White Paper, Stand 2015/06: Dieses Dokument wurde im Rahmen des DFG Projektes PA 631-2 erstellt und ist eine grundlegende Überarbeitung nachfolgender Masterthesis.

- MASTERTHESIS -

Entwicklung eines Simulators zur energetischen Bewertung von Prozessketten der spanenden Bauteilfertigung mit MATLAB/SimEvents®

angefertigt von

Artur Schmidt, B.Eng.

an der

Hochschule Wismar

Fachhochschule für Technik, Wirtschaft und Gestaltung

am

Fachbereich Maschinenbau

---

1. Betreuer: Prof. Dr. -Ing. Thorsten Pawletta, FIW/MVU, FG CEA

2. Betreuer: Dr. Olaf Hagendorf, FIW/EUI, FG CEA

Eingereicht am: 23. April 2012 in Wismar

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Programm-Listings

Abkürzungsverzeichnis

A	Anlassen
ALD	Außenlängsdrehen
ARS	Außenrundschleifen
ARSH	Außenrundschleifhärten
ASCII	American Standard Code for Information Interchange
BBM	Black Box Modell
BBT	Black Box Testing
BM	Basismodell
CNC	Computerized Numerical Control
EF	Experiment File
EG	Enabled Gate
GC	Gate Control
HA	Hauptanforderung
IH	Induktionshärten
KSS	Kühlschmierstoff
LS	Lokale Steuerung
MB	Modellbibliothek
MFK	Materialflusskomponenten

NC	Numerical Control
PAP	Programmablaufplan
Q	Quelle
RG	Release Gate
SA	Signalanalyse
SE	Steuerentität
SK	Senke
SP	Systemparameter
VH	Vakuumhärten
ViZ	Verweildauer im Zustand
WE	Werkstückentität

1. Einleitung

Steigende Rohstoff- und Energiepreise stellen produzierende Unternehmen vor neue Herausforderungen. Zunehmend werden Methoden der Modellbildung und Simulation eingesetzt, um den Ressourcenverbrauch, die Energieverteilung und Energieumsetzung quantitativ zu erfassen und Potentiale zur Einsparung aufzuzeigen. Vor diesem Hintergrund werden zunehmend Methoden der Modellbildung und Simulation eingesetzt, um den Ressourcenverbrauch von fertigungstechnischen Prozessketten quantitativ zu untersuchen. Eine fertigungstechnische Prozesskette entspricht nach [NBL07] mehreren aufeinanderfolgenden Fertigungsprozessen zur Herstellung eines Bauteil oder einer Bauteileigenschaft. Oft steht die Untersuchung von logistischen, produktionssteuerungstechnischen und gebäudetechnischen Aspekten [Jun08, PS11] beziehungsweise die Untersuchung einzelner sehr energieintensiver Fertigungsprozesse [MBS06] im Vordergrund. Gemäß der Forderung in [NWK+08] auch die Ressourceneffizienz der Fertigungsprozesse in Prozessketten eingehend zu analysieren, wurde in [LBM+11, Lar12] eine Modellbibliothek für spezielle Fertigungsprozesse in der ereignisdiskreten Simulationsumgebung MATLAB/SimEvents [Mat11a] entwickelt. Die Modellierung basiert auf dem transaktionsorientierten Ansatz gemäß [ZPK00]. Danach werden die Werkstücke als aktive Komponenten (Entitäten) und Fertigungsprozesse als passive Komponenten (Blöcke) abgebildet. Die auf diese Weise entwickelten konfigurierbaren dynamischen Modelle, in folge Basismodelle genannt, können in MATLAB/SimEvents zu Prozessketten komponiert, parametriert und ausgeführt werden. Auf diese Weise wird nicht nur die Berechnung von Ressourcenverbräuchen, sondern auch die zeitliche Bewertung von Verbrauchsspitzen fertigungstechnischer Prozessketten ermöglicht.

Jedoch weisen die entwickelten Basismodelle einige Schwächen bezüglich der Modellkomplexität, Laufzeitverhalten und Mehrfachverwendbarkeit typengleicher Basismodelle auf. Somit beschränkt sich die Verwendung der entwickelten Basismodelle ausschließlich auf die Modellierung und Simulation sequenzieller Prozessketten. Ziel dieser Arbeit ist die Restrukturierung der in [Lar12] entwickelten Basismodelle in MATLAB/SimEvents und der Behebung der erwähnten Schwächen, damit neben den sequentiellen Prozessketten

auch komplexe verzweigter Fertigungsstrukturen modelliert und bezüglich des Ressourcenverbrauches simulationstechnisch analysiert werden können. Als Voraussetzung zur Untersuchung von Prozessketten im Kontext einer Fertigungsstruktur soll ein Konzept zur Erweiterung der Basismodelle um Steuerschnittstellen erweitert werden.

Ausgehen von einer Systemanalyse im Kapitel 2 und einer konkreten Erfassung der Schwachstellen der Basismodelle werden im 3. Kapitel konzeptionelle Vorschläge zur Restrukturierung erarbeitet. Die Umsetzung der konzeptionellen Überlegungen erfolgt im vierten Kapitel. Eine Validierung der neu implementierten Basismodelle wird im fünften Kapitel präsentiert. Kapitel 6 beinhaltet zwei Beispiele und zeigt die neu gewonnenen Möglichkeiten. Die Arbeit endet mit einer Zusammenfassung und einem Ausblick.

2. Systemanalyse auf Basis von Vorarbeiten

Im zweiten Kapitel erfolgt eine Systemanalyse der in [Lar12] entwickelten und implementierten Modellbibliothek. Ausgehend von der Übersicht zu den gewählten Prozessketten, Fertigungsverfahren und Ressourcenarten wird auf die ereignisdiskrete Modellierung der einzelnen Fertigungsverfahren in MATLAB/SimEvents eingegangen. Dabei wird ein besonderer Fokus auf die von [Lar12] vorgestellten Modellstruktur gelegt. Anschließende Simulationsexperimente sollen einen Gesamtüberblick zu den Möglichkeiten und Grenzen der Modellbibliothek liefern. Abschließend erfolgt eine Diskussion der durchgeführten Simulationsexperimente.

2.1. Ausgewählte Prozessketten und Fertigungsverfahren

Für das Herstellen eines rotationssymmetrischen Werkstückes mit fein bearbeiteten Funktionsflächen aus zwei alternativen Werkstoffen 100Cr6 und 42CrMo4 (Abb. 2.1) wurde in [Lar12] drei alternativen Prozessketten ausgewählt (Abb. 2.2).

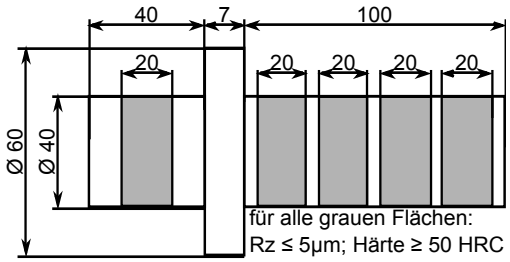


Abbildung 2.1.: Schematische Darstellung des rotationssymmetrischen Werkstückes mit qualitätsbezogenen Randbedingungen.

Die einzelnen Prozessketten bestehen aus folgenden Fertigungsverfahren:

1. Außenlängsdrehen (ALD), Vakuumhärten (VH), Anlassen (A) und Außenrundschleifen (ARS)
2. Außenlängsdrehen, Induktionshärten (IH), Anlassen und Außenrundschleifen
3. Außenlängsdrehen und Außenrundschleifhärten (ARSH)

Abbildung 2.2 zeigt weiterhin die zu betrachtenden Ressourcenarten elektrische Energie, Kühlschmierstoff, Werkstoff und Werkzeugverschleiß. Jedem Fertigungsverfahren ist mindestens eine Ressourcenart zu geordnet.

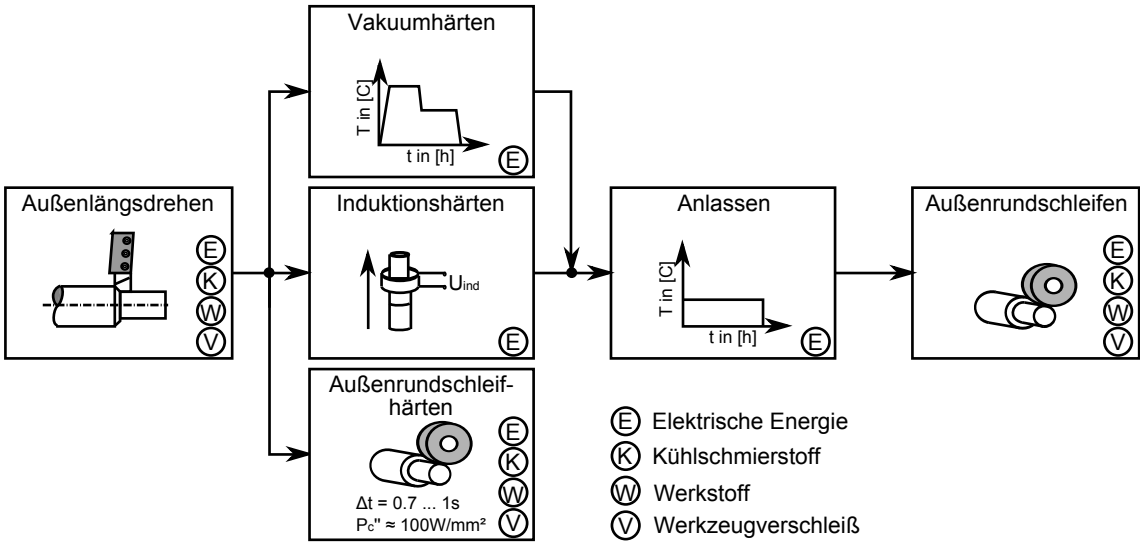


Abbildung 2.2.: Ausgewählte Prozessketten und Fertigungsverfahren mit dem zu betrachtenden Ressourcenverbräuchen nach[Lar12].

Die betrachteten Fertigungsverfahren und Ressourcenarten bilden die Grundlage für die Modellierung der einzelnen Fertigungsverfahren als ereignisdiskrete Basismodelle (BM) in MATLAB/SimEvents. Basismodelle stellen formal nach [ZPK00] Modellelemente dar, die nicht weiter zerlegt werden dürfen oder können. Durch die horizontale Kopplung der einzelnen Basismodelle sollen Prozessketten nach Abbildung 2.2 komponiert und simuliert werden. Nach einem Simulationslauf stehen dem Nutzer Informationen über die Ressourcenverbräuche als Verlauf über der Zeit oder kumulierte Werte zur Verfügung. Die einzelnen Diagramme können dann Aufschluss über die zeitliche Lastverteilung, auftretende Lastspritzen, Werkzeugverschleiß oder verbrauchtes Volumen des Kühlschmierstoffs liefern.

Die konkrete Umsetzung der Fertigungsverfahren mit den zugehörigen Ressourcenarten als ereignisdiskrete Basismodelle in MATLAB/SimEvents erfolgt im nachfolgendem

Abschnitt.

2.2. Modellierung der Fertigungsverfahren als Basismodelle

Die ausgewählten Fertigungsverfahren wurden als ereignisdiskrete BMs in MATLAB/SimEvents modelliert. Nach [Lar12] „ gleicht das äußere Verhalten der Basismodelle dem von Servern als Abbild von Arbeitsplätzen oder Maschinen in einem Materialflussmodell“ und beinhalten die eigentliche Modelldynamik. Aus der Sicht des Autors können die BMs in drei Grundtypen klassifiziert werden:

- CNC-satzbasierte Basismodelle
- Messdatenbasierte Basismodelle
- Hilfsmodelle

Zu den CNC-satzbasierten BMs zählen ALD, ARS und ARSH. Der Ressourcenverbrauch wird durch die sequenzielle Verarbeitung von CNC-Sätzen bestimmt. Den messdatenbasierten BMs gehören VH, IH und A an. Ihr Ressourcenverbrauch wird unmittelbar aus Messdaten der realen Fertigungseinrichtungen identifiziert. Zu den Hilfsmodellen gehören eine Quelle (Q) und eine Senke (SK). Q und SK dienen der Generierung und Terminierung von Werkstücken, welche als Entitäten [1] abgebildet werden. Entitäten wandern zwischen den einzelnen BMs und lösen Ereignisse aus. Abbildung 2.3 zeigt die Modellbibliothek mit den BMs als Subsysteme in MATLAB/SimEvents und die Einteilung in die genannten Grundtypen.

[1] Entitäten bilden Objekte der ereignisdiskreten Simulation in MATLAB/SimEvents ab [Mat11a]. Im Kontext der transaktionsorientierten Simulation nach [ZPK00] entsprechen Entitäten formal den aktiven Komponenten.

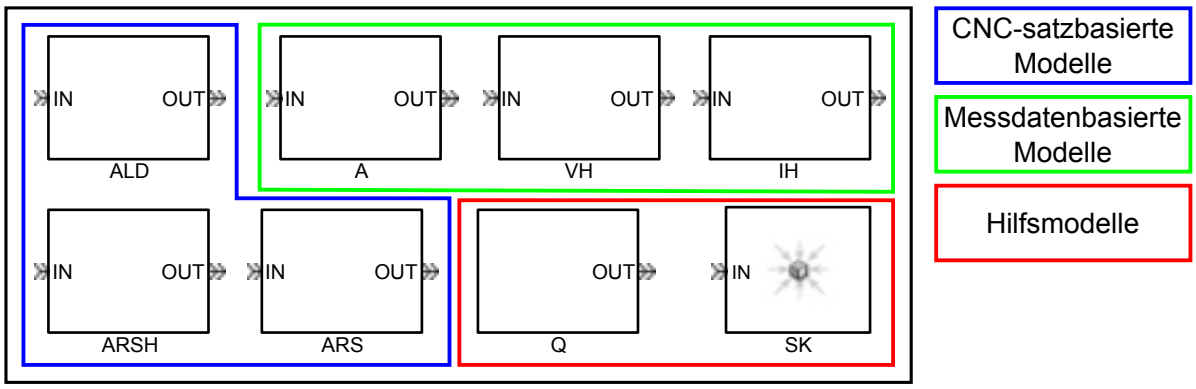


Abbildung 2.3.: Klassifikation der Basismodelle in drei Grundtypen.

Sowohl die CNC-satzbasierten als auch die Messdatenbasierten BMs besitzen einen nahezu identischen strukturellen Aufbau. Die Modellstruktur ist auf der obersten Ebene in vier Subsysteme *Ereignissteuerung*, *Zustandsberechnung*, *Komponentenabbilder* und *Signalanalyse* aufgeteilt. Abbildung 2.4 zeigt die Modellstruktur aller BMs, wobei die Anzahl Komponentenabbilder variieren kann.

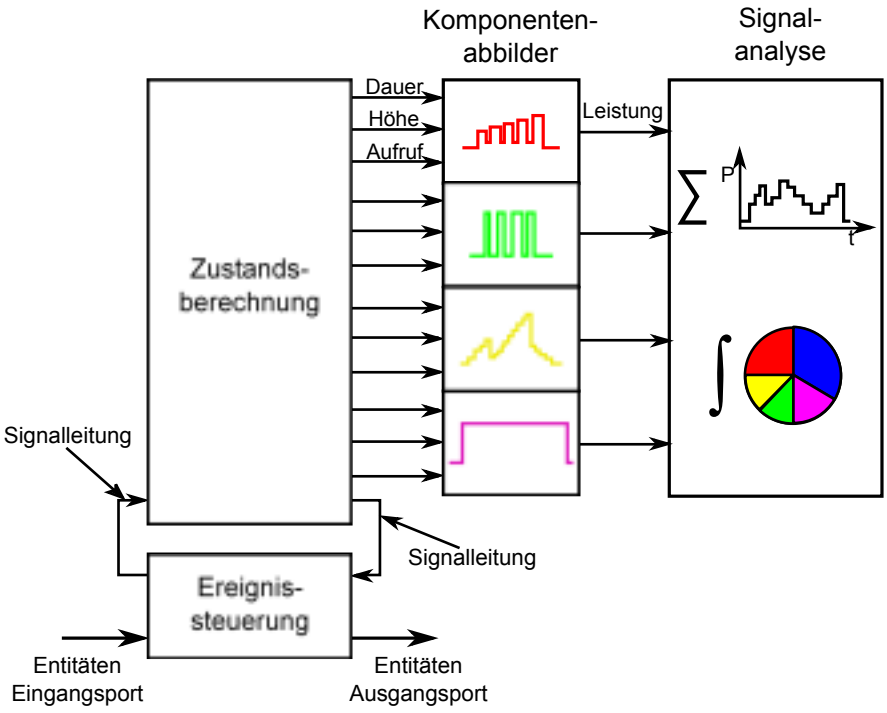


Abbildung 2.4.: Schematische Darstellung der Modellstruktur aller Modelle nach [Lar12].

In weiteren werden die einzelnen Subsysteme näher erläutert.

Ereignissteuerung

Steuert den Aufruf des Subsystems Zustandsberechnung während der Simulation durch das zyklische erzeugen von Zeitereignissen oder Zustandsereignissen beim Eintreten einer Entität in das BM. Je nach BM Typ (CNC-satzbasiert oder Messdatenbasiert) unterscheidet sich die Komplexität der Ereignissteuerung (vgl. A.1.2 auf Seite IV und A.2.2 auf Seite IX).

Zustandsberechnung

Beinhaltet eine verfahrensspezifische MATLAB Funktion, die den Ressourcenverbrauch eines BMs berechnet. Die Zustandsberechnung wird von der Ereignissteuerung aufgerufen. Es werden für alle Komponentenabbilder die Zustandsgröße sowie die Verweildauer im Zustand berechnet und an die Komponentenabbilder übergeben. Weiterhin wird die Verweildauer im Zustand (ViZ) an die Ereignissteuerung übergeben, die daraufhin ein Zeitereignis einplant.

Komponentenabbilder

Repräsentieren die im System befindlichen Verbraucher. Sie werden von der *Zustandsberechnung* aufgerufen. Die aufgerufene Komponente erhält den Werte der Zustandsgröße und die ViZ. Anhand dieser Werte wird ein entsprechendes rechteckiges Lastsignal berechnet und an das Subsystem *Signalanalyse* übermittelt.
Die Komponentenabbilder können zwei Betriebszustände annehmen:

1. Deren zeitliches Ende zu Beginn bestimmbar ist.
   Dieser Zustand tritt ein, wenn die von der *Zustandsberechnung* übergebene ViZ ungleich Null ist. Das Komponentenabbild setzt anhand des ViZs ein Zeitereignis. Nach dem Eintreten des Zeitereignisses wird der Wert der entsprechenden Zustandsgröße auf Null gesetzt.
2. Deren zeitliches Ende zu Beginn nicht bestimmbar ist.
   Der Zustand wird eingenommen, wenn die von der *Zustandsberechnung* übergebene ViZ gleich Null ist. Der Wert der entsprechenden Zustandsgröße bleibt bis zum nächsten Aufruf des Komponentenabbildes konstant.

Der Inhalt der Komponentenabbilder ist für jeden Grundtyp identisch.

Signalanalyse

Nimmt Zustandsgrößen entgegen, verarbeitet diese und übermittelt sie als Verlauf über der Zeit, kumulierte Werte oder kumuliertes Integral über der Zeit an die Simulationsumgebung. Dies geschieht durch die von MATLAB/SimEvents zur Verfügung gestellten *Discrete Event Signal to Workspace* Blöcke.

Im Anhang A auf Seite I ist beispielhaft die umgesetzte Modellstruktur eines CNC-satzbasierten und eines Messdatenbasierten Basismodells zu sehen.

Zum vollständigen CNC-satzbasierten oder Messdatenbasierten BM gehören neben einem MATLAB/SimEvents Block gemäß Abbildung 2.3 verschiedene MATLAB Funktionen. Sie Verarbeiten Messdaten oder CNC-Programme und berechnen den Ressourcenverbrauch der entsprechenden Komponentenabbilder. Bei den CNC-Satzbasierten BMs lauten die Funktionen *nc_calc.m*, *G1.m*, *G2.m* sowie *part_shape.m*. Die Messdatenbasierten BMs besitzen nur die nc_clac.m. Je nach BM besitzen diese Funktionen einen anderen Bezeichner. Beim BM Außenlängsdrehen besitzt diese Funktionen beispielsweise den Bezeichner *D_* und lauten somit *D_nc_calc.m*, *D_G1.m* und *D_G0.m*. Anhang C.2 auf Seite XIX gibt eine Übersicht zu den einzelnen MATLAB Funktionen wieder.
CNC-satzbasierte BMs besitzen noch ein numerisches Werkstückmodell. Es stellt die Ausgangsgeometrie des Rohlings in Form einer Matrix dar. Mit Hilfe des numerischen Werkstückmodells kann in jedem Prozessschritt der flächenbezogene Energieverbrauch bestimmt und gespeichert werden. Für nähere Information zum numerischen Werkstückmodell soll auf den Anhang auf Seite XVII verwiesen werden.

Der nachfolgende Abschnitt beschäftigt sich mit Simulationsexperimenten der gezeigten Modellbibliothek und den darin enthaltenen Basismodellen.

2.3. Simulationsexperimente

Ein Simulationsexperiment besteht grundlegend aus einem Experiment-File (EF) und der Modellbibliothek (MB) mit den verfahrensspezifischen MATLAB Funktionen. Mit den Modellkomponenten der MB kann ein gewünschtes MATLAB/SimEvents Modell der Prozessketten oder einzelner Fertigungsverfahren nach Abbildung 2.2 auf Seite 4 erstellt und mit Hilfe des EFs simuliert werden. Das EF muss Informationen über die notwendige Fertigungsaufgabe mit Randbedingungen für das jeweilige Fertigungsverfahren, Steuerparameter eines Simulationslaufes sowie Anzeigemethoden zur Visualisierung

der Simulationsergebnisse beinhalten (vgl. D auf Seite XXI).

Bevor detaillierter auf die Simulationsexperimente eingegangen wird, soll zu nächst ein Simulationslauf nach [Lar12] Schrittweise erläutert werden. Abbildung 2.5 zeigt ein Programmablaufplan (PAP) eines Simulationslaufes.

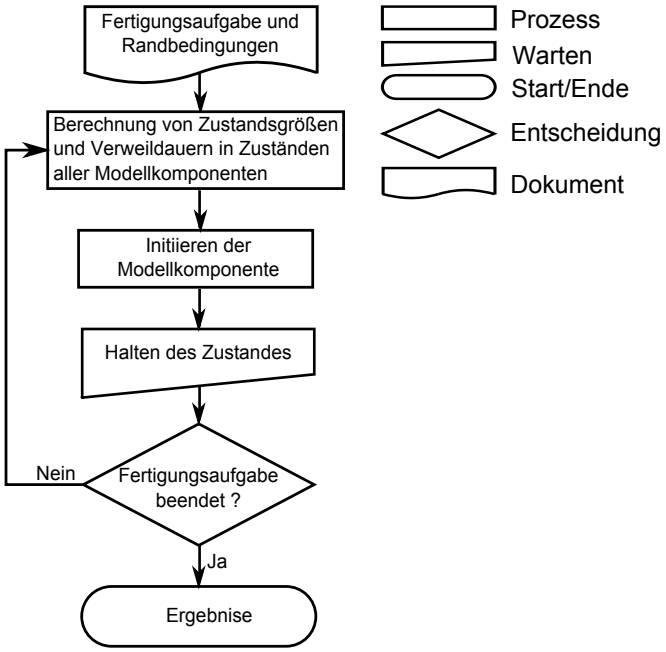


Abbildung 2.5.: PAP eines Simulationsexperimentes nach [Lar12].

1. Ereignisgesteuerter Eintritt einer Entität in die Ereignissteuerung und zustandsbasierte Aktivierung der Zustandsberechnung.

   a) Bei CNC-satzbasierten BMs werden Werkstückdaten sowie Maschinenparameter aus der Simulationsumgebung eingelesen.

   b) Messdatenbasierte BMs lesen die zugehörigen Messdaten aus der Simulationsumgebung ein.

2. Berechnung von Zustandsgrößen und Verweildauern in Zuständen (ViZ) für den aktuellen Bearbeitungsschritt in der Zustandsberechnung.

   a) Berechnung der Zustandsgrößen und ViZ durch das zeilenweise Einlesen und Interpretieren der CNC-Sätze.

   b) Messdatenbasierte BMs berechnen die Zustandsgrößen und ViZs aus den eingelesenen Messdaten.

3. Zeitgleiche Aktivierung alle Komponentenabbilder, welche eine Zustandsänderung erfahren. Daraufhin erzeugen die Komponentenabbilder anhand der übergebenen Werten der Zustandsgröße sowie der ViZ ein wertdiskretes Lastsignal.
4. Die Ereignissteuerung wird gleichzeitig mit den Komponentenabbildern durch das setzen eines Zeitereignisses aktiviert.
5. Nach dem Eintreten des Zeitereignisses in der Ereignissteuerung wird die Zustandsberechnung erneut aktiviert und es geht mit dem 2. Schritt weiter.
6. Ist die Fertigungsaufgabe beendet, verlässt die Entität das BM über den Ausgangsport der Ereignissteuerung und aktiviert das nachstehende BM oder wird in einer Entitäten-Senke terminiert.

Die Signalanalyse zeichnet alle Zustandsgrößen während eines Simulationslaufes auf und übergibt diese nach dem Simulationsende an die Simulationsumgebung.

Die nachfolgenden Unterabschnitte beschäftigen sich mit einfachen und komplexen Simulationsexperimenten mit der MB. Während der Durchführung der Experimente wird jedes BM sowie die daraus komponierten Prozesskettenmodelle auf Mehrfachverwendung der einzelnen BMs innerhalb eines MATLAB/SimEvents Modells, Mehrfachausführung sowie Laufzeitverhalten und Parametrierung eines Experimentes untersucht. Die Auswertung der zu genannten Punkten erfolgt erst im nächsten Abschnitt 2.4 auf Seite 14.

2.3.1. Beispiel eines Fertigungsverfahrens

In diesem Unterabschnitt soll ein einfaches MATLAB/SimEvents Modell erstellt und simuliert werden, welches nur das Fertigungsverfahren ALD beinhaltet.
Zunächst muss das Arbeitsverzeichnis mit der MB und den verfahrensspezifische Funktionen in MATLAB ausgewählt und mit den dort enthaltenen BMs ein neues MATLAB/SimEvents Modell per *Drag and Drop* erstellt werden. Die nachstehende Abbildung 2.6 zeigt das gewünschte MATLAB/SimEvents Modell.

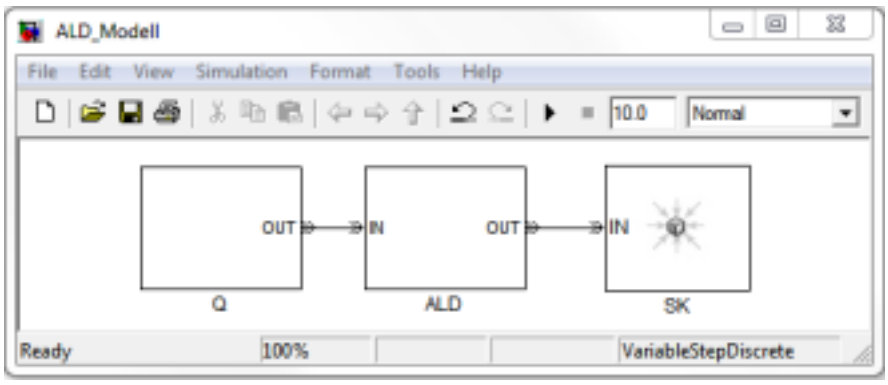


Abbildung 2.6.: MATLAB/SimEvents Modell mit dem Fertigungsverfahren ALD.

Im ALD Block muss der zu bearbeitende Werkstoff manuell gesetzt werden. Für die Simulation wird der Werkstoff 100Cr6 ausgewählt. Der Q Block ermöglicht das Setzen der gewünschte Anzahl der zu bearbeitenden Werkstücke. Für das aktuelle Beispiel wird die Anzahl der Werkstücke auf 3 gesetzt und das Modell unter dem Namen *ALD_Modell* im selben Verzeichnis gespeichert. Das zugehörige EF ist im Anhang D.2 auf Seite XXII gezeigt.
Nach der Simulation liefert das Subsystem Signalanalyse Strukturen mit dem Verlauf der einzelnen Zustandsgrößen. Die Strukturen für das Fertigungsverfahren ALD lauten

- D_simout_E für die Energie
- D_simout_KSS für den Kühlschmierstoff
- D_simout_Q für den Werkstoffverbrauch
- D_simout_tool für den Werkzeugverschleiß

Mittels der in von MATLAB zur Verfügung gestellten Anzeigemethoden können die einzelnen Verläufe graphisch angezeigt werden (Abb. 2.7).

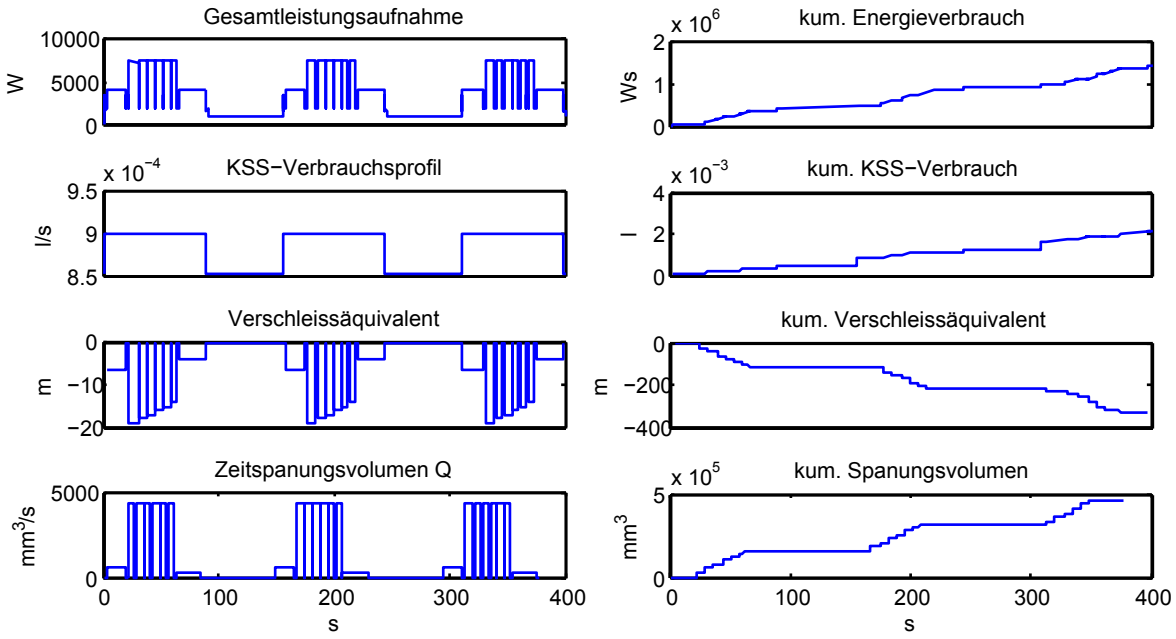


Abbildung 2.7.: Simulationsergebnisse vom ALD_Modell.

Für detaillierte Übersicht zur Simulation einzelner Fertigungsverfahren in MATLAB/SimEvents sei an dieser Stelle auf den Anhang E auf Seite XXVI verwiesen.

2.3.2. Beispiel einer erweiterten Prozesskette

Analog zum vorherigem Unterabschnitt wird in diesem Unterabschnitt die Simulation der ersten Prozesskette nach Abbildung 2.2 auf Seite 4 gezeigt.
Das MATLAB/SimEvents Modell der ersten Prozessketten wird per *Drag and Drop* erstellt, die BMs miteinander verkoppelt und im Arbeitsverzeichnis abgespeichert. Die Darstellung der ersten Prozesskette in Form eines MATLAB/SimEvents Modelles ist in Abbildung 2.8 zu sehen.

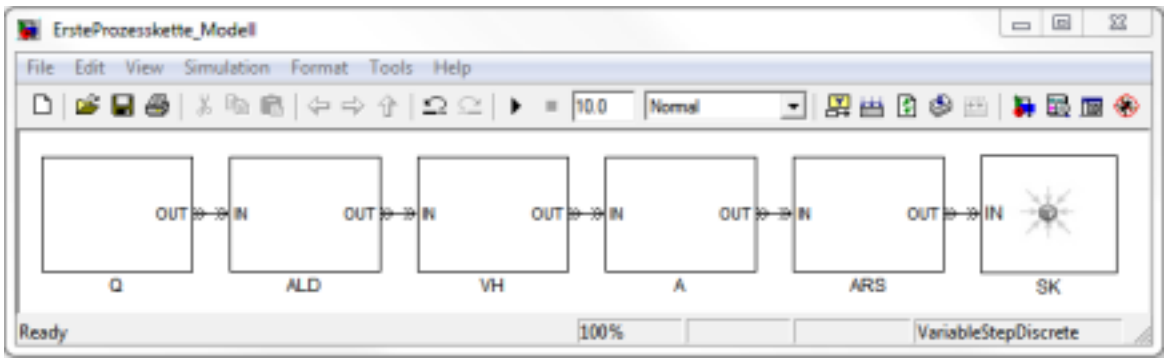


Abbildung 2.8.: MATLAB/SimEvents Modell der ersten Prozesskette.

Das Experiment-File für das aktuelle Beispiel befindet sich im Anhang D.3 auf Seite XXIV. Die nachfolgende Abbildung 2.9 zeigt als mögliches Simulationsergebnis die Gesamtleistungsaufnahme der einzelnen Fertigungsverfahren der ersten Prozesskette für das herstellen von 12 Werkstücken unter Verwendung des Werkstoffes 100Cr6.

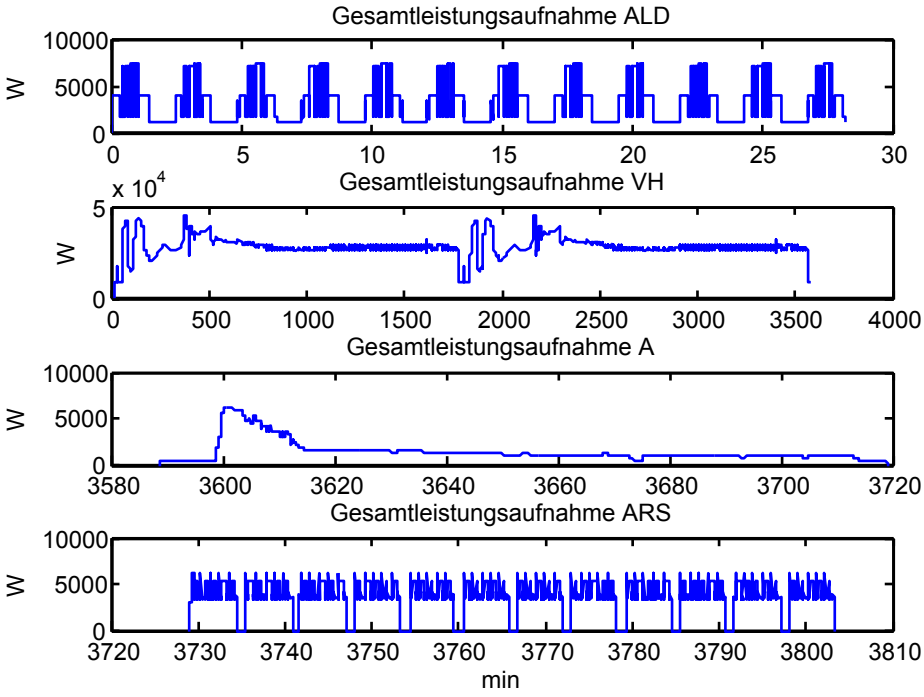


Abbildung 2.9.: Gesamtleistungsaufnahme der einzelnen Fertigungsverfahren von *Erste-Prozesskette_Modell*.

2.4. Bewertung des Status Quo

Mit der Modellbibliothek von Larek wurden umfangreiche Simulationsexperimente durchgeführt und auf die Punkte Parametrierung, Mehrfachverwendung und Mehrfachausführung der BMs sowie der Simulationslaufzeit geachtet. Zu Beginn ist auf gefallen, dass die Fertigungsverfahren als eine Kombination aus Server + Puffer realisiert wurden und nicht als Single bzw. N-Server. Aus diesem Grund sind in den gezeigten Simulationsmodellen (Abbildungen 2.6 und 2.8) keine Puffer zwischen den einzelnen BMs zu sehen.

Die Parametrierung der BMs mittels globalen Variablen in einem Experiment-File erscheint auf den ersten Blick übersichtlich. Bei näherer Betrachtung wurden jedoch eine Vielzahl globaler sowie persistenter Variablen in den verfahrensspezifischen Funktionen (*nc_calc.m, G0.m, G1.m*) entdeckt. Die meisten von ihnen werden zum Informationsaustausch zwischen den Funktionen oder zum Speichern von Zwischenwerten verwendet. In der nachstehenden Tabelle sind alle globalen und persistenten Variablen den einzelnen Fertigungsverfahren zugeordnet aufgezählt.

Tabelle 2.1.: Aufzählung der globalen und persistenten Variablen.

Basismodelle	globale Variablen	persistente Variablen	Summe
Außenlängsdrehen	24	9	33
Außenrundschleifen	25	18	43
Außenrundschleifhärten	25	18	43
Vakuumhärten	1	15	16
Induktionshärten	1	15	16
Anlassen	1	15	16

Einige Variablen (global und persistent) werden nach einem Simulationslauf nicht auf ihre Anfangszustände zurückgesetzt und besitzen bei einem nachfolgenden Simulationslauf Anfangszustände, die den Endzuständen der vorhergehenden Simulation entsprechen. Es ist also problematisch ein und dasselbe MATLAB/SimEvents Modell mehrmals hintereinander auszuführen, da nur die Ergebnisse der ersten Simulation als richtig angesehen werden können. Möchte man ein Simulationsmodell mehrmals hintereinander ausführen, so ist der MATLAB-Befehl *clear all* vor einem erneuten Simulationslauf zu verwenden. Bei einem solchen Vorgang gehen jedoch neben den globalen und persistenten Variablen auch alle vorhandenen Simulationsergebnisse verloren. Diese müssen im Vorfeld in einer **.mat*-Datei oder Ähnlichen gesichert werden.

Die Mehrfachverwendung der einzelnen BMs wird durch die Verwendung der globalen und persistenten Variablen verhindert. Alle Instanzen einer BM Klasse greifen unkontrolliert sowohl lesend als auch schreibend auf dieselben globalen und persistenten Variablen zu. Somit kommt es bei Mehrfachverwendung typengleichet BMs zu *race conditions* [KY99] Problemen. Ein Simulationslauf mit mehreren gleichen BMs liefert somit keine brauchbare Ergebnisse.

Die Simulationslaufzeit wird über die vorzugebende Simulationszeit gesteuert. Erst wenn die vordefinierte Simulationszeit abgelaufen ist, endet die Simulation. Es entsteht das Problem, dass die Simulationszeit sehr gut geschätzt werden muss. Wird die sie als zu gering gewählt, so endet der Simulationslauf bevor die geforderte Anzahl der Werkstücke die gesamte Prozesskette oder ein Fertigungsverfahren durchlaufen haben. Bei einer hohen Simulationszeit werden zwar alle Werkstücke bearbeitet und in der SK terminiert die Simulation jedoch nicht beendet. Das MATLAB/SimEvents Modell verweilt bist die Simulationszeit abgelaufen ist im Leerlauf, was sich wiederum negativ auf die Simulationslaufzeit und die gesamte Simulation auswirkt.
Des Weiteren beinhaltet das Subsystem *Zustandsberechnung* ein *Embedded Matlab Function* Block, der die verfahrensspezifische Funktion *nc_calc.m,* über den MATLAB-Befehl *eml.extrinsic,* in MATLAB/SimEvents lädt. Bei einem Simulationslauf kompiliert MATLAB/SimEvents die extrinsische Funktion nicht mit, sondern führt sie zur Laufzeit in MATLAB aus und liefert die Ausgabeparameter an den *Embedded Matlab Funktion* Block zurück. Ein solches Vorgehen wirkt sich wiederum negativ auf die Simulationslaufzeit aus.

Ein weiterer Nachteil der aktuellen Implementierung der BMs ist, dass die MATLAB/SimEvents-Statistik nicht genutzt werden kann. Beispielsweise können statistische Größen wie mittlere Auslastung oder mittlere Bedienzeit[2] der CNC-Satzbasierten sowie Messdatenbasierten BMs nicht ermittelt werden. Der Grund dafür liegt in dem Aufbau der Ereignissteuerung der BMs. Dort werden eintretende Werkstück-Entitäten (WEs) als Steuer-Entitäten verwendet, die die Zustandsberechnung zyklisch aktivieren. Abbildung 2.10 zeigt einen Ausschnitt der Ereignissteuerung. Die Komplette Darstellung der Ereignissteuerung befindet sich im Anhang A.1.2 und A.2.2.

[2]Die Bedienzeit entspricht der Verweildauer einer Entität in einem Server. Sie kann auch als Bearbeitungszeit angesehen werden.

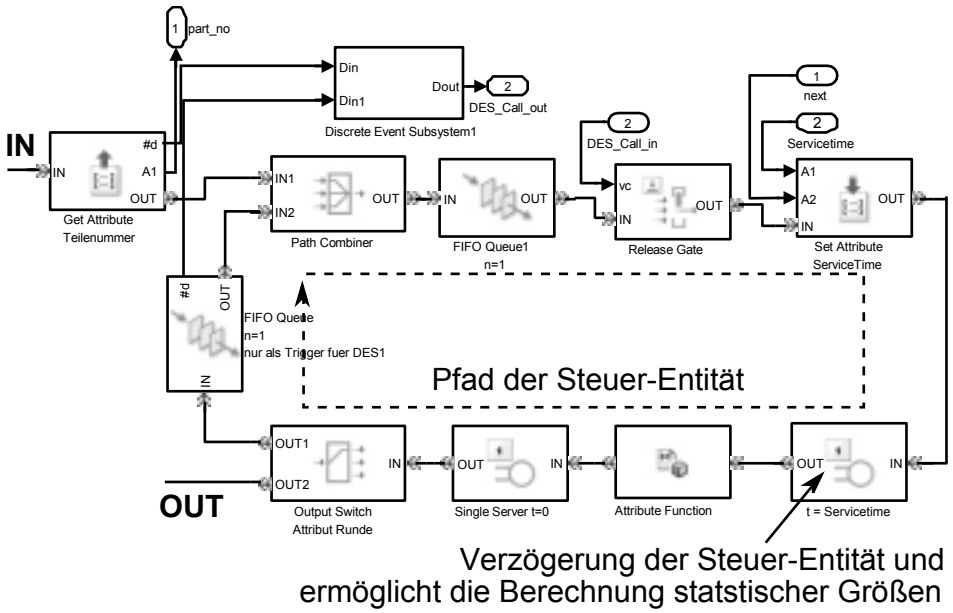


Abbildung 2.10.: Ausschnitt der Ereignissteuerung nach [Lar12].

WEs gelangen einzeln über den Get Attribute Block in den Steuerungskreislauf (gestrichelt dargestellt). Im Steuerungskreislauf wird die aktuelle WE nun als Steuer-Entität (SE) verwendet und verweilt solange im Steuerungskreislauf bis die Fertigungsaufgabe abgearbeitet ist. Im gekennzeichneten Single Server erfolgt die Verzögerung der SE um die Zeitdauer eines Arbeitsganges. Weiterhin kann derselbe Single Server die genannten statistischen Größen berechnen und an die Simulationsumgebung weiterleiten. Durch den Steuerungskreislauf ist die Statistik jedoch unbrauchbar. Der Single Server besitzt keine Information über das gesamt Modell. Bei jeder Aktivierung wird angenommen, dass eine neue Entität verzögert werden soll. In diesem Fall handelt es sich jedoch um dieselbe Entität, die den Single Server aktiviert und somit die Statistik verfälscht bzw. unbrauchbar macht.

Grundlegend kann jedes Fertigungsverfahren sowohl allein als auch innerhalb der in Abbildung 2.2 auf Seite 4 gezeigten Prozessketten erfolgreich simuliert werden. Die Auswertung der durchgeführten Simulationsexperimente hat jedoch gezeigt, dass die einzelnen Basismodelle ein großes Potential für Verbesserungen bieten. Im nächsten Kapitel werden Verbesserungsvorschläge für die Restrukturierung der einzelnen BMs entwickelt.

3. Überlegungen zur Restrukturierung der Basismodelle

Basierend auf der Auswertung der durchgeführten Simulationsexperimenten aus Abschnitt 2.4 werden in diesem Kapitel Konzepte zur Restrukturierung der Basismodelle erarbeitet. Am Anfang erfolgt die Darstellung der grundlegenden Ansätze mit Verbesserungsvorschlägen zur Restrukturierung. Es sei dabei angemerkt, dass die Restrukturierung sich ausschließlich auf die Struktur der BMs ausgerichtet ist. Die dynamikbeschreibenden verfahrensspezifischen Funktionen werden bei der Restrukturierung nicht verändert.
Danach werden alle Verbesserungsvorschläge auf alle Basismodelle präzisiert. Im letzten Abschnitt wird eine Idee zur Integration der Steuerschnittstellen diskutiert, die das An- und Abschalten der Grundlast der Basismodelle während eines Simulationslaufes ermöglicht.

3.1. Grundlegender Ansatz

Jedes BM soll grundlegend als ein parametrierbares Black-Box-Modell (BBM) mit eindeutigem Ein-/Ausgangsverhalten angesehen werden. Ein solches BBM nimmt unbearbeitete Werkstück-Entitäten (WEs) entgegen und leitet diese nach einer erfolgreichen Bearbeitung aus dem BBM hinaus. Ausgewählte Ressourcenverbräuche sollen aus dem BBM an die Simulationsumgebung übermittelt werden. Die Parametrierung der BMs soll nicht mehr über globale sowie persistente Variablen erfolgen, sondern über eine Parametermaske. Diese muss für das jeweilige BM alle wichtigen Parameter enthalten, sodass vor einem Simulationslauf die Parametrierung problemlos von dem Anwender durchgeführt werden kann. Abbildung 3.1 zeig schematisch ein BM als Black-Box-Modell.

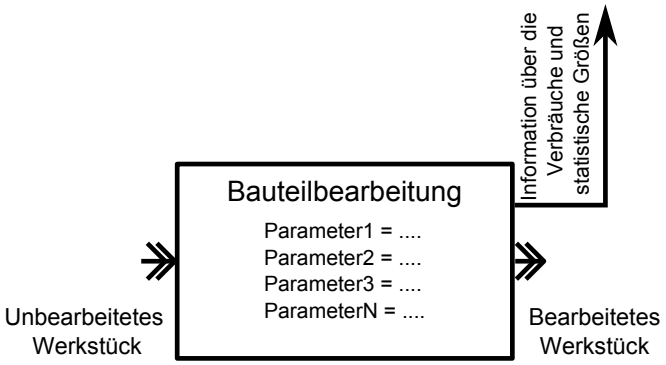


Abbildung 3.1.: Darstellung einer BMs als Black-Box-Modell.

Die Parametermaske liefert einen weiteren Vorteil und zwar die Kapselung von Daten innerhalb eines BMs. Alle verwendeten Variablen sind dann nur innerhalb eines BMs gültig. Weiterhin erfolgt der Vorschlag alle restlichen Variablen globaler und persistenter Natur als Entitäten-bezogene Attribute abzubilden. Bei einem solchen Vorgehen werden alle Zustandsgrößen sowie Informationen innerhalb einer Entität gekapselt, sodass sie sich bei einer Mehrfachverwendung bzw. Mehrfachausführung der BMs nicht mehr gegenseitig beeinflussen. Abschließend soll die Anzahl der Blöcke eines BMs auf deren Notwendigkeit überprüft werden. Eine Reduktion der Anzahl der Blöcke wirkt sich positiv auf die Simulationslaufzeit aus [Mat11b]. Zuletzt soll die Verwendung der MATLAB/SimEvents-Statistik ermöglicht werden.

Für die Restrukturierung der BMs ergeben sich folgende Hauptanforderungen:

1. Implementierung der BMs als parametrierbare Black-Box-Modelle
2. Implementierung klassenspezifischer Parametermasken
3. Abbildung von globalen und persistenten Variablen als Entitäten-bezogene Attribute
4. Reduktion der Anzahl der Blöcke
5. Verwendung der MATLAB/SimEvents-Statistik in den BMs ermöglichen

In den nachfolgenden Abschnitten erfolgt eine Präzision der Hauptanforderungen für die einzelnen BM Klassen. Des Weiteren werden Vorschläge zur Senkung der Simulationslaufzeit erarbeitet.

3.2. Klassenspezifische Restrukturierungsvorschläge

In diesem Kapitel werden klassenspezifische Restrukturierungsvorschläge vorgestellt.

3.2.1. Hilfsmodelle

In der aktuellen Implementierung generiert die Quelle WEs mit einer eindeutigen Identifikationsnummer. An Hand dieser Identifikationsnummer wird die Bearbeitungsaufgabe sowie die Werkstückeigenschaften mittels globaler Variablen aus der Simulationsumgebung ausgelesen. Die Senke entspricht der Senke aus der MATLAB/SimEvents Bibliothek. Analog zu Abschnitt 3.1 sollen auch die Hilfsmodelle einem BBM gleichen. Die Quelle soll WEs generieren und diese mit allen werkstoffspezifischen Eigenschaften, wie Werkstofftyp und Werkstückgeometrie, attributieren. Eine Parametermaske soll die wichtigsten Werkstoffeigenschaften kodieren.
Die Senke soll soweit erweitert werden, dass sie einen zustandsbasierten Simulationsabbruch ermöglicht. Dabei soll die Anzahl der terminierten WEs auf die gewünschte Anzahl zu fertigender WEs verglichen werden. Wurde beispielsweise die notwendige Anzahl der WEs terminiert, so soll die Simulation abgebrochen werden. Ein zustandsbasierter Simulationsabbruch soll zur Senkung der Simulationslaufzeit beitragen.

3.2.2. CNC-Satzbasierte Basismodelle

Die Restrukturierung der CNC-satzbasierten BMs soll unter der Berücksichtigung der Hauptanforderungen aus Abschnitt 3.1 erfolgen. Um den Hauptanforderung gerecht zu werden, wird eine neue Modellstruktur (Abb. 3.2) vorgeschlagen.

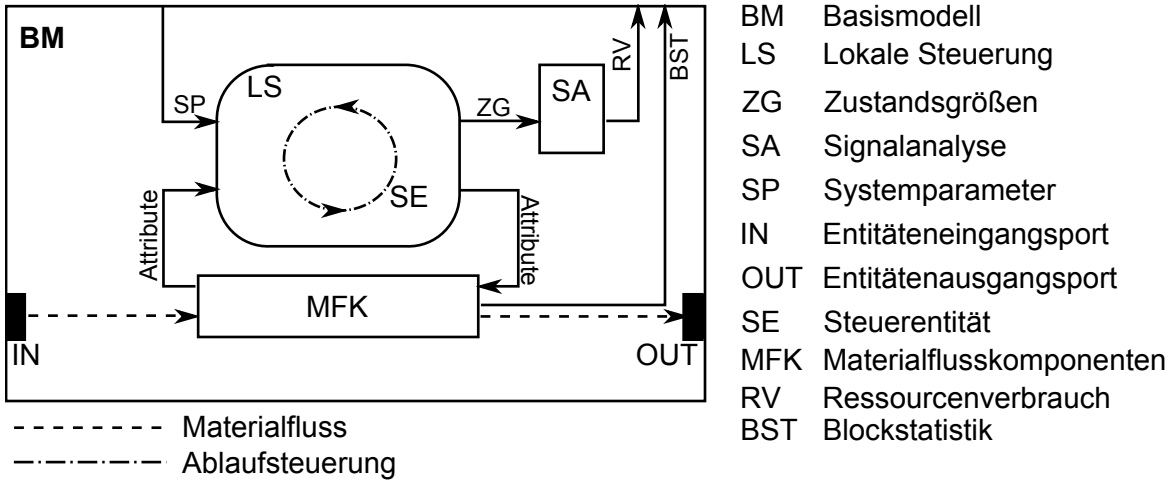


Abbildung 3.2.: Schematische Darstellung der neuen Modellstruktur der CNC-Satzbasierten BMs.

Anders als in [Lar12] soll bei der neuen BM-Struktur der Materialfluss von der Ablauf-

steuerung getrennt und die Subsysteme Zustandsberechnung sowie Ereignissteuerung zu dem Subsystem lokale Steuerung (LS) kombiniert werden. In dem Subsystem LS soll eine Steuerentität (SE) auf dem Pfad der Ablaufsteuerung solange verweilen bis die Fertigungsaufgabe beendet wurde. Weiterhin soll die SE globale und persistente Variablen als Entitäten bezogene Attribute kapseln und einen Austausch ausgewählter Attribute zwischen einer Werkstückentität (WE) , die auf dem Materialflusspfad wandert, und einer SE ermöglichen.
Die in Abbildung 3.2 gezeigte Signalanalyse wird aus [Lar12] weitgehend übernommen.

Nachfolgend wird die grundlegende Abarbeitung der neuen CNC-Satzbasierten BMs erläutert.

1. Eintritt einer WE in das BM
2. Übermitteln ausgewählter Attribute der WE an das Subsystem LS
3. Verharren der WE im BM und Aktivierung von LS
   a) Generierung SE mit im LS
   b) Attributierung der SE mit den notwendigen Attributen der WE und Systemparametern
   c) Abarbeitung der Fertigungsaufgabe und Berechnung der entsprechenden Zustandsgrößen
   d) Übermittlung der Zustandsgrößen an die Signalanalyse
   e) Ist die Fertigungsaufgabe abgeschlossen, werde ausgewählte Attribute von der SE auf die WE übertragen
   f) Vernichtung der SE
4. WE verlässt das BM

3.2.3. Messdatenbasierte Basismodelle

Messdatenbasierte Basismodelle lesen im vor Feld eines Simulationslaufes Messdaten ein, die den entsprechenden Ressourcenverbrauch kodieren. Die Messdaten werden zur Laufzeit vom BM sukzessiv abgearbeitet und werden nach der Simulation an die Simulationsumgebung übermittelt. Der Ressourcenverbrauch beschränkt sich bei dieser Modellklasse ausschließlich auf die elektrische Leistung.

Als Simulationsergebnis erhält man den zeitlich versetzten Ressourcenverbrauch, was die Abbildung 3.3 schematisch verdeutlichen soll.

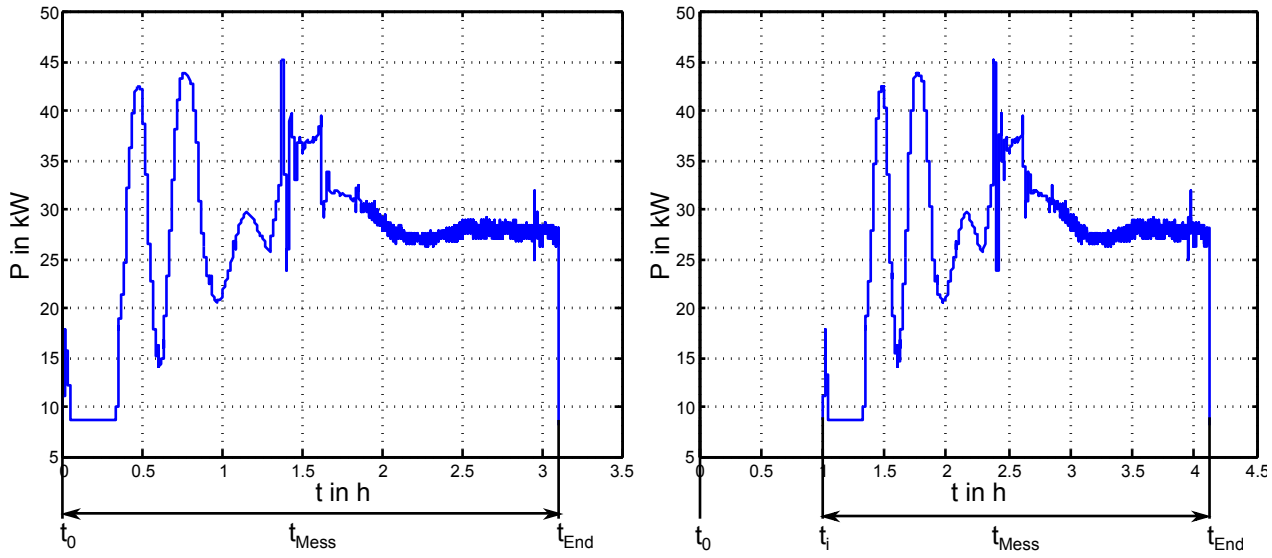


Abbildung 3.3.: Schematische Darstellung der Messdaten (links) und des Simulationsergebnisses (rechts) vom BM Vakuumhärten.

Dabei stellt t_i den Ereigniszeitpunkt der Aktivierung des BMs und t_{End} stellt den Ereigniszeitpunkt dar, bei dem die WE das BM wieder verlässt . Die Zeitspanne t_{Mess} stellt die Messdauer und die Bedienzeit des BMs dar. In der Abbildung ist auch zu erkennen, dass eine sukzessive Abarbeitung der Messdaten nicht notwendig ist. Im BM müssen die Messdaten nur auf der Zeitachse um t_i verschoben und die WE um t_{Mess} verzögert werden.
Abbildung 3.4 zeigt neue Modellstruktur der Messdatenbasierten BMs.

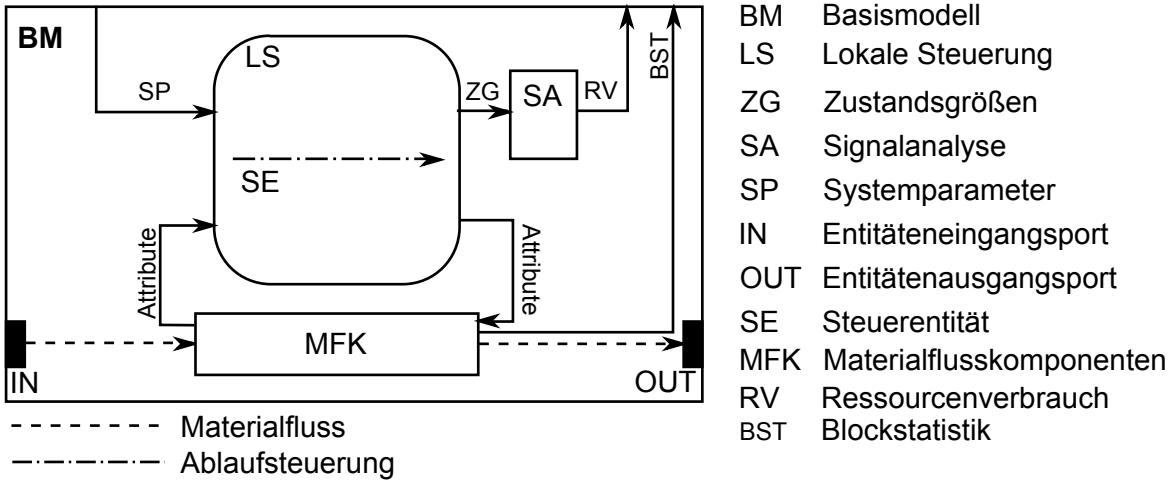


Abbildung 3.4.: Schematische Darstellung der neuen Modellstruktur der Messdatenbasierten BMs.

Wie auch bei den CNC-Satzbasierten BMs werden auch hier eine SE und WE zur Aktivierung BMs und der Abarbeitung der Fertigungsaufgabe eingesetzt. Auch die SA wird aus weitestgehend unverändert aus [Lar12] übernommen.

Die Abarbeitung der Messdatenbasierter BMs wird nachfolgend gezeigt.

1. Eintritt der WE in das BM
2. Übermitteln ausgewählter Attribute der WE an LS
3. Verharren der WE im BM und Aktivierung von LS
   a) Generierung einer SE im LS
   b) Attributierung der SE mit den notwendigen Attributen der WE und Systemparametern
   c) Ermittlung von t_i
   d) Verschieben der Messdaten um t_i
   e) Setzen der Bedienzeit auf t_{Mess}
   f) Nach Ablauf von t_{Mess} werden ausgewählte Attribute von der SE auf die WE übermittelt
   g) Vernichtung der SE
4. WE verlässt das BM

3.3. Integration von Steuerschnittstellen

Die Basismodelle werden durch das Eintreten einer Werkstück-Entität (WE) aktiviert und durch das raus leiten dieser wieder in den Ruhemodus versetzt. Dabei besitzen die keine Kenntnisse über

- die Anzahl der Basismodelle im Simulationsmodell
- die Gesamtzahl der zu fertigenden WEs
- die Ankunft der nächsten WE

Speziell der letzte Punkt ist an dieser Stelle zu beachten.
Bei einer langen Ankunftszeit einer WE befindet sich ein Basismodell im Ruhemodus. Nur die Grundlast der Fertigungsmaschine kann als einziger Verbraucher angesehen werden. Auch bei unendlich langen Ankunftszeiten besteht keine Möglichkeit die Fertigungsmaschine auszuschalten, was in der Realität der Fall sein müsste.
An einem Beispiel soll dieser Sachverhalt verdeutlicht werden.

Es sollen fünf WEs durch das Basismodell Außenlängsdrehen bearbeitet werden. Die Ankunftszeit der einzelnen Entitäten wird auf 90 Minuten gesetzt und durch einen dem Basismodell vorgeschalteten *Single Server* realisiert. In der nachstehenden Abbildung 3.5 ist das Lastprofil des Basismodells Außenlängsdrehen graphisch dargestellt.

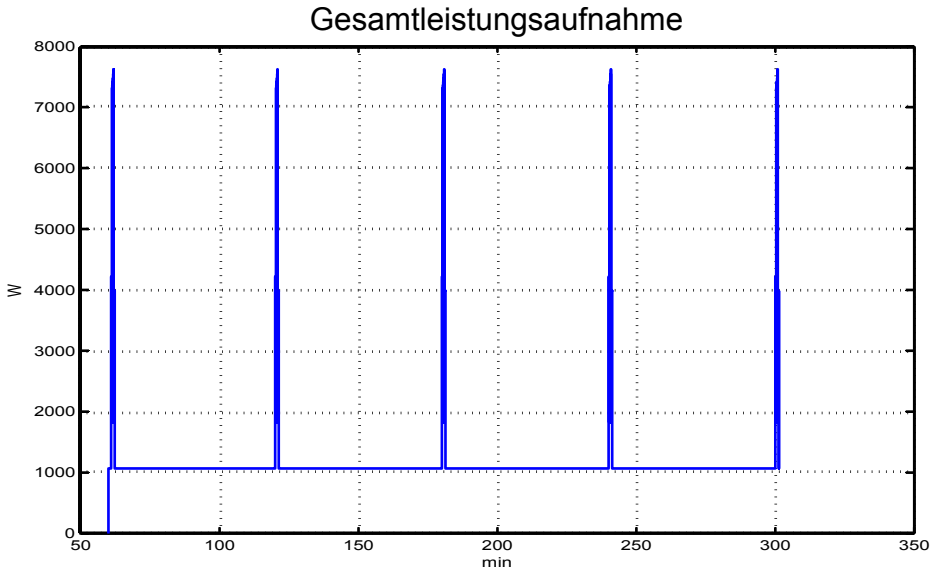
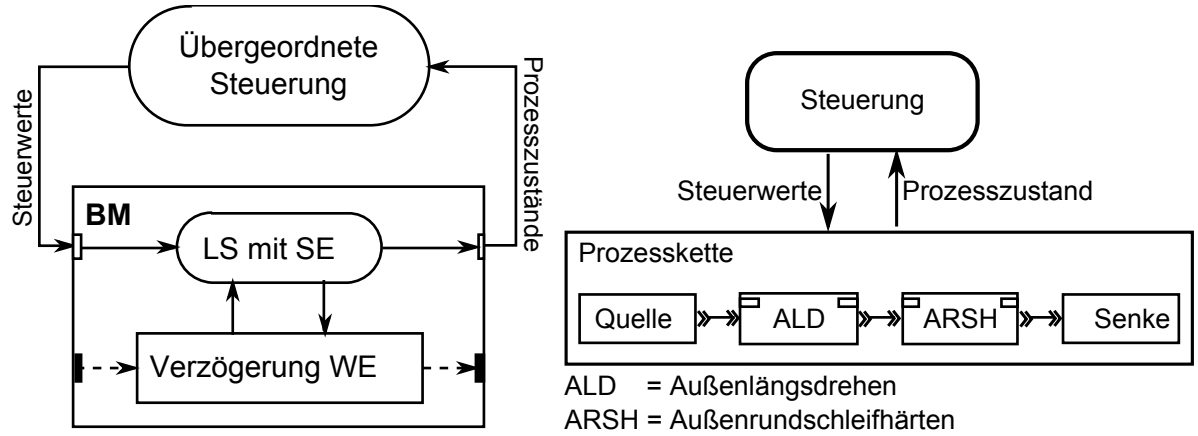


Abbildung 3.5.: Leistungsprofil des Außenlängsdrehen Basismodells.

In der gezeigten Abbildung machen die gut zu sehenden Leerlaufphasen bis zu 86% des Gesamtverbrauches aus.

Um diese Leerlaufzeiten zu senken und somit den Gesamtverbrauch zu reduzieren, sollen Steuerschnittstellen in die Basismodelle implementiert werden. Die Steuerschnittstellen sollen das Ein- und Ausschalten der einzelnen Basismodelle in einer Prozesskette durch eine übergeordnete Steuerung zur Laufzeit ermöglichen.

Abbildung 3.6 zeigt ein Basismodell und eine Prozesskette mit Steuerschnittstellen und einer übergeordneter Steuerung.



(a) Basismodells mit Steuerung und Steuerschnittstellen.

(b) Prozesskette mit Steuerung.

Abbildung 3.6.: Basismodell und Prozesskette mit Steuerschnittstellen und einer Steuerung.

3.4. Zusammenfassung

In diesem Kapitel wurden grundlegende Konzepte zur Restrukturierung der Basismodell erarbeitet. Ausgehend von dem grundlegendem Ansatz jedes Basismodell als ein parametrierbares Black-Box-Modell zu reimplementieren, wurden zwei neue Modellstrukturen zu den Messdaten sowie CNC-Satzbasierten Basismodellen vorgeschlagen. Weiterhin wurden Hauptanforderungen an die neuen Strukturen konzipiert, welche in nach in der Tabelle 3.1 zusammengefasst sind.

Tabelle 3.1.: Zusammenfassung der Hauptanforderungen

ID	Hauptanforderungen
HA1	*Abbildung der globalen und persistenten Variablen sowie der Werkstofftypen 100Cr6 und 42CrMo4 als Entitäten-bezogene Attribute*
HA2	*Implementierung einer klassenspezifischen Parametermaske*
HA3	*Implementierung der verfahrenspezifischen Funktionen in MATLAB/SimEvent*
HA4	*Realisierung eines zustandsbasierten Simulationsabbruches*
HA5	*Implementierung von Steuerungsschnittstellen*
HA6	*Reduzierung der Anzahl der Blöcke*

Die konkrete Umsetzung der hier vorgestellten Hauptanforderung und Konzepte zur Restrukturierung der BMs erfolgt im nächsten Kapitel.

4. Reimplementierung der Basismodelle in MATLAB/SimEvents

Kapitel 4 beschäftigt sich mit der Implementierung der im Kapitel 3 konzipierten Überlegungen bezüglich der Restrukturierung der einzelnen Basismodelle. Zunächst wird auf die Implementierung der Hilfsmodelle also der Quelle und Senke eingegangen. Danach auf die restrukturierten CNC-Satzbasierten und Messdatenbasierten Basismodelle.

4.1. Hilfsmodelle

Für die Restrukturierung der Hilfsmodelle wurden hauptsächlich die Hauptanforderungen HA1 , HA2 und HA4 aus der Tabelle 3.1 auf der vorherigen Seite herangezogen. Auf die jeweiligen Änderungen wird in den nächsten Unterabschnitten näher eingegangen.

4.1.1. Parametermasken

Jedes Hilfsmodell ist mit einer Parametermaske zur übersichtlichen Parametrierung versehen (vgl. Abb. 4.1).

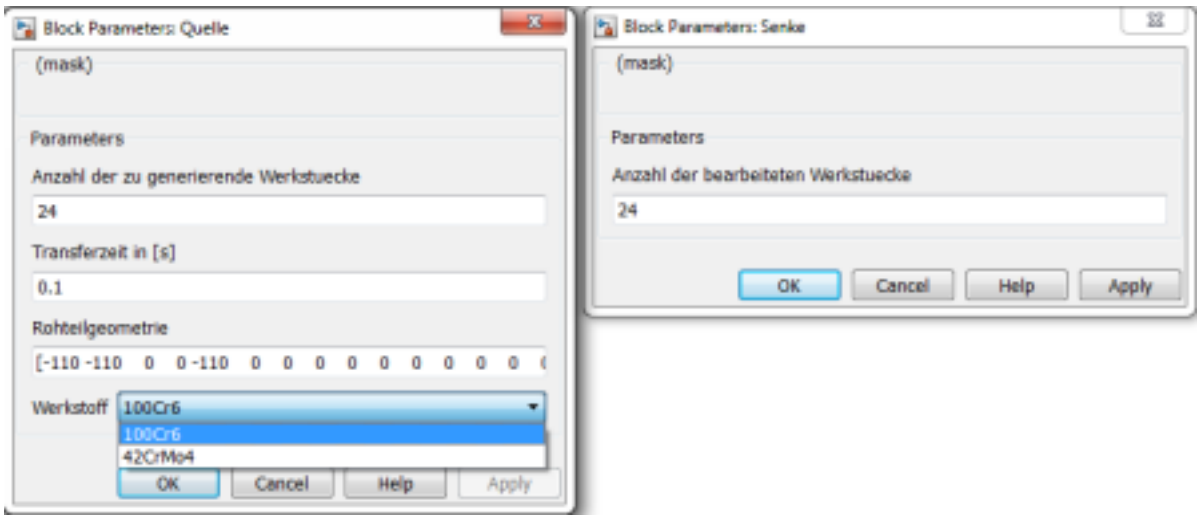


Abbildung 4.1.: Darstellung der Paramtermasken der Hilfsmodelle Quelle (links) und Senke (rechts).

Der Anwender kann in der Parametermaske der Quelle genau angeben wie viele Werkstückentitäten (WEs) und mit welcher Transferzeit[1] generiert werden sollen. Darüber hinaus kann eine gewünschte Rohteilgeometrie der Werkstücke vergeben und deren Werkstofftyp 100Cr6 oder 42CrMo4 ausgewählt werden. Die Rohteilgeometrie entspricht dem in [Lar12] beschriebenen numerischen Werkstückmodell, die zur Simulationszeit von den Basismodellen eingelesen und anhand der Fertigungsaufgabe abgeändert wird.
In der Parametermaske der Senke kann der Anwender im Feld *Anzahl der bearbeiteten Werkstücke* einen numerischen Wert übergeben. Dieser Wert ist für einen zustandsbasierten Simulationsabbruch (HA4) notwendig. Nach der Terminierung der dort eingetragenen Anzahl von Werkstückentitäten wird die Simulation abgebrochen und somit zur Reduzierung der gesamt Simulationslaufzeit beitragen.

Aus den gezeigten Parametermasken geht hervor, dass 24 Werkstückentitäten aus dem Werkstoff 100Cr6 und einer geforderten Rohteilgeometrie alle 0.1 Sekunden generiert werden soll. Die Simulation soll solange andauern, bis alle 24 Werkstückentitäten in der Senke terminiert wurden. Soll die Simulation erst abbrechen, wenn die vordefiniert Simulationszeit abgelaufen ist, so muss in der Parametermaske *inf*[2] einzutragen.

Jedes Feld der Parametermaske ist nur innerhalb des jeweiligen Hilfsmodells gültig und kann nicht mit gleichartigen BMs innerhalb eines Simulationsmodells kollidieren.

[1] Die Transferzeit entspricht der Intergenerationszeit eines zeitbasierten Entitätengenerators in MATLAB/SimEvents.

[2] Did Abkürzung *inf* steht in MATLAB/Simulink für *Infinity*.

Abbildung 4.2 zeigt beispielhaft den Maskeneditor von Simulink und die interne Kodierung der einzelnen Variablen innerhalb des Hilfsmodells Quelle.

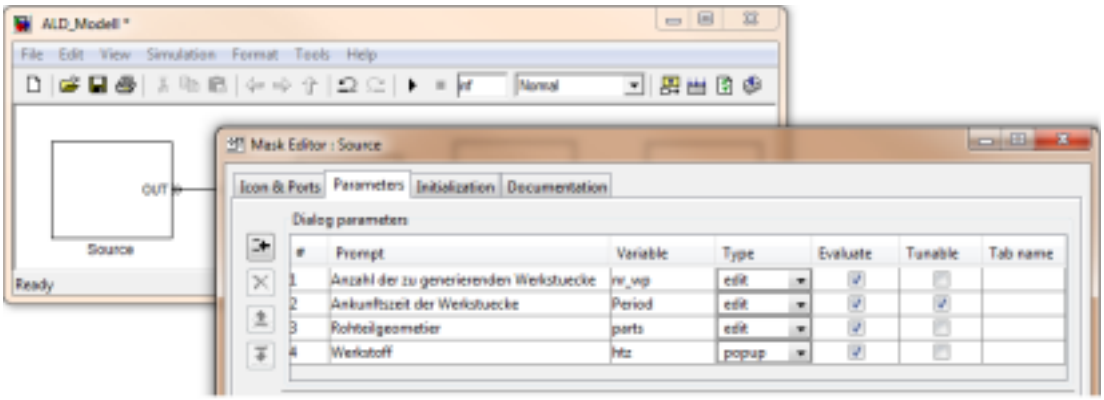


Abbildung 4.2.: Maskeneditor von Simulink und interne Kodierung der einzelnen Variablen der Quelle.

Im Weiteren wird nur auf einige wenige Aspekte des Maskeneditors von Simulink eingegangen. Detaillierte Informationen zur Erstellung von Parametermasken sind in [Mat11b] zu finden.

Im Bereich *Prompt* wird der Anzeigetext der einzelnen Felder der Parametermaske angegeben (vgl. Abb. 4.1). Der Type kodiert die Feldeigenschaft. Dabei erlaubt *edit* die Erstellung editierbarer Felder, in denen der Nutzer Werte eingeben kann. Der übergebene Wert wird dann der Variablen, die im Bereich *Variable* definiert wurde, gespeichert und dann über diese innerhalb des Subsystems verwendet werden. Zum Beispiel wird der Wert 24 aus dem Feld *Anzahl der zu generierende Werkstuecke* aus Abbildung 4.1 intern auf der Variablen *nr_wp* gespeichert. Der Geltungsbereich der Variablen einer Parametermaske ist ausschließlich auf das jeweilige Subsystem (hier Quelle) begrenzt. Weiterhin wird der Zugriff auf die Variablen mit „nur lesend“ eingeschränkt. Im nächsten Abschnitt wird detailliert auf die strukturellen Änderungen der jeweiligen Hilfsmodelle eingegangen.

4.1.2. Quelle

Die Struktur der Quelle (vgl. Abb 4.3) wurde weitgehend aus [Lar12] übernommen und um HA1 sowie HA2 der Tabelle 3.1 auf Seite 25 erweitert.

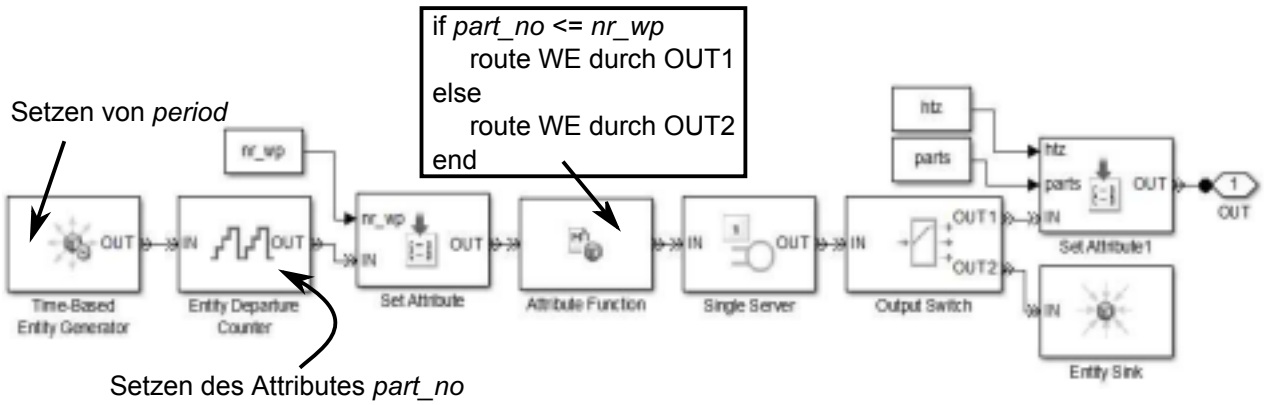


Abbildung 4.3.: Interne Struktur der Quelle.

Der *Time-Based Entity Generator* generiert unter Verwendung der Maskenvariablen (vgl. Abb. 4.2) *period* Werkstückentitäten (WE), die im nachfolgenden Block mit einem fortlaufenden Identifikationsattribut *part_no* attributiert wird. Weiterhin erfolgt die Attributierung der WE mit dem Attribut *nr_wp*. Der zugehörige Wert wird aus der Parametermaske entnommen. Im *Attribute Function Block* wird geprüft, ob die gewünschte Anzahl der WEs generiert wurden. Ist das nicht der Fall, so werden die WEs über den ersten Port des *Output Switches* geroutet. In anderen Fall werden die WEs über den Ausgangsport *OUT2* des *Output Switches* zur *Entity Sink* durchgeleitet. Um sicher zu stellen, dass der *Entity Generator* nicht weiterhin WEs mit der Zeitschrittweite *period* generiert und diese zum gleichen Ereigniszeitpunkt in der *Entity Sink* terminiert werden, muss die Funktionalität *Input port available for entity arrivals* der *Entity Sink* deaktiviert werden. Auf diese Weise blockiert der Eingangsport der Sink und es werden keine WEs angenommen. Da die WEs nicht terminiert werden, kann auch der Generator keine weiteren WEs generieren. Somit wird sichergestellt, dass nur die gewünschte Anzahl der WEs die Quelle verlassen und im gesamten Simulationsmodell im Umlauf sind.
Alle WEs, die das Hilfsmodell Quelle verlassen, werden zusätzlich mit den Attributen *htz* und *parts* der Parametermaske attributiert und besitzen die in der Tabelle 4.1 aufgeführten Attribute.

Tabelle 4.1.: Attribute der WE nach dem Verlassen der Quelle.

Attribut	Beschreibung
nr_wp	Anzahl der zu generierenden Werkstücke
part_no	fortlaufende Identifikationsnummer
parts	Rohteilgeometrie
htz	Werkstofftyp

4.1.3. Senke

In der Senke werden WEs nach der Bearbeitung der vorgelagerten BMs terminiert. Weiterhin ist in der Senke die Möglichkeit zum zustandsbasierten Simulationsabbruch realisiert (HA4) worden. Abbildung 4.4 zeigt die neue interne Struktur der Senke mit der Realisierung des zustandsbasierten Simulationsabbruches.

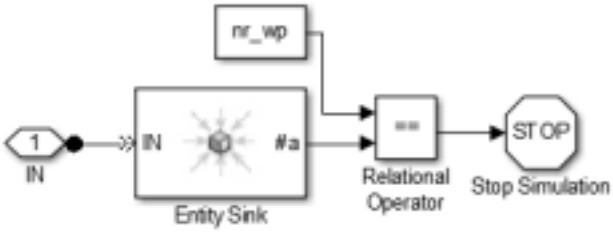


Abbildung 4.4.: Interne Struktur der Senke.

Wie auch das Hilfsmodell Quelle besitzt auch die Senke eine Parametermaske (vgl. Abschnitt 4.1.1). Der Anwender kann durch einen nummerischen Wert, der intern auf der lokalen Variablen *nr_wp* gespeichert wird, wann die Simulation abgebrochen werden soll. Über den *IN* Eingangsport gelangen WEs in die *Entity Sink* und werden gleich terminiert. Der Ausgabeport *#a* der Entity Sink gibt die Anzahl der terminierten WEs aus. Dieser Wert wird dan mit *nr_wp* (Anzahl der bearbeiteten Werkstücke) verglichen. Ist die Bedingung wahr, so bricht die Simulation ab. Möchte der Anwender diese Möglichkeit des zustandbasierten Simulationsabbruches nicht in Anspruch nehmen, so ist in der Parametermaske der Senke der Wert *inf* einzutragen. Die Simulation bricht dann erst ab, wenn die vordefinierte Simulationszeit abgelaufen ist.

4.2. CNC-Satzbasierte Basismodelle

Bei den CNC-satbasierten Basismodellen wurden die im dritten Kapitel beschriebenen Hauptanforderungen HA1-3 und HA6 umgesetzt. H4 wird bei der Restrukturierung der CNC-satzbasierten BMs nicht herangezogen, da diese ausschließlich für das Hilfsmodell Senke wichtig ist. Die Darstellung der Implementierung der CNC-satzbasierten Basismodellen soll nur an Hand des Basismodells Außenlängsdrehen erfolgen. Gemäß der Forderung der isomorphen Implementierung aus Kapitel 3, gilt die vorgestellte neue Struktur für alle CNC-satzbasierte BMs. Abbildung 4.5 zeigt die Implementierung des Basismodells Außenlängsdrehen.

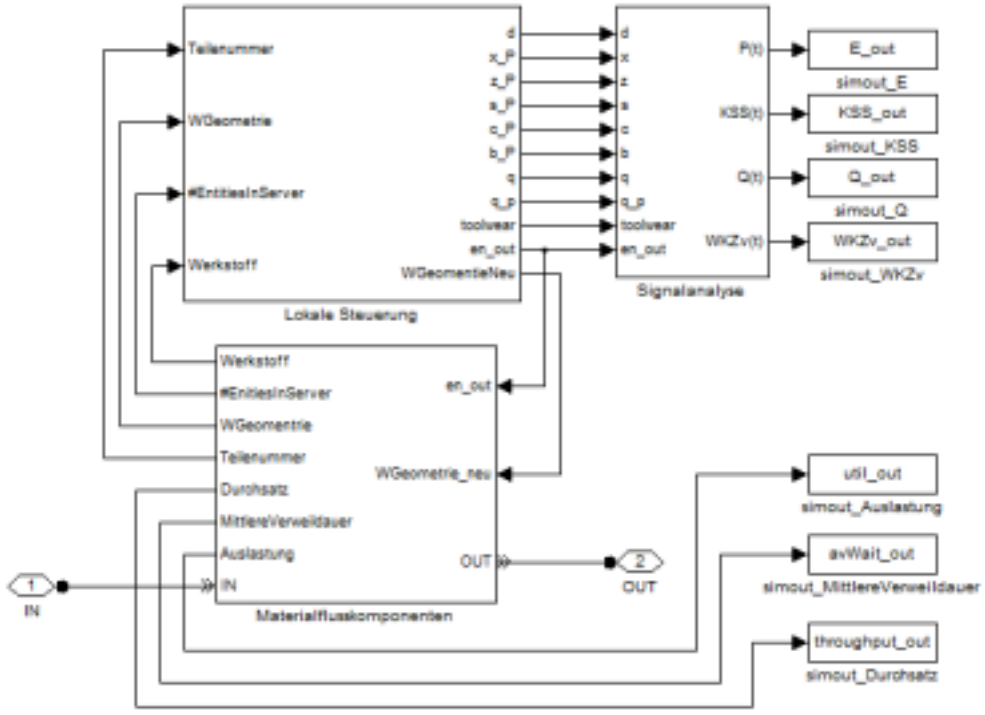


Abbildung 4.5.: Exemplarische Darstellung der neuen Struktur eines CNC-satzbasierten Basismodells.

Das Basismodell ist in drei Subsysteme: (i) *Lokale Steuerung (LS)*, (ii) *Materialflusskomponenten (MFK)* und (iii) *Signalanalyse (SA)* eingeteilt. Wobei letzteres ohne Änderungen aus [Lar12] übernommen wurde. Über den *IN* Eingangsport gelangen WEs in das Subsystem Materialflusskomponenten. Dort werden WE bezogene Attribute wie Werkstoff, Werkstoffgeometrie und Teilenummer ausgelesen und dem Subsystem LS übergeben. Weiterhin berechnet die MFK statistische Größen, wie Durchsatz, mittlere Verweildauer der WE im BM und die Auslastung des BMs, die über die entsprechenden Ausgabeports ausgegeben werden.
Die LS erhält als Eingabewert die von der WE ausgelesenen Attribute und ein *#EntitiesInServer* Wert mit dem Wertebereich 1 und 0. Tritt eine WE in die MFK ein, so gibt die MFK eine 1 über den Port *#EntitiesInServer* an die LS weiter. Diese beginnt dann mit der Berechnung der Ressourcenverbräuchen und gibt diesen an die Signalanalyse zur weiteren Verarbeitung an die Signalanalyse weiter. Block mit den Bezeichnern *simout_* entsprechen den to Workspace Blöcken aus der MATLAB/Simulik Bibliothek. Alle berechneten Größen werden an diese Blöcke übergeben und dort gespeichert. Nach dem Ende der Simulation werden alle Ergebnisse der Simulationsumgebung für das Weitere verarbeiten zur Verfügung gestellt.

Auf den folgenden Seiten wird detailliert auf die zwei Subsysteme MFK und LS sowie auf die Parametermaske eingegangen.

4.2.1. Parametermaske

Zur übersichtlichen Parametrierung wurden alle wesentlichen Systemparameter eines CNC-satzbasiertes Basismodell in einer Parametermaske zusammengefasst. Somit erhält der Anwender ein Gesamtüberblick über die Einstellungsmöglichkeiten des entsprechenden Basismodells. Es sei drauf hingewiesen, dass folgend nur die Parametermaske des Außenlängsdrehen BMs (vgl. Abb. 4.6) als Beispiel für alle weiteren CNC-satzbasierten BM verwendet wird.

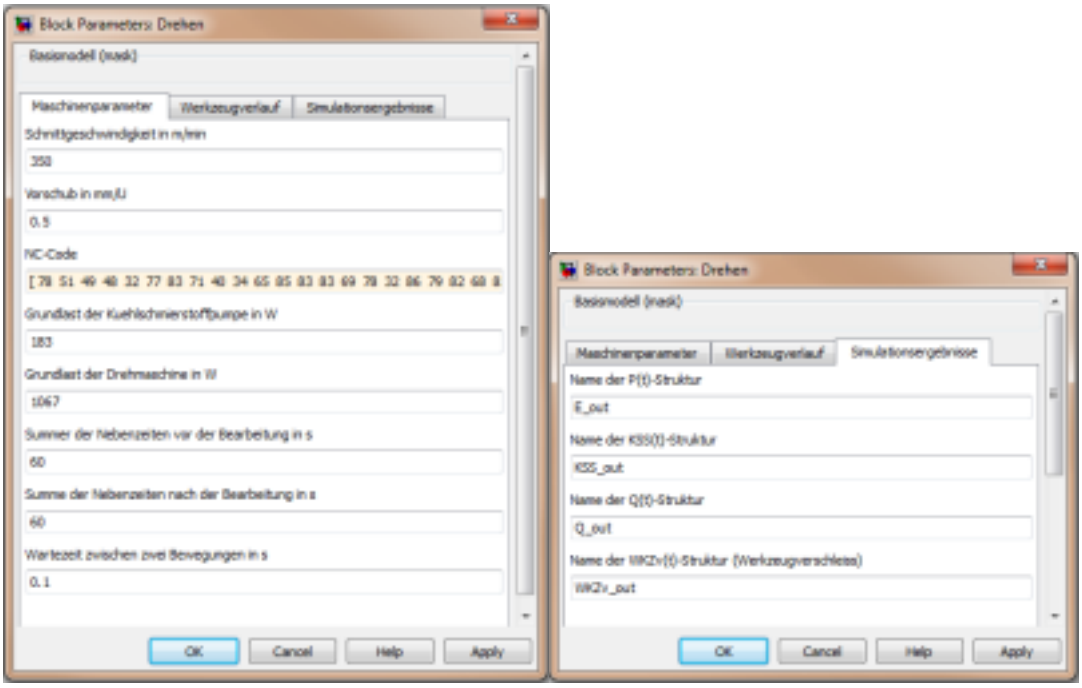


Abbildung 4.6.: Parametermaske des Basismodells Außenlängsdrehen.

Die Parametermaske beinhaltet drei Reiter: (i) *Maschinenparameter*, (ii) *Werkzeugverlauf* und (iii) *Simulationsergebnisse*. Im ersten Reiter werden Maschinenspezifische Parameter wie Schnittgeschwindigkeit, Vorschub, NC-Programm etc. gesetzt. Letzteres muss jedoch zu nächst ASCII kodiert werden. Warum eine ASCII Kodierung des NC-Programms erfolgen muss wird im Abschnitt 4.2.3 näher beleuchtet. Im Reiter Werkzeugverlauf kann eine leere 2xn Matrix zur Speicherung der X-Z-Werzeugverlaufes angegeben werden. Nach der Simulation kann zum Beispiel eine Analyse der Werkzeugverfahrwege erfolgen (vgl.

[Lar12]) . Im Reiter Simulationsergebnisse müssen Variablennamen für die *to Workspace* Blöcke (simout_## in Abbildung 4.5) vergeben werden, die zur Simulationslaufzeit die Ergebnisse speichern und nach der Simulation der Simulationsumgebung zur Verfügung stellen. Die in den entsprechenden Feldern eingetragenen Zeichenketten müssen gültige MATLAB Variablennamen sein, da sie nach der Simulation im MATLAB Workspace angelegt werden. Sind die Übergebenen Zeichenketten keine gültigen Variablennamen von MATLAB, so können die Simulationsergebnisse nach der Simulation nicht an das MATLAB Workspace übergeben werden. Es sei darauf hingewiesen, dass die in Abbildung 4.6 im Reiter Simulationsergebnisse nicht alle Felder angezeigt sind. Nicht dargestellt sind die Felder für die Vergabe der Variablennamen der statistischen Größen (vgl. Abb. 4.5).

Die interne Kodierung der einzelnen Felder erfolgt nach dem selben Schema wie im Abschnitt 4.1.1. in Abbildung 4.2 gezeigt und beschrieben ist.

4.2.2. Subsystem *Materialflusskomponenten*

Abbildung 4.7 zeigt die innere Struktur des Subsystems Materialflusskomponenten (MFK). Die MFK durchlaufen ausschließlich Werkstückentitäten (WEs). Die WEs treten durch den IN Eingangsport in das Subsystem. Im *Get Attribute* Block werden WE spezifische Attribute wie hier zum Beispiel Werkstoff, Werkstückgeometrie (WGeometrie) und die Teilenummer ausgelesen und über die Ausgabeports an die lokale Steuerung übermittelt (vgl. Abb. 4.5).

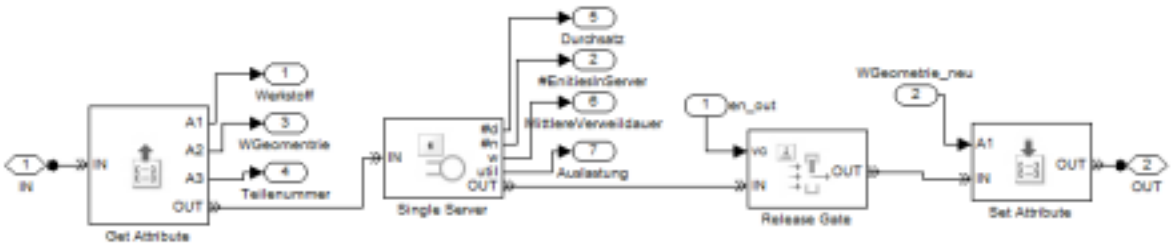


Abbildung 4.7.: Subsystem Materialflusskomponenten eines CNC-satzbasierten Basismodells.

Danach gelangt die WE in den *Single Server* mit der Bedienzeit von 0 Zeiteinheiten, der die statistischen Größen berechnet und auch ausgibt. Der *Single Server* ist Bestandteil der MATLAB/SimEvents Bibliothek. Dem Single Server nachgelagerte *Release Gate*

verhindert die automatische Weiterleitung der WE. Somit verweilt die WE solange im Single Server bis die lokale Steuerung das Ereignis *en_out* generiert. Dieses Ereignis wird zum Öffnen des Release Gates genutzt und somit kann die WE den Single Server verlassen. Bevor die WE das Subsystem MFK verlässt besteht auch die Möglichkeit neue Attribute auf die WE im *Set Attribut* Block zu setzen. Zum Beispiel wird am Anfang die Werkstückgeometrie ausgelesen und der lokalen Steuerung übergeben. Diese arbeitet das NC-Programm ab und ändert dabei auch die Werkstückgeometrie. Verlässt jetzt die WE das Subsystem MFK, so wird die neue Werkstückgeometrie auf der WE gespeichert bzw. aktualisiert und die WE kann jetzt in das nächste BM ereignisorientiert betreten.

4.2.3. Subsystem *Lokale Steuerung*

Die lokale Steuerung bildet die eigentliche Berechnungseinheit der Ressourcenverbräuche. Zur Berechnung der Zustandsgrößen wird ein Entitäten basierter Ansatz gewählt. Dabei wir eine Steuererntität (SE) erzeugt, mit den gewünschten Parametern bzw. Attributen und die Berechnung aktiviert. Eine Steuerentität kann mit dem Datentyp *Struktur* gleichgesetzt werden. Felder der Strukturen entsprechen den Attributen der SE. Diese werden von den *Attribute Function* Blöcken (vgl. Abb.4.8) ausgelesen, manipuliert und erneut als Attribute auf der SE geschrieben.

In der Abbildung 4.8 wird die interne Struktur der lokalen Steuerung gezeigt. Zur besseren Nachvollziehbarkeit der Abarbeitung der lokalen Steuerung ist ein Pseudocode 4.1 auf Seite 37 angegeben.
Durch das eintreten des *#EntitiesInServer* Ereignisses, dass durch die MFK generiert wird, wird im Subsystem lokale Steuerung (LS) eine SE erzeugt. Anschließend erfolgt die Initialisierung der SE Attribute, die den globalen und persistenten in [Lar12] entsprechen. Nach dem selben Ansatz, wie im Abschnitt 4.1.1 beschrieben, werden die Werte der Parametermaske als entitätenbezogene Attribute abgebildet. Es ist nur zu beachten, dass die Attribute ausschließlich nummerische