinsamen Ereignisse Σ_{ce} besteht aus allen Ereignissen, die sowohl in Σ_1 als auch in Σ_2 auftreten können. Dagegen treten die privaten Ereignisse Σ_{pe} ausschließlich in Σ_1 oder Σ_2 auf. [Wenck (2006), S. 39]

2.6.3. Zusammenschaltung zweier Generatoren

Zwei Generatoren $G_1 = (Q_1, \Sigma_1, \delta_1, q_{10}, F_1)$ und $G_2 = (Q_2, \Sigma_2, \delta_2, q_{20}, F_2)$ können mittels Komposition gekoppelt werden. Kompositionsoperatoren werden immer nur für zwei Generatoren definiert. Sollen drei oder mehr Generatoren gekoppelt werden, können die Operationen nacheinander ausgeführt werden.

Zwei Kompositionsoperatoren sind die strenge Synchronisation (SPC) und das synchrone Produkt (SYPC). Die strenge Synchronisation $G_{SPC} = G_1||_{SPC}G_2$ beschreibt den totalen Gleichschritt zweier gekoppelter Generatoren. Es werden alle möglichen gemeinsamen Ereignisse synchronisiert, private Ereignisse können nicht stattfinden. Für die Zustandsübergangsfunktion von $G_{SPC} = (Q, \Sigma, \delta, q_0, F) = G_1||_{SPC}G_2$ gilt mit $a \in \Sigma$ und $q \in Q$:

$$\delta(q, a) = \begin{cases} \delta_1(q_1, a) \times \delta_2(q_2, a) & \text{falls } \delta_1(q_1, a)! \text{ und } \delta_2(q_2, a)!, \\ \text{nicht definiert} & \text{sonst.} \end{cases} \tag{2.27}$$

Befindet sich ein komponiertes System G_{SPC} im Zustand q, führt ein Ereignis a genau dann zu einem Folgezustand $\delta(q, a)$, wenn beide Generatoren G_1 und G_2 das Ereignis in ihrem aktuellen Zustand generieren können (Bedingung 1 aus Gleichung (2.27)). In allen anderen Fällen ist $\delta(q, a)$ nicht definiert (Bedingung 2 aus Gleichung (2.27)).

Das synchrone Produkt $G_{SYPC} = G_1||_{SYPC}G_2$ synchronisiert wie die strenge Synchronisation die gemeinsamen Ereignisse, lässt aber zusätzlich auch private Ereignisse zu. Für die Zustandsübergangsfunktion von $G_{SYPC} = (Q, \Sigma, \delta, q_0, F) = G_1||_{SYPC}G_2$ gilt mit $a \in \Sigma$ und $q \in Q$:

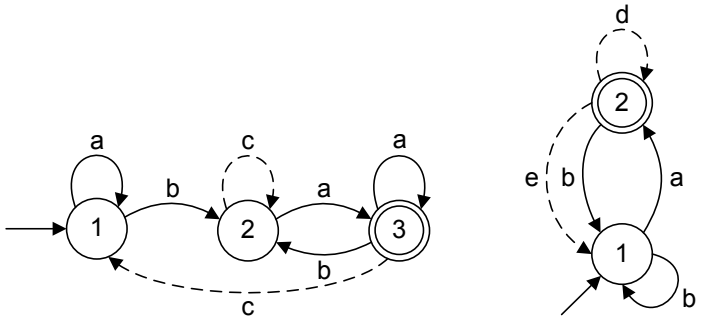
$$\delta(q, a) = \begin{cases} \delta_1(q_1, a) \times \delta_2(q_2, a) & \text{falls } \delta_1(q_1, a)! \text{ und } \delta_2(q_2, a)!, \\ \delta_1(q_1, a) \times \{q_2\} & \text{falls } \delta_1(q_1, a)! \text{ und } \neg\delta_2(q_2, a)! \text{ und } a \notin (\Sigma_1 \cap \Sigma_2), \\ \{q_1\} \times \delta_2(q_2, a) & \text{falls } \neg\delta_1(q_1, a)! \text{ und } \delta_2(q_2, a)! \text{ und } a \notin (\Sigma_1 \cap \Sigma_2), \\ \text{nicht definiert} & \text{sonst.} \end{cases} \tag{2.28}$$

Bedingung 1 aus Gleichung (2.28) stimmt mit Bedingung 1 aus Gleichung (2.27) überein. Zusätzlich zum SPC Operator führt das Ereignis a im Zustand q zu einem Folgezustand, wenn lediglich einer der Generator das Ereignis im aktuellen Zustand generieren kann und das Ereignis kein gemeinsames Ereignis ist (Bedingung 2 und 3 aus Gleichung (2.28)). In allen anderen Fällen ist $\delta(q, a)$ nicht definiert (Bedingung 4 aus Gleichung (2.28)).

Bei der strengen Synchronisation und dem synchronen Produkt gibt es zwei Sonderfälle, die als „Meet“ und „Shuffle“ bezeichnet werden. Das synchrone Produkt wird auch „Meet“ genannt, wenn beide Generatoren die gleichen Ereignisalphabete haben ($\Sigma_1 = \Sigma_2$). In

diesem Fall gilt $G_1||_{SPC}G_2 = G_1||_{SYPC}G_2$. Das synchrone Produkt wird „Shuffle“ genannt, wenn die gemeinsamen Ereignisse der leeren Menge entsprechen ($\Sigma_{ce} = \Sigma_1 \cap \Sigma_2 = \varnothing$). In diesem Fall bewegen sich G_1 und G_2 völlig unabhängig voneinander. [Wenck (2006), S. 39, 41 S. 101ff]

Abbildung 2.10 zeigt zwei Generatoren G_1 und G_2. Die gemeinsamen Ereignisse der Generatoren sind $\Sigma_{ce} = \{a, b\}$ und die privaten Ereignisse $\Sigma_{pe} = \{c, d, e\}$. Private Ereignisse sind zur Unterscheidung von gemeinsamen Ereignissen in den Automatengraphen mit einer gestrichelten Kante dargestellt.

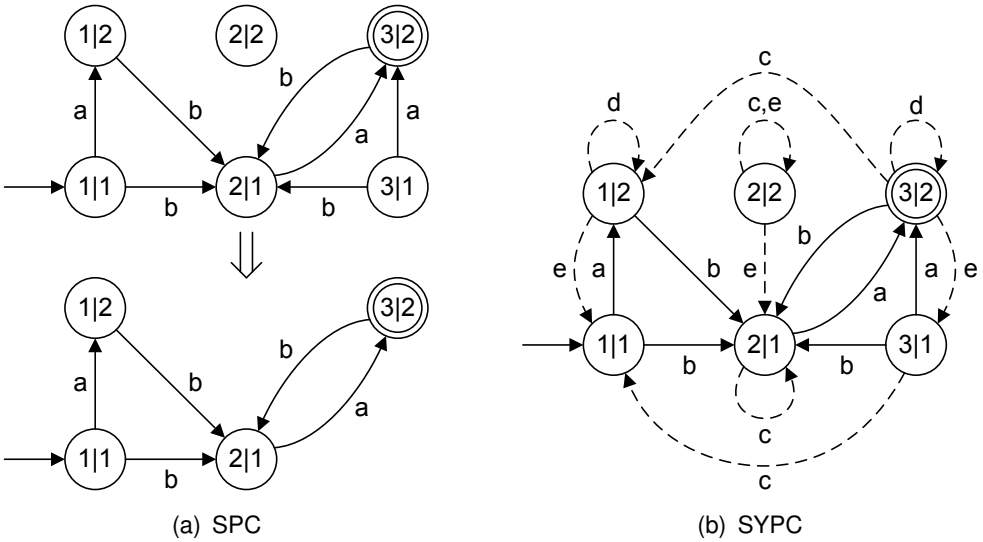


(a) G1 (b) G2

Abbildung 2.10.: Zwei Generatoren nach [Lunze (2012b), S. 218]

Bei der Kopplung der beiden Generatoren mit dem Kompositionsoperator SPC (Abbildung 2.11 (a)) werden zunächst alle möglichen Zustände als Knoten dargestellt. Die erste Zahl[1] in einem Knoten gibt den aktuellen Zustand von G_1 an, die zweite den Zustand von G_2. Anschließend werden die Zustandsübergänge gemeinsamer Ereignisse Σ_{ce} eingetragen. In diesem Beispiel bedeutet eine waagerechte Bewegung des komponierten Systems ein Schalten von G_1, eine senkrechte Bewegung ein Schalten von G_2 und eine diagonale Bewegung ein Schalten beider Generatoren. Das System $G_{SPC} = G_1||_{SPC}G_2$ kann anschließend vereinfacht werden, indem alle nicht erreichbaren Knoten und Kanten aus dem System entfernt werden. Bei der Kopplung der beiden Generatoren mit dem Kompositionsoperator SYPC (Abbildung 2.11 (b)) können alle Knoten und Kanten, die vor der Vereinfachung bei der Kopplung mit dem Kompositionsoperator SPC entstanden sind, übernommen werden. Zusätzlich werden die Kanten für private Ereignisse Σ_{pe} eingetragen. Der Generator, bei dem das private Ereignis nicht auftritt, bleibt in seinem letzten aktuellen Zustand. [Lunze (2012b), S. 217f, 227]

[1] Statt Zahlen können auch Buchstaben oder Symbole zur Markierung der Zustände verwendet werden.



(a) SPC (b) SYPC

Abbildung 2.11.: Komposition der Generatoren G_1 und G_2 mit SPC und SYPC nach [Lunze (2012b), S. 218, 227]

2.7. Steuerungsentwurf

Zum Steuerungsentwurf wird in Abschnitt 2.7.1 zunächst der Begriff der Steuerung definiert. In Abschnitt 2.7.2 werden Grundlagen zum modellbasierten Steuerungsentwurf eingeführt. Abschließend erfolgt die Definition einer minimal restriktiven Steuerung in Abschnitt 2.7.3.

2.7.1. Begriff der Steuerung

Nach DIN IEC 60050-351 ist der Begriff „Steuerung“ mit dem Begriff „Steuern“ gleichzusetzen und wird beschrieben als

> *„Vorgang in einem System, bei dem eine oder mehrere variable Größen als Eingangsgrößen andere variable Größen als Ausgangsgrößen aufgrund der dem System eigenen Gesetzmäßigkeiten beeinflussen.“*
> (Quelle: [DIN IEC 60050-351 (2014)], 351-47-02)

Es wird unter dem Begriff also die Gesamtheit aus steuerndem System und Steuerstrecke verstanden, die auch Steuerkreis genannt wird. Da in dieser Thesis die Aufteilung des Steuerkreises von Bedeutung ist, wird von der DIN Norm abgewichen und als Steuerung das steuernde System bezeichnet.

Abbildung 2.12 zeigt die Struktur eines industriellen Steuerkreises. Auf die Steuerung wirken Eingangssignale, wie zum Beispiel von überlagerten Steuerungen oder durch Bedienoberflächen vom Menschen, die Sensorik, die den Prozesszustand der Steuerstrecke erfasst, und der Zustand der Aktorik. Die daraus gewonnenen Informationen werden an die Aktorik weitergegeben, die auf die Steuerstrecke einwirkt. Störungen wirken auf die Steuerung, auf die Aktorik und Sensorik sowie auf die Steuerstrecke. [Litz und Frey (1999), S. 145f]

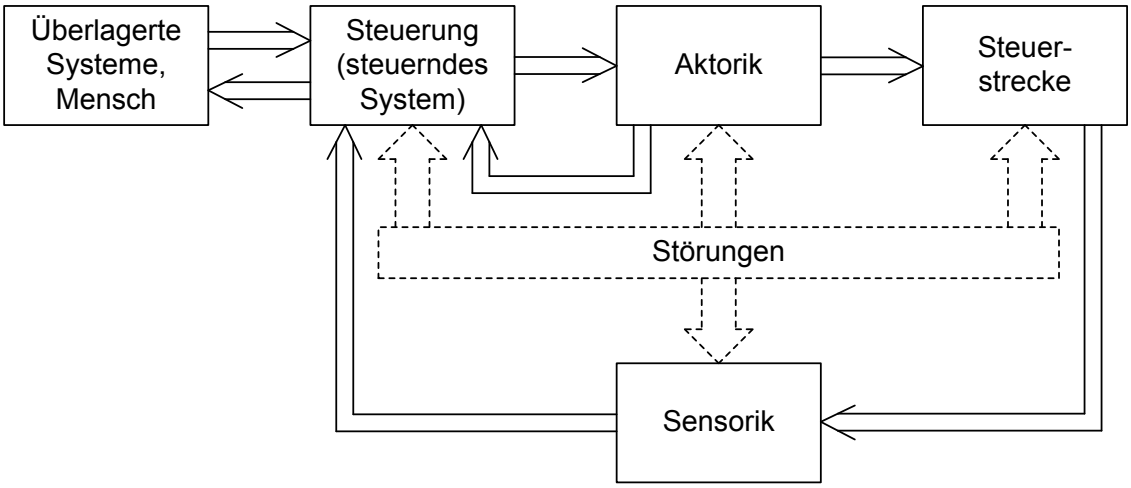


Abbildung 2.12.: Struktur eines industriellen Steuerkreises nach [Litz und Frey (1999), S. 145]

2.7.2. Modellbasierter Steuerungsentwurf

Als Basis für einen Steuerungsentwurf gibt es fast immer eine informelle Spezifikation der Steuerungsaufgabe. Dazu können unter anderem eine Beschreibung des ungesteuerten Prozesses, Anforderungen an den gesteuerten Prozess oder Anforderungen an den Steuerungsalgorithmus gehören. Zum Steuerungsentwurf wird im industriellen Alltag meist eine direkte Umsetzung dieser informellen Spezifikation realisiert. Eine Validierung der Steuerung erfolgt erst während der Inbetriebnahme.
Eine direkte Umsetzung der informellen Spezifikation kann zur Folge haben, dass die Inbetriebnahme zeitaufwändig ist. Außerdem kann es vorkommen, dass nur ein Teil der Fehler gefunden und korrigiert wird. Während des späteren Betriebes können dann Fehler auftreten, die in der Inbetriebnahme nicht detektiert wurden.

Aus diesem Grund werden im universitären Bereich verstärkt modellbasierte Verfahren zum Steuerungsentwurf entwickelt. Die Basis ist hierbei die Trennung der Streckenmodellierung

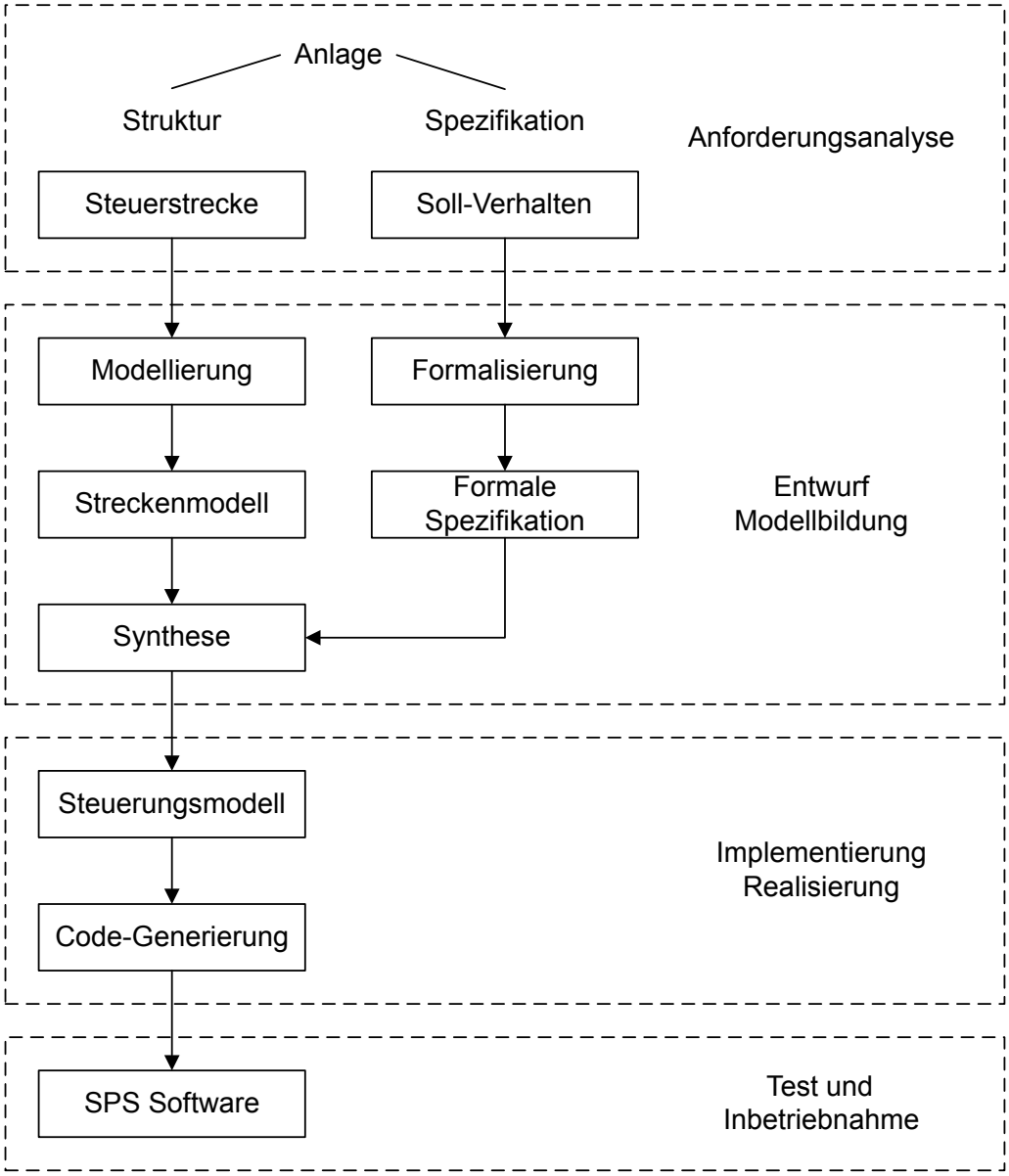


Abbildung 2.13.: Vorgehensweise beim modellbasierten Steuerungsentwurf nach [Uhlig (2006), S. 11]

und des Soll-Verhaltens. Abbildung 2.13 auf Seite 34 zeigt eine Vorgehensweise beim modellbasierten Steuerungsentwurf durch Synthese. In der Anforderungsanalyse wird das System auf mögliche Blockierungen untersucht. In der nächsten Phase geht es um die Modellbildung. Dabei ist darauf zu achten, dass das ungesteuerte Streckenmodell vollständig beschrieben wird. Parallel wird aus dem Soll-Verhalten eine formale Spezifikation der Steuerung entwickelt. Unter die formale Spezifikation fallen unter anderem die Definition von verbotenen Zuständen oder Zustandsfolgen des ungesteuerten Systems, die Schäden an Personen, Anlagenteilen oder der Umwelt verursachen können, sowie die Definition des gewünschten Verhaltens des gesteuerten Systems. Mit dem Streckenmodell und der formalen Spezifikation wird in der Synthese das Steuerungsmodell berechnet. In der dritten Phase wird dieses über eine Code-Generierung implementiert und realisiert. Die Code-Generierung kann dabei automatisch erfolgen. In der letzten Phase wird die Steuerung zum Beispiel auf einer speicherprogrammierbaren Steuerung (SPS) getestet und in Betrieb genommen. [Abel und Bollig (2006), S. 243f; Litz (2013), S. 195; Uhlig (2006), S. 10f]

2.7.3. Minimal restriktive Steuerung

Beim Entwurf von Steuerungen wird die Steuerstrecke als Automat A und das Soll-Verhalten durch einen Spezifikationsautomaten K dargestellt. Der Automat des Steuerkreises, der aus Steuerung und Steuerstrecke entsteht, wird über die strenge Synchronisation (SPC) von A und K gebildet. Existieren im Prozess nicht steuerbare Ereignisse Σ_{uc}, die von der Steuerung nicht verhindert werden können, kann es vorkommen, dass das gesteuerte System über diese nicht steuerbaren Ereignisse in verbotene Zustände gelangt. Aus diesem Grund muss die Steuerung auch Ereignisse verhindern, die nicht direkt, aber über weitere nicht steuerbare Ereignisse zu verbotenen Zuständen oder Ereignisfolgen führen. Eine minimale restriktive Steuerung erfüllt zum einen die durch K vorgegebene Spezifikation und schränkt zum anderen das Verhalten des gesteuerten Systems so wenig wie möglich ein. [Lunze (2012b), S. 220, 253]

2.8. Supervisory Control Theory

Die Supervisory Control Theory (SCT) ist eine Ansammlung von Methoden zur formalen Synthese von Steuerungen, die erstmalig von P. J. Ramadge in [Ramadge (1983)] im Jahr 1983 formalisiert wurde. Sie wurde in zahlreichen anderen Publikationen, zum Teil in Zusammenarbeit mit W. M. Wonham, weiter entwickelt. Bei der SCT werden das Streckenmodell M und die formale Spezifikation K des Soll-Verhaltens als Automat oder Petrinetz modelliert.

Zusätzliche Eingänge ermöglichen der Steuerung, Zustandsübergänge im Prozess zu erlauben oder zu verbieten. Da die Steuerung den Prozess überwacht, wird sie auch Supervisor genannt. Der von der SCT entworfene Supervisor ist minimal restriktiv.

Mit der SCT kann der Supervisor entweder durch „Steuerungsentwurf mittels Ereignisrückführung“ oder durch „Steuerungsentwurf mittels Zustandsrückführung“ entworfen werden. In dieser Thesis wird ausschließlich die Variante mittels Ereignisrückführung behandelt. Dazu wird vorausgesetzt, dass alle Ereignisse des Streckenmodells M, das durch einen Generator $G = (Q, \Sigma, \delta, q_0, F)$ dargestellt wird, beobachtbar sind. Die Ereignismenge Σ kann entsprechend Gleichung (2.18) in steuerbare Σ_c und nicht steuerbare Ereignisse Σ_{uc} aufgeteilt werden. Für den Entwurf des Supervisors können verschiedene Ansätze verwendet werden, die in den nachfolgenden Abschnitten beschrieben sind. [Wenck (2006), S. 48ff]

2.8.1. Monolithischer Steuerungsentwurf

Abbildung 2.14 zeigt den Steuerkreis S/G beim monolithischem Ansatz. Der Supervisor S befindet sich hier im Rückführzweig des Steuerkreises. Die Ereignisfolge s, die die Steuerstrecke G generiert, wird von dem Supervisor S überwacht. Der Supervisor aktiviert und deaktiviert daraufhin über die Steuereingriffe $S(s)$ die erlaubten Ereignisse für den nächsten Schritt. Nicht steuerbare Ereignisse werden dabei immer zugelassen. [Abel und Bollig (2006), S. 249; Wenck (2006), S. 45, 48ff]

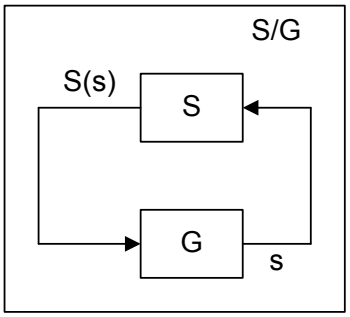


Abbildung 2.14.: Steuerkreis mit monolithischem Ansatz nach [Wenck (2006), S. 51]

Das Verhalten $L(S/G)$ des Steuerkreises S/G ist rekursiv definiert mit $s \in \Sigma^*$ und $\sigma \in \Sigma$:

1. $\varepsilon \in L(S/G)$,
2. $(s \in L(S/G) \cap \sigma \in S(s) \cap s\sigma \in L(G)) \Rightarrow s\sigma \in L(S/G)$,
3. keine anderen Ereignisfolgen gehören zu $L(S/G)$.

Bedingung 1 besagt, dass das Verhalten $L(S/G)$ nichtleer ist, da es immer das leere Wort ε enthält. Bedingung 2 besagt, dass Ereignisfolgen s des Steuerkreises S/G nur dann durch Folgeereignisse σ erweitert werden dürfen, wenn der Supervisor S die Ereignisse erlaubt und sie in der Steuerstrecke G möglich sind.

Das markierende Verhalten des Steuerkreises S/G ist definiert als

$$L_m(S/G) = L(S/G) \cap L_m(G). \tag{2.29}$$

Damit die angestrebte Steuerungsaufgabe erfüllt werden kann, ist es notwendig, dass der Supervisor S bezüglich der Steuerstrecke G nichtblockierend ist. Dies ist der Fall, wenn auch der Steuerkreis S/G nichtblockierend ist:

$$\overline{L_m(S/G)} = L(S/G). \tag{2.30}$$

Damit ein Supervisor S für eine Steuerstrecke G und eine Spezifikation $K \in \Sigma^*$ existiert, muss K bezüglich G steuerbar sein. In diesem Fall gilt:

$$\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}. \tag{2.31}$$

Ist die Spezifikation K nicht steuerbar bezüglich der Steuerstrecke G, wird eine supremale steuerbare Teilsprache $K^{\uparrow C}$ gesucht, die diese Eigenschaft erfüllt und möglichst wenig von der ursprünglichen Spezifikation K abweicht. Dazu wird die Klasse der bezüglich G steuerbaren Teilsprachen mit $K \subseteq L(G)$ gebildet:

$$C_{in}(K, L(G)) = \{E \subseteq K \quad | \, \overline{E}\Sigma_{uc} \cap L(G) \subseteq \overline{E}\}. \tag{2.32}$$

Durch Vereinigung aller Elemente von $C_{in}(K, L(G))$ entsteht die supremale steuerbare Teilsprache $K^{\uparrow C}$.

Damit ein nichtblockierender Supervisor S für eine Steuerstrecke G und eine Spezifikation K, die nach Gleichung (2.31) steuerbar bezüglich G ist, existiert, muss für die Spezifikation K gelten:

$$K = \overline{K} \cap L_m(G). \tag{2.33}$$

Diese Existenzbedingung wird auch $L_m(G)$-Abgeschlossenheit genannt.

Für die Supervisory Control Theory gibt es mehrere Standardprobleme mit Lösungen, die in [Wonham (2004)] dargestellt sind. Das Hauptproblem dieser Thesis ist das Basic Supervisory Control Problem - Nichtblockierend (BSCP-NB), das wie folgt definiert wird.

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

1. $L_m(S/G) \subseteq K$,
2. $L_m(S/G)$ ist maximal, also für jede andere nichtblockierende Steuerung S' mit $L_m(S'/G) \subseteq K$ gilt:

$$L_m(S'/G) \subseteq L_m(S/G). \tag{2.34}$$

Zur Lösung dieses Problems muss der Supervisor so gewählt werden, dass

$$L_m(S/G) = \overline{K^{\uparrow C}}. \tag{2.35}$$

Der entstandene Supervisor kann entweder als Liste oder als Akzeptor R der Sprache $\overline{K^{\uparrow C}}$ realisiert werden. Die Realisierung als Liste ist in [Wenck (2006), S. 74] beschrieben. Bei der Realisierung als Akzeptor wird $R = \{Y, \Sigma, \zeta, y_0, Y_m\}$ so konstruiert, dass

- $Y = Y_m$,
- Σ = Ereignisalphabet der Steuerstrecke G,
- R = Trim(R),
- ζ, sodass $L_m(R) = L(R) := \overline{K^{\uparrow C}}$.

Für den Steuerkreis S/G gilt dann:

$$L(G||_{SPC}R) = L(G) \cap L(R) = L(G) \cap \overline{K^{\uparrow C}} = \overline{K^{\uparrow C}} = L(S/G), \tag{2.36}$$

$$L_m(G||_{SPC}R) = L_m(G) \cap L_m(R) = \overline{K^{\uparrow C}} \cap L_m(G) = L(S/G) \cap L_m(G) = L_m(S/G). \tag{2.37}$$

Durch die beschriebene Konstruktion von R als Akzeptor und die Komposition von R und G nach Gleichung (2.36) und Gleichung (2.37) verhält sich der Steuerkreis wie gewünscht. [Wenck (2006), S. 50ff, 85f]

2.8.2. Modularer Steuerungsentwurf

Statt eines einzelnen Supervisors können auch mehrere Supervisor eingesetzt werden, die die Steuerungsaufgabe übernehmen. Dieser Ansatz wird als modularer Ansatz bezeichnet. Abbildung 2.15 zeigt den Steuerkreis S_{mod}/G beim modularem Ansatz beispielhaft für zwei Supervisor. Bei diesem Ansatz wird die Steuerungsaufgabe in mehrere Teile aufgeteilt und je von einem Supervisor umgesetzt. Voraussetzung für den Ansatz ist, dass jeder Supervisor alle Ereignisse der Steuerstrecke G beobachten kann. Mit diesem Ansatz soll die Modellkomplexität der Supervisor gegenüber einem monolithischen Supervisor reduziert werden. Des Weiteren soll die Wartung bestehender Supervisor erleichtert werden.

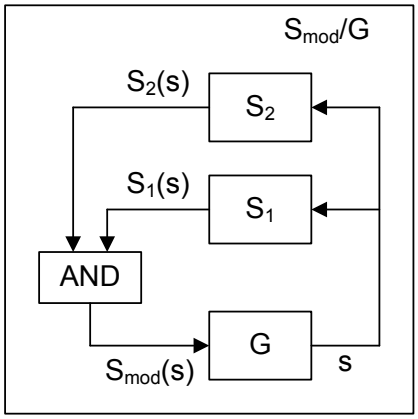


Abbildung 2.15.: Steuerkreis mit modularem Ansatz nach [Wenck (2006), S. 153]

Ein Ereignis wird nur dann ausgeführt, wenn es von keinem Supervisor verboten wird. Sind $S_1, S_2, ..., S_n$ Supervisor bezüglich der Steuerstrecke G, setzen sich die Steuereingriffe des allgemeinen modularen Supervisors $S_{mod}(s)$ aus der Schnittmenge der einzelnen Supervisor zusammen:

$$S_{mod}(s) = S_1(s) \cap S_2(s) \cap ... \cap S_n(s). \tag{2.38}$$

Damit ergibt sich für die Sprachen des modularen Steuerkreises S_{mod}/G:

$$L(S_{mod}/G) = L(S_1/G) \cap L(S_2/G) \cap ... \cap L(S_n/G), \tag{2.39}$$

$$L_m(S_{mod}/G) = L_m(S_1/G) \cap L_m(S_2/G) \cap ... \cap L_m(S_n/G). \tag{2.40}$$

Die Gesamtspezifikation K des Steuerkreises S_{mod}/G muss in Form von präfix-abgeschlossenen Teilspezifikationen zerlegbar sein, da Steuerbarkeit unter Schnittmengenbildung nur erhalten bleibt, wenn die beteiligten Verhalten präfix-abgeschlossen sind.

Für die Spezifikationen gilt dann:

$$(K_1 \cap K_2)^{\uparrow C} = K_1^{\uparrow C} \cap K_2^{\uparrow C}. \tag{2.41}$$

Beim modularen Steuerungsentwurf muss das Modular Supervisory Control Problem (MSCP) gelöst werden, das wie folgt definiert wird.

MSCP:
Gegeben ist ein Generator G mit einem Ereignisalphabet Σ, $\Sigma_{uc} \subseteq \Sigma$ und der Spezifikation $K = K_1 \cap K_2 \cap ... \cap K_n$ mit $K_i = \overline{K_i}$ für $i = 1, ..., n$. Bestimme einen modularen Supervisor, sodass gilt:

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

Da die Teilspezifikationen präfix-abgeschlossen sind, können zur Lösung dieses Problems die Supervisor S_i unabhängig entworfen werden, so dass

$$L(S_i/G) = K_i^{\uparrow C} \tag{2.43}$$

für $i = 1, ..., n$ gilt.

Der modulare Supervisor S_{mod} ergibt sich nach Gleichung (2.38). Für zwei nichtblockierende Supervisor S_1 und S_2 muss zusätzlich die Bedingung

$$\overline{L_m(S_1/G) \cap L_m(S_2/G)} = L_m(S_1/G) \cap L_m(S_2/G) \tag{2.44}$$

erfüllt sein, damit der entstandene modulare Supervisor S_{mod} nichtblockierend ist. [Wenck (2006), S. 153ff]

2.8.3. Lokal-modularer Steuerungsentwurf

Abbildung 2.16 zeigt den Steuerkreis S_{mod}/G beim lokal-modularem Ansatz beispielhaft für zwei Supervisor. Er ist eine Erweiterung des modularen Ansatzes und teilt zusätzlich die Steuerstrecke in mehrere Teile auf. Da reale Systeme oftmals aus mehreren Komponenten bestehen, wird diese Aufteilung beim lokal-modularen Ansatz ausgenutzt. Die Supervisor beobachten und steuern dabei jeweils nur einen Teil der Steuerstrecke.

Zunächst wird das Kompositionssystem gebildet. Ein Kompositionssystem ist jedes Gesamtmodell G eines System, das aus der SYPC-Komposition von n' Teilmodellen G_i' mit

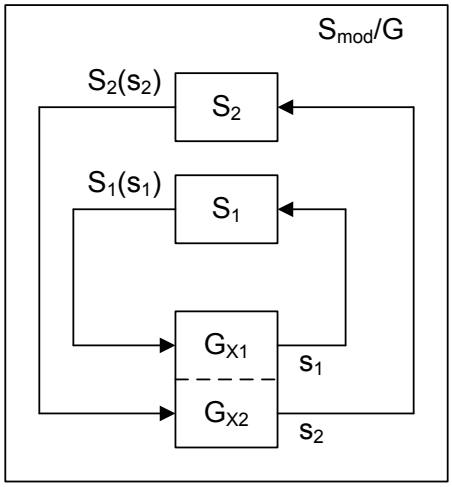


Abbildung 2.16.: Steuerkreis mit lokal-modularem Ansatz nach [Wenck (2006), S. 156]

$i \in N' = \{1, ..., n'\}$ entstanden ist. Aus dem Kompositionssystem ergeben sich die Steuerstrecke G und das Ereignisalphabet Σ, welches aus der Vereinigung der n' Ereignismengen entsteht. Sind die Komponenten eines Kompositionssystems zueinander asynchron, also besitzen sie keine gemeinsamen Ereignisse, wird das Kompositionssystem Produktsystem genannt. Das feinste Produktsystem ist das Produktsystem mit der größtmöglichen Anzahl asynchroner Komponenten.
Durch die Bildung des feinsten Produktsystems wird die Steuerstrecke asynchronisiert. Fordert eine Spezifikation allerdings eine Synchronisierung von Komponenten, werden diese Komponenten zu so genannten lokalen Systemen zusammengefasst. Lokale Systeme sind wie folgt definiert:

Lokale Systeme:
Sei G ein Produktsystem mit den Komponenten G_i mit $i \in N = \{1, ..., n\}$ und seien K_{Xj} beliebige gegebene Spezifikationen mit $j = 1, ..., m$, dann sind die lokalen Systeme G_{Xj} definiert als:

$$G_{Xj} = ||_{i \in N_{Xj}} G_i \quad \text{mit } N_{Xj} = \{k \in N \quad | \Sigma_k \cap \Sigma_{Xj} \neq \varnothing\}. \tag{2.45}$$

Der Operator $||$ ist hier und im Folgenden dem synchronen Produkt gleichzusetzen. Ein lokales System G_{Xj} fasst nach Gleichung (2.45) nur die Komponenten G_i zusammen, die gemeinsame Ereignisse mit der Spezifikation K_{Xj} besitzen. Komponenten, die nicht durch eine Spezifikation synchronisiert werden und keine gemeinsamen Ereignisse mit anderen Komponenten besitzen, sind für den lokal-modularen Supervisor nicht relevant und können weggelassen werden. Relevante lokale Systeme werden zum eingeschränkten System G_e

zusammengefasst mit:

$$G_e = ||_{j=1}^{m} G_{Xj} \quad \text{und } \Sigma_e = \bigcup_{j=1}^{m} \Sigma_{Xj}. \tag{2.46}$$

Anschließend werden die Spezifikationen K_{Xj} bezüglich der lokalen Systeme, des eingeschränkten Systems und des Gesamtsystems ausgedrückt:

$$E_{Xj} = K_{Xj} || L_m(G_{Xj}), \tag{2.47}$$
$$E_{Xje} = K_{Xj} || L_m(G_e), \tag{2.48}$$
$$E_X = K_{Xj} || L_m(G). \tag{2.49}$$

Im nachfolgenden Beispiel werden diese Begriffe veranschaulicht. Ein Kompositionssystem G ist durch seine Komponenten G_i' mit $i \in N' = \{1, ..., 5\}$ gegeben. $K_{X1} \in \Sigma_{X1}^*$ und $K_{X1} \in \Sigma_{X1}^*$ seien zwei gegebene Spezifikationen. Abbildung 2.17 zeigt in einem Venn-Diagramm die Relationen zwischen den Ereignismengen.

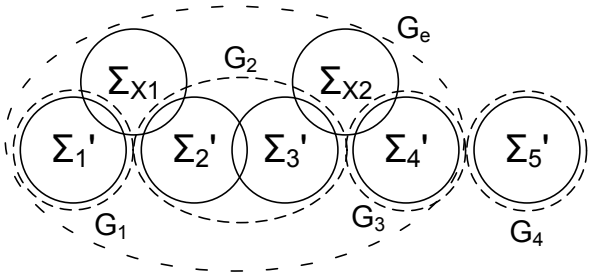


Abbildung 2.17.: Venn-Diagramm der Ereignismengen nach [Wenck (2006), S. 158]

Das feinste Produktsystem in diesem Beispiel ist $G_1 = G_1'$, $G_2 = G_2'||G_3'$, $G_3 = G_4'$ und $G_4 = G_5'$. Für die lokalen Systeme ergeben sich $G_{X1} = G_1||G_2$ und $G_{X2} = G_2||G_3$. Die angepassten Spezifikationen sind $E_{X1} = K_{X1}||L_m(G_{X1})$ und $E_{X2} = K_{X2}||L_m(G_{X2})$. Das eingeschränkte System ist $G_e = G_1||G_2||G_3$.

Zum Supervisor-Entwurf wird lokale Modularität gefordert. Sie ist definiert durch:

Lokale Modularität:
Sei I eine beliebige Indexmenge und $L_i \subseteq \Sigma_i^*$ mit $i \in I$, dann ist die Menge $\{L_i\}$ mit $i \in I$ lokal modular, wenn

$$||_{i \in I} \overline{L_i} = \overline{||_{i \in I} L_i}. \tag{2.50}$$

Für zwei lokale Supervisor gilt nach [Wenck (2006), S. 159] das nachfolgende Theorem.

> **Theorem:**
> Gegeben sei ein Kompositionssystem G mit n' Komponenten G_i' und zwei lokale Spezifikationen K_{X1} und K_{X2}. Es sei $supC(E_{X1}, G_{X1})$ die supremale steuerbare Teilsprache von E_{X1} bezüglich G_{X1} und $supC(E_{X2}, G_{X2})$ die supremale steuerbare Teilsprache von E_{X2} bezüglich G_{X2}. Sind diese supremalen steuerbaren Teilsprachen lokal-modular, dann gilt
>
> $$supC(E_{X1e} \cap E_{X2e}, G_e) = supC(E_{X1}, G_{X1}) \,||\, supC(E_{X2}, G_{X2}). \tag{2.51}$$

Nach Gleichung (2.51) ist es ausreichend, die einzelnen lokalen Supervisor für die lokalen Systeme mittels

$$L(S_1/G_{X1}) = supC(E_{X1}, G_{X1}) \text{ und} \tag{2.52}$$
$$L(S_2/G_{X2}) = supC(E_{X2}, G_{X2}) \tag{2.53}$$

zu berechnen.

Anschließend muss die lokale Modularität über

$$\overline{supC(E_{X1}, G_{X1})} \,||\, \overline{supC(E_{X2}, G_{X2})} = \overline{supC(E_{X1}, G_{X1}) \,||\, supC(E_{X2}, G_{X2})} \tag{2.54}$$

überprüft werden.

Zur Verringerung des Speicherbedarfs der Steuerung können die Steuerungen nach [Su und Wonham (2004)] minimiert werden. Dabei bleibt die Eigenschaft der lokalen Modularität nicht erhalten. Eine Reduzierung darf daher erst nach der Prüfung auf lokale Modularität erfolgen. [Wenck (2006), S. 155ff]

2.8.4. Dezentraler Steuerungsentwurf

Abbildung 2.18 zeigt den Steuerkreis S_{dez}/G beim dezentralem Ansatz beispielhaft für zwei Supervisor. Dieser Ansatz ist ähnlich wie der modulare Ansatz, unterscheidet sich aber in der Hinsicht, dass der Supervisor nicht alle Ereignisse beobachten kann. Ursachen dafür können zum Beispiel sein, dass Sensorereignisse nicht an jedem Ort zur Verfügung stehen oder eine Ansammlung aller Ereignisse an einem (monolithischer Ansatz) oder mehreren Orten (modularer Ansatz) nicht möglich ist. Die eingeschränkte Verfügbarkeit aller Ereignisse wird für n Supervisor durch natürliche Projektionen P_i mit $i = 1, \ldots, n$ dargestellt.

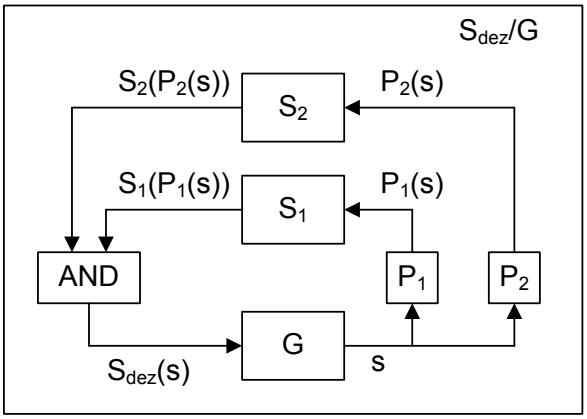


Abbildung 2.18.: Steuerkreis mit dezentralem Ansatz nach [Wenck (2006), S. 162]

Beim dezentralen Steuerungsentwurf werden zwei Problemklassen unterschieden. Das lokale dezentrale Steuerungsproblem besagt, dass jeder Supervisor seine eigene Steuerungsaufgabe erfüllen muss. Das Steuerungsproblem ist gelöst, wenn jeder Supervisor isoliert betrachtet seine Steuerungsaufgabe erfüllt. Bei globalen dezentralen Steuerungsproblemen können Spezifikationen nicht in lokale Spezifikationen aufgeteilt werden, sodass lediglich eine globale Spezifikation zur Verfügung steht. Diese Steuerungsprobleme sind gelöst, wenn die einzelnen Supervisor gemeinsam die Spezifikationen erfüllen.

Gleichzeitig gibt es die Unterscheidung zwischen Problemen ohne und mit Toleranz. Unter einem Problem ohne Toleranz ist zu verstehen, dass das Verhalten $L(S/G)$ des gesteuerten Systems S/G exakt der Spezifikation K entsprechen muss:

$$L(S/G) = K \subseteq L(G). \tag{2.55}$$

Bei einem Problem ohne Toleranz kann entweder ein maximal zulässiges Verhalten L_a angegeben werden, sodass das gesteuerte System S/G lediglich ein Verhalten erzeugen muss, das in der Spezifikation L_a enthalten ist:

$$L(S/G) \subseteq L_a \subseteq L(G). \tag{2.56}$$

Andererseits kann auch durch die Angabe einer minimal erforderlichen Sprache L_r ein bestimmter Bereich festgelegt werden, in dem das Verhalten des gesteuerten Systems liegt:

$$L_r \subseteq L(S/G) \subseteq L_a \subseteq L(G). \tag{2.57}$$

Für den dezentralen Steuerungsansatz gibt es daher die vier Steuerungsprobleme lokales dezentrales Problem mit Toleranz (LDPT), lokales dezentrales Problem ohne Toleranz (LDPZT), globales dezentrales Problem mit Toleranz (GDPT) und globales dezentrales Problem ohne Toleranz (GDPZT). Die Lösungen dieser Probleme sind in [Wenck (2006); S. 163ff] beschrieben. [Wenck (2006), S. 17, 161ff]

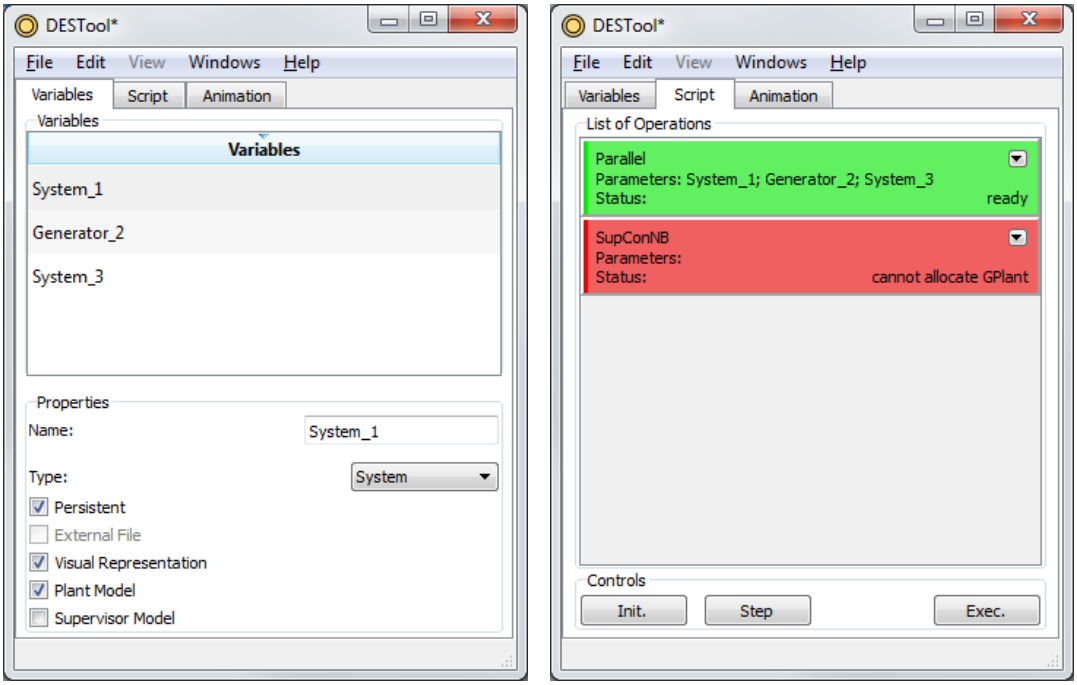
2.9. Software zum Steuerungsentwurf

Zum Steuerungsentwurf gibt es diverse verschiedene Software. In Kapitel 4 werden einige Tools miteinander verglichen. In diesem Abschnitt wird die Software, die in dieser Thesis verwendet wird, vorgestellt. Abschnitt 2.9.1 beschreibt „DESTool“ und Abschnitt 2.9.2 den Codegenerator „ACArrow“.

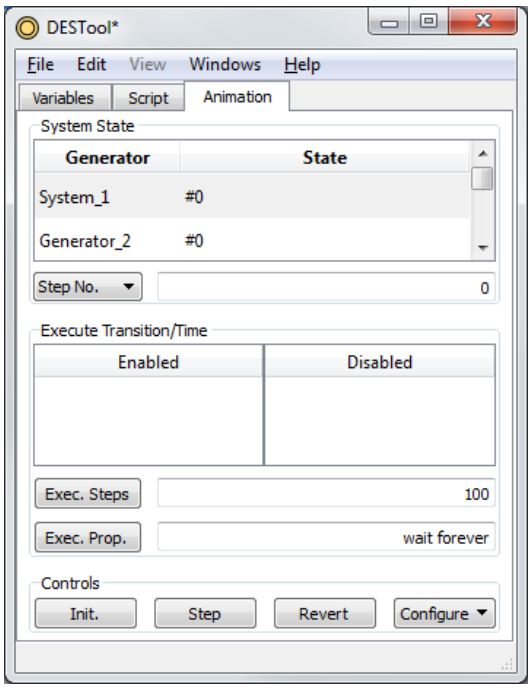
2.9.1. Analyse- und Synthesetool „DESTool“

„DESTool“ ist ein grafisches Tool zur Analyse und Synthese von ereignisdiskreten Systemen. Es wurde an der Friedrich-Alexander-Universität Erlangen-Nürnberg von Thomas Moor und seinem Team entwickelt. Die aktuelle Version des Tools ist Version 0.82 vom 01.03.2016. Da das Tool sehr umfangreich ist, wird in dieser Thesis lediglich auf die hier relevanten Funktionen eingegangen. Weitere Funktionen können auf der Internetseite von „DESTool“ [Moor (2016)] nachgelesen werden.

Das Hauptfenster des Tools (siehe Abbildung 2.19 auf Seite 46) zeigt unter dem Reiter „Variables“ die angelegten Variablen. Neue Variablen können über einen Rechtsklick in dem Fenster hinzugefügt werden. Generatoren werden innerhalb dieser Thesis immer als Variablen vom Typ „System“ angelegt, da bei Variablen des Typs „Generator“ die Ereignisse nicht als steuerbar oder beobachtbar deklariert werden können. Außerdem können Variablen der Typen „Alphabet“ und „Boolean“ hinzugefügt werden. Erstere beschreiben eine Ansammlung von Ereignissen, die beispielsweise für das Einfügen von Schlingen genutzt werden können. Letztere dienen als Ergebnis für Operationen, wie zum Beispiel die Prüfung auf Nichtblockieren. Streckenmodelle werden im unteren Bereich des Hauptfensters als „Plant Model“ und Supervisor-Modelle als „Supervisor Model“ deklariert. Diese Deklaration ist wichtig für den späteren Import in den Codegenerator „ACArrow“ (siehe Abschnitt 2.9.2).
Im Reiter „Script“ können verschiedene Operationen hinzugefügt und über die Buttons „Init“, „Step“ und „Exec.“ schrittweise oder komplett ausgeführt werden. Tabelle 2.3 zeigt eine Auswahl von Operationen in „DESTool“. Im dritten Reiter „Animation“ können Generatoren oder Systeme simuliert werden.



(a) Reiter „Variables“ (b) Reiter „Script“



(c) Reiter „Animation“

Abbildung 2.19.: Hauptfenster von „DESTool“

Operation	Beschreibung	Ergebnis
IsNon-blocking(G_1, G_2)	Prüft, ob zwei Generatoren oder Systeme G_1 und G_2 nichtblockierend sind.	Boolean
Parallel(G_1, G_2)	Berechnet aus zwei Generatoren oder Systemen G_1 und G_2 das synchrone Produkt (SYPC).	System / Generator[2]
Product(G_1, G_2)	Berechnet aus zwei Generatoren oder Systemen G_1 und G_2 die strenge Synchronisation (SPC).	System / Generator[2]
SelfLoop(G, Σ)	Ergänzt jeden Zustand eines Generators oder Systems G mit Ereignissen eines Alphabets Σ als Schlinge.	System / Generator[2]
SubConNB(G_{Plant}, G_{Spec})	Berechnet aus einem Streckenmodell G_{Plant} und einer Spezifikation G_{Spec} die supremale steuerbare nichtblockierende Teilsprache.	System / Generator[2]
SubReduce(G_{Plant}, G_{Sup})	Reduziert die Zustände eines Supervisors G_{Sup} bezüglich eines Streckenmodells G_{Plant}.	System / Generator[2]

Tabelle 2.3.: Operationen in „DESTool“

Eine Variable kann im Reiter „Variables“ mit der Auswahl von „Show *Name*“ im Kontextmenü der rechten Maustaste geöffnet werden. In einem separaten Fenster wird die Variable angezeigt. Bei einem System sind auf der rechten Seite die Zustände und Transitionen grafisch dargestellt. Links gibt es drei Reiter für die Transitionen („Transitions“), die Zustände („States“) und die Ereignisse („Alphabet“). Bei den Zuständen können der Initialzustand („I“) und die markierten Zustände („M“) ausgewählt werden. Für den späteren Import in den Codegenerator „ACArrow“ (siehe Abschnitt 2.9.2) müssen die Zustände nummeriert sein. Wird der Codegenerator nicht verwendet, können die Zustände auch Buchstaben oder andere Zeichen enthalten. Bei den Ereignissen kann unter anderem angegeben werden, ob das Ereignis steuerbar („C“) oder beobachtbar („O“) ist.
Mit einem Rechtsklick auf die Freifläche, auf der die Zustände und Transitionen grafisch dargestellt sind, können diese mit „Re-arrange Graph via Grid“ oder „Re-arrange Graph via Dot“ im Raster oder punktförmig auf der Zeichenfläche angeordnet werden.

2.9.2. Codegenerator „ACArrow“

Der Codegenerator „ACArrow“ wurde im Jahr 2014 im Rahmen der Masterthesis von Nadine Gohert entwickelt. Seine Aufgabe ist es, Automaten eines „DESTool“ Projektes oder Petri-

[2] Wird als Ergebnis ein Name eines noch nicht vorhandenen Systems eingegeben, wird dieses als Generator erzeugt. Alternativ muss ein bereits vorhandenes System als Ausgabe gewählt werden.

netze eines Matlab Protokolls in Quelltext für eine SPS umzuformen. Wie im Abschnitt 2.9.1 „DESTool“ wird in dieser Thesis lediglich auf die hier relevanten Funktionen eingegangen. Weitere Informationen zum Codegenerator können in [Gohert (2014), S. 58ff] nachgelesen werden.

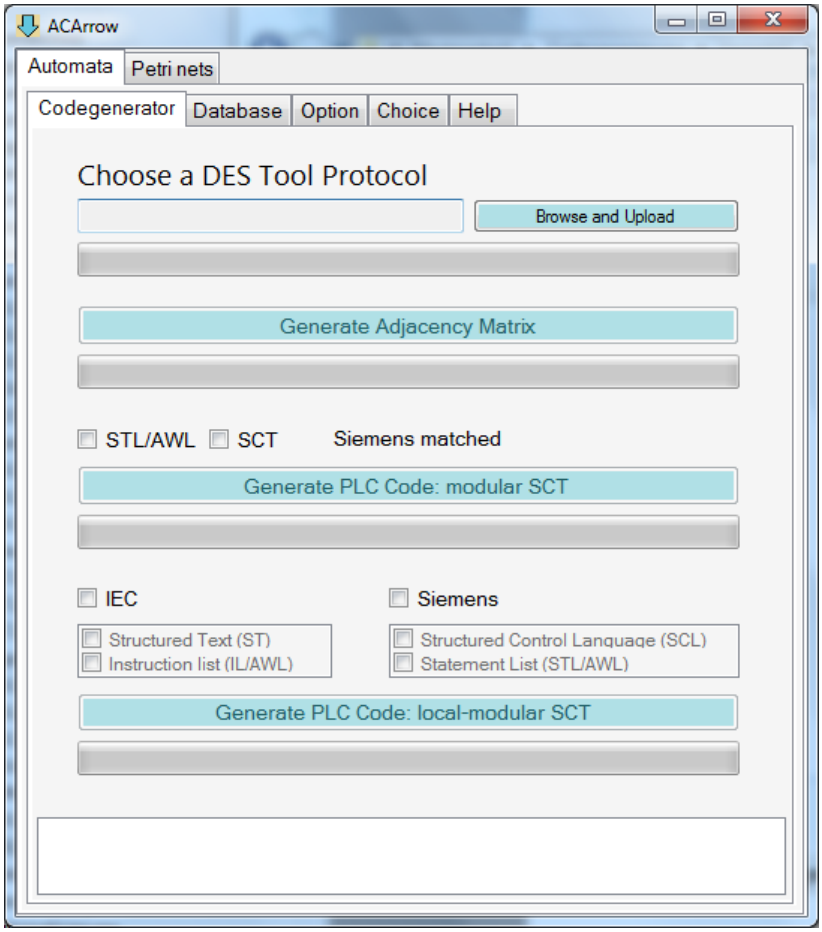
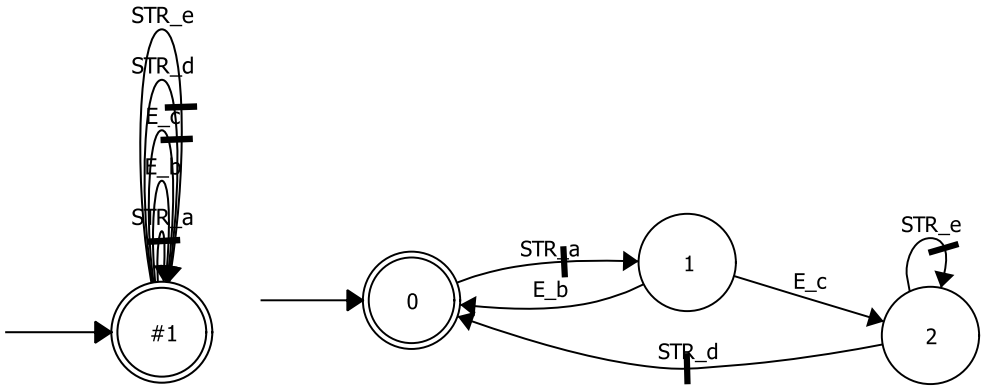


Abbildung 2.20.: Hauptfenster des Codegenerators „ACArrow“

Nach dem Start des Codegenerators wird das Hauptfenster angezeigt (siehe Abbildung 2.20). Im oberen Bereich kann das „DESTool“ Projekt ausgewählt und importiert werden. Es werden nur Generatoren importiert, die als „Plant Model“ oder „Supervisor Model“ im „DESTool“ Projekt gekennzeichnet sind. Befinden sich mehrere steuerbare Ereignisse an einem Zustand der Automaten, wird nach dem Import der Reiter „Choice“ geöffnet. Dort kann der Bediener mit einem Haken auswählen, dass das Auswahlproblem für diesen Zustand zufällig gelöst wird. Anderenfalls findet automatisch eine priorisierte Auswahl statt. Im Reiter „Option“ kann die Startadresse der Merker vorgegeben werden. Außerdem wird definiert, welche Anfangskennungen zum Beispiel Ein- und Ausgänge sowie steuerbare Ereignisse

besitzen. Die Anfangskennungen müssen mit der Bezeichnung der Ereignisse in „DESTool“ übereinstimmen.
Nach dem Import und der Auswahl der Optionen wird im Reiter „Codegenerator“ die Codegenerierung gestartet. Es muss ausgewählt werden, ob der Quelltext in SCL oder AWL erstellt werden soll. Beim lokal-modularen Ansatz kann zusätzlich ausgewählt werden, ob die IEC oder Siemens Syntax verwendet wird. Die Codegenerierung für den monolithischen oder modularen Entwurfsansatz wird über den Button „Generate PLC Code: modular SCT“ gestartet, für den lokal-modularen Entwurfsansatz über den Button „Generate PCL Code: local-modular SCT“. In dieser Thesis wird nur SCL Quelltext mit Siemens Syntax erstellt und verwendet.
Der generierte Quelltext wird in dem Ordner „CODE“, der sich im selben Verzeichnis wie der Codegenerator befindet, gespeichert. Für den lokal-modularen Ansatz werden zusätzlich zu den SCL-, ST- oder AWL-Quellen Excel-Dateien mit den Variablen generiert, die beispielsweise im TIA-Portal importiert werden können. Für den modularen Ansatz müssen die Variablen manuell eingefügt werden.

Der Quelltext, der bei dem Codegenerator „ACArrow“ entsteht, soll anhand eines Beispiels erklärt werden. Das Beispiel endet mit dem Symbol □. Abbildung 2.21 (a) zeigt ein einfaches Streckenmodell mit einem Zustand, an dem alle Ereignisse Schlingen sind. Es gibt drei steuerbare Ereignisse a, d und e sowie zwei nicht steuerbare Ereignisse b und c. Abbildung 2.21 (b) zeigt einen Supervisor für dieses Streckenmodell, der aus drei Zuständen besteht. Damit der Codegenerator angewendet werden kann, müssen alle steuerbaren Ereignisse mit „STR“ und alle nicht steuerbaren Ereignisse mit „E“ gekennzeichnet werden.



(a) Streckenmodell (b) Supervisor

Abbildung 2.21.: Streckenmodell und Supervisor für ein System

Nach der Codegenerierung werden alle Zustände und Ereignisse als Variablen dargestellt. Die modulare Codegenerierung erzeugt einen SCL Quelltext aus Listing 2.1 für den Supervisor. Für jede abgehende Kante an einem Zustand gibt es eine Abfrage, die ausgeführt

wird, wenn der Zustand aktiv und das Ereignis möglich ist. Ist die Abfrage wahr, wird die Variable des Nachfolgezustandes gesetzt und die des aktuellen Zustandes zurückgesetzt. Am Ende werden die steuerbaren Ereignisse abhängig von den aktiven Zuständen gesetzt beziehungsweise zurückgesetzt. Die Variable „block_Supervisor“ ist nicht dokumentiert. Aus diesem Grund werden die Zeilen, in denen die Variable gesetzt wird, auskommentiert und die Variable auf null initialisiert.

```
//Code generated by ACArrow Generator:Supervisor
//Z1
IF Supervisor_Z1 AND NOT block_Supervisor AND E_b THEN
  Supervisor_Z0:=1;
  Supervisor_Z1:=0;
  block_Supervisor:=TRUE;
END_IF;

//Z1
IF Supervisor_Z1 AND NOT block_Supervisor AND E_c THEN
  Supervisor_Z2:=1;
  Supervisor_Z1:=0;
  block_Supervisor:=TRUE;
END_IF;

//Z0
IF Supervisor_Z0 AND NOT block_Supervisor AND STR_a THEN
  Supervisor_Z1:=1;
  Supervisor_Z0:=0;
  block_Supervisor:=TRUE;
END_IF;

//Z2
IF Supervisor_Z2 AND NOT block_Supervisor AND STR_d THEN
  Supervisor_Z0:=1;
  Supervisor_Z2:=0;
  block_Supervisor:=TRUE;
END_IF;

//Set controllable Events
STR_a_Plant:=Supervisor_Z0;
STR_d_Plant:=Supervisor_Z2;
STR_e_Plant:=Supervisor_Z2;
```

Listing 2.1: Vom Codegenerator generierter Quelltext für den Supervisor

Die lokal-modulare Codegenerierung erzeugt zwölf SCL Quelltexte. In der Hauptfunktion „Main_SCT“ werden alle untergeordneten Funktionen aufgerufen. Außerdem gibt es einen mit Kommentaren gekennzeichneten Bereich, in dem die steuerbaren Ereignisse der untergeordneten Steuerung zugewiesen werden können. Die Funktionen „Init_SCT“, „Read_Input_SCT“ und „Write_Output_SCT“ sind für die Initialisierung, das Einlesen der Eingänge beziehungsweise das Schreiben der Ausgänge zuständig. In den Funktionen „C_Events_Plant_SCT“, „UC_Events_Plant_SCT“ und „UC_Events_Super_SCT“ gibt es wie beim modularen Entwurf für jede abgehende Kante an einem Zustand eine Abfrage. Die Abfragen sind in steuerbare und nicht steuerbare Ereignisse der Steuerstrecke sowie in nicht steuerbare Ereignisse des Supervisors eingeteilt. Die Zuweisung der steuerbaren Ereignisse abhängig von den aktiven Zuständen erfolgt in der Funktion „C_Events_Enable_SCT“. Die restlichen Funktionen sind für die Art der Codegenerierung zuständig und werden in dieser Thesis nicht weiter behandelt. Weitere Informationen sind in der Masterthesis von Nadine Gohert [Gohert (2014)] zu finden. □

Enthält ein „DESTool“ Projekt Automaten mit nur einem Zustand, muss im SCL Quelltext, den der Codegenerator erzeugt hat, die Reihenfolge des Setzens und Rücksetzens von Zuständen getauscht werden. Anderenfalls funktioniert der Quelltext nicht, da die Markierung verloren geht. Listing 2.2 zeigt die Reihenfolge, die der Codegenerator generiert, Listing 2.3 zeigt die Reihenfolge, bei der die Markierung nicht verloren geht.

```
IF "G5_60_Z1" AND "STR_60_Abs1Erlauben_0" THEN
        "G5_60_Z1" := 1;
        "G5_60_Z1" := 0;
        "STR_60_Abs1Erlauben" := 1;
END_IF;
```

Listing 2.2: Vom Codegenerator generierter Quelltext

```
IF "G5_60_Z1" AND "STR_60_Abs1Erlauben_0" THEN
        "G5_60_Z1" := 0;
        "G5_60_Z1" := 1;
        "STR_60_Abs1Erlauben" := 1;
END_IF;
```

Listing 2.3: Korrigierter Quelltext

Beim lokal-modularen Entwurfsansatz werden die Eingänge in der Funktion „Read_Input_SCT“ eingelesen. Alle Eingänge sind jeweils nur für einen Zyklus aktiv, da die steigende Flanke abgefragt wird. Gerade bei Eingangssignalen, die länger anstehen können, ist eine Flankenauswertung an dieser Stelle nicht geeignet und sollte durch eine einfache Zuweisung ersetzt werden.

In der Funktion „Init_SCT“ beim lokal-modularen Ansatz werden mit dem Codegenerator „ACArrow“ nur die ersten Zustände aller Generatoren initialisiert. Bei Fehlern während des Betriebes könnte dies dazu führen, dass nach einem Neustart zwei Zustände markiert sind. Aus diesem Grund sollte die Initialisierung erweitert werden, sodass alle Zustände mit eins oder null initialisiert werden.

Der generierte und gegebenenfalls veränderte Quelltext kann nun im TIA-Projekt[3] importiert werden. Nach dem Anlegen der benötigten Variablen müssen die Ein- und Ausgänge sowie die steuerbaren Ereignisse des generierten Quelltextes mit den Variablen aus dem TIA-Projekt verknüpft werden. Dieser Schritt entfällt, wenn die Variablen denselben Namen besitzen. Anschließend muss die Hauptfunktion, die der Codegenerator erzeugt hat, im TIA-Projekt aufgerufen werden. Beim Ausschalten des Automatikbetriebs sollte für den lokal-modularen Ansatz auch das Bit „InitBit“ zurückgesetzt werden, damit bei einem Neustart keine alten Zustände markiert sind. Beim modularen Ansatz muss die Initialisierung manuell erfolgen.

[3] Informationen zum TIA-Portal und dem TIA-Projekt sind im Abschnitt 3.2 zu finden. Die Einbindung des Quelltextes in das TIA-Portal wird in Abschnitt 6.2 beschrieben.

3. Beschreibung der Modellfabrik

Die Modellfabrik, die von der Firma KÖSTER Systemtechnik GmbH hergestellt wurde, stellt an sechs Stationen Relais für Windkraftanlagen im Gehäuse mit Verschluss her. Die Produkte bestehen aus einem Unterbau, einer ein- oder zweipoligen Platine und einem Deckel. Abbildung 3.1 zeigt eine schematische Darstellung der Modellfabrik. Rechts befindet sich Station 10 mit dem Transportsystem. Links ist Station 20 mit Zufuhr- und Auswurfbändern sowie der Linearachse dargestellt. Oben innerhalb des Schutzkäfigs befindet sich Station 30 mit dem 6-Achsen-Knickarmroboter, den zwei Roboterbändern, dem Magazin und einem Handling. Im oberen Bereich des Transportsystems ist Station 40 mit der elektrischen Prüfvorrichtung und unten Station 50 mit der pneumatischen Presse dargestellt. Station 60 mit Handling, Transportband, Abschiebern und Kamera befindet sich unten innerhalb des Schutzkäfigs. Die einzelnen Stationen sind in Abschnitt 3.1 näher erläutert. Abschnitt 3.2 beschreibt die Automatisierung der Modellfabrik und in Abschnitt 3.3 werden die vorgenommenen Optimierungen an der Modellfabrik erläutert.

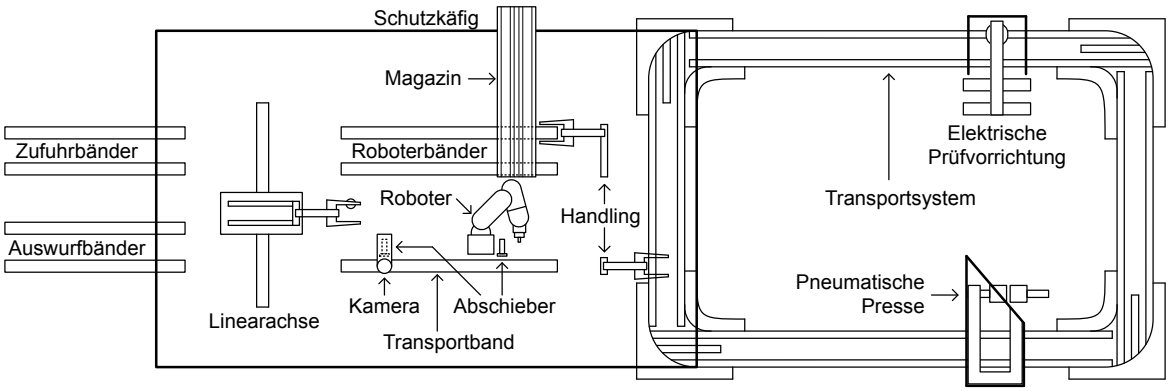


Abbildung 3.1.: Schematische Darstellung der Modellfabrik

Für jede Station gibt es ein Bedienpult mit Industrietastern, -schaltern, Not-Halt und einem Touchpanel zur Bedienung. Station 20, 30 und 60 befinden sich teilweise oder ganz innerhalb eines geschlossenen Schutzkäfigs aus Plexiglas, um Verletzungen durch die Linearachse oder den Roboter zu vermeiden. Schiebetüren am Schutzkäfig ermöglichen einen Eingriff in die Modellfabrik. Die Ruheposition der Linearachse und des Roboters werden mit einem

Sensor überwacht. Sind beide Sensoren aktiv, können die Schiebetüren geöffnet werden, anderenfalls geht die Modellfabrik beim Öffnen der Türen in den Not-Halt. Station 40 und 50 sind mit Plexiglas gegen Berührungen geschützt.

3.1. Beschreibung der Stationen

Die folgenden Abschnitte beschreiben die Stationen 10 bis 60. Station 10 ist der Leitstand, Station 20 das Lager, Station 30 der Roboter, Station 40 die elektrische Prüfung, Station 50 die pneumatische Presse und Station 60 die Bildverarbeitung.

3.1.1. Station 10: Leitstand

Abbildung 3.2 zeigt das Transportsystem des Leitstandes (Station 10). Es besteht aus vier Transferbändern, auf denen die Werkstücke auf Transportwagen von Station 30 über Station 40 und 50 zu Station 60 transportiert werden. An diesen Stationen und zwischen den Stationen 40 und 50 sowie zwischen 50 und 60 befinden sich Vereinzeler, die die Transportwagen an der Weiterfahrt hindern. Die Transportwagen sind mit einem RFID-Chip (engl.: Radio-Frequency IDentification) ausgestattet, der Informationen über das Werkstück, das sich auf dem Transportwagen befindet, speichert. RFID-Schreib-/Lesegeräte befinden sich an den Stationen 30 bis 60. An den Stationen 40 und 50 gibt es Fixiervorrichtungen für die Transportwagen, um ein präzises Arbeiten am Werkstück zu ermöglichen.

Der Leitstand steuert die Vereinzeler, die Fixiervorrichtungen sowie die Transferbänder des Transportsystems. Die Bänder können nur gemeinsam angesteuert werden. Das Auslesen und Beschreiben der RFID-Chips erfolgt ebenfalls über den Leitstand. Ein RFID-Chip kann mit einem Byte beschrieben werden und kann folgende Werte annehmen:

- 1 (dezimal) / 0x01 (hexadezimal): Werkstück einpolig ungeprüft
- 2 (dezimal) / 0x02 (hexadezimal): Werkstück zweipolig ungeprüft
- 11 (dezimal) / 0x0B (hexadezimal): Werkstück einpolig Platine fehlerfrei
- 12 (dezimal) / 0x0C (hexadezimal): Werkstück zweipolig Platine fehlerfrei
- 241 (dezimal) / 0xF1 (hexadezimal): Werkstück einpolig Platine fehlerhaft
- 242 (dezimal) / 0xF2 (hexadezimal): Werkstück zweipolig Platine fehlerhaft
- 243 (dezimal) / 0xF3 (hexadezimal): Werkstück einpolig Deckel fehlerhaft
- 244 (dezimal) / 0xF4 (hexadezimal): Werkstück zweipolig Deckel fehlerhaft

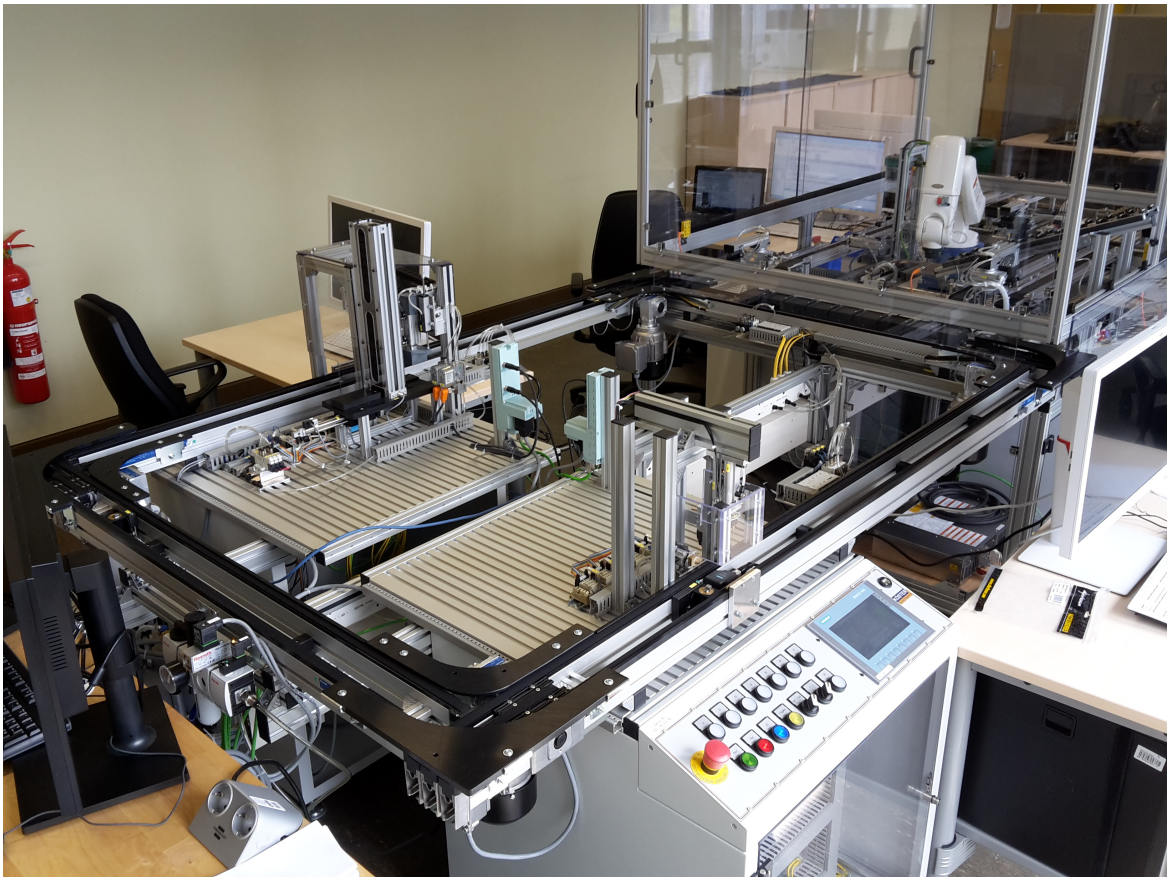


Abbildung 3.2.: Station 10: Leitstand

Die Information, was auf den RFID-Chip geschrieben werden soll, wird von den Stationen 30 bis 60 entschieden und an Station 10 übertragen. Die gesamte Kommunikation zwischen den einzelnen Stationen erfolgt mittels interner Kommunikation über Station 10. Das bedeutet es gibt keine direkte Kommunikation zwischen den Stationen 20 bis 60.

3.1.2. Station 20: Lager

Abbildung 3.3 zeigt das Lager (Station 20) mit den vier Transportbändern auf der linken Seite. Auf den zwei oberen Zufuhrbändern werden Unterteile in die Modellfabrik hinein und auf den zwei unteren Auswurfbändern fertige Produkte aus der Modellfabrik heraus geführt. Auf der rechten Seite sind oben die beiden Roboterbänder von Station 30 und unten das Transportband von Station 60 zu sehen. Bei den Zufuhrbändern von Station 20 und den Roboterbändern von Station 30 ist im Bild jeweils das obere für ein- und das untere für zweipolige

Werkstücke vorgesehen. Auf den Auswurfbändern von Station 20 und dem Transportband von Station 60 wird keine Unterscheidung zwischen ein- und zweipoligen Produkten vorgenommen. Das obere Auswurfband wird so lange gefüllt bis beide Sensoren am Bandanfang und am Bandende aktiv sind. Dann wird das untere Auswurfband gefüllt. Sind beide Bänder voll, werden keine Werkstücke von Station 60 abgeholt.

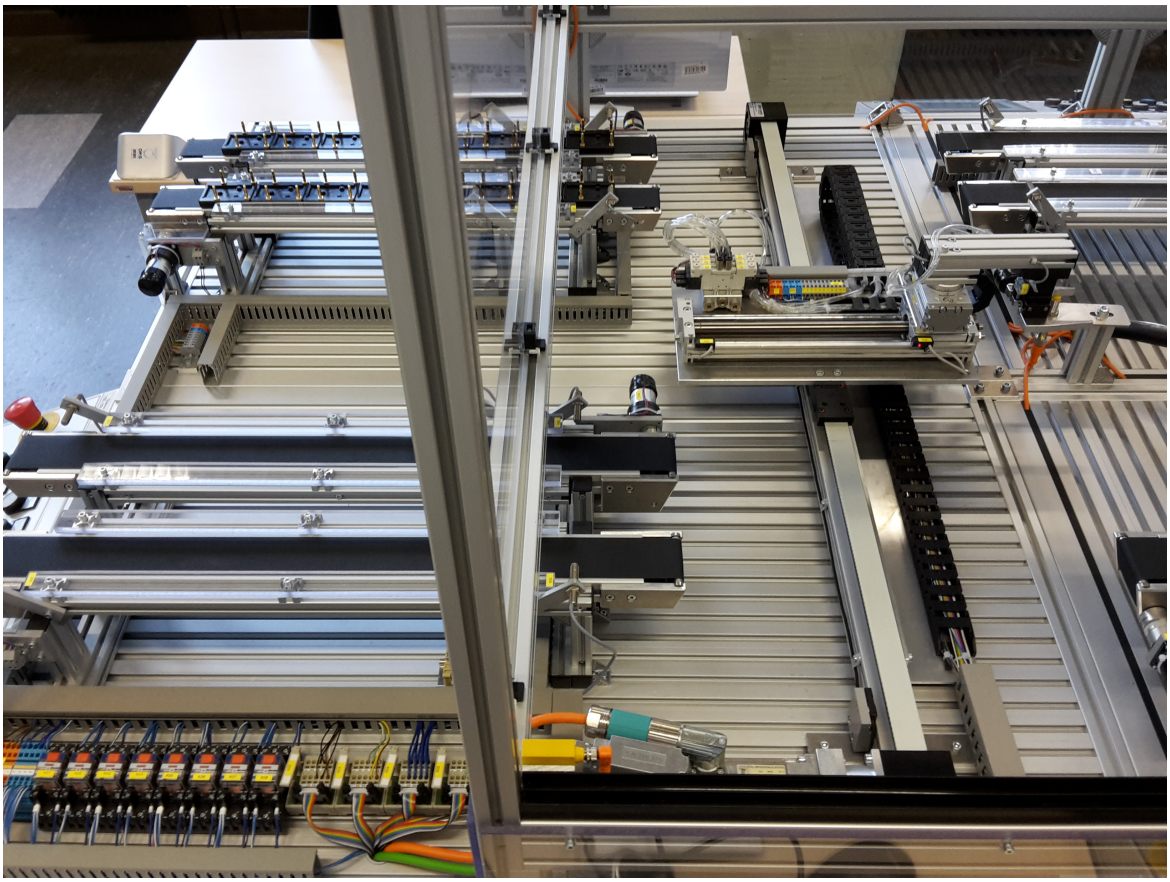


Abbildung 3.3.: Station 20: Lager

Im mittleren rechten Bereich des Bildes ist die Linearachse in ihrer Ruheposition dargestellt. Sie ist mit Servomotor und Absolutwertgeber ausgestattet und transportiert die Werkstücke bzw. Produkte zwischen den Transportbändern von Station 20, 30 und 60. Die Abarbeitung der Transportaufträge erfolgt immer in der gleichen Reihenfolge. Die höchste Priorität haben Werkstücke, die zur Abholung an Station 60 bereit liegen. Mittlere Priorität haben einpolige Werkstücke, die von Station 20 zu Station 30 transportiert werden. Die geringste Priorität haben zweipolige Werkstücke, die von Station 20 zu Station 30 transportiert werden. Bereits begonnene Aufträge werden immer erst zu Ende ausgeführt, ehe ein neuer Auftrag gestartet wird. Ob die Roboterbänder von Station 30 frei sind oder ein Werkstück an Station 60 zur

Abholung bereit liegt, wird über ein Bit von Station 30 bzw. 60 über Station 10 an Station 20 übertragen.

Station 20 steuert außerdem die Produktzähler. Es gibt vier Zähler: zwei Sollwertzähler und zwei Istwertzähler jeweils für ein- und zweipolige Produkte. Die Zähler sind so implementiert, dass beim Start der Automatikfunktion der vom Benutzer vorgegebene Wert am Touchpanel in den Soll- und Istwertzähler übernommen wird. Sobald die Linearachse ein Unterteil von Station 20 zu Station 30 gebracht hat, wird der Istwertzähler für ein- bzw. zweipolige Produkte um eins dekrementiert. Die Linearachse bringt, vorausgesetzt Station 30 ist frei, so lange Unterteile zu Station 30 bis der Istwertzähler den Wert null erreicht hat.

3.1.3. Station 30: Roboter

Abbildung 3.4 zeigt die Roboterstation (Station 30). Der 6-Achsen-Knickarmroboter von der Firma Denso Wave Incorporated ist in der Mitte des Bildes in seiner Ruheposition dargestellt. Vor dem Roboter befinden sind zwei Transportbänder (Roboterbänder), auf denen die Unterteile von Station 20 ankommen. Ein Gleitlagermagazin, welches vorne in der Mitte des Bildes dargestellt ist, führt die Platinen in Greifweite des Roboters. Der rechte Teil des Magazins ist für ein- und der linke für zweipolige Platinen vorgesehen. Zwei Sensoren jeweils am Ende des Magazins zeigen an, ob sich Platinen im Magazin befinden. Am Ende der Transportbänder befindet sich ein Handling. Es transportiert die Werkstücke von den Transportbändern auf die Transportwagen des Transportsystems, die links dargestellt sind.

Über Sensoren am Bandanfang erkennt der Roboter, ob das Unterteil in der richtigen Ausrichtung auf dem Transportband liegt. Durchbricht das Werkstück die Lichtsensoren in der Mitte des Transportbandes, stoppt das Transportband, solange das Werkstück in Bearbeitung ist. Liegt das Werkstück verkehrt herum, dreht der Roboter es um. Anschließend entnimmt er dem Magazin eine Platine und positioniert sie auf dem Unterteil. Sobald der Roboter an seiner Ruheposition angekommen ist, startet das Transportband. Nachdem das Handling das Werkstück auf einem Transportwagen abgelegt hat, sendet Station 30 an Station 10 die Information, ob es sich um ein ein- oder zweipoliges Werkstück handelt. Station 10 schreibt diese Information auf den RFID-Chip und gibt den Transportwagen über den Vereinzeler frei. Gleichzeitig wird das Transportband von Station 30, von dem das Werkstück entnommen wurde, für ein neues Werkstück freigegeben.

3.1.4. Station 40: Elektrische Endkontrolle

Abbildung 3.5 zeigt die Station der elektrischen Endkontrolle (Station 40). Die Station besteht aus einer Messvorrichtung mit vier Taststiften und einem Berührschutz aus Plexiglas.

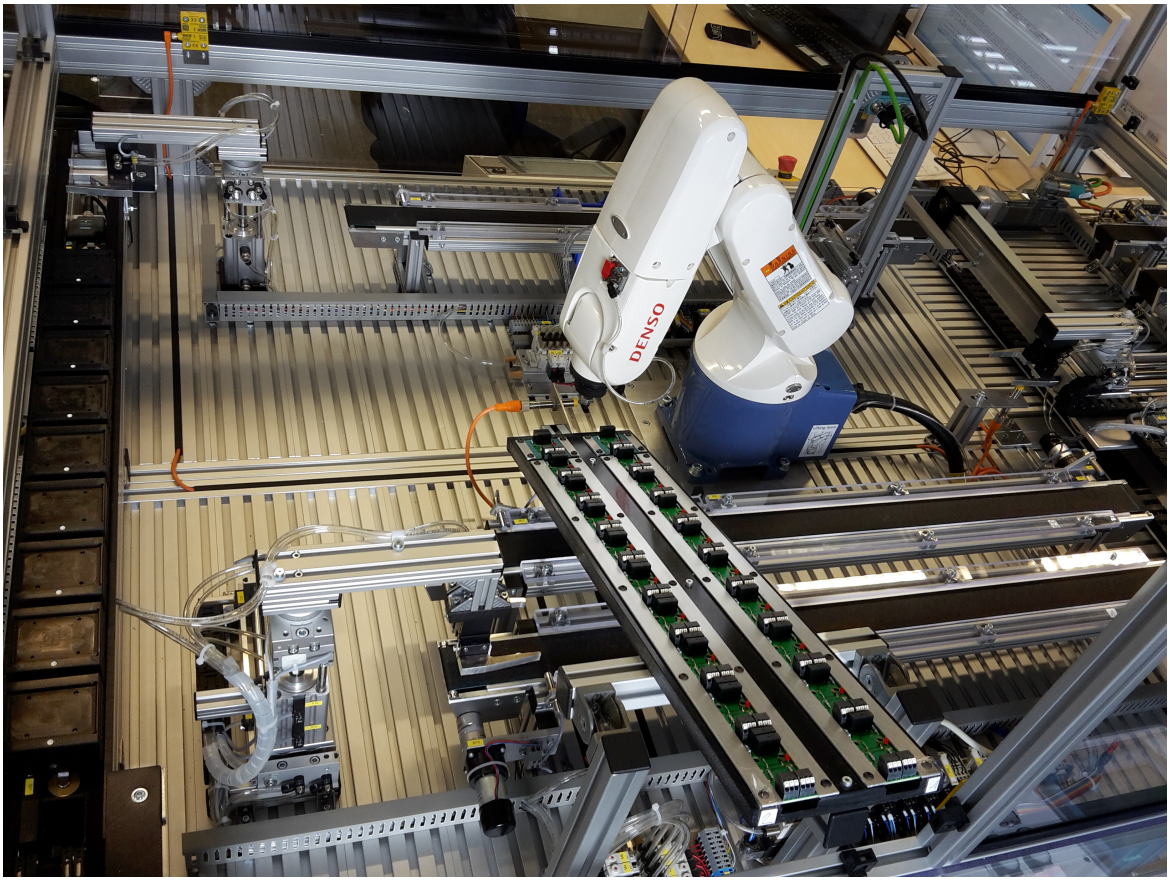


Abbildung 3.4.: Station 30: Roboter

Das Werkstück auf dem Transportwagen fährt in die Station hinein, der Transportwagen wird von dem Vereinzeler angehalten und von der Spannvorrichtung eingespannt. Dann fährt die Messvorrichtung mit den Taststiften herunter und misst die Spannung der Platine. Die gemessene Spannung wird mit einer Kontrollspannung verglichen. Ist die gemessene Spannung größer als die Kontrollspannung, ist das Werkstück fehlerfrei, ist sie kleiner handelt es sich um eine fehlerhafte Platine. Das Ergebnis der elektrischen Endkontrolle wird an Station 10 übertragen, dort auf den RFID-Chip geschrieben und der Transportwagen entspannt und über den Vereinzeler freigegeben.

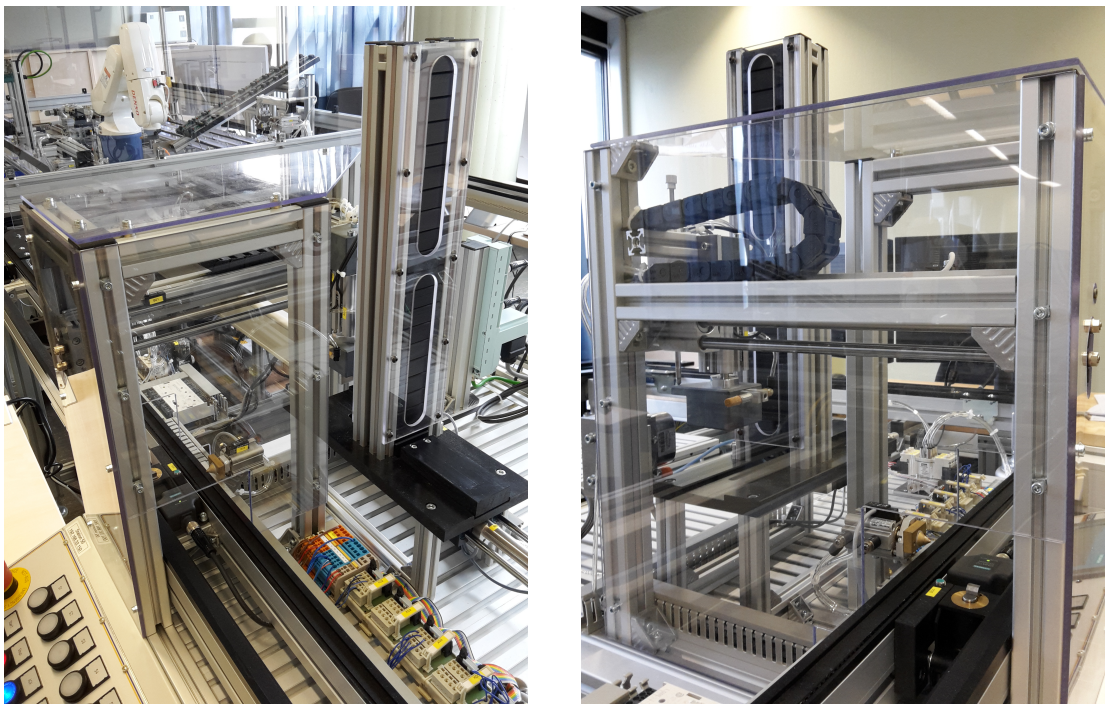
3.1.5. Station 50: Pneumatische Presse

Abbildung 3.6 zeigt die Station der pneumatischen Presse (Station 50). In der Ansicht von rechts ist auf der rechten Seite des Bildes das Magazin mit Deckeln dargestellt. Der unterste

Abbildung 3.5.: Station 40: elektrische Endkontrolle

Deckel kann mit einem Ausschieber aus dem Magazin heraus geschoben werden. In der Ansicht von links ist das Handling mit Unterdrucksauger in seiner Ruheposition dargestellt. Es saugt den Deckel an, befördert ihn zum Werkstück und presst ihn dort auf.

Wie bei Station 40 fährt das Werkstück auf dem Transportwagen in die Station hinein, wird angehalten und eingespannt. Über Station 10 wird der RFID-Chip des Transportwagens ausgelesen und der Wert an Station 50 übertragen. Befindet sich auf dem Transportwagen ein fehlerhaftes Werkstück, wird der Wagen wieder entspannt, über den Vereinzeler freigegeben und fährt zur nächsten Station. Ist das Werkstück auf dem Transportwagen fehlerfrei, wird über den Ausschieber ein Deckel aus dem Magazin heraus geschoben. Sobald der Ausschieber wieder eingefahren ist, rutschen aus dem Magazin automatisch Deckel nach. Das Handling fährt herunter zum ausgeschobenen Deckel und saugt diesen an. Mit dem angesaugten Deckel fährt es nach oben und zum Werkstück. Dort presst es den Deckel auf das Werkstück, schaltet seinen Sauger aus und fährt wieder in seine Ruheposition. Anschließend wird der Wert, der auf den RFID-Chip geschrieben werden soll, an Station 10 über-



(a) Ansicht von rechts (b) Ansicht von links

Abbildung 3.6.: Station 50: pneumatische Presse

tragen und dort geschrieben. Der Transportwagen wird entspannt und über den Vereinzeler freigegeben.

3.1.6. Station 60: Bildverarbeitung

Abbildung 3.7 zeigt die Bildverarbeitungsstation (Station 60). Auf der rechten Seite ist das Handling in seiner Ruheposition dargestellt. Es hebt die Werkstücke von den Transportwagen des Transportsystems herunter und legt sie auf dem Transportband von Station 60 ab. Zwei Abschieber am Transportband können Werkstücke in die blauen Kisten abschieben. Auf der linken Seite des Bildes befindet sich über dem Transportband eine Kamera, die Bilder von den fertigen Werkstücken aufnehmen und auswerten kann. Sensoren am Bandanfang, -ende und an den Abschiebern ermöglichen eine Steuerung der Station.

Sobald ein Werkstück auf einem Transportwagen an Station 60 angekommen ist, wird über Station 10 der Status auf dem RFID-Chip des Transportwagens ausgelesen. Das Handling nimmt das Werkstück vom Transportwagen herunter und legt es auf das Transportband. Sobald es sich wieder in Ruheposition befindet, startet das Transportband. Ist das Werkstück

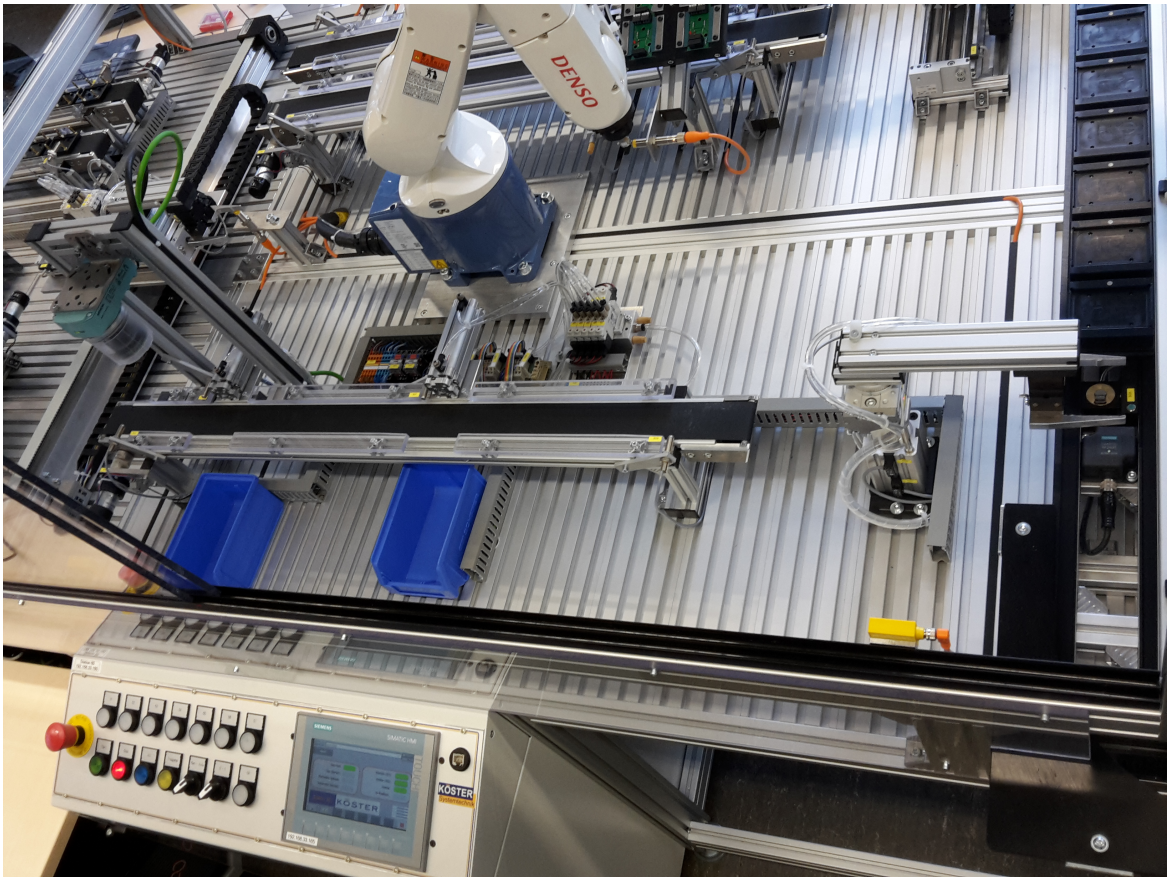


Abbildung 3.7.: Station 60: Bildverarbeitung

am ersten Abschieber angekommen und handelt es sich um ein fehlerhaftes Werkstück, wird das Transportband gestoppt und das Werkstück über den Abschieber abgeschoben. Handelt es sich um ein fehlerfreies Werkstück, fährt es am Abschieber vorbei. Das Transportband stoppt nicht. Erreicht das Werkstück den zweiten Abschieber unter der Kamera, wird die Kamera angetriggert und nimmt ein Bild von dem Werkstück auf. Eine Bildverarbeitung in der Kamera wertet aus, ob der Deckel richtig herum ausgerichtet ist, indem es das aufgenommene mit gespeicherten Bildern vergleicht. Ist der Deckel fehlerhaft verbaut, wird das Transportband gestoppt und das Werkstück über den zweiten Abschieber in die Kiste abgeschoben. Fehlerfreie Endprodukte werden am Ende des Transportbandes von der Linearachse von Station 20 abgeholt. Dazu wird von Station 60 über Station 10 ein Bit an Station 20 übertragen, welches für drei Sekunden anzeigt, dass ein Produkt zur Abholung bereit liegt.
Nach dem Abschieben eines Werkstücks oder dem Setzen des Bits zur Abholung wird der

RFID-Chip auf dem Transportwagen von Station 10 mit null überschrieben. Der Wagen wird über den Vereinzeler freigegeben und das nächste Werkstück kann bearbeitet werden.

3.2. Automatisierung

Die Automatisierung je Stationen erfolgt über eine speicherprogrammierbare Steuerung (SPS; engl.: Programmable Logic Controller, PLC) SIMATIC S7 1500 der Siemens AG mit analogen und digitalen Ein- und Ausgabebaugruppen. Abbildung 3.8 zeigt eine Übersicht der Verschaltung der einzelnen Komponenten. Die SPS und die Touchpanel der sechs Stationen sowie weitere Komponenten der Stationen, wie die RFID-Schreib-/Lesegeräte, die Kamera und die beiden Umrichter für die Linearachse bzw. die Transferbänder des Transportsystems, sind über Profinet (Process Field Network) miteinander verbunden. Über ein IO-Device sind die Vereinzeler und Fixiervorrichtungen des Transportsystems an einem Aktor-Sensor-Interface Bus (AS-i, engl.: Actuator-Sensor-Interface) verbunden.

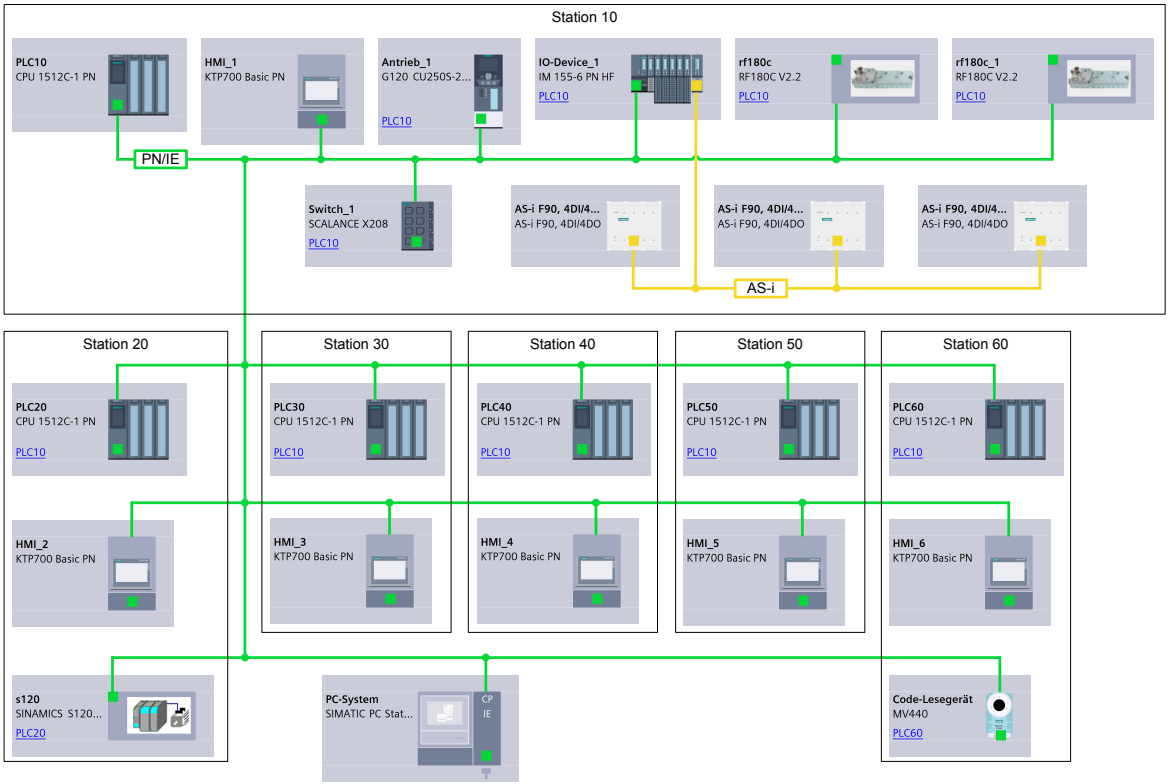


Abbildung 3.8.: Verschaltungsübersicht[4]

[4] Die grau hinterlegten Zeichnungen sind aus der Geräteansicht des TIA-Portals entnommen.

Die Programmierung der SPS erfolgt über das TIA-Portal (Totally Integrated Automation) der Siemens AG, welches unter anderem die Software SIMATIC STEP 7 und SIMATIC WinCC (Windows Control Center) miteinander verknüpft. Das TIA-Gesamtprojekt für die Modellfabrik umfasst unter anderem die Steuerung und Zusammenschaltung der einzelnen Stationen sowie der externen Komponenten, wie den Touchpaneln, der Kamera oder den RFID-Sensoren. Die Programmierung im TIA-Portal kann entweder in Funktionsbausteinsprache (FBS), Anweisungsliste (AWL), Kontaktplan (KOP) oder strukturiertem Text (SCL) erfolgen.
Die sechs SPS der einzelnen Stationen kommunizieren über interne Ein- und Ausgänge miteinander. Im TIA-Gesamtprojekt der Modellfabrik ist festgelegt, dass die SPS der Stationen 20 bis 60 jeweils über vier Ein- und Ausgangsbytes mit der SPS von Station 10 verbunden sind. Dabei sind die Eingangsbytes der Station 10 mit den Ausgangsbytes der Stationen 20 bis 60 verbunden und die Ausgangsbytes der Station 10 mit den Eingangsbytes der Stationen 20 bis 60. Im ersten der vier Bytes werden Statusmeldungen der SPS übertragen und empfangen. Ein Byte dient zum Übertragen und Empfangen der Stationsmeldungen, wie zum Beispiel der Meldung, dass ein Roboterband bereit für neue Werkstücke ist oder ein Werkstück zur Abholung an Station 60 bereit liegt. Im dritten und vierten Byte werden Informationen übertragen und empfangen, die beispielsweise auf den RFID-Chip geschrieben werden sollen, vom RFID-Chip gelesen wurden oder an den Produktzähler von Station 20 übergeben werden sollen.

3.3. Optimierung der Modellfabrik

Der in diesem Abschnitt beschriebene Ausgangszustand der Modellfabrik beschreibt den Auslieferungszustand, der vertraglich zwischen der Hochschule für angewandte Wissenschaften Hamburg (HAW) und der Firma Köster Systemtechnik GmbH vereinbart wurde. Inhalt des Vertrages ist unter anderem, dass die Software der Modellfabrik lediglich die grundlegende Funktion der Stationen zeigt. Aus diesem Grund sind einige Funktionen nicht optimal umgesetzt. Eine Optimierung soll durch die Studierenden der HAW erfolgen. Alle vertraglichen Vereinbarungen wurden bei Auslieferung erfüllt.
In Abschnitt 3.3.1 bis 3.3.6 werden verschiedene Teile der Modellfabrik, bei denen es Optimierungsbedarf gibt, genauer erläutert und die Optimierung beschrieben und analysiert. Weitere Veränderungen, die umgesetzt werden können, werden in Kapitel 8 erläutert.

3.3.1. Parallele Prozesse an Station 30

Ausgangszustand
Die Modellfabrik ist so programmiert, dass die Funktion der Stationen 20 bis 60 in einer Schrittkette nacheinander abgearbeitet wird. Dies führt dazu, dass an jeder Station nur ein

Werkstück zurzeit bearbeitet werden kann. Bei Station 20 ist die Linearachse eine so genannte „shared resource“. Das bedeutet, dass diese Ressource für mehrere Aufträge genutzt wird. In diesem Fall ist sie für alle Transportaufträge erforderlich, sodass keine parallelen Prozesse möglich sind. Bei den Stationen 40 und 50 kann aufgrund ihres Aufbaus nur ein Werkstück zurzeit bearbeitet werden.
Optimierungsbedarf besteht besonders bei der Station 30. Die Linearachse kann zwar auf beide Roboterbänder jeweils ein Werkstück legen, durch die Schrittkette wird das Werkstück auf dem zweiten Transportband aber erst mit einer Platine bestückt, wenn das Handling das Werkstück von dem ersten Band auf dem Transportwagen abgelegt hat und wieder in seiner Ruheposition angekommen ist. Während dieser Zeit könnte das Werkstück auf dem zweiten Band schon mit einer Platine bestückt werden. Weiterer Optimierungsbedarf besteht bei der Ablage der Werkstücke auf den Transportwagen. Ist das Transportband nicht eingeschaltet, würde der Transportwagen nach der Freigabe von Station 10 an Station 30 stehenbleiben und das Handling das nächste Werkstücke auf einen belegten Transportwagen pressen. Dieser Vorgang kann zur Beschädigung des Handlings führen.

Maßnahmen
Um parallele Prozesse an Station 30 zu ermöglichen, muss die Schrittkette dupliziert werden, sodass für jedes Transportband eine Schrittkette existiert. Die beiden „shared resources“, der Roboter und das Handling, dürfen nur von einer Schrittkette gleichzeitig genutzt werden. Aus diesem Grund wird für beide Ressourcen jeweils eine Schlüsselmarke in Form einer Variablen eingefügt. Vor der Verwendung der Ressource wird die Variable abgefragt. Ist sie gesetzt, wird die Ressource gerade verwendet und die Schrittkette wartet auf Freigabe. Ist sie nicht gesetzt, wird die Variable gesetzt und die jeweilige Schrittkette kann die Ressource verwenden. Nach der Verwendung wird die Variable wieder zurückgesetzt. Für den Bereich des Handlings wird zusätzlich abgefragt, ob das Transportsystem an Station 10 eingeschaltet ist. Ist das Transportsystem ausgeschaltet, warten die Werkstücke nach der Bestückung mit einer Platine nicht nur auf die Freigabe der Schlüsselmarke für das Handling sondern auch auf das Einschalten des Transportbandes.

Ergebnis
Durch die Duplizierung der Schrittkette und das Einfügen von Schlüsselmarken kann der Roboter auf dem zweiten Transportband bereits eine Platine einsetzen, während das Handling noch ein Werkstück auf dem ersten Band bearbeitet. Die Werkstücke werden in kürzerer Zeit auf die Transportwagen gelegt. Für spätere Erweiterungen, bei denen der Roboter benötigt wird, kann jederzeit auf den Roboter zugegriffen werden, wenn dieser gerade nicht verwendet wird. Beispielsweise muss beim Entfernen fehlerhafter Platinen an Station 60 nicht auf die Freigabe der Bänder gewartet werden, obwohl für diesen Auftrag kein Transportband benötigt wird. Durch die Abfrage, ob das Transportband eingeschaltet ist, wird sichergestellt,

dass der Transportwagen nach der Bestückung die Station 30 verlässt und es zu keiner Beschädigung kommt.

3.3.2. Abholen der Werkstücke von Station 60 durch die Linearachse

Ausgangszustand
Die Befehlsübergabe, dass die Linearachse von Station 20 ein Werkstück von Station 60 abholen soll, erfolgt über ein Bit, welches von Station 60 über Station 10 an Station 20 gesendet wird. Ist die Linearachse gerade in ihrer Ruheposition, wird das Werkstück sofort abgeholt. Arbeitet sie gerade einen anderen Auftrag ab, wird die Abholanforderung von Station 20 gespeichert und nach dem aktuellen Auftrag abgearbeitet. An Station 60 wird das Bit nach drei Sekunden automatisch zurückgesetzt. Ist das Werkstück nach den drei Sekunden noch nicht abgeholt worden und bearbeitet Station 60 ein neues Werkstück, wird das Transportband gestartet und das Werkstück, was zur Abholung bereit liegt, fällt vom Band. Im schlimmsten Fall könnte sich das Werkstück unter der Linearachse verkeilen, wenn diese das Werkstück abholen will. Um das heruntergefallene Werkstück aus der Modellfabrik zu entfernen, muss die Schiebetür des Schutzkäfigs geöffnet werden. Befinden sich der Roboter oder die Linearachse nicht in ihrer Ruheposition oder wollen diese während der Öffnungszeit der Tür verlassen, geht die Modellfabrik in den Zustand Not-Halt und die Produktion von Werkstücken innerhalb des Schutzkäfigs wird unterbrochen.

Maßnahmen
Damit die Werkstücke nicht vom Transportband fallen, muss das Bit an Station 60 so lange gesetzt bleiben, bis die Linearachse von Station 20 das Werkstück abgeholt hat. Solange das Bit gesetzt ist, muss der Start des Transportbandes verhindert werden. Das Rücksetzen des Bits erfolgt von Station 20 über Station 10 an Station 60. Es wird zurückgesetzt, nachdem die Linearachse an Station 60 angekommen ist, den Greifer geschlossen hat und das Werkstück vom Band genommen hat, also der Sensor „B07“ („Linearachse an Station 20“) aktiv ist. Ein Rücksetzen erfolgt auch, wenn der Automatikbetrieb von Station 60 ausgeschaltet wird. Solange das Bit gesetzt ist, befindet sich das Transportband im Status „Stop“. Startanforderungen werden erst ausgeführt, wenn der Befehl „Stop“ nicht mehr ansteht. Zusätzlich zu dem Bit, welches angibt, dass ein Werkstück zur Abholung bereit liegt, gibt es zwei weitere Bits, die von Station 60 über Station 10 an Station 20 melden, ob das Werkstück, was zur Abholung bereit liegt, ein- oder zweipolig ist. Diese Information wird zum einen für den Produktzähler verwendet (siehe Abschnitt 3.3.6) und zum anderen ermöglicht es ein späteres Sortieren der Werkstücke auf den Auswurfbändern. Die beiden Bits an Station 60, die die Art des Werkstücks melden, werden wie das Bit, das meldet, ob ein Werkstück an Station 60 zur Abholung bereit liegt, zurückgesetzt, wenn die Linearachse das Werkstück abgeholt hat.

Ergebnis
Nach der Umsetzung aller beschriebenen Maßnahmen fällt kein Werkstück mehr vom Transportband. Die Gefahr des Verkeilens unter der Linearachse ist damit ebenfalls nicht mehr vorhanden. Liegt ein Werkstück an Station 60 zur Abholung bereit und bearbeitet die Station ein neues Werkstück, wird das Transportband erst nach der Abholung gestartet.

3.3.3. Optimierung von Station 60

Ausgangszustand
An Station 60 wird das Transportband erst gestartet, wenn das Handling dort ein Werkstück abgelegt hat und wieder an seiner Ruheposition angekommen ist. Das Handling braucht für die Bewegung vom Transportband zurück in seine Ruheposition circa sieben Sekunden. Für die Werkstücke in der Produktion entsteht eine vermeidbare Wartezeit, die die Produktionsdauer verlängert. Startet das Transportband sofort nach Ablage des Werkstücks, würde die Produktionsdauer reduziert werden.

Maßnahmen
Das Transportband soll direkt nach Ablegen eines Werkstücks gestartet werden. Dazu wird der Sensor am Bandanfang abgefragt. Ist er aktiv und liegt kein Werkstück zur Abholung bereit, startet das Transportband. Parallel dazu bewegt sich das Handling zurück in seine Ruheposition am Transportsystem. In der Schrittkette von Station 60 wird die Startbedingung des zweiten Schritts von „Handling zurück am Transportsystem“ geändert in „Sensor am Bandanfang aktiv“. Damit ein neues Werkstück erst dann bearbeitet werden kann, wenn das Handling frei ist, wird die Startbedingung von Schritt eins um diesen Befehl ergänzt.

Ergebnis
Für den Fall, dass ein Werkstück an Station 60 zur Abholung bereit liegt und das Transportband nicht sofort starten kann, bleibt die Produktionszeit eines Werkstücks unverändert. In allen anderen Fällen startet das Transportband sofort und das Werkstück befindet sich zwischen Abschieber eins und zwei, wenn das Handling seine Ruheposition erreicht hat. Die vermeidbare Wartezeit wird eliminiert und die Produktionsdauer der Werkstücke verringert sich durch die getroffenen Maßnahmen im besten Fall um circa sieben Sekunden pro Werkstück.

3.3.4. Einbindung der Kamera

Ausgangszustand
Ein lauffähiges Programm für die Kamera ist nicht Bestandteil des Auslieferungszustandes

der Modellfabrik. Aus diesem Grund ist nur die Anbindung und nicht die Entscheidung der Kamera in der Software implementiert. Die Erstellung eines Programms für die Kamera erfolgt über einen Webserver. Dazu wird ein Computer, der mit der Kamera verbunden ist und über einen Internet-Browser verfügt, benötigt. Ohne die Einbindung der Kamera ist dieser Teil der Modellfabrik außer Funktion.

Maßnahmen
Da die Einbindung der Kamera eine Einarbeitung in den Webserver erfordert, soll lediglich eine Alternative zur Kamera Entscheidung umgesetzt werden. Die Umsetzung erfolgt über zwei Taster am Bedienpult von Station 60. Zur Umsetzung wird der vorbereitete Baustein für die Einbindung der Kamera dupliziert, der originale Baustein über den Enable-Eingang deaktiviert und der kopierte abgeändert. Die Ein- und Ausgänge der Bausteine sind bis auf zwei extra Ausgänge beim kopierten Baustein identisch. Diese Vorgehensweise vereinfacht ein späteres Einbinden der realen Kamera.
In dem kopierten Baustein wird eine Schrittkette implementiert. Sobald der Sensor unterhalb der Kamera aktiv ist, fangen die Taster „S2“ und „S3“ an zu blinken. Der Bediener der Anlage hat von diesem Zeitpunkt an drei Sekunden Zeit, einen von beiden zu betätigen. „S2“ bedeutet, dass das Werkstück fehlerfrei ist, und „S3“, dass das Werkstück fehlerhaft ist und abgeschoben werden muss. Nach Ablauf der drei Sekunden wird das Werkstück automatisch als fehlerfrei befunden. Der Timer von drei Sekunden dient dazu, den Ablauf für den Bediener zu vereinfachen. Ohne Timer müsste der Bediener für jedes Werkstück vorgeben, ob es fehlerfrei oder fehlerhaft ist. Vergisst der Bediener, eine Taste zu drücken, stauen sich die Werkstücke vor Station 60. Bei der Implementierung mit Timer gibt es keinen Rückstau. Der Bediener hat aber weiterhin die Möglichkeit, Werkstücke als fehlerhaft zu markieren, oder die Entscheidung durch Drücken von „S2“ zu beschleunigen.

Ergebnis
Nach der Umsetzung der Kamera Alternative gibt es für den Bediener eine Möglichkeit, selbst zu entscheiden, ob das Werkstück fehlerfrei oder fehlerhaft ist. Mit dieser Erweiterung kann der zweite Abschieber in Betrieb genommen werden. Werkstücke mit fehlerhaft verbautem Deckel werden weiterhin nicht automatisch erkannt. Sobald ein Programm für die reale Kamera geschrieben wird, ist das Einbinden vereinfacht, da der Ersatzbaustein fast dieselben Ein- und Ausgänge besitzt wie der originale Baustein und somit einfach ausgetauscht werden kann.

3.3.5. Verdeckelung Station 50

Ausgangszustand
Die Deckel, die an Station 50 auf das Werkstück gepresst werden, bestehen aus dem ei-

gentlichen Deckel und einem aufgeklebten Logo. Durch das Ansaugen der Deckel kann es vorkommen, dass sich der Aufkleber dabei vom Deckel löst und nur der Aufkleber auf dem Werkstück verpresst wird. Die Deckel ohne Aufkleber bleiben entweder vor dem Ausschieber an Station 50 liegen oder fallen aus geringer Höhe dorthin zurück. Mit dem Ausschieben des nächsten Deckels fällt der Deckel ohne Aufkleber unter die Station 50. Eine Beschädigung der Station ist sehr unwahrscheinlich, da sich keine Komponenten, die beschädigt werden könnten, unterhalb der Station befinden. Werkstücke mit Aufkleber können von dem Sensor am Bandende von Station 60 nicht erkannt werden. Selbst wenn die Kamera eingebunden wäre, könnte es passieren, dass diese ein fehlerfreies Werkstück erkennt, da das Logo erkannt wird. Werkstücke mit Aufkleber fallen am Bandende von Station 60 vom Transportband und könnten sich unter der Linearachse verkeilen.

Maßnahmen
Um zu ermitteln, ob der Deckel oder nur der Aufkleber verpresst wurde, wird die Zeit ermittelt, die die Presse für das Verpressen des Deckels benötigt. Ein Deckel benötigt mindestens 1500 Millisekunden (minimaler Wert bei 18 Messungen), ein Aufkleber nur maximal 1200 Millisekunden (maximaler Wert bei sechs Messungen). Der Schwellwert für die Erkennung wird bei 1300 Millisekunden gesetzt. Ist die gemessene Zeit kleiner als 1300 Millisekunden, geht die Steuerung davon aus, dass nur ein Aufkleber verpresst wurde. Diese Information wird auf dem RFID-Chip des Transportwagens geschrieben und an Station 60 ausgelesen. Das Werkstück wird am zweiten Abschieber abgeschoben. Gleichzeitig wird hier die Information an Station 20 übermittelt, dass ein fehlerhaftes Werkstück erkannt wurde, sodass ein neues produziert werden kann.

Ergebnis
Wird ein Aufkleber auf ein Werkstück verpresst, wird dieses Werkstück am zweiten Abschieber der Station 60 abgeschoben und automatisch ein neues Werkstück produziert. Ein fehlerhaftes Werkstück mit Aufkleber ist gegenüber einem Werkstück mit fehlerhafter Platine irreparabel, da keine Komponente innerhalb der Modellfabrik dieses Werkstück demontieren kann.

3.3.6. Produktzähler

Ausgangszustand
In diesem Abschnitt wird die Problematik der Produktzähler anhand der Zähler für einpolige Werkstücke dargestellt. Analog dazu gibt es jeden Zähler für zweipolige Werkstücke. Bei den Produktzählern gibt es für drei Anwendungsfälle Optimierungsbedarf. Im ersten Fall geht es um den Istwertzähler, der angibt, wie viele einpolige Werkstücke noch produziert werden müssen, um den Sollwert zu erreichen. Die Problematik dabei ist, dass der Bediener

der Modellfabrik im Normalfall die Anzahl der fertigen Produkte wissen möchte. Die zweite Problematik gilt, wenn ein fehlerhaftes Werkstück produziert wird. In diesem Fall stimmt die Anzahl der gewünschten nicht mit den real produzierten Produkten überein, da eine automatische Neuproduktion nicht stattfindet. Die dritte Problematik besteht bei der Beauftragung von neuen Produkten. Diese werden am Touchpanel von Station 20 vorgegeben. Läuft die Produktion bereits, ist eine Erweiterung des Sollwertes nicht möglich. Für neu zu produzierende Werkstücke muss immer zunächst die gesamte Produktion gestoppt und anschließend wieder gestartet werden. Produkte, die sich bereits in der Produktion befinden, werden dann in der Zählung nicht weiter berücksichtigt.

Maßnahmen

Um die vorhandenen Produktzähler zu optimieren ist es notwendig, die bestehenden Zähler zu erweitern. Neben dem bereits vorhandenen Soll- und Istwertzähler wird ein Auftragszähler eingeführt, der die gesamte Funktion des Istwertzählers übernimmt. Der Begriff des Istwertzählers ist durch die Einführung des Auftragszählers frei für das Zählen der fertigen Produkte. Der neue Istwertzähler wird jedes Mal um eins inkrementiert, wenn einer der Sensoren am Anfang der Auswurfbänder aktiv wird und Station 60 meldet, dass das Werkstück einpolig ist. Der aktuelle Wert des Istwertzählers kann auf dem Touchpanel von Station 20 abgelesen werden und wird zurückgesetzt, wenn der Automatikbetrieb deaktiviert oder im Automatikbetrieb der Tasten „S6“ am Bedienpult von Station 20 gedrückt wird.
Die zweite Problematik, dass die Anzahl der gewünschten teilweise nicht mit den fertigen Produkten übereinstimmt, wird gelöst, indem Station 40 und 60 durch jeweils drei Bits melden, dass ein Werkstück fehlerhaft ist. Das erste Bit wird für die Information, dass das Werkstück fehlerhaft ist, verwendet. Das zweite und dritte Bit geben an, ob es sich um ein ein- oder zweipoliges fehlerhaftes Werkstück handelt. Handelt es sich um ein einpoliges Werkstück, wird der Auftragszähler um eins inkrementiert und es wird automatisch ein weiteres einpoliges Werkstück produziert.
Für die Übertragung der Stationsmeldungen zwischen den SPS steht, wie in Abschnitt 3.2 beschrieben, ein Byte, also acht Bits, zur Verfügung. Durch die bestehende und erweiterte Kommunikation von Station 30, 40 und 60 werden an Station 20 mindestens elf Eingangsbits benötigt (davon zwei für Station 30, drei für Station 40 und sechs für Station 60). Aus diesem Grund werden die zu übertragenden Ein- und Ausgangsbytes zwischen Station 10 und 20 von vier auf fünf erweitert, sodass dort zwei Bytes für die Stationsmeldungen zur Verfügung stehen. Das erste Byte wird für Meldungen zwischen Station 20, 30 und 40 verwendet, das zweite für Meldungen zwischen Station 20 und 60.
Für die dritte Problematik, dass eine Erweiterung der bestehenden Aufträge nicht möglich ist, müssen die Sollwert- und Auftragszähler verändert werden. Der Istwertzähler ist von dieser Maßnahme nicht betroffen, da er keinen Einfluss auf die neu zu produzierenden Produkte hat. Beim Ausschalten der Automatikfunktion sollen auch nach der Erweiterung alle Produktzähler auf null gesetzt werden. Aus diesem Grund wird der Taster „S5“ am Bedien-

pult von Station 20 verwendet, um neue Aufträge zu starten. Beim Drücken des Tasters wird der aktuelle Stand des Auftragszählers mit den neu beauftragten einpoligen Werkstücken addiert und dieser Wert in den Sollwertzähler geschrieben. Im selben Programmzyklus wird der Auftragszähler mit diesem Wert neu geladen.

Ergebnis

Nach der Implementierung aller beschriebenen Maßnahmen gibt es drei Zähler. Über den Istwertzähler hat der Bediener die Information, wie viele Produkte bereits produziert wurden. Die ehemalige Funktion des Istwertzählers wird durch den Auftragszähler übernommen. Er gibt an, wie viele Produkte die Linearachse noch zu Station 30 bringen muss, um den Sollwert zu erreichen. Der Sollwertzähler gibt die Anzahl der Werkstücke an, die nach dem Starten eines Auftrages produziert werden sollen. Wird ein Fehlteil von Station 40 oder 60 erkannt, wird automatisch ein neues Produkt produziert, sodass es zu keinen Abweichungen zwischen gewünschten und real produzierten Produkten kommt. Neue Aufträge können im Bedienpanel von Station 20 vorgegeben und mit dem Taster „S5“ beauftragt werden. Die Produktion muss nicht mehr gestoppt werden.

4. Konzeption

In diesem Kapitel werden verschiedene Entwicklungstools (siehe Abschnitt 4.1), Beschreibungsformen (siehe Abschnitt 4.2) und Steuerungsansätze (siehe Abschnitt 4.3) miteinander verglichen. In einer Tabelle werden verschiedene Eigenschaften und Kriterien aufgelistet und bewertet. Die Eigenschaften und Kriterien werden in der Spalte „Relevanz“ von eins „nicht relevant“ bis fünf „sehr relevant“ gewichtet. Im rechten Teil der Tabelle erfolgt eine Bewertung der Eigenschaften und Kriterien in Bezug auf das Entwicklungstool, die Beschreibungsform oder den Steuerungsansatz. Die Bewertung erfolgt von eins „trifft nicht zu“ bis fünf „trifft zu“. Ist die Eigenschaft oder das Kriterium nicht relevant, wird sie mit fünf bewertet. Anschließend werden die Bewertungen mit der Relevanz multipliziert und diese Produkte aller Eigenschaften und Kriterien summiert. Anhand des Ergebnisses erfolgt eine Entscheidung für ein Entwicklungstools, eine Beschreibungsform und einen Steuerungsansatz.

4.1. Auswahl des Entwicklungstools

Für die Erstellung und Synthese der Modelle gibt es verschiedene Entwicklungstools. Im Folgenden werden die Tools „DESTool“, „TCT“ und „SPNBOX“ miteinander verglichen und bewertet. „DESTool“ wird in Abschnitt 2.9.1 vorgestellt. „TCT“ wurde von Karen Rudie, Pablo Iglesias, Jimmy Wong und Pok Lee entwickelt und wird von W. M. Wonham weiter gepflegt. Es ist ein Tool zur Supervisor-Synthese von Systemen. Als Darstellungsform werden endliche Automaten verwendet. Das Tool bietet eine umfangreiche Funktionalität, enthält dafür aber keine grafische Oberfläche. [Feng und Wonham (2006)]
„SPNBOX“ ist eine Matlab-Toolbox zur Supervisor-Synthese. Als Darstellungsform werden Petrinetze verwendet. Zur Ausführung wird entweder Matlab, Octave, FreeMat oder SciLab benötigt. Die Petrinetze werden als Matrizen in das Programm eingegeben. [Iordache (2013)]

Tabelle 4.1 zeigt verschiedene Eigenschaften und Kriterien für die Auswahl des Entwicklungstools. Die Eigenschaften „Funktionalität“ und „Codegenerator vorhanden“ werden mit fünf gewichtet, da das Tool ohne die gewünschte Funktionalität nicht infrage kommt. Ohne

Codegenerator vergrößert sich der Aufwand bei der Implementierung, da zunächst ein Codegenerator entwickelt werden muss. Die Eigenschaften „Dokumentation“ und „keine Erweiterungen notwendig“ werden mit vier gewichtet, da eine gute Dokumentation die Bedienung erleichtert und notwendige Erweiterungen die Implementierung erschweren. Mit drei werden die Eigenschaften „einfache Installation“ und „kostenloses Tool“ bewertet. Eine einfache Installation ist einmalig und daher nicht so wichtig wie die Bedienung des Tools. Die Kosten sind mäßig wichtig, da die Tools an Universitäten oder Hochschulen meist kostenlos zur Verfügung stehen. Für Privatanwendungen sind sie dafür eher wichtig. Die Benutzeroberfläche und die Bedienbarkeit des Tools sind eher unwichtig, da der geübte Anwender Erfahrung hat, das Tool zu bedienen.

Eigenschaften und Kriterien	**Relevanz**	**„DESTool“**	**„TCT“**	**„SPNBOX“**
Funktionalität	5	5 (++)	5 (++)	5 (++)
Benutzeroberfläche	2	5 (++)	2 (-)	2 (-)
Bedienbarkeit	2	4 (+)	2 (-)	3 (o)
Dokumentation	4	5 (++)	4 (+)	2 (-)
Einfache Installation	3	4 (+)	3 (o)	3 (o)
Keine Erweiterungen notwendig	4	4 (+)	5 (++)	2 (-)
Kostenloses Tool	3	5 (++)	5 (++)	3 (o)
Codegenerator vorhanden	5	5 (++)	1 (- -)	5 (++)
		131	**94**	**98**

Tabelle 4.1.: Auswahl des Entwicklungstools

Die Funktionalität ist bei allen drei Tools gegeben. Die Benutzeroberfläche von „DESTool“ ist grafisch und sehr übersichtlich. Bei „TCT“ und „SPNBOX“ erfolgt die Eingabe in einem Konsolenfenster, was die Übersichtlichkeit negativ beeinflusst. Die Bedienbarkeit ist bei „DESTool“ am besten, da die Automaten grafisch dargestellt und nachträglich bearbeitet werden können. Da das Tool bei der Ausführung einiger Funktionen abstürzen kann, wird es nur mit vier statt mit fünf bewertet. Bei „TCT“ und „SPNBOX“ ist die Bedienbarkeit nicht sehr gut, da die Automaten oder Petrinetze nicht grafisch dargestellt werden und es schnell zu Fehlern führen kann. Bei Fehlern müssen die Automaten oder Petrinetze erneut eingegeben werden. Die Darstellung von „SPNBOX“ ist ein wenig besser als bei „TCT“, da die Matrizen nach der Eingabe angezeigt werden. Die Dokumentation ist bei „DESTool“ und „TCT“ sehr gut, allerdings ist die Dokumentation bei „DESTool“ besser, da sie öffentlich zur Verfügung steht. Die Dokumentation bei „SPNBOX“ ist schwer zu finden und kann erst nach der Installation eingesehen werden. Die Installation bei „DESTool“ ist einfach, da es direkt von der Homepage heruntergeladen werden kann. Eine Erweiterung, die etwas kompliziert bei der Installation

ist, ist optional für die grafische Anordnung der Zustände der Automaten. „TCT“ kann von der Homepage von W. M. Wonham [Wonham (2016)] heruntergeladen werden. Es erfordert allerdings einen Login, der angefragt werden muss. Erweiterungen sind keine notwendig. Bei „SPNBOX“ muss zunächst Matlab installiert werden, was sehr lange dauert. Anschließend müssen Erweiterungen getätigt werden, um das Tool nutzen zu können. „DESTool“ und „TCT“ sind kostenlose Tools, für „SPNBOX“ wird Matlab benötigt, wofür hohe Lizenzgebühren anfallen. Es gibt kostenlose Alternativen, für die aber ebenfalls Erweiterungen geschrieben werden müssen. Ein Codegenerator ist für „DESTool“ und „SPNBOX“ verfügbar (siehe Abschnitt 2.9.2), für das Tool „TCT“ muss ein Codegenerator entwickelt werden.

Nach der Auswertung der Eigenschaften und Kriterien ist „DESTool“ am besten geeignet. Die Nutzung des Tools erfordert allerdings Automaten als Beschreibungsform. Sollen Petrinetze verwendet werden, muss auf „SPNBOX“ ausgewichen werden.

4.2. Auswahl der Beschreibungsform

Als Beschreibungsform für die Strecken- und Spezifikationsmodelle können entweder Automaten oder Petrinetze verwendet werden. Tabelle 4.2 listet verschiedene Eigenschaften und Kriterien auf, die bei der Entscheidung für eine Beschreibungsform wichtig sind. Mit einer fünf werden die Eigenschaften „einfache Darstellung sequenzieller Prozesse“, „Synthesetool vorhanden“, „Codegenerator vorhanden“ und „Anwendbarkeit auf „DESTool““ gewichtet. Sequenzielle Prozesse werden innerhalb der Modellfabrik sehr häufig verwendet. Aus diesem Grund muss auch die Beschreibungsform für diese geeignet sein. Ein Synthesetool muss vorhanden sein, damit der Steuerungsentwurf mit diesem vorgenommen werden kann. Der vorhandene Codegenerator verringert den Aufwand bei der Implementierung, da dieser nicht erst entwickelt werden muss. In Abschnitt 4.1 bietet „DESTool“ die meisten Vorteile gegenüber den anderen Entwicklungstools. Daher wird auch die Anwendbarkeit auf dieses Tool mit „sehr relevant“ gewichtet. Die Kosten für das Synthesetool werden mit drei gewichtet, da es lediglich für private Anwendung relevant ist und im universitären Bereich die Tools meist kostenlos zur Verfügung stehen. Die Bedienbarkeit des Synthesetools wird mit zwei gewichtet, da dieses Kriterium nach der Einarbeitung kaum noch wichtig ist. Auch die einfache Darstellung paralleler Prozesse ist eher nicht relevant, da keine bis wenige Prozesse an der Modellfabrik parallel ausgeführt werden müssen. In anderen Anlagen kann dieser Punkt allerdings ausschlaggebend für die Wahl der Beschreibungsform sein.

Sowohl bei Automaten als auch bei Petrinetzen können sequenzielle Prozesse sehr gut dargestellt werden. Für parallele Prozesse eignen sich allerdings Petrinetze besser. Mit Automaten können diese schlecht bis gar nicht dargestellt werden. Ein Synthesetool sowie ein Codegenerator sind für beide Beschreibungsformen vorhanden. Die Bedienbarkeit der Synthesetools für Automaten ist allerdings ein wenig besser als bei denen für Petrinetze (siehe

Eigenschaften und Kriterien	Relevanz	Automaten	Petrinetze
Einfache Darstellung sequenzieller Prozesse	5	5 (++)	5 (++)
Einfache Darstellung paralleler Prozesse	2	2 (-)	5 (++)
Synthesetool vorhanden	5	5 (++)	5 (++)
Codegenerator vorhanden	5	5 (++)	5 (++)
Bedienbarkeit der Synthesetools	2	4 (+)	3 (o)
Kostenloses Synthesetool	3	5 (++)	3 (o)
Anwendbarkeit auf „DESTool“	5	5 (++)	1 (- -)
		127	**105**

Tabelle 4.2.: Auswahl der Beschreibungsform

Abschnitt 4.1). Auch stehen diese kostenlos zur Verfügung. Die Anwendbarkeit auf „DESTool“ ist nur bei Automaten gegeben, da das Tool keine Petrinetze verarbeiten kann.

Nach Tabelle 4.2 sind Automaten für diese Anwendung besser geeignet als Petrinetze. Besonders bei der Anwendbarkeit auf „DESTool“ zeigt diese Beschreibungsform seine Vorteile.

4.3. Auswahl des Steuerungsansatzes

Als Steuerungsansätze stehen der monolithische, der modulare, der lokal-modulare und der dezentrale Steuerungsansatz zur Verfügung, die in Abschnitt 2.8 erläutert sind. Die vier verschiedenen Steuerungsvarianten, die in Abschnitt 5.3 beschrieben sind, werden unabhängig voneinander entworfen. Aus diesem Grund ist es möglich, für jede Variante den besten Steuerungsansatz zu wählen. Die Tabellen 4.3 und 4.4 zeigen verschiedene Streckeneigenschaften und andere Kriterien, die die Auswahl des Steuerungsansatzes beeinflussen. Die Streckenmodelle der Varianten zwei bis vier haben ähnliche Eigenschaften. Aus diesem Grund wird die Auswahl des Ansatzes zusammengefasst.

Als „sehr relevant“ werden Streckeneigenschaften gewichtet, die ausschlaggebend für den Steuerungsentwurf sind. Beispielsweise muss das Streckenmodell für den lokal-modularen Entwurfsansatz strukturierbar sein oder die Spezifikation für den modularen, lokal-modularen

Eigenschaften und Kriterien	Relevanz	Monolithisch	Modular	Lokal-modular	Dezentral
Streckenmodell strukturierbar	5	5 (++)	5 (++)	5 (++)	5 (++)
Geringe Komplexität des Streckenmodells	5	5 (++)	5 (++)	5 (++)	5 (++)
Spezifikationen sind nicht partitionierbar	5	5 (++)	1 (- -)	1 (- -)	1 (- -)
Verteilte Implementierung der Steuerung nicht möglich	5	5 (++)	1 (- -)	1 (- -)	1 (- -)
Alle Ereignisse stehen global zur Verfügung	5	4 (+)	4 (+)	5 (++)	4 (+)
Geringer Implementierungsaufwand	4	4 (+)	4 (+)	1 (- -)	1 (- -)
Codegenerator vorhanden	4	5 (++)	5 (++)	5 (++)	1 (- -)
Geringe Speicherauslastung der SPS	1	5 (++)	5 (++)	3 (o)	3 (o)
		161	**121**	**112**	**91**

Tabelle 4.3.: Auswahl des Steuerungsansatzes für Variante eins

und dezentralen Ansatz partitionierbar sein. Der Implementierungsaufwand sowie das Vorhandensein eines Codegenerators werden mit vier bewertet, da es die Realisierung erleichtert, aber nicht zwingend notwendig ist. Die geringe Speicherauslastung der SPS wird mit eins bewertet, da sechs SPS zur Verfügung stehen, die maximal zu 20 Prozent ausgelastet sind, und somit noch ausreichend freie Kapazitäten zur Verfügung stehen. Die Eigenschaften und Kriterien in den Tabellen 4.3 und 4.4 unterscheiden sich in der Partitionierbarkeit der Spezifikationen und der Möglichkeit der verteilten Implementierung. Diese Eigenschaften sind nur für die Varianten zwei bis vier vorhanden.

Für alle Varianten ist das Streckenmodell strukturierbar und weist eine geringe Komplexität auf. Die Spezifikation für Variante eins kann nicht partitioniert werden. Dies ist beim modularen, lokal-modularen und dezentralen Ansatz Voraussetzung. Bei Variante zwei bis vier sind die Spezifikationen partitionierbar, sodass hier eine verteilte Implementierung der Steuerung möglich ist. Für den monolithischen und modularen Entwurf müssen alle Ereignisse zur Verfügung stehen. Dies ist bei Variante eins nach kleineren Erweiterungen der Fall. Bei Variante zwei bis vier können die Ereignisse zwar auch global zur Verfügung gestellt werden, dies würde aber zu einem vermeidbaren hohen Datenaustausch zwischen den Stationen führen. Ein Codegenerator ist nur für den monolithischen, modularen und lokal-modularen Entwurfsansatz vorhanden. Für den dezentralen Ansatz müsste ein Codegenerator entwickelt werden, da der modellbasierte Entwurfsansatz mit Synthese voraussetzt, dass ein Codegenerator

Eigenschaften und Kriterien	Relevanz	Monolithisch	Modular	Lokal-modular	Dezentral
Streckenmodell strukturierbar	5	5 (++)	5 (++)	5 (++)	5 (++)
Geringe Komplexität des Streckenmodells	5	5 (++)	5 (++)	5 (++)	5 (++)
Spezifikationen sind partitionierbar	5	1 (- -)	5 (++)	5 (++)	5 (++)
Verteilte Implementierung der Steuerung möglich	5	1 (- -)	4 (+)	4 (+)	4 (+)
Alle Ereignisse stehen global zur Verfügung	5	2 (-)	2 (-)	4 (+)	3 (o)
Geringer Implementierungsaufwand	4	2 (-)	2 (-)	4 (+)	1 (- -)
Codegenerator vorhanden	4	5 (++)	5 (++)	5 (++)	1 (- -)
Geringe Speicherauslastung der SPS	1	5 (++)	5 (++)	3 (o)	3 (o)
		103	**138**	**154**	**121**

Tabelle 4.4.: Auswahl des Steuerungsansatzes für Variante zwei bis vier

verwendet wird, um Fehler während der Implementierung zu vermeiden. Dieser Aufwand ist gegenüber den anderen Ansätzen, für die es bereits einen Codegenerator gibt, sehr groß. Die Speicherauslastung der SPS ist beim monolithischen und modularen Ansatz eher gering, da der Codegenerator „ACArrow“ nur eine SCL-Quelle erstellt, die den Automaten abbildet. Beim lokal-modularen Ansatz werden zwölf SCL-Quellen generiert, die mehr Speicher der SPS benötigen. Zusätzlich zu den Modellen für die Steuerung werden hier auch die Modelle der Produktsysteme implementiert. Beim dezentralen Ansatz wird dieses Kriterium neutral mit einer drei bewertet, da kein Codegenerator zur Verfügung steht, anhand dessen eine Bewertung möglich ist.

Für Variante eins ist nach der Auswertung in Tabelle 4.3 der monolithische Ansatz geeignet. Da die Spezifikation nicht partitionierbar ist, können die anderen Ansätze nicht verwendet werden. Für die Varianten zwei bis vier eignet sich nach Tabelle 4.4 der lokal-modulare Ansatz. Auch der modulare Ansatz könnte gewählt werden, allerdings werden dabei sehr viele Signale übertragen, die beim lokal-modularen Ansatz nicht benötigt werden. Der monolithische Ansatz eignet sich nicht, da die Steuerungen nicht verteilt implementierbar sind. Der dezentrale Ansatz ist wegen des hohen Aufwandes bei der Implementierung nicht geeignet.

5. Modellbildung und Steuerungsentwurf

In diesem Kapitel erfolgen die Modellbildung und der Steuerungsentwurf. Annahmen, die sowohl für die Modellbildung als auch für die Auswertung in Kapitel 7 gelten, sind in Abschnitt 5.1 aufgelistet. Abschnitt 5.2 beschreibt die Modelle für die Steuerstrecke und Abschnitt 5.3 für die Spezifikationen. Der Entwurf der Steuerungen wird in Abschnitt 5.4 behandelt. Alle Modelle, die in diesem Kapitel entworfen werden, sind als „DESTool“ Projekte und in gedruckter Form im Anhang zu finden.

5.1. Voraussetzungen für die Modellbildung

Für die Modellbildung und die Auswertung in Kapitel 7 werden einige Annahmen getroffen, damit zum einen die Komplexität der Modelle auf das Wesentliche reduziert wird, zum anderen aber auch die Zeiten und Situationen in der Auswertung miteinander verglichen werden können. Diese Annahmen sind:

1. Alle Hardware Komponenten arbeiten korrekt. Hardware Fehler werden nicht berücksichtigt. Kommt es dennoch zu einem Defekt, wird der Fehler behoben und die Messung erneut durchgeführt.
2. Sensoren werden nur durch Werkstücke ausgelöst. Jeder Sensor erkennt zu jeder Zeit ein vorbeikommendes Werkstück. Wird ein Werkstück nicht erkannt, muss die Messung wiederholt werden.
3. Ein manueller Eingriff in den Prozess ist nicht vorgesehen. Zu Testzwecken dürfen Werkstücke außerhalb des Schutzkäfigs aufgehalten, beschleunigt oder aus dem Prozess genommen werden. Beim erneuten Eingliedern müssen sie an der Stelle, an der sie entnommen wurden, wieder in der richtigen Lage eingegliedert werden. Für Zeitmessungen dürfen Werkstücke zu keiner Zeit aus dem Prozess genommen, beschleunigt oder aufgehalten werden.
4. Die Schiebetüren des Schutzkäfigs dürfen während der Produktion nicht geöffnet werden, damit die Produktion nicht durch einen Not-Halt unterbrochen wird.

5. Die Modellfabrik muss gemäß Anleitung in Anhang A in Betrieb genommen und gestartet werden. Vor dem Start der Modellfabrik dürfen sich keine Werkstücke im Prozess befinden und die Komponenten müssen sich in ihrer Ruheposition[5] befinden.

6. Der Roboter saugt die Werkstücke jederzeit korrekt an. Wird eine Platine nicht korrekt auf dem Unterteil abgelegt, kann der Benutzer die für das Unterteil vorgesehene Platine manuell auf das Unterteil legen, wenn er dabei nicht gegen Punkt 3 verstößt. Anderenfalls muss die Messung wiederholt werden.

7. Auf den Zufuhrbändern und in den Magazinen für die Platinen und Deckel befinden sich jederzeit ausreichend Unterteile, Platinen beziehungsweise Deckel.

8. Die Unterteile auf den Zufuhrbändern sowie die Platinen und Deckel in den Magazinen befinden sich in der richtigen Lage.

9. Auf den Auswurfbändern ist jederzeit ausreichend Platz für neue Werkstücke. Bei gleichen Messungen mit unterschiedlichen Steuerungsvarianten werden Werkstücke auf den Auswurfbändern immer zum selben Zeitpunkt entfernt. Auf diese Weise wird sichergestellt, dass die Linearachse beispielsweise bei zehn Werkstücken die Werkstücke eins bis sechs auf Band eins und die Werkstücke sieben bis zehn auf Band zwei ablegt. Nur wenn die Reihenfolge identisch ist, können die Messungen miteinander verglichen werden.

10. Bei allen Zeitmessungen darf die Zeit, die sich ein Werkstück unter der Kamera befindet, nicht durch Drücken des Tasters „S2“ beschleunigt werden. Auch dürfen keine Unterteile durch Drücken von „S3“ aus dem Prozess abgeschoben werden.

11. Geht Station 50 während einer Zeitmessung durch einen defekten Deckel in Störung oder wird nur ein Aufkleber auf das Werkstück gepresst, ist gegebenenfalls die Störung zu beseitigen und die Messung zu wiederholen.

5.2. Modellbildung der Steuerstrecke

Da bereits eine untergeordnete Steuerung, die in Kapitel 3 beschrieben ist, existiert, die die grundlegenden Funktionen der Modellfabrik steuert, müssen bei der übergeordneten Steuerung, die in diesem Kapitel entworfen wird, lediglich die relevanten Komponenten modelliert werden. Insgesamt gibt es 14 Modelle für die Steuerstrecke, von denen zehn aus nur einem Zustand bestehen und alle Ereignisse Schlingen sind. Tabelle 5.1 listet diese Modelle mit Ereignissen in Zeile eins bis 23 auf. In Zeile 24 bis 35 sind die anderen Modelle, die aus mehr

[5] Die Ruhepositionen der Komponenten sind in den Bildern und Beschreibungen der Stationen (siehe Kapitel 3) dargestellt beziehungsweise beschrieben.

als einem Zustand bestehen, mit Ereignissen aufgelistet. Im Laufe dieses Absatzes werden diese Modelle beschrieben. Die Zahlen in den Modell- und Ereignisnamen zeigen an, an welcher Station die Ereignisse generiert werden. „E60“ und „EA30“ bedeuten zusätzlich, dass diese Signale Eingangssignale von Station 60 beziehungsweise Ein- und Ausgangssignale von Station 30 sind. Für den Codegenerator „ACArrow“ müssen die steuerbaren Ereignisse mit „STR“ und die nicht steuerbaren Ereignisse mit „E“ gekennzeichnet werden. Da die Ereignisnamen sehr lang sind, zeigt die dritte Spalte die Abkürzung der Ereignisnamen, die im nachfolgenden Text verwendet wird. Einige Ereignisse in den Modellen „20_Linearachse“ und „60_EA30_RoboAuftraege“ gelten nicht für alle Steuerungsvarianten. Daher werden in der Spalte „Ereignisname“ in Klammern die Varianten angegeben, für die dieses Ereignis existiert.

Nr.	Modellname	Ereignisname	Name im Text
1	20_Auftraege	E_20_Auftrag1Mgl	Auftrag1Mgl
2		E_20_NurAuftrag2Mgl	NurAuftrag2Mgl
3	20_Linearachse	STR_20_Auftrag20_30_Bd1_erl (V2, V3, V4)	ST20_Auftr20_30_Bd1
4		STR_20_Auftrag20_30_Bd2_erl (V2, V3, V4)	ST20_Auftr20_30_Bd2
5		STR_20_Auftrag60_20_Bd1_erl (V3)	Auftr60_20_Bd1
6		STR_20_Auftrag60_20_Bd2_erl (V3)	Auftr60_20_Bd2
7		STR_20_Auftrag60_30_Bd1_erl (V3)	Auftr60_30_Bd1
8		STR_20_Auftrag60_30_Bd2_erl (V3)	Auftr60_30_Bd2
9	20_E60_Linear-achse	STR_20_E60_Auftrag20_30_Bd1_erl	ST60_Auftr20_30_Bd1
10		STR_20_E60_Auftrag20_30_Bd2_erl	ST60_Auftr20_30_Bd2
11	60_Abschieber1-Erlauben	STR_60_Abs1Erlauben	Abs1Erlauben
12	60_Lager	STR_60_LagerLeer	LagerLeer
13	60_EA30_Robo-Auftraege	E_60_RoboAusgelagert (V2, V4)	RoboAusgelagert
14		E_60_RoboEingelagert (V2, V4)	RoboEingelagert
15		STR_60_RoboST60Bd1 (V1, V4)	RoboST60Bd1
16		STR_60_RoboST60Bd2 (V1, V4)	RoboST60Bd2
17		STR_60_RoboST60Lager (V2, V4)	RoboST60Lager
18		STR_60_RoboST60Platine (V3)	RoboST60Platine
19	60_Roboter-Erlauben	STR_60_RoboterErlauben	RoboterErlauben
20	60_Schleuse-Erlauben	STR_60_SchleuseErlauben	SchleuseErlauben

Nr.	Modellname	Ereignisname	Name im Text
21	60_Timer	E_60_TimerAbgelaufen	TimerAbgelaufen
22	60_Unterteil	STR_60_UnterteilBd1	UnterteilBd1
23		STR_60_UnterteilBd2	UnterteilBd2
24	20_E60_Fehlteil	E_20_E60_FehlteilBd1	FehlteilBd1
25		E_20_E60_FehlteilBd2	FehlteilBd2
26		E_20_E60_FehlteilInPos	FehlteilInPos
27		E_20_E60_NotFehlteilInPos	NotFehlteilInPos
28	20_E60_Lager	E_20_E60_LagerLeer	LagerLeer
29		E_20_E60_NotLagerLeer	NotLagerLeer
30	60_Fehlteil	E_60_Fehlteil	Fehlteil
31		E_60_NotFehlteil	NotFehlteil
32	20_EA30_Robo-Auftraege	E_20_RoboDoneLagerLeer	RoboDoneLagerLeer
33		E_20_RoboDoneNotLagerLeer	RoboDoneNotLager-Leer
34		STR_20_RoboLagerBd1	RoboLagerBd1
35		STR_20_RoboLagerBd2	RoboLagerBd2

Tabelle 5.1.: Übersicht der Streckenmodelle mit Ereignissen

Die Streckenmodelle „20_E60_Fehlteil“, „20_E60_Lager“ und „60_Fehlteil“ bestehen jeweils aus zwei Zuständen. Die Zustandsübergänge werden durch die Ereignisse „FehlteilInPos“ und „NotFehlteilInPos“, „LagerLeer“ und „NotLagerLeer“ beziehungsweise „Fehlteil“ und „NotFehlteil“ ausgelöst. Bei dem Modell „20_E60_Fehlteil“ sind in beiden Zuständen jeweils die Ereignisse „FehlteilBd1“ und „FehlteilBd2“ möglich.

Das Streckenmodell „20_EA30_RoboAuftraege“ besteht aus drei Zuständen. Die Ereignisse „RoboLagerBd1“ beziehungsweise „RoboLagerBd2“ lösen einen Zustandswechsel zum Zustand zwei beziehungsweise drei aus. Der Zustandswechsel zurück zum Zustand eins erfolgt beim Auftreten der Ereignisse „RoboDoneLagerLeer“ oder „RoboDoneNotLagerLeer“.

5.3. Modellbildung der Spezifikationen

Um defekte Platinen von Unterteilen zu entfernen, gibt es innerhalb der Modellfabrik nur die Möglichkeit, diese mit dem Roboter abzunehmen. Um Unterteile von Station 60 zu den Roboterbändern von Station 30 zu bringen, gibt es dagegen zwei Möglichkeiten. Die erste Möglichkeit ist, dass der Roboter das Unterteil ansaugt und es auf den Roboterbändern ablegt. Die zweite Möglichkeit ist, dass das Unterteil bis zum Ende des Transportbandes an

Station 60 fährt und dort von der Linearachse zu den Roboterbändern gebracht wird. Kann ein Unterteil nicht erneut mit einer Platine bestückt werden, gibt es die Möglichkeit, es über den Abschieber an Station 60 abzuschieben. Das kann zum Beispiel der Fall sein, wenn keine neuen Aufträge vorhanden oder aktuell möglich sind. Alternativ dazu wird ein Lager mit einer Kapazität von vier Unterteilen geschaffen, in dem der Roboter Unterteile einlagern kann.

Aus den verschiedenen Möglichkeiten, mit Fehlteilen umzugehen, werden vier übergeordnete Steuerungsvarianten gebildet, die in Kapitel 6 umgesetzt und in Kapitel 7 miteinander verglichen werden. Die Variante, Fehlteile über den Abschieber abzuschieben, ist bereits in der untergeordneten Steuerung aus Kapitel 3 realisiert.
In der ersten Steuerungsvariante, die in Abschnitt 5.3.1 beschrieben ist, wird geprüft, ob ein Auftrag möglich ist. In diesem Fall wird die defekte Platine entfernt, das Unterteil vom Roboter direkt zu dem entsprechenden Roboterband gebracht und mit einer neuen Platine bestückt. Sind beide Aufträge möglich, wird das Unterteil, wie auch bei den Varianten zwei bis vier, auf Band eins abgelegt. Ist kein Auftrag möglich, wird das Werkstück vom Abschieber abgeschoben.
Die zweite Steuerungsvariante wird in Abschnitt 5.3.2 beschrieben. Hier geprüft, ob das Lager voll ist. In diesem Fall wird das Werkstück abgeschoben. Anderenfalls wird die defekte Platine entfernt und das Unterteil vom Roboter im Lager eingelagert. Befindet sich ein Unterteil im Lager und ist ein Auftrag möglich, holt der Roboter dieses aus dem Lager und legt es auf das entsprechende Roboterband.
In Variante drei aus Abschnitt 5.3.3 wird geprüft, ob ein Auftrag möglich ist. In diesem Fall nimmt der Roboter die defekte Platine ab und das Unterteil fährt bis zum Ende des Transportbandes von Station 60. Dort wird es von der Linearachse zu dem entsprechenden Roboterband an Station 30 gebracht und wie ein Unterteil, das von den Zufuhrbändern kommt, behandelt. Ist kein Auftrag möglich, wird das Werkstück abgeschoben.
Variante vier aus Abschnitt 5.3.4 ist eine Kombination aus den Varianten eins und zwei. Ist ein Auftrag möglich, wird die defekte Platine abgenommen und das Unterteil vom Roboter auf direktem Weg zu den Roboterbändern gebracht. Ist kein Auftrag möglich, wird geprüft, ob das Lager voll ist. In diesem Fall wird das Werkstück abgeschoben. Ist das Lager nicht voll, wird die defekte Platine entfernt und das Unterteil dort eingelagert.

5.3.1. Variante eins: Direkter Weg Station 60 » Station 30

Die Steuerung für Variante eins wird in Station 60 realisiert. Abbildung 5.1 zeigt das Modell für die Spezifikation „SPEC_V1_60_Fehlteil“, das aus fünf Zuständen besteht. Nicht steuerbare Ereignisse werden in jedem Zustand erlaubt, sind aber der Übersichtlichkeit halber in dieser und in den folgenden Abbildungen nicht dargestellt. Im Startzustand eins ist das Werkstück kein Fehlteil und die steuerbaren Ereignisse „ST60_Auftr20_30_Bd1“ und

„ST60_Auftr20_30_Bd2“ werden erlaubt. Sobald das nicht steuerbare Ereignis „Fehlteil“ auftritt, ist der Zustand zwei aktiv und die beiden steuerbaren Ereignisse werden verboten, damit das Unterteil des Fehlteils wieder in den Prozess eingegliedert werden kann. Über das Auftreten des nicht steuerbaren Ereignisses „Auftrag1Mgl“ beziehungsweise „NurAuftrag2Mgl“ erfolgt ein Zustandswechsel von Zustand zwei zum Zustand drei beziehungsweise vier. In diesem Zustand werden die steuerbaren Ereignisse „RoboterErlauben“ und „RoboST60Bd1“ beziehungsweise „RoboST60Bd2“ erlaubt. Gleichzeitig wird auch der jeweils andere Auftrag „ST60_Auftr20_30_Bd2“ beziehungsweise „ST60_Auftr20_30_Bd1“ wieder erlaubt, um parallel neue Werkstücke vom Zufuhrband holen zu können. Ist kein Auftrag möglich, tritt nach einer einstellbaren Zeit, die in der untergeordneten Steuerung vorgegeben werden kann, das Ereignis „TimerAbgelaufen“ auf und der Automat wechselt in den Zustand fünf, in dem der Abschieber erlaubt wird. Sind keine Aufträge mehr vorhanden, tritt dieses Ereignis ohne Zeitverzögerung auf. Aus den Zuständen drei bis fünf erfolgt ein Zustandswechsel zum Zustand eins nach Auftreten des nicht steuerbaren Ereignisses „NotFehlteil“.

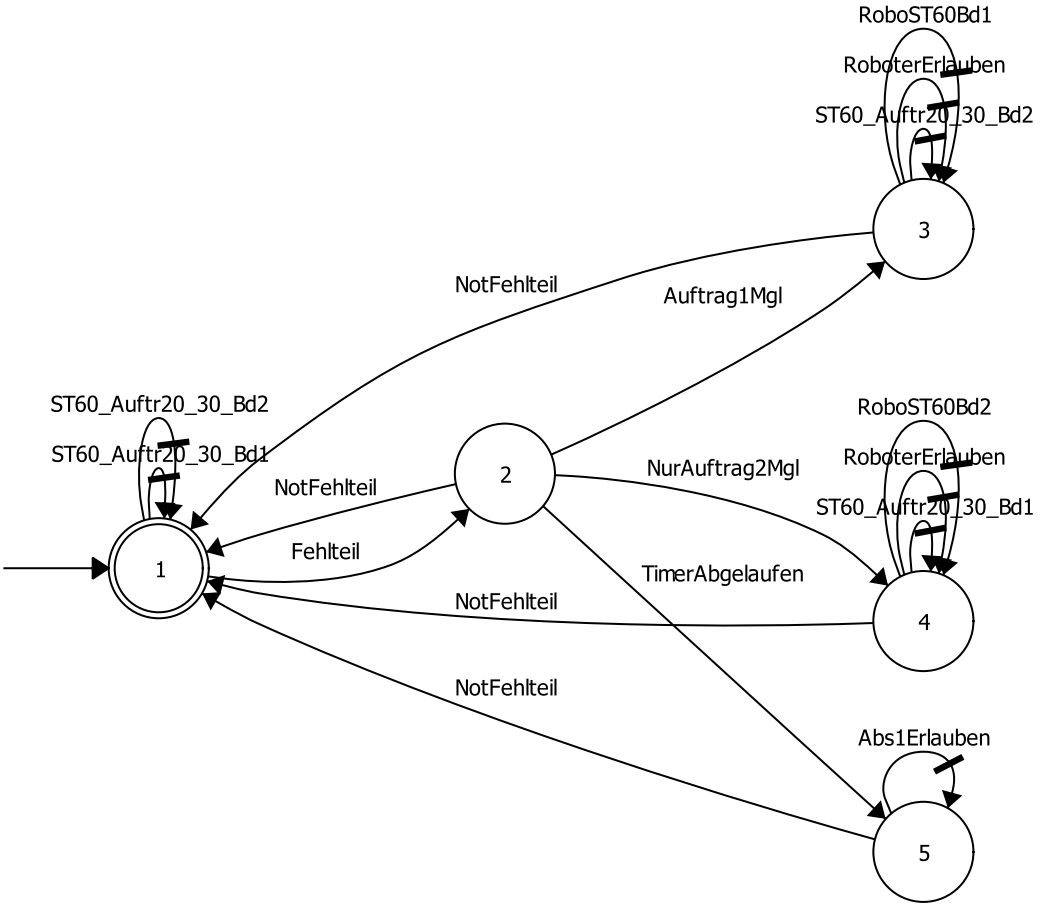


Abbildung 5.1.: Spezifikation „SPEC_V1_60_Fehlteil“

5.3.2. Variante zwei: Lager

Die Steuerung für Variante zwei ist in zwei Steuerungen aufgeteilt, die in den Stationen 60 und 20 realisiert sind. Es gibt die drei Spezifikationen „SPEC_V2_60_Lager“, „SPEC_V2_60_Fehlteil“ und „SPEC_V2_20_Eingliedern“, von denen die ersten beiden über die strenge Synchronisation (SPC) gekoppelt werden, sodass ein Ereignis immer dann möglich ist, wenn es in beiden Automaten erlaubt ist.

Der Automat „SPEC_V2_60_Fehlteil“ (siehe Abbildung 5.2) gibt an, ob das aktuelle Werkstück an Station 60 ein Fehlteil ist. Er besteht aus zwei Zuständen. Ein Zustandswechsel erfolgt bei den nicht steuerbaren Ereignissen „Fehlteil“ und „NotFehlteil“. Im Startzustand eins ist das Werkstück kein Fehlteil und die steuerbaren Ereignisse „ST60_Auftr20_30_Bd1“, „ST60_Auftr20_30_Bd2“ und „LagerLeer“ werden erlaubt. Im zweiten Zustand ist das Werkstück ein Fehlteil und die steuerbaren Ereignisse „ST60_Auftr20_30_Bd1“, „ST60_Auftr20_30_Bd2“, „RoboterErlauben“, „RoboST60Lager“, „LagerLeer“ und „Abs1Erlauben“ werden erlaubt.

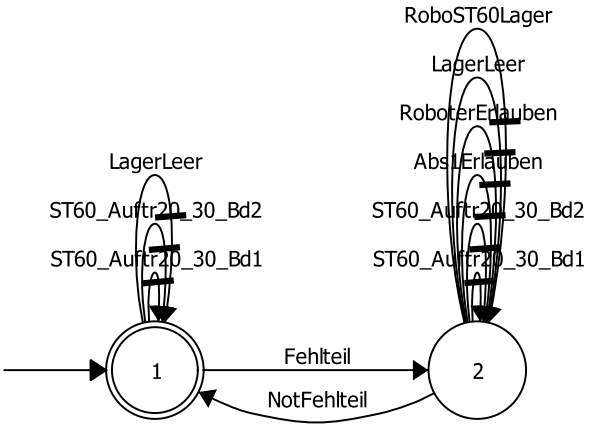


Abbildung 5.2.: Spezifikation „SPEC_V2_60_Fehlteil“

Der Zustand des Lagers wird von dem Automaten „SPEC_V2_60_Lager“ (siehe Abbildung 5.3) beschrieben. Er besteht aus fünf Zuständen, da es vier Lagerplätze gibt. Die Zustände eins bis fünf repräsentieren in aufsteigender Reihenfolge die Zustände „Lager leer“, „ein Lagerplatz belegt“, ..., „Lager voll“. Ein Zustandswechsel zum nächst höheren Zustand erfolgt, wenn das nicht steuerbare Ereignis „RoboEingelagert“ auftritt. Eine Verringerung des Zustandes erfolgt beim Auftreten des nicht steuerbaren Ereignisses „RoboAusgelagert“. Ein Einlagern, welches über die steuerbaren Ereignisse „RoboterErlauben“ und „RoboST60Lager“ gesteuert wird, ist in den Zuständen eins bis vier erlaubt. Im fünften Zustand ist der Abschieber über das steuerbare Ereignis „Abs1Erlauben“ erlaubt. Die steuerbaren Ereignisse „ST60_Auftr20_30_Bd1“ und „ST60_Auftr20_30_Bd2“ werden

in den Zuständen eins und zwei erlaubt, da das Verbieten der Aufträge von der Spezifikation „SPEC_V2_20_Eingliedern“ übernommen wird. In den anderen Zuständen werden diese Ereignisse verboten, damit zuerst die Unterteile aus dem Lager statt von den Zufuhrbändern verwendet werden. Befindet sich der Automat im Startzustand eins, ist das steuerbare Ereignis „LagerLeer“ erlaubt. Dieses gibt an, dass das Lager leer ist und wird für die Steuerung an Station 20 benötigt.

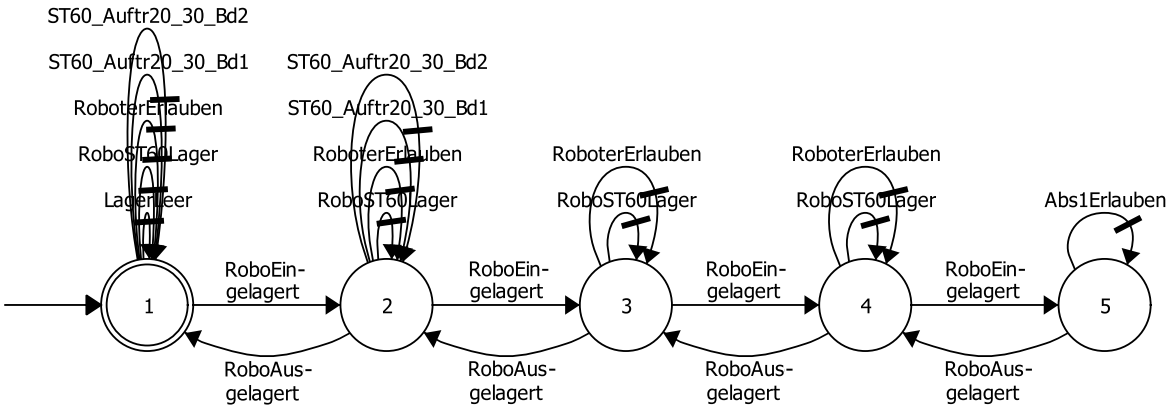


Abbildung 5.3.: Spezifikation „SPEC_V2_60_Lager“

Der Automat „SPEC_V2_20_Eingliedern“ (siehe Abbildung 5.4) besteht aus vier Zuständen. Im Startzustand eins ist das Lager leer. In diesem Zustand werden die steuerbaren Ereignisse „ST20_Auftr20_30_Bd1“ und „ST20_Auftr20_30_Bd2“ erlaubt. Tritt das nicht steuerbare Ereignis „NotLagerLeer“ auf, wechselt der Automat in den Zustand zwei. Hier werden alle steuerbaren Ereignisse verboten. Über die nicht steuerbaren Ereignisse „Auftrag1Mgl“ oder „NurAuftrag2Mgl“ erfolgt ein Zustandswechsel zum Zustand drei beziehungsweise vier. In diesem Zustand werden die steuerbaren Ereignisse „RoboLagerBd1“ und „ST20_Auftr20_30_Bd2“ beziehungsweise „RoboLagerBd2“ und „ST20_Auftr20_30_Bd1“ erlaubt. In Kombination mit der Steuerung an Station 60 ist so sichergestellt, dass das Lager zunächst komplett geleert wird, ehe ein Unterteil vom Zufuhrband dem Prozess hinzugefügt wird.

5.3.3. Variante drei: Linearachse

Die Steuerung für Variante drei erfolgt über zwei Steuerungen an den Stationen 60 und 20. Die Spezifikationen sind „SPEC_V3_60_Fehlteil“ und „SPEC_V3_20_Linearachse“. Der erste Automat entspricht grundsätzlich dem Automaten „SPEC_V1_60_Fehlteil“ aus Variante eins (siehe Abbildung 5.1). In den Zuständen drei und vier, in denen ein- beziehungsweise zweipolige Aufträge möglich sind, werden statt des Ereignisses „RoboST60Bd1“ be-

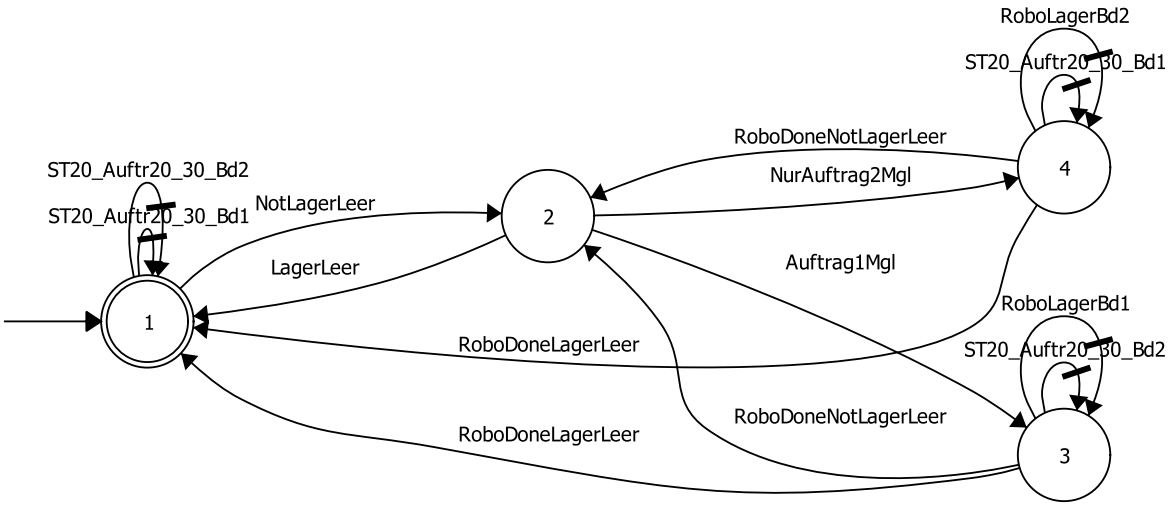


Abbildung 5.4.: Spezifikation „SPEC_V2_20_Eingliedern“

ziehungsweise „RoboST60Bd2“ die Ereignisse „RoboST60Platine“, „SchleuseErlauben“ und „UnterteilBd1“ beziehungsweise „UnterteilBd2“ erlaubt.

Der zweite Automat „SPEC_V3_20_Linearachse“ (siehe Abbildung 5.5) besteht aus fünf Zuständen. Im Startzustand eins ist das Werkstück an Station 60 kein Fehlteil und es werden die steuerbaren Ereignisse „ST20_Auftr20_30_Bd1“, „ST20_Auftr20_30_Bd2“, „Auftr60_20_Bd1“ und „Auftr60_20_Bd2“ erlaubt. Beim Auftreten des nicht steuerbaren Ereignisses „FehlteilBd1“ beziehungsweise „FehlteilBd2“ erfolgt ein Zustandswechsel zum Zustand zwei beziehungsweise drei. In diesem Zustand wird nur der Auftrag für das jeweils andere Band, also „ST20_Auftr20_30_Bd2“ im Zustand zwei und „ST20_Auftr20_30_Bd1“ im Zustand drei, erlaubt, damit die Linearachse kein Unterteil vom Zufuhrband holen kann, während das Unterteil an Station 60 zum Bandende gefahren wird. Sobald das Unterteil am Bandende angekommen ist, tritt das nicht steuerbare Ereignis „FehlteilInPos“ auf und es erfolgt ein Zustandswechsel von Zustand zwei zum Zustand vier beziehungsweise von Zustand drei zum Zustand fünf. Im Zustand vier werden die steuerbaren Ereignisse „ST20_Auftr20_30_Bd2“ und „Auftr60_30_Bd1“ beziehungsweise im Zustand fünf die Ereignisse „ST20_Auftr20_30_Bd1“ und „Auftr60_30_Bd2“ erlaubt. Mit dem Auftreten des nicht steuerbaren Ereignisses „NotFehlteilInPos“ erfolgt der Zustandswechsel vom Zustand vier beziehungsweise fünf zum Startzustand eins.

Die beiden Steuerungen an Station 20 und 60 stellen zusammen sichern, dass die Linearachse bei einem Fehlteil an Station 60 keine Unterteile von den Zufuhrbändern holen kann. Sobald das Fehlteil an Station 60 erkannt wird, wird ein entsprechender Auftrag der Linearachse verboten. Bevor die Schrittkette an Station 60 beendet wird, übernimmt Station 20 das Verbieten des Auftrages, solange bis das Werkstück die Roboterbänder an Station 30

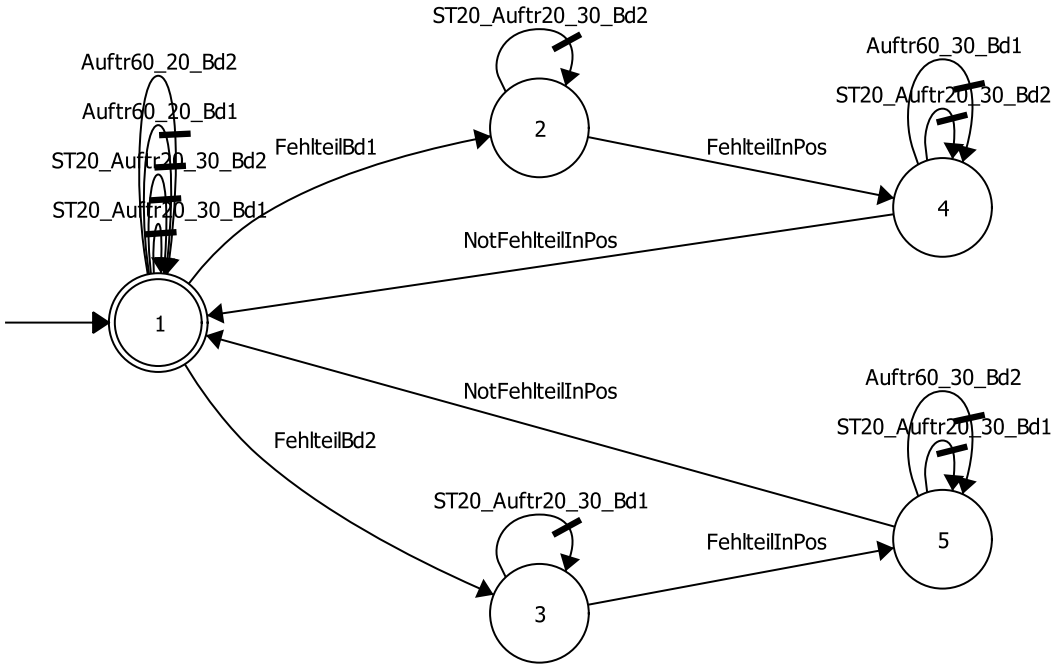


Abbildung 5.5.: Spezifikation „SPEC_V3_20_Linearachse“

erreicht hat. Die Überschneidung, dass beim Beenden der Schrittkette an Station 60 beide Stationen den Auftrag der Linearachse verbieten, ist notwendig, da sonst für einen Zyklus kein Verbot anliegt und es zu einer Fehlbearbeitung kommen kann.

5.3.4. Variante vier: Direkter Weg und Lager

Die Steuerung für Variante vier ist in zwei Steuerungen aufgeteilt, die in den Stationen 60 und 20 realisiert sind. Es gibt die drei Spezifikationen „SPEC_V4_60_Lager“, „SPEC_V4_60_Fehlteil“ und „SPEC_V4_20_Eingliedern“, von denen die ersten beiden wie bei Variante zwei über die strenge Synchronisation (SPC) gekoppelt werden. Da die vierte Variante eine Kombination der Varianten eins und zwei ist, gibt es bei den Modellen große Übereinstimmungen mit den bestehenden Modellen.

Der erste Automat „SPEC_V4_60_Lager“ entspricht grundsätzlich dem Automaten „SPEC_V2_60_Lager“ aus Variante zwei (siehe Abbildung 5.3). Zusätzlich sind an jedem Zustand die steuerbaren Ereignisse „RoboST60Bd1“ und „RoboST60Bd2“ möglich. Im Zustand vier muss außerdem das steuerbare Ereignis „RoboterErlauben“ möglich sein, damit der Roboter gegebenenfalls Platinen von den Unterteilen entfernen kann.

Der zweite Steuerungsautomat „SPEC_V4_60_Fehlteil“ ist eine Erweiterung des Automaten „SPEC_V1_60_Fehlteil“ aus Variante eins (siehe Abbildung 5.1). An jedem Zustand wird zusätzlich das steuerbare Ereignis „LagerLeer“ erlaubt. Außerdem werden im Zustand fünf, wenn kein Auftrag möglich ist, die steuerbaren Ereignisse „RoboterErlauben“ und „RoboST60Lager“ für das Einlagern von Werkstücken erlaubt.

Der Automat „SPEC_V4_20_Eingliedern“ ist identisch mit dem Automaten „SPEC_V2_20_Eingliedern“ aus Variante zwei (siehe Abbildung 5.4). Die Ereignisse „Auftrag1Mgl“ und „NurAuftrag2Mgl“ beinhalten in der untergeordneten Steuerung allerdings zusätzlich die Bedingung, dass der Roboter frei ist und sich an Station 60 kein Fehlteil befindet. Ohne diese Bedingungen kann es dazu führen, dass zu viele Werkstücke produziert werden, da sowohl die Steuerung an Station 60 als auch die Steuerung an Station 20 das Unterteil für das selbe Band vorsieht.

5.4. Entwurf der Steuerungen

Zum Entwurf der Steuerungen müssen zunächst die benötigten Strecken- und Spezifikationsmodelle aus Abschnitt 5.2 und 5.3 nach Abschnitt 2.9.1 in „DESTool“ erstellt werden. Für jede der vier Steuerungsvarianten wird dabei ein eigenes „DESTool“ Projekt erstellt. Die Abbildungen 5.6 und 5.7 zeigen, welche Strecken- und Spezifikationsmodelle für die einzelnen Varianten benötigt werden. Bei den Spezifikationsmodellen müssen alle nicht steuerbaren Ereignisse, die im Gesamtmodell der Steuerstrecke (Variante eins) beziehungsweise in den lokalen Systemen (Variante zwei bis vier) vorkommen, als Schlingen an jedem Zustand eingefügt werden. Existiert an dem Zustand bereits eine abgehende Kante mit diesem Ereignis, muss keine Schlinge eingefügt werden. Anschließend werden die Steuerungen entworfen.

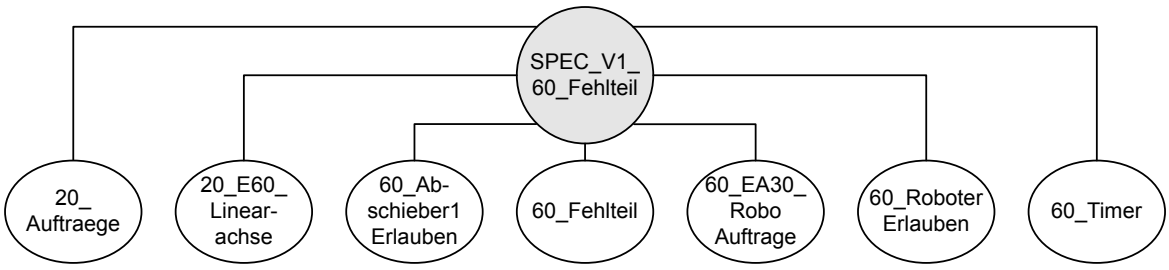


Abbildung 5.6.: Strecken- und Spezifikationsmodelle für Steuerungsvariante eins

Die erste Steuerungsvariante wird nach dem monolithischen Ansatz entworfen. Dazu muss zunächst das Gesamtmodell G der Steuerstrecke gebildet werden. Dies geschieht mit der Funktion „Parallel(G_1, G_2)“ von „DESTool“. Die benötigten Streckenmodelle aus

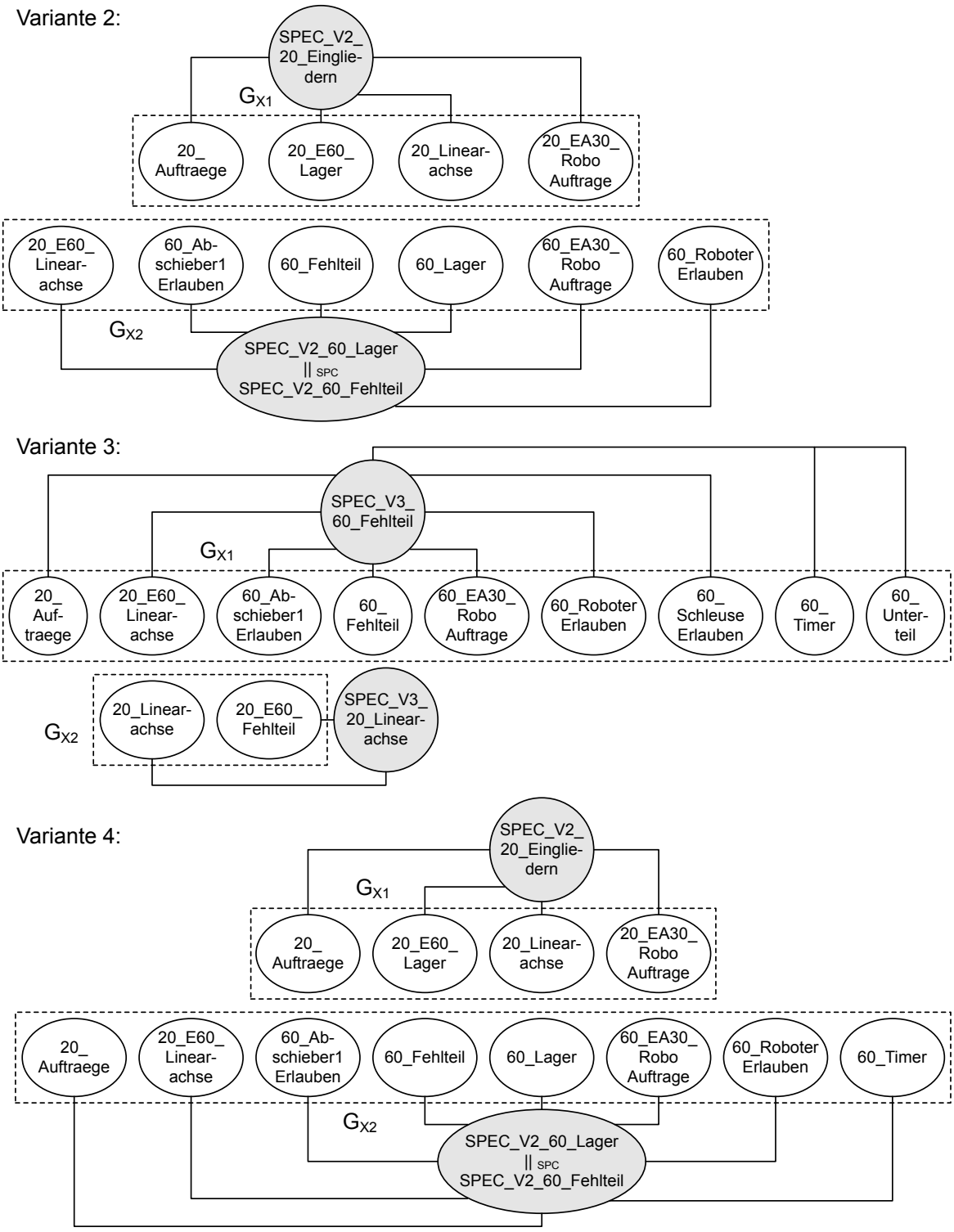


Abbildung 5.7.: Strecken- und Spezifikationsmodelle für Steuerungsvariante zwei bis vier

Abbildung 5.6 werden nacheinander der Funktion übergeben, bis das Gesamtmodell G, das in Abbildung 5.8 dargestellt ist, entstanden ist. Mit zwei Zuständen ist es nicht besonders komplex, sodass sich der monolithische Ansatz eignet. Anschließend wird mit der Funktion „SupConNB(G_{Plant}, G_{Spec})“ die supremale steuerbare nichtblockierende Teilsprache S gebildet. Der Funktion wird in G_{Plant} das Streckenmodell G und in G_{Spec} die Spezifikation K, die in Abschnitt 5.3.1 dargestellt ist, übergeben. Der entstandene Supervisor S wird anschließend über die Funktion „SubReduce(G_{Plant}, G_{Sup})“ reduziert, sodass der reduzierte Supervisor S_{Red} entsteht. Die Übergabeparameter an die Funktion sind das Gesamtmodell G der Steuerstrecke für G_{Plant} sowie der entstandene Supervisor S für G_{Sup}.

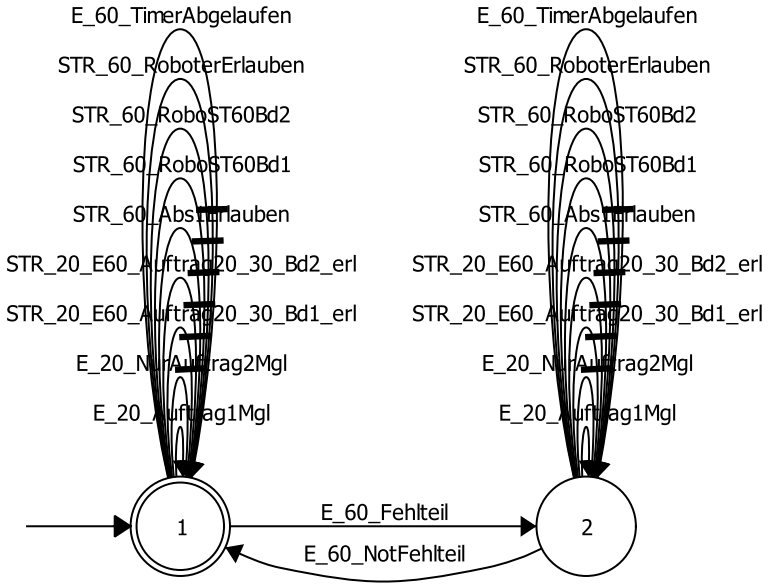


Abbildung 5.8.: Gesamtmodell der Steuerstrecke für Variante eins

Für die Steuerungsvarianten zwei bis vier müssen analog zum Gesamtmodell der Steuerstrecke von Variante eins zunächst die lokalen Systeme G_{X1} und G_{X2} gebildet werden. Abbildung 5.7 zeigt die Modelle, die für die einzelnen lokalen Systeme benötigt werden sowie die Information, welches die zugehörige Spezifikation ist. Anschließend werden die angepassten Spezifikationen E_{X1} und E_{X2} gebildet. Dazu wird die Funktion „Parallel(G_1, G_2)“ verwendet. Ihr werden das lokale System G_{Xj} sowie die zugehörige Spezifikation K_{Xj} aus Abschnitt 5.3.2 bis Abschnitt 5.3.4 für $j = 1, 2$ übergeben. Die Supervisor S_1 und S_2 werden über die Funktion „SupConNB(G_{Plant}, G_{Spec})“ mit G_{Xj} als G_{Plant} und E_{Xj} als G_{Spec} gebildet. Nachdem alle Supervisor erstellt wurden, wird über die Funktion „IsNonblocking(G_1, G_2)“ mit S_1 als G_1 und S_2 als G_2 auf lokale Modularität geprüft. Ist die lokale Modularität gegeben, können die reduzierten Supervisor S_{Red1} und S_{Red2} über die Funktion „SubReduce(G_{Plant}, G_{Sup}) gebildet werden.

6. Realisierung

Die Realisierung der übergeordneten Steuerung ist in zwei Abschnitte unterteilt. In Abschnitt 6.1 wird die untergeordnete Steuerung, die in Kapitel 3 beschrieben ist, erweitert. Im Abschnitt 6.2 werden die Modelle aus Kapitel 5 in Quelltext umgewandelt und in die untergeordnete Steuerung integriert. Das Roboterprogramm und das TIA-Gesamtprojekt nach diesen Änderungen und Erweiterungen befinden sich im Anhang. Zusätzlich sind dort auch alle Quelltexte der Stationen als XPS-Dokumente zu finden.

6.1. Erweiterung der untergeordneten Steuerung

Da eine minimal restriktive Steuerung (siehe Abschnitt 2.7.3) das Gesamtsystem in seiner Funktion einschränkt, muss die bestehende untergeordnete Steuerung aus Kapitel 3 erweitert werden. Funktionserweiterungen sind in den Stationen 20, 30 und 60 notwendig und werden in den folgenden Abschnitten erläutert. In Station 10 müssen die zu übertragenden Variablen zwischen den Stationen angepasst werden. Diese Variablen sind in Tabelle 6.1 aufgelistet.

6.1.1. Erweiterungen in Station 20

Visualisierung
Auf dem Touchpanel von Station 20 werden sechs neue Bilder erstellt. Das Bild „Steuerungen“ dient zur Übersicht und Beschreibung der vier Steuerungsvarianten. Vier Buttons in diesem Bild führen zu den Detailbildern, in denen die Steuerungsautomaten aus Kapitel 5 für jede Variante visualisiert sind. Ist die jeweilige Steuerungsvariante aktiv, wird in dem Detailbild der aktuelle Zustand grün hervorgehoben sowie die steuerbaren und nicht steuerbaren Ereignisse angezeigt und grün hervorgehoben, wenn sie möglich sind. Im Bild „Auswahl Steuerung“ kann die aktive Steuerung ausgewählt werden. Die Auswahl ist nur dann möglich, wenn sich sowohl Station 20 als auch Station 60 nicht im Automatikbetrieb befinden, um im laufenden Betrieb keine undefinierten Zustände zu erzeugen. Außerdem ist so eine Initialisierung der Steuerungen im ausgeschalteten Automatikbetrieb möglich.

Übertragung	Name	Kommentar
ST 20 » ST30	RoboLagerBd1	Roboter Auftrag Lager » Station 30 (Band 1)
	RoboLagerBd2	Roboter Auftrag Lager » Station 30 (Band 2)
ST 20 » ST 60	Auftrag1Mgl	Einpoliger Auftrag ist möglich
	NurAuftrag2Mgl	Nur zweipoliger Auftrag ist möglich
	AuftraegeFertig	Keine neuen Aufträge vorhanden
	SteuerungBit0	Auswahl der Steuerung (codiert)
	SteuerungBit1	Auswahl der Steuerung (codiert)
	SteuerungBit2	Auswahl der Steuerung (codiert)
ST 30 » ST 20	RoboDoneST20	Roboter Auftrag von ST 20 fertig bearbeitet
	RoboIdle	Roboter ist frei
ST 30 » ST 60	RoboDoneST60	Roboter Auftrag von ST 60 fertig bearbeitet
ST 60 » ST 20	Auftrag1Verbieten	Einpoligen Auftrag verbieten
	Auftrag2Verbieten	Zweipoligen Auftrag verbieten
	LagerST60Leer	Das Lager an Station 60 ist leer
	FehlteilST60	Aktuelles Werkstück an ST 60 ist Fehlteil
ST 60 » ST 30	RoboST60Bd1	Roboter Auftrag ST 60 » Station 30 (Band 1)
	RoboST60Bd2	Roboter Auftrag ST 60 » Station 30 (Band 2)
	RoboST60Lager	Roboter Auftrag ST 60 » Lager
	RoboST60Platine	Roboter Auftrag Platine an ST 60 abnehmen

Tabelle 6.1.: Erweiterte Kommunikation zwischen den Stationen

SPS Station 20

Da die Linearachse künftig Unterteile von Station 60 zu den Roboterbändern der Station 30 bringen soll, muss eine neue Funktion „SK_60>30“ mit diesem Fahrauftrag programmiert werden. Dazu kann eine der beiden bestehenden Funktionen mit einem Fahrauftrag „SK_20>30“ oder „SK_20<60“ dupliziert und angepasst werden. Wie bei den bestehenden Fahraufträgen kann mit der neuen Funktion „SK_60>30“ entweder Band 1 oder Band 2 der Roboterbänder angefahren werden. Welches Band angefahren wird, ist abhängig von der Variable, mit der die Schrittkette in der Funktion gestartet wird. Die Programmierung und Variablenbezeichnung orientiert sich an den beiden bestehenden Funktionen.

In der Funktion „Auto“ werden die vier bereits bestehenden Variablen, die Fahraufträge starten, gesetzt. Für die neue Funktion „SK_60>30“ werden zwei neue Variablen eingefügt. Die erste wird gesetzt, wenn Station 60 meldet, dass ein Unterteil für Band 1 in Position liegt, die zweite, wenn ein Unterteil für Band 2 in Position liegt. Damit jeweils nur ein Fahrauftrag zurzeit aktiv ist, muss die gegenseitige Verriegelung aller Variablen, die einen Fahrauftrag starten, ergänzt werden. Die übergeordneten Steuerungen an den Stationen 20 und 60 ha-

ben die Möglichkeit, zeitweise einzelne Fahraufträge zu verbieten. Aus diesem Grund müssen alle Variablen, die einen Fahrauftrag starten, um die entsprechenden Variablen, die in der übergeordneten Steuerung gesetzt werden, erweitert werden.

Die Auswahl der Steuerung, die über das Touchpanel im Bild „Auswahl Steuerung“ erfolgt, wird in der Funktion „Auto“ codiert und in drei Bits über Station 10 an Station 60 übertragen. Die Codierung erfolgt, damit die Anzahl der zu übertragenden Bits gering bleibt und bei Bedarf erweitert werden kann. Die Codierung setzt sich wie folgt zusammen (die erste Zahl entspricht „SteuerungBit0“, die zweite „SteuerungBit1“ und die dritte „SteuerungBit2“):

- 000: Keine Steuerung
- 001: Steuerung Variante 1
- 010: Steuerung Variante 2
- 011: Steuerung Variante 3
- 100: Steuerung Variante 4

Neben dieser Übertragung werden in der Funktion „Auto“ die Variablen „Auftrag1Mgl“, „NurAuftrag2Mgl“ und „AuftraegeFertig“ gebildet und über Station 10 an Station 60 übertragen, da die Variablen als Eingangsereignisse für die übergeordneten Steuerungen an Station 20 und 60 benötigt werden. Ein Auftrag ist möglich, wenn der Automatikbetrieb aktiv ist, die Linearachse gerade kein Werkstück zum entsprechenden Roboterband bringt, Station 30 meldet, dass das zugehörige Band frei ist, und noch ein unbearbeiteter Auftrag dieser Art vorhanden ist. Die Variable „NurAuftrag2Mgl“ wird nur gesetzt, wenn „Auftrag1Mgl“ nicht gesetzt ist. Sind alle Aufträge abgearbeitet, wird die Variable „AuftraegeFertig“ gesetzt.

6.1.2. Erweiterungen in Station 30

Roboterbefehle
Für die Erweiterungen an Station 30 muss das Roboterprogramm erweitert werden. In der ursprünglichen Variante kann der Roboter nur die Aufträge „Einpolige Platine einsetzen“ und „Zweipolige Platine einsetzen“ annehmen und ausführen. Im Rahmen einer nicht veröffentlichten Bachelorthesis wurde diese Erweiterung umgesetzt. Anschließend ist der Roboter in der Lage, die Aufträge aus Tabelle 6.2 anzunehmen und zu bearbeiten.

Bei allen Aufträgen, bei denen der Roboter ein Werkstück an Station 60 abholen muss, muss er die Werkstücke anschließend präzise ablegen. Aus diesem Grund muss sich das Werkstück vor dem Ansaugen an einer definierten Position befinden. Da die kapazitiven Sensoren an dem Transportband von Station 60 nicht präzise genug sind, muss der Roboter das Werkstück vor dem Ansaugen in Position schieben. Auf Höhe des ersten Abschiebers an Station

Auftrag	Codierung (Port 11, 10, 9, 8)
Kein neuer Auftrag	0000
Einpolige Relaiskarte einsetzen	0001
Zweipolige Relaiskarte einsetzen	0010
Band 1 Unterteil wenden	0011
Band 2 Unterteil wenden	0100
Band 60 Relaiskarte entfernen	0101
Band 60 Unterteil einlagern	0110
Unterteil aus Lager zu Band 1	0111
Unterteil aus Lager zu Band 2	1000
Unterteil Band 60 zu Band 1	1001
Unterteil Band 60 zu Band 2	1010
Relaiskarte entfernen und...	
... Band 60 Unterteil einlagern	1011
... Unterteil Band 60 zu Band 1	1100
... Unterteil Band 60 zu Band 2	1101

Tabelle 6.2.: Erweitertes Roboterprogramm

60 ist die Bandbegrenzung, die für das Schieben benötigt wird, an beiden Seiten unterbrochen. Dadurch ist hier kein Schieben möglich. Aus diesem Grund muss das Werkstück nach dem ersten Abschieber gestoppt werden. Hier ist die Bandbegrenzung vorhanden, sodass ein Schieben durch den Roboter möglich ist.

Befindet sich der Roboter am Band der Station 60, ist der Ausgangsport 28 („Roboter am Band ST 60“) des Roboters gesetzt. Befindet er sich am Lager, ist der Ausgangsport 29 („Roboter am Lager“) gesetzt. Diese Ergänzung dient zur Beschleunigung an Station 60. Ohne dieses Bit würde das Band an Station 60 erst freigegeben werden, wenn der Roboter das Werkstück eingelagert oder an Station 30 bestückt hat. In dieser Zeit könnte bereits das nächste Werkstück an Station 60 bearbeitet werden. Ein Rückstau vor Station 60 wird durch das Setzen dieses Bits vermieden. Das Bit für das Lager dient hauptsächlich zur einheitlichen Programmierung.

Damit der Roboter Unterteile zuverlässig auf den Roboterbändern ablegen kann, werden jeweils in der Mitte eines Bandes Aussparungen an den Plexigas-Begrenzungen eingefräst. Durch diese Maßnahme wird das Band an dieser Stelle etwas breiter. Ohne diese Aussparungen verkantet sich das Unterteil beim Ablegen auf den Bändern manchmal zwischen den Begrenzungen, sodass ein manueller Eingriff in den Prozess notwendig ist.

SPS Station 30
Damit der Roboter Platinen entfernen und Werkstücke einlagern kann, müssen zwei neue Funktionen „WSPlatineEntfernen“ und „WSEinlagern“ erstellt werden. Die Schrittketten in den Funktionen werden gestartet, wenn Station 60 meldet, dass eine Platine entfernt beziehungsweise ein Werkstück eingelagert werden soll. Dann wird geprüft, ob die Variable für die Schlüsselmarke des Roboters gerade gesetzt ist. In diesem Fall wartet die Schrittkette, bis die Variable nicht mehr gesetzt ist. Ist die Variable nicht gesetzt, wird sie gesetzt und dem Roboter wird der Befehl zum Entfernen der Platine beziehungsweise zum Einlagern übermittelt. Setzt der Roboter das Ausgangsbit „Roboter am Band ST 60“ zurück, wird dieser Befehl über Station 10 an Station 60 übertragen, damit diese das Transportband für das nächste Werkstück freigeben kann. Wenn der Roboter fertig ist, wird die Variable für die Schlüsselmarke wieder zurückgesetzt und die Funktion beendet.

In die Funktionen „Band1“ und „Band2“ werden die Aufträge „Unterteil von Station 60 nach Station 30 bringen“ und „Unterteil vom Lager nach Station 30 bringen“ integriert. Auch hierbei wird beim Rücksetzen des Ausgangsbits „Roboter am Band ST 60“ die Information an Station 60 beziehungsweise beim Rücksetzen des Ausgangsbits „Roboter am Lager“ die Information an Station 20 übermittelt. Da die Funktionen „WSPlatineEntfernen“ und „WSEinlagern“ priorisiert werden sollen, dürfen die Funktionen „Band1“ und „Band2“ den Roboter nur verwendet, wenn kein Werkstück eingelagert oder keine Platine entfernt werden sollen. Anderenfalls käme es zu einem Rückstau an Station 60.

In der Funktion „Auto“ werden ähnlich zu den Variablen, die die Fahraufträge der Linearachse starten, Variablen eingefügt, die die Roboteraufträge starten. Die Aufträge „Platine entfernen“ und „Werkstück einlagern“ benötigen keine Verriegelung, sodass hier auch keine Variablen erstellt werden müssen. Für die ein- und zweipoligen Aufträge „Platine bestücken“, „Unterteil von Station 60 nach Station 30 bringen“ und „Unterteil vom Lager nach Station 30 bringen“ ist eine Verriegelung notwendig. Es werden jeweils die drei ein- und zweipoligen Aufträge gegeneinander verriegelt, da immer nur ein Werkstück zurzeit auf einem Transportband sein darf. Ein- und zweipolige Aufträge können gleichzeitig bearbeitet werden und benötigen keine gegenseitige Verriegelung.

6.1.3. Erweiterungen in Station 60

Lager
Um Unterteile innerhalb des Schutzkäfigs aufzubewahren, muss ein Lager gebaut werden. Die einfachste Konstruktion dafür ist eine Platte, auf der die Unterteile abgelegt werden. Die Platte kann beispielsweise aus Plexiglas bestehen. Die Unterteile im Lager müssen präzise abgelegt werden können und gegen Verrutschen, zum Beispiel bei Stößen gegen die Modellfabrik, gesichert sein, damit der Roboter diese ohne Verkanten auf den Roboterbändern

ablegen kann. Aus diesem Grund wird auf dem Lager eine zweite Plexiglas-Platte befestigt, die mit Aussparungen für die Unterteile versehen ist. Die Aussparungen müssen etwas größer als die Unterteile sein, damit der Roboter diese im Lager ablegen kann und sie sich dabei nicht verkanten. Sind die Aussparungen zu groß, kann der Roboter die Unterteile nicht mehr präzise genug auf den Roboterbändern ablegen.

Damit der Roboter Unterteile im Lager einlagern und entnehmen kann, muss es sich in Greifweite des Roboters befinden. Abbildung 6.1 zeigt die nähere Umgebung des Roboters mit dem Lager in der Mitte des Bildes. Aus diesem Blickwinkel wird deutlich, dass nur auf der rechten Seite des Roboters ausreichend Platz für ein Lager ist. Der Roboter kann den linken Bereich durch die Begrenzung seiner Drehachsen nicht erreichen. Der Platz ober- und unterhalb des Roboters ist durch die Transportbänder begrenzt. Der Platz auf der rechten Seite des Roboters ist gut geeignet, da der Roboter, wenn er von Station 60 zu seinem Home-Sensor fährt, am Lager vorbeifährt. Da sich am Boden unterhalb des neuen Lagers Luftschläuche befinden, wird das Lager auf Höhe der Transportbänder montiert. So können auch später noch Leitungen und Schläuche unterhalb des Lagers entlanggeführt werden.

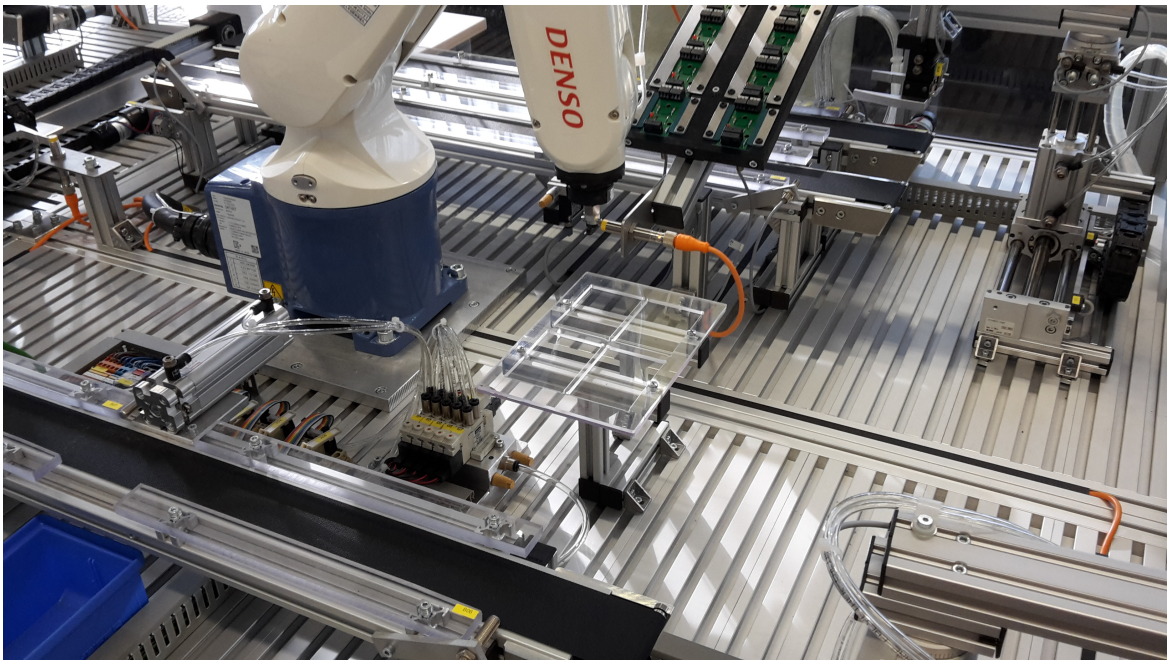


Abbildung 6.1.: Lager für Unterteile in der Modellfabrik

Das Lager soll ausreichend Platz für Unterteile bieten. Gleichzeitig ist es aufgrund der physikalischen Begrenzungen durch andere Komponenten und der Greifweite des Roboters begrenzt. Die Lagerkapazität wird auf vier festgelegt, da so eine kompakte und nahezu quadratische Bauweise des Lagers möglich ist, was es stabiler macht.

Weitere Hardware-Änderungen

Bei der Variante, bei der die Linearachse die Unterteile von Station 60 zu den Roboterbändern von Station 30 bringt, müssen alle Sensoren in der Lage sein, diese zuverlässig zu erkennen. Ist ein Sensor auf Höhe der Pins des Unterteils montiert, erkennt er bei Werkstücken mit Deckel eine steigende und eine fallende Flanke. Bei Unterteilen dagegen erkennt er entweder zwei Mal eine steigende und eine fallende Flanke oder gar nichts, wenn der Sensor zu schwach eingestellt ist. Ist der Sensor dicht über dem Transportband montiert, erkennt er bei Werkstücken mit Deckel und Unterteilen jeweils eine steigende und eine fallende Flanke. Der Sensor am zweiten Abschieber befindet sich auf Höhe der Pins. Damit Unterteile und Werkstücke mit Deckel immer an einer ähnlichen Position gestoppt werden, wenn sie sich am zweiten Abschieber befinden, muss der Sensor so verändert werden, dass er dicht über dem Transportband montiert ist. Bei der Positionierung des Sensors muss darauf geachtet werden, dass sich die Werkstücke nicht verkanten können, wenn sie abgeschoben werden. Der Sensor am Ende des Transportbandes befindet sich auch auf Höhe der Pins. Durch die Bandbegrenzung ist es nicht möglich, den Sensor dicht über dem Band zu montieren. Daher muss hier eine Software-Lösung entwickelt werden. Der Sensor muss so eingestellt werden, dass er die Pins des Unterteils zuverlässig erkennt. Gleichzeitig darf er nicht zu intensiv eingestellt werden, damit die fallende Flanke des Sensors das Transportband stoppt, bevor die Werkstücke mit Deckel vom Transportband fallen.

Visualisierung

Auf dem Touchpanel von Station 60 werden sechs neue Bilder erzeugt. Wie bei Station 20 gibt es das Bild „Steuerungen“ mit der Übersicht und Beschreibung der Steuerungsvarianten sowie vier Detailbilder, in denen die Steuerungsautomaten für jede Variante visualisiert sind. Der Steuerungsautomat der vierten Variante besitzt 25 Zustände. Eine Visualisierung aller Zustände und Zustandsübergänge ist aus Platzgründen nicht möglich. Daher sind in dem Detailbild der vierten Variante die zwei einzelnen Automaten mit je fünf Zuständen und ihren Zustandsübergängen dargestellt. In einem weiteren Bild sind alle 25 Zustände ohne Zustandsübergänge abgebildet.
In dem Bild „Automatik“ wird eine bedienbare Variable eingefügt, mit der die Zeit, die das Werkstück am ersten Abschieber wartet, bevor es abgeschoben wird, eingestellt werden kann. Dies ermöglicht dem Bediener, die Zeit in Sekunden von außen vorzugeben.

SPS Station 60

In die Schrittkette der Funktion „Auto“ muss eine Alternative zum ersten Abschieber einprogrammiert werden. Dazu wird die bestehende Schrittkette erweitert. Werkstücke, die keine Fehlteile sind, sind von den Änderungen nicht betroffen und werden wie zuvor bearbeitet. Für die Bearbeitung von Fehlteilen werden die Variablen „Abs1Erlauben“, „RoboterErlauben“ und „SchleuseErlauben“ eingefügt, die von der übergeordneten Steuerung gesetzt wer-

den. Je nachdem welche übergeordnete Steuerungsvariante aktiv ist, wird bei Fehlteilen der Abschieber oder der Roboter benutzt und das Unterteil gegebenenfalls bis zum Ende des Transportbandes durchgeschleust. Ohne die Auswahl einer übergeordneten Steuerung ist der Abschieber aktiv.
Befindet sich ein Fehlteil am ersten Abschieber, kann entweder der Roboter zum Entfernen der Platine oder der Abschieber benutzt werden. So lange der Sensor am ersten Abschieber aktiv ist, wird eine Zeit gemessen, wie lange sich das Werkstück dort befindet. Nach einer einstellbaren Zeit wird die Variable „TimerAbs1“ gesetzt und in der übergeordneten Steuerung verarbeitet. Ist die Variable „RoboterErlauben“ oder „Abs1Erlauben“ gesetzt, wartet die Schrittkette darauf, dass der Abschieber oder der Roboter fertig ist. Wurde das Werkstück durch den Roboter oder den Abschieber entfernt, wird die Schrittkette beendet. Anderenfalls fährt das Unterteil weiter bis zum zweiten Abschieber. Unterteile werden von der Kamera nicht erfasst und ausgewertet, daher muss in der Schrittkette dieser Schritt übersprungen werden, wenn die Variable „SchleuseErlauben“ gesetzt ist. Anschließend meldet Station 60 über Station 10 an Station 20, dass das Werkstück ein Unterteil für Band 1 beziehungsweise Band 2 ist. Die Entscheidung, auf welches Band das Unterteil gelegt wird, wird von der übergeordneten Steuerung getroffen. Zum Schluss wird die Schrittkette beendet.

Für die Bearbeitung von Fehlteilen mit dem Roboter wird eine neue Funktion „Fehlteil_Roboter“ erstellt. Die Schrittkette in dieser Funktion startet, wenn die Variable „RoboterErlauben“ aktiv ist und sich das Werkstück am ersten Abschieber befindet. Während der Bearbeitung der Schrittkette ist die Variable „Roboter_w“ gesetzt. Da der Roboter die Werkstücke erst nach dem ersten Abschieber bearbeiten kann, fährt das Werkstück so weit vor, bis der Sensor am ersten Abschieber eine fallende Flanke detektiert. Über jeweils eine Variable wird das Transportband gestoppt, wenn das Werkstück vom Transportband entfernt werden soll, beziehungsweise pausiert, wenn das Werkstück zum Bandende durchgeschleust werden soll. Hat das Band angehalten, wird der entsprechende Roboterbefehl über Station 10 an Station 30 gesendet. Welcher Befehl gesendet wird, wird von der übergeordneten Steuerung entschieden. Nach der Rückmeldung von Station 30, dass das Werkstück bearbeitet wurde, wird die Schrittkette beendet und das Transportband gegebenenfalls wieder gestartet.

In der Funktion „Main“ muss die Steuerung des Transportbandes um die Befehle „Band stoppen“ und „Band pausieren“ erweitert werden. Der Baustein für den ersten Abschieber darf nur ausgeführt werden, wenn auch die Variable „Abs1Erlauben“ gesetzt ist. Damit Unterteile am Bandende in der richtigen Position gestoppt werden, darf bei diesen erst auf die zweite fallende Flanke des Sensors am Bandende reagiert werden. Dazu ist eine Detektion der Flanken notwendig. Die Realisierung erfolgt über drei Setzen-Rücksetzen-Bausteine, die nur gesetzt werden, wenn Station 60 meldet, dass das Werkstück ein Fehlteil ist und wenn der jeweils vorherige Baustein gesetzt ist. Der erste Baustein wird gesetzt, wenn eine fallende Flanke am Sensor detektiert wird. Der zweite Baustein wird bei einer positiven Flanke und

der dritte bei einer negativen gesetzt. Der Baustein für die positive Flanke ist notwendig, da sonst in einem SPS-Zyklus sowohl der erste als auch der dritte Baustein gleichzeitig gesetzt würden. Die Bausteine werden zurückgesetzt, wenn Station 20 meldet, dass das Werkstück abgeholt wurde.
Die Zeit, die sich das Werkstück maximal am ersten Abschieber befindet, wird in der Funktion „Main“ für die übergeordneten Steuerungsvarianten mit Lager (Variante zwei und vier) standardmäßig auf null Sekunden festgelegt. Bei den Varianten ohne Lager (Variante eins und drei) ist die Standard-Zeit 40 Sekunden. Die 40 Sekunden ergeben sich daher, dass es bei einer externen Robotergeschwindigkeit von 50 Prozent circa 35 Sekunden dauert, bis der Roboter ein Werkstück bestückt hat, es auf einem Transportwagen abgelegt wird und dieser über den Vereinzeler freigegeben wird. Eine Standard-Zeit wird eingefügt, damit die Zeit nicht undefiniert ist, der Bediener aber dennoch die Möglichkeit hat, diese zu ändern. Durch die Art der Implementierung kann eine Änderung der Standard-Zeit im Touchpanel von Station 60 nur bei eingeschaltetem Automatikbetrieb erfolgen.

6.2. Einbindung der übergeordneten Steuerung in das TIA-Portal

Um die in Kapitel 5 erstellten Modelle in der SPS umzusetzen, wird der Codegenerator „ACArrow“ verwendet. Die Anwendung des Codegenerators auf ein „DESTool“ Projekt ist in Abschnitt 2.9.2 beschrieben. Bei Steuerungsvariante eins wird vor dem Import in „ACArrow“ im „DESTool“ Projekt das Gesamtmodell G der Steuerstrecke als „Plant Model“ und der reduzierte Supervisor S_{Red} als „Supervisor Model“ deklariert. Bei den Steuerungsvarianten zwei bis vier wird die Codegenerierung für jeden reduzierten Supervisor separat ausgeführt, um diese später auf verschiedenen Steuerungen zu implementieren. In diesem Fall gibt es jeweils zwei Supervisor pro Variante, sodass die Codegenerierung insgesamt sechs Mal durchgeführt werden muss. Dabei werden für $j = 1, 2$ die zugehörigen lokalen Systeme G_{Xj} als „Plant Model“ und der reduzierte Supervisor $S_{Red\,j}$ als „Supervisor Model“ deklariert. Anschließend wird die Codegenerierung für Variante eins mit dem modularen und für Variante zwei bis vier mit dem lokal-modularen Ansatz gestartet. Der modulare Ansatz für Variante eins kann verwendet werden, da der monolithische ein Sonderfall des modularen Ansatzes ist.

Der durch die Codegenerierung entstandene SCL Quelltext wird im TIA-Portal unter „Externe Quellen“ eingefügt. Über „Bausteine aus Quelle generieren“ im Kontextmenü der rechten Maustaste wird ein Baustein aus der externen Quelle erzeugt. Die bei der lokal-modularen Codegenerierung entstandenen Excel-Dateien mit Variablen werden unter „PLC-Variablen“ importiert. Für den monolithischen Ansatz müssen die Variablen manuell erstellt werden.

Da die Variablen in der untergeordneten Steuerung, die in Kapitel 3 beschrieben ist, andere Namen haben als die Eingangsvariablen aus der übergeordneten Steuerung, die durch die Codegenerierung entstanden ist, muss für jede Steuerungsvariante eine Funktion eingefügt werden, die die Zuweisung übernimmt. Da jeweils nur eine Steuerungsvariante gleichzeitig laufen kann, können für verschiedene Varianten dieselben Variablen aus der untergeordneten Steuerung genutzt werden. Dadurch reduziert sich die Anzahl der benötigten Variablen. In den Funktionen wird neben der Zuweisung der Eingänge auch die Hauptfunktion der Steuerungsvariante aufgerufen. Eine Zuweisung der steuerbaren Ereignisse aus der übergeordneten Steuerung zu den Variablen in der untergeordneten Steuerung erfolgt beim monolithischen Ansatz nach dem Aufruf der Hauptfunktion. Beim lokal-modularen Ansatz erfolgt die Zuweisung in der Hauptfunktion. Damit beim Ausschalten des Automatikbetriebes die steuerbaren Ereignisse der übergeordneten Steuerung keine unerwarteten Vorgänge auslösen, werden diese sowie das Bit „InitBit“ beim Ausschalten des Automatikbetriebes zurückgesetzt. Die Funktionen für die Steuerungsvarianten werden aus dem Hauptprogramm der untergeordneten Steuerung aufgerufen. Dazu gibt es eine Funktion „Steuerungen“, die abhängig von der Eingabe auf dem Touchpanel von Station 20 die jeweilige Funktion der Steuerungsvariante aufruft. Der Aufruf der Funktion „Steuerungen“ muss vor dem Aufruf der Funktion „Auto“ erfolgen, da es sonst dazu kommen kann, dass für einen Zyklus falsche Variablen von der Steuerung gesetzt sind. Das führt zum Beispiel dazu, dass die Linearachse in dem Zyklus den Befehl bekommt, ein neues Werkstück von Station 20 zu Station 30 zu bringen, obwohl die Steuerung dies im nächsten Zyklus verbietet.

7. Auswertung

Für die in Kapitel 6 eingefügte übergeordnete Steuerung werden in diesem Kapitel verschiedene Testfälle entworfen und untersucht. Für alle Messungen gelten die in Abschnitt 5.1 beschriebenen Voraussetzungen. Sobald eine Voraussetzung nicht erfüllt ist, ist die Messung ungültig und kann nicht ausgewertet werden. Variable Stellgrößen sind die Robotergeschwindigkeit, die maximale Wartezeit am ersten Abschieber an Station 60, die Steuerungsvariante und die Anzahl beziehungsweise Anordnung der fehlerhaften Platinen im Magazin.

Zur Auswertung werden in die beiden Funktionen „Auto“ von Station 20 und 30 insgesamt drei Timer-Bausteine eingefügt. Der erste Baustein misst die Zeit, wie lange ein Auftragsbit für die Linearachse gesetzt ist. Ein zweiter misst die Zeit, wie lange die Variable für die Schlüsselmarke des Roboters gesetzt ist. Der dritte Baustein misst so lange die Zeit, bis die beiden Istwertzähler für ein- und zweipolige Werkstücke den Wert der Sollwertzähler erreicht haben und die Linearachse den letzten Auftrag beendet hat, indem sie wieder in ihrer Ruheposition angekommen ist.

In den Abschnitten 7.1 und 7.2 werden zunächst die Wiederholgenauigkeit der Messungen und der Einfluss der Zeit, die ein Werkstück am ersten Abschieber wartet, bevor es abgeschoben oder eingelagert wird, untersucht. Abschnitt 7.3 vergleicht in verschiedenen Szenarien die vier Steuerungsvarianten miteinander. Abschnitt 7.4 fasst das Ergebnis der Messungen zusammen und gibt anhand dessen eine Empfehlung, welche Variante am besten zur Steuerung geeignet ist.

7.1. Wiederholgenauigkeit der Messungen

Neben den Voraussetzungen für den Ablauf der Messungen muss für den Vergleich sichergestellt sein, dass die Timer-Bausteine bei gleichen Eingangsvoraussetzungen ähnliche Zeiten messen. Dazu werden fünf Durchläufe gemessen, bei denen jeweils fünf ein- und fünf zweipolige Platinen in das Magazin eingelegt werden. Die Robotergeschwindigkeit wird auf 50 Prozent eingestellt, die maximale Wartezeit am ersten Abschieber und die Steuerungsvariante sind nicht relevant, da keine fehlerhaften Platinen bearbeitet werden. Tabelle 7.1 zeigt die Zeiten, die die drei Timer-Bausteine gemessen haben. Der Prozentwert gibt an wie lange die Linearachse oder der Roboter bezogen auf die Gesamtzeit aktiv sind.

Nr.	Var.	Zeit gesamt	Linearachse		Roboter	
			Zeit	Prozent	Zeit	Prozent
1	-	10m 38s 469ms	8m 44s 708ms	82,2%	2m 37s 120ms	24,6%
2	-	10m 38s 576ms	8m 44s 913ms	82,2%	2m 37s 129ms	24,6%
3	-	10m 38s 291ms	8m 45s 081ms	82,3%	2m 37s 124ms	24,6%
4	-	10m 38s 134ms	8m 45s 368ms	82,3%	2m 37s 084ms	24,6%
5	-	10m 38s 108ms	8m 45s 411ms	82,3%	2m 37s 120ms	24,6%

Tabelle 7.1.: Wiederholgenauigkeit der Messungen

An den Werten lassen sich die maximalen Zeitunterschiede zwischen den Messungen ablesen. Bei der Gesamtzeit beträgt er 468 Millisekunden, bei der Linearachse 703 Millisekunden und beim Roboter 45 Millisekunden. Bezogen auf die Mittelwerte der fünf Messungen liegt die Abweichung bei maximal 0,2 Prozent. Die Werte können also als vergleichbar angesehen werden.

7.2. Einfluss der maximalen Wartezeit am ersten Abschieber

In den nachfolgenden Messungen wird der Einfluss der Zeit, die ein Werkstück am ersten Abschieber von Station 60 wartet, bevor es abgeschoben oder eingelagert wird, untersucht. Im Magazin befinden sich fünf ein- und fünf zweipolige Platinen. Die zweite und dritte Platine ist jeweils fehlerhaft, sodass sich insgesamt vier fehlerhafte Platinen im Prozess befinden. Die übergeordneten Steuerungsvarianten eins, drei und vier werden jeweils untersucht. Für Variante zwei hat die Zeit keinen Einfluss, da jedes Werkstück sofort eingelagert wird, sofern das Lager nicht voll ist. Tabelle 7.2 zeigt die Zeiten der drei Timer-Bausteine sowie Informationen zur Steuerungsvariante, der maximalen Wartezeit am ersten Abschieber und dem Ausschuss an Unterteilen bei einer Robotergeschwindigkeit von 50 Prozent. Tabelle 7.3 zeigt dieselben Parameter bei einer Robotergeschwindigkeit von 100 Prozent. Es wird jeweils nur die minimale (null Sekunden) und maximale Wartezeit (40 Sekunden) am ersten Abschieber miteinander verglichen, da bei einer externen Robotergeschwindigkeit von mindestens 50 Prozent innerhalb 40 Sekunden in jedem Fall ein Roboterband frei wird, es aber gleichzeitig nicht bedeutet, dass das Werkstück immer 40 Sekunden am ersten Abschieber wartet. Sobald ein Auftrag möglich ist, erfolgt sofort eine Bearbeitung des Werkstücks. Sind keine Aufträge mehr vorhanden, wird der Befehl zum Abschieben beziehungsweise Einlagern auch bei einer Wartezeit von 40 Sekunden sofort erteilt.

Nr.	Var.	Zeit Abs1	Zeit gesamt	Linearachse		Roboter		Ausschuss
				Zeit	Prozent	Zeit	Prozent	
1	1	0s	12m 12s 741ms	09m 38s 561ms	79,0%	4m 45s 253ms	38,9%	2
2	1	40s	11m 44s 498ms	08m 50s 707ms	75,3%	3m 26s 597ms	29,3%	-
3	3	0s	12m 53s 297ms	10m 16s 114ms	79,7%	4m 05s 964ms	31,8%	1
4	3	40s	13m 01s 690ms	10m 10s 825ms	78,1%	4m 14s 696ms	32,6%	-
5	4	0s	12m 39s 548ms	08m 47s 369ms	69,4%	6m 03s 702ms	47,9%	-
6	4	40s	11m 44s 498ms	08m 50s 707ms	75,3%	3m 26s 597ms	29,3%	-

Tabelle 7.2.: Einfluss der maximalen Wartezeit am ersten Abschieber bei 50% Robotergeschwindigkeit

Nr.	Var.	Zeit Abs1	Zeit gesamt	Linearachse		Roboter		Ausschuss
				Zeit	Prozent	Zeit	Prozent	
1	1	0s	10m 53s 454ms	9m 11s 248ms	84,4%	3m 03s 639ms	28,1%	1
2	1	40s	10m 29s 705ms	8m 42s 915ms	83,0%	2m 54s 247ms	27,7%	-
3	3	0s	11m 52s 478ms	10m 18s 361ms	86,8%	2m 23s 108ms	20,1%	1
4	3	40s	11m 13s 397ms	10m 14s 686ms	91,3%	2m 28s 752ms	22,1%	-
5	4	0s	10m 52s 522ms	8m 42s 627ms	80,1%	3m 29s 974ms	32,2%	-
6	4	40s	10m 29s 705ms	8m 42s 915ms	83,0%	2m 54s 247ms	27,7%	-

Tabelle 7.3.: Einfluss der maximalen Wartezeit am ersten Abschieber bei 100% Robotergeschwindigkeit

An den Zeiten von Variante eins ist zu erkennen, dass die Gesamtproduktionszeit bei einer längeren Wartezeit am ersten Abschieber um circa 28 Sekunden bei 50 Prozent Robotergeschwindigkeit und um circa 24 Sekunden bei 100 Prozent Robotergeschwindigkeit reduziert

wird, da die Linearachse nicht erst ein neues Unterteil von den Zufuhrbändern holen muss. Des Weiteren ist der Ausschuss bei einer längeren Wartezeit um zwei beziehungsweise eins kleiner. Bei Variante drei und 50 Prozent Robotergeschwindigkeit ist die Gesamtzeit bei 40 Sekunden Wartezeit am ersten Abschieber circa neun Sekunden langsamer als bei null Sekunden. Wird jedoch die Robotergeschwindigkeit auf 100 Prozent erhöht, ist die Gesamtzeit mit längerer Wartezeit circa 40 Sekunden schneller als bei kürzerer Wartezeit. Auch der Ausschuss bei längerer Wartezeit ist jeweils um eins geringer. Bei der vierten Variante ist die Gesamtzeit bei einer längeren Wartezeit und 50 Prozent Robotergeschwindigkeit um circa 56 Sekunden und bei 100 Prozent Robotergeschwindigkeit um circa 23 Sekunden schneller als beim direkten Einlagern. Bei beiden Wartezeiten gibt es keinen Ausschuss, da Werkstücke im Lager eingelagert werden können.

Aufgrund der Erkenntnisse der Messungen, dass bei einer längeren Wartezeit am ersten Abschieber sowohl der Ausschuss als auch fast immer die Gesamtproduktionszeit reduziert wird, werden für die Steuerungsvarianten ein und drei jeweils eine Wartezeit von 40 Sekunden eingestellt. Bei Variante vier wird eine Wartezeit von null Sekunden gewählt, damit sich gegenüber 40 Sekunden Wartezeit kein Rückstau vor Station 60 bilden kann, obwohl ein Lager vorhanden ist. Außerdem gäbe es kaum einen Unterschied zu Variante eins, wenn hier eine Wartezeit von 40 Sekunden eingestellt wird. Der einzige Unterschied wäre die Möglichkeit, Unterteile einzulagern, wenn keine neuen Aufträge vorhanden sind.

7.3. Vergleich der vier Steuerungsvarianten

In diesem Abschnitt werden die vier übergeordneten Steuerungsvarianten miteinander verglichen. Als Referenzzeit werden bei jeder Messung die Zeiten angegeben, die die einzelnen Timer-Bausteine ohne Steuerungsvariante messen. Ist keine fehlerhafte Platine im Prozess, erzielen alle Steuerungsvarianten dieselben Ergebnisse. Aus diesem Grund werden ausschließlich Situationen mit Fehlteilen aufgenommen und miteinander verglichen. Alle Messungen werden bei einer Robotergeschwindigkeit von 100 Prozent aufgenommen. Zunächst wird ein üblicher Produktionsprozess betrachtet, bei dem von zehn Platinen eine fehlerhafte vorhanden ist. Dies entspricht einer Ausschussquote von zehn Prozent. Tabelle 7.4 zeigt die Zeiten der drei Timer-Bausteine für den Fall, dass sich im Magazin fünf ein- und fünf zweipolige Platinen befinden. Die vierte einpolige Platine ist jeweils fehlerhaft.

An den Gesamtzeiten ist zu erkennen, dass die Produktionszeiten der einzelnen Varianten sehr ähnlich sind. Bezogen auf die Referenzzeit mit Abschieber sind alle Varianten mindestens 24 Sekunden schneller. Variante eins und vier sind gleich schnell, da kein Unterteil während der Produktion eingelagert wird. Die Variante mit Linearachse ist von den vier Varianten am langsamsten. Auch die Auslastung der Linearachse liegt bei fast 100 Prozent, sodass hier so gut wie keine Reserven vorhanden sind. Die Auslastung des Roboters ist bei

Nr.	Var.	Zeit Abs1	Zeit gesamt	Linearachse		Roboter		Ausschuss
				Zeit	Prozent	Zeit	Prozent	
1	-	0s	10m 46s 165ms	9m 12s 587ms	85,5%	1m 39s 754ms	15,4%	4
2	1	40s	10m 01s 642ms	8m 44s 700ms	87,2%	1m 58s 655ms	19,7%	-
3	2	0s	10m 09s 414ms	8m 45s 172ms	86,2%	2m 06s 242ms	20,7%	-
4	3	40s	10m 21s 940ms	9m 75s 915ms	99,0%	1m 45s 256ms	16,9%	-
5	4	0s	10m 02s 037ms	8m 45s 628ms	87,3%	1m 58s 696ms	19,7%	-

Tabelle 7.4.: Unterschied der Steuerungsvarianten bei einem Fehlteil

allen Varianten mit circa 15 bis 21 Prozent sehr gering. Am schnellsten sind die Varianten eins, zwei und vier. Auch die Auslastung der Linearachse bietet mit circa 85 Prozent noch Reserven.

In einem zweiten Durchlauf wird die Anzahl der Fehlteile erhöht. Von fünf ein- und fünf zweipoligen Platinen im Magazin sind jeweils die zweite und dritte fehlerhaft. Dieser Fall könnte in der Realität eintreten, wenn für kurze Zeit ein Fehler in der Produktion der Platinen ist, sodass fehlerhafte Platinen dem Prozess zugeführt werden. Nach einiger Zeit wird dieser Fehler entdeckt und behoben, sodass die Produktion mit fehlerfreien Platinen fortgesetzt wird. Tabelle 7.5 zeigt die Zeiten der drei Timer-Bausteine für die vier Steuerungsvarianten mit diesem Szenario.

Auch bei diesem Szenario ist zu erkennen, dass die Gesamtzeit ohne Steuerungsvariante am langsamsten ist. Die dritte Variante liegt allerdings zeitlich nur circa zwei Sekunden dahinter. Allerdings ist der Ausschuss ohne Steuerungsvariante mit vier Unterteilen deutlich höher als bei Variante drei, bei der es keinen Ausschuss gibt. Am schnellsten ist die Variante eins, die circa zwölf Sekunden schneller ist als die zweitschnellste Variante zwei. Die Auslastung der Linearachse ist bei den Varianten eins, zwei und vier, bei denen der Roboter verwendet wird, reduziert, sodass mehr Reserven vorhanden sind. Bei diesen Varianten wird dafür der Roboter um circa 50 bis 90 Sekunden länger verwendet. Da seine Auslastung jedoch nur bei maximal 33,7 Prozent liegt, sind immer noch ausreichend Reserven vorhanden.

In einem dritten Durchlauf wird jeweils eine Variante mit Lager mit einer ohne verglichen. Dabei ist von fünf ein- und fünf zweipoligen Platinen im Magazin die letzte zweipolige Platine fehlerhaft. Da am Ende der Produktion keine neuen Aufträge vorhanden sind, kann für die Auswertung ohne Lager sowohl Variante eins als auch Variante drei genutzt werden. Für die

Nr.	Var.	Zeit Abs1	Zeit gesamt	Linearachse		Roboter		Ausschuss
				Zeit	Prozent	Zeit	Prozent	
1	-	0s	11m 14s 172ms	10m 30s 302ms	93,5%	2m 06s 727ms	18,8%	4
2	1	40s	10m 29s 705ms	08m 42s 915ms	83,0%	2m 54s 247ms	27,7%	-
3	2	0s	10m 41s 947ms	08m 43s 057ms	81,5%	3m 36s 175ms	33,7%	-
4	3	40s	11m 11s 951ms	10m 12s 213ms	91,1%	2m 28s 554ms	22,1%	-
5	4	0s	10m 52s 522ms	08m 42s 627ms	80,1%	3m 29s 974ms	32,2%	-

Tabelle 7.5.: Unterschied der Steuerungsvarianten bei vier Fehlteilen

Auswertung mit Lager kann Variante zwei oder vier verwendet werden. Nachdem alle zehn Produkte produziert worden sind, wird erneut ein einpoliges Produkt in Auftrag gegeben. Ist ein Lager vorhanden, kann das zuvor eingelagerte Unterteil verwendet werden, anderenfalls muss ein neues Unterteil von den Zufuhrbändern geholt werden. Tabelle 7.6 zeigt die Zeiten der drei Timer-Bausteine. Zeile eins und drei sind jeweils die Zeiten nach den ersten zehn Produkten, Zeile zwei und vier jeweils die Zeiten, nachdem ein weiteres Werkstück in Auftrag gegeben wurde.

Nr.	Var.	Zeit Abs1	Zeit gesamt	Linearachse		Roboter		Ausschuss
				Zeit	Prozent	Zeit	Prozent	
1	1/3	40s	11m 11s 520ms	09m 10s 969ms	82,0%	1m 39s 398ms	14,8%	1
2	1/3	40s	14m 26s 982ms	10m 05s 517ms	69,8%	1m 48s 637ms	12,5%	1
3	2/4	0s	11m 11s 978ms	09m 10s 978ms	82,0%	1m 55s 999ms	17,3%	-
4	2/4	0s	14m 10s 030ms	09m 37s 810ms	68,0%	2m 15s 137ms	15,9%	-

Tabelle 7.6.: Vergleich der Varianten mit und ohne Lager

Der Tabelle ist zu entnehmen, dass die Gesamtproduktionszeit nach den ersten zehn Produkten gleich ist, da das Einlagern parallel zu der Bearbeitung des letzten Werkstücks erfolgt. Die Zeit, die der Roboter aktiv ist, ist bei der Variante mit Lager etwas größer, da ein Unterteil

eingelagert werden muss. Der Ausschuss bei der Variante ohne Lager beträgt ein Unterteil. An den Ergebnissen nach dem elften Produkt ist zu erkennen, dass nicht nur der Ausschuss bei einer Variante mit Lager reduziert wird, sondern auch die gesamte Produktionszeit um circa 17 Sekunden reduziert wird. Aus diesem Grund ist es sinnvoll, ein Lager einzusetzen, wenn der Prozess zeitweise unterbrochen und zu einem späteren Zeitpunkt wieder fortgesetzt werden soll.

7.4. Ergebnis

Anhand der Auswertungen in Abschnitt 7.3 können alle Varianten zur Steuerung von Fehlteilen in der Modellfabrik verwendet werden, da die Gesamtproduktionszeiten nur geringfügig voneinander abweichen. Sollen freie Reserven bei der Linearachse vorhanden sein, eignet sich Variante drei nicht, da die Auslastung in den Messungen mit 82 bis 99 Prozent durchgehend sehr hoch ist. Die beste Gesamtproduktionszeit in allen Messungen erzielt Variante eins mit einer Wartezeit am ersten Abschieber von 40 Sekunden. Um den Ausschuss zu reduzieren, sollte jedoch zusätzlich ein Lager verwendet werden. Dies entspricht Variante vier mit einer Wartezeit am ersten Abschieber von 40 Sekunden. Diese Variante erzielt die schnellsten Ergebnisse, gleichzeitig steht aber auch ein Lager zur Verfügung.

8. Zusammenfassung und Ausblick

In dieser Thesis wurde die Steuerung der Modellfabrik optimiert und erweitert. An den Stationen 30 und 60 wurden Betriebsabläufe beschleunigt und parallelisiert. An Station 60 wurde eine Alternative für die Kamera entwickelt, sodass der Bediener über die Taster am Bedienpult nun in der Lage ist, Werkstücke als fehlerhaft zu markieren und abzuschieben. Der Produktzähler an Station 20 wurde optimiert, sodass beispielsweise der Istwertzähler die bereits produzierten und nicht die fehlenden Werkstücke zählt, was der Bediener intuitiv vermutet. Bei fehlerhaften Werkstücken, die aus dem Prozess abgeschoben werden, wird automatisch ein neues Werkstück produziert. Außerdem können neue Aufträge erteilt werden, ohne den Prozess zu unterbrechen. An Station 50 wurde eine Erkennung für fehlende Deckel hinzugefügt, sodass diese automatisch aus dem Prozess abgeschoben und neu produziert werden können.

Zusätzlich wurden vier Möglichkeiten entwickelt, bei elektrisch fehlerhaften Werkstücken die Platine zu entnehmen und die Unterteile wieder dem Prozess zuzuführen. In der ersten Steuerungsvariante wurde der direkte Weg von Station 60 zu Station 30 über den Roboter umgesetzt. Für die zweite Steuerungsvariante wurde ein Lager entworfen, in dem die Unterteile zwischengelagert werden können. Auch hier wird der Roboter für den Transport verwendet. In der dritten Steuerungsvariante entnimmt der Roboter die Platine, das Unterteil wird jedoch von der Linearachse von Station 60 zu Station 30 transportiert. Die letzte Variante ist die Kombination aus dem direkten Weg und dem Lager.
In der Auswertung wurden die verschiedenen Ansätze in unterschiedlichen Anwendungsfällen getestet und ausgewertet. Es wurde unter anderem der Einfluss der Zeit, die das Werkstück am ersten Abschieber wartet, bevor es abgeschoben wird, untersucht. Anschließend wurden die Steuerungsansätze bei gleichen Ausgangssituationen miteinander verglichen. Es hat sich herausgestellt, dass die Gesamtproduktionszeit bei einer längeren Wartezeit geringer und der Ausschuss weniger wird. Die erste Steuerungsvariante über den direkten Weg mit einer Wartezeit am ersten Abschieber von 40 Sekunden war bei allen Messungen am schnellsten. Aus diesem Grund ist dieser Ansatz zu empfehlen, allerdings sollte zur Verringerung des Ausschusses ein Lager verwendet werden. Dies entspricht Steuerungsvariante vier mit einer Wartezeit am ersten Abschieber von 40 Sekunden.

In weiterführenden Arbeiten kann die Kamera in den Prozess eingebunden werden, sodass optisch fehlerhafte Werkstücke automatisch erkannt werden. Außerdem kann untersucht

werden, inwieweit Werkstücke, bei denen nur ein Aufkleber verpresst wurde, von der Kamera erkannt werden können. Zusätzlich kann die Optimierung der Modellfabrik fortgesetzt werden. Beispielsweise kann untersucht werden, ob die Linearachse beschleunigt werden kann. Des Weiteren können die Schrittketten an den Stationen 30 und 60 weiter aufgelöst werden, sodass mehrere Werkstücke parallel auf den Transportbändern bearbeitet werden können. Als Steuerungsvarianten können die Kombination aus der zweiten Variante über das Lager und der dritten Variante über die Linearachse oder eine neue Variante, bei der sowohl der Roboter als auch die Linearachse für den Transport der Unterteile zu Station 30 verwendet werden, untersucht werden. Außerdem kann ein Lager für Unterteile in Reichweite der Linearachse geschaffen werden, sodass Unterteile auch nach dem Entfernen der Platine eingelagert werden können. Gegebenenfalls kann geprüft werden, ob die Linearachse Unterteile wieder auf die Zufuhrbänder von Station 20 legen kann. In diesem Fall müssen am Bandanfang der Zufuhrbänder allerdings zusätzliche Sensoren angebracht werden, sodass keine Unterteile vom Transportband geschoben werden können.

Literaturverzeichnis

[Abel und Bollig 2006] ABEL, Dirk ; BOLLIG, Alexander: *Rapid Control Prototyping Methoden und Anwendungen*. Springer-Verlag Berlin Heidelberg, 2006. – ISBN 978-3-540-29524-2

[Cassandras und Lafortune 2008] CASSANDRAS, Christos G. ; LAFORTUNE, Stéphane: *Introduction to Discrete Event Systems*. Second Edition. Springer Science+Business Media, LLC, New York, 2008. – ISBN 978-1-4419-4119-0

[Cramer und Nešlehová 2015] CRAMER, Erhard ; NEŠLEHOVÁ, Johanna: *Vorkurs Mathematik*. 6., überarbeitete Auflage. Springer-Verlag Berlin Heidelberg, 2015. – ISBN 978-3-662-46399-4

[DIN IEC 60050-351 2014] DEUTSCHES INSTITUT FÜR NORMUNG E.V. (DIN): *DIN IEC 60050-351 ; Internationales Elektrotechnisches Wörterbuch - Teil 351: Leittechnik*. September 2014

[Erk und Priese 2008] ERK, Katrin ; PRIESE, Lutz: *Theoretische Informatik*. 3., erweiterte Auflage. Springer-Verlag Berlin Heidelberg, 2008. – ISBN 978-3-540-76319-2

[Feng und Wonham 2006] FENG, Lei ; WONHAM, W. M.: TCT: A Computation Tool for Supervisory Control Synthesis. In: *Proceedings - Eighth International Workshop on Discrete Event Systems, WODES 2006*, 08 2006, S. 388 – 389

[Gohert 2014] 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: 21.04.2017

[Güting und Erwig 1999] GÜTING, Ralf H. ; ERWIG, Martin: *Übersetzerbau Techniken, Werkzeuge, Anwendungen*. Springer-Verlag Berlin Heidelberg, 1999. – ISBN 978-3-540-65389-9

[Hofmann und Lange 2011] HOFMANN, Martin ; LANGE, Martin: *Automatentheorie und Logik*. Springer-Verlag Berlin Heidelberg, 2011. – ISBN 978-3-642-18089-7

[Hopcroft u. a. 2011] HOPCROFT, John E. ; MOTWANI, Rajeev ; ULLMAN, Jeffrey D.: *Einführung in Automatentheorie, Formale Sprachen und Berechenbarkeit*. 3. aktualisierte Auflage. Pearson Education Deutschland GmbH München, 2011. – ISBN 978-3-86894-082-4

[Iordache 2013] IORDACHE, Ph. D. M.: *SPNBOX - A Toolbox for the Supervisory Control of Petri Nets*. 2013. – URL `http://www.letu.edu/people/marianiordache/abs/spnbox/`. – Zugriffsdatum: 20.11.2017

[León und Kiencke 2013] LEÓN, Fernando P. ; KIENCKE, Uwe: *Ereignisdiskrete Systeme Modellierung und Steuerung verteilter Systeme*. 3. vollständig überarbeitete und ergänzte Auflage. Oldenbourg Wissenschaftsverlag GmbH München, 2013. – ISBN 978-3-486-73574-1

[Litz 2013] LITZ, Lothar: *Grundlagen der Automatisierungstechnik Regelungssysteme - Steuerungssysteme - Hybride Systeme*. 2. aktualisierte Auflage. Oldenbourg Wissenschaftsverlag GmbH München, 2013. – ISBN 978-3-486-70888-2

[Litz und Frey 1999] LITZ, Lothar ; FREY, Georg: *Methoden und Werkzeuge zum industriellen Steuerungsentwurf - Historie, Stand, Ausblick*. Automatisierungstechnik 47 (1999) Nr. 4, R. Oldenbourg Verlag. 1999. – URL `http://www.aut.uni-saarland.de/uploads/media/LL_GF_at_apr_1999.pdf`. – Zugriffsdatum: 04.07.2017

[Lunze 2012a] LUNZE, Jan: *Automatisierungstechnik Methoden für die Überwachung und Steuerung kontinuierlicher und ereignisdiskreter Systeme*. 3. überarbeitete Auflage. Oldenbourg Wissenschaftsverlag GmbH München, 2012. – ISBN 978-3-486-71266-7

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

[Modler und Kreh 2014] MODLER, Florian ; KREH, Martin: *Tutorium Analysis 1 und Lineare Algebra 1*. 3. Auflage. Springer-Verlag Berlin Heidelberg, 2014. – ISBN 978-3-642-37365-7

[Moor 2016] MOOR, Thomas: *DESTool*. 2016. – URL `http://www.rt.eei.uni-erlangen.de/FGdes/destool/`. – Zugriffsdatum: 14.07.2017

[Ramadge 1983] RAMADGE, P. J.: *Control and Supervision of Discrete Event Processes*, University of Toronto, Kanada. Dep. of Electrical and Computer Engineering, Dissertation, 1983

[Su und Wonham 2004] SU, R. ; WONHAM, W. M.: *Supervisor Reduction for Discrete-Event Systems*. Discrete Event Dynamic Systems: Theory and Applications 14 Nr. 1. 2004

[Uhlig 2006] UHLIG, Reiner: *SPS - Modellbasierter Steuerungsentwurf für die Praxis*. Oldenbourg Industrieverlag GmbH München, 2006. – ISBN 3-486-63088-1

[Wenck 2006] WENCK, Florian: *Modellbildung, Analyse und Steuerungsentwurf für gekoppelte ereignisdiskrete Systeme*, Ruhr-Universität Bochum, Dissertation, 2006

[Wilhelm u. a. 2012] WILHELM, Reinhard ; SEIDL, Helmut ; HACK, Sebastian: *Übersetzerbau*. Bd. 2: Syntaktische und semantische Analyse. Springer-Verlag Berlin Heidelberg, 2012. – ISBN 978-3-642-01134-4

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

[Wonham 2016] WONHAM, W. M.: *W.M. Wonham's Homepage*. 2016. – URL `http://www.control.toronto.edu/~wonham/wonham.html`. – Zugriffsdatum: 20.11.2017

[Zander 2015] ZANDER, Hans-Joachim: *Steuerung ereignisdiskreter Prozesse*. Springer Fachmedien Wiesbaden, 2015. – ISBN 978-3-658-01381-3

Anhang

Der Anhang dieser Thesis befindet sich zum Teil auf DVD und kann beim Erst- oder Zweiprüfer eingesehen werden.

Anhang A: Bedienung der Modellfabrik (Thesis und DVD)
Dieser Anhang beinhaltet eine Anleitung, wie die Modellfabrik ein- und ausgeschaltet wird sowie in den Standby Betrieb gebracht beziehungsweise von dort wieder gestartet wird. Die Anleitung dient zur gezielten Inbetriebnahme (siehe Abschnitt 5.1) sowie als Leitfaden für Studierende, die an der Modellfabrik arbeiten. Innerhalb der Anleitung gibt es Verweise auf die Bedienung des Roboters, die in Anhang B zu finden ist.

Anhang B: Bedienung des Roboters (Thesis und DVD)
In diesem Anhang wird die Bedienung des Roboters erklärt.

Anhang C: DESTool Modelle (DVD)
Dieser Anhang beinhaltet für jede der vier Steuerungsvarianten das zugehörige „DESTool“ Projekt. Die Modelle in diesen Projekten sind in Kapitel 5 beschrieben.

Anhang D: Strecken- und Spezifikationsmodelle als PDF (DVD)
Dieser Anhang beinhaltet alle Strecken- und Spezifikationsmodelle aus Kapitel 5 als PDF.

Anhang E: TIA-Gesamtprojekt der Modellfabrik (DVD)
Das in dieser Thesis optimierte und erweiterte TIA-Gesamtprojekt aus Kapitel 6 befindet sich in diesem Anhang.

Anhang F: Quelltexte aus dem TIA-Gesamtprojekt (DVD)
Dieser Anhang enthält alle Quelltexte für die einzelnen Stationen der Modellfabrik aus dem TIA-Gesamtprojekt. Pro Station gibt es ein XPS-Dokument, in dem alle Objektbausteine, Funktionen und Funktionsbausteine dargestellt sind. Für die Funktionen der übergeordneten Steuerungen aus der Codegenerierung an Station 20 und 60 gibt es eigene XPS-Dokumente. Zur Übersicht gibt es ein separates Inhaltsverzeichnis, welches die Funktionen in den XPS-Dokumenten mit Seitenzahl auflistet.

Anhang G: Roboterprogramm der Modellfabrik (DVD)
Das zugehörige Roboterprogramm zu dem TIA-Gesamtprojekt der Modellfabrik befindet sich in diesem Anhang. Im Rahmen dieser Thesis wurden keine Änderungen im Roboterprogramm vorgenommen.

A. Bedienung der Modellfabrik

Modellfabrik einschalten

1. Hauptschalter einschalten
2. Warten bis Kamera einmal aufgeblinkt hat (anschließend leuchtet die Power LED grün und die Ethernet LED blinkt grün)
3. ggf. Programme auf SPS laden
4. nach Laden ggf. Umrichter quittieren
    a. am Umrichter von ST 10 über „OK“ Taste Fehler anzeigen lassen und über „Acknowledge all“ Fehler quittieren, dann über „ESC“ Taste zurück zum Hauptbildschirm
    b. am Umrichter von ST 20 „FN“ Taste drücken)
5. Umrichter an Station 20 über den „0-1“ Schalter kurz einschalten (steigende Flanke)
6. Roboter über Kippschalter an Robotersteuerung einschalten (während des Startvorgangs blinken die Tasten „MOTOR“ und „LOCK“ auf dem Programmierhandgerät)
7. Stationen freigeben und quittieren
    a. ST 20 und ST 60 können nur über ST 30 freigegeben werden
    b. beim Quittieren von ST 10 wird Luft auf die Anlage gegeben
8. ggf. an Station 20 Steuerung über Touchpanel auswählen („Steuerungen“ → „Auswahl Steuerung“)
9. Prüfen, ob sich alle Komponenten in Home-Position befinden und ob sich keine Werkstücke mehr in der Anlage befinden. Wenn nein, Komponenten in Home-Position fahren beziehungsweise Werkstücke aus der Anlage entfernen
10. Stationen 30 bis 60 in beliebiger Reihenfolge starten
11. Roboter starten (siehe Anleitung „Bedienung des Roboters“)
12. Station 10 starten
13. Transportsystem an Station 10 über den „0-1“ Schalter einschalten
14. Umrichter an Station 20 über den „0-1“ Schalter einschalten
15. Über Touchpanel von Station 20 im Bild „Automatikbetrieb“ Auftrag vorgeben (Button „ST 20“ muss ausgewählt sein)
16. Station 20 starten

Modellfabrik in Standby bringen

1. Transportsystem an Station 10 über den „0-1“ Schalter ausschalten
2. Roboterprogramm stoppen (siehe Anleitung „Bedienung des Roboters“)
3. Umrichter an Station 20 über den „0-1“ Schalter ausschalten

Modellfabrik aus Standby starten

1. Roboterprogramm starten (siehe Anleitung „Bedienung des Roboters“)
2. Transportsystem an Station 10 über den „0-1“ Schalter einschalten
3. Umrichter an Station 20 über den „0-1“ Schalter einschalten
4. Über Touchpanel von Station 20 im Bild „Automatikbetrieb“ Auftrag vorgeben (Taste „ST 20“ muss ausgewählt sein)
5. Taste „S5“ drücken (wenn Automatikbetrieb bereits aktiv war) sonst Taste „Start“ drücken

Modellfabrik ausschalten

1. Alle Stationen stoppen
2. Transportsystem an Station 10 über den „0-1“ Schalter ausschalten
3. Umrichter an Station 20 über den „0-1“ Schalter ausschalten
4. Roboter ausschalten (siehe Anleitung „Bedienung des Roboters“)
5. ggf. über „NOTAUS“ an Station 10 Luft abblasen („NOTAUS“ danach wieder entsperren)
6. PCs herunterfahren
7. Hauptschalter ausschalten

B. Bedienung des Roboters

Bedienung des Roboters

Roboter starten:

1. Kippschalter an Robotersteuerung ein

Programm starten:

1. Roboter am Home-Sensor? Nein → erst dort hinfahren!
2. Schlüsselschalter am Programmierhandgerät auf „Auto“
3. „F1“ oder „Programm“ drücken
4. „Programm für Modellfabrik“ auswählen
5. „Start“ drücken
6. „Einzelzyklus“ oder „Kontinuierlich“ auswählen (egal)
7. Über „Speed“ Geschwindigkeit auswählen und bestätigen

Programm stoppen und resetten:

1. „Halt“ oder Taste „Stop“ drücken
2. Taste „Motor“ drücken, um den Motor auszuschalten
3. „Reset“ drücken
4. „Alle Programme“ auswählen

Roboter manuell in Home Position fahren:

1. Schlüsselschalter am Programmierhandgerät auf „Manuell“
2. Taste „M-Mod“ drücken
3. Fahrmodus „Achse“ und Werkzeug Koordinaten „T1“ auswählen
4. Taste „Speed“ drücken
5. Geschwindigkeit einstellen (ca. 10% für Anfänger) und mit „OK“ bestätigen
6. Totmannschalter am Programmierhandgerät drücken
7. Warten bis „Motor“ leuchtet
8. Roboter mit J1 – J6 „+“ oder „-“ in die Nähe des Home Sensors steuern
9. Unter „Variablen“ „P“ auswählen, dort „P4“ markieren
10. Taste „Fahre zu“ drücken
11. „PtP“ auswählen
12. „OK“ drücken, bis Roboter an Home Position ist

Roboter ausschalten:

1. Alle Programme stoppen
2. Motor über Taste „Motor“ ausschalten
3. Über „Cancel“ auf den Home Bildschirm
4. Kippschalter an Robotersteuerung aus

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, 19. Dezember 2017	
Ort, Datum	Unterschrift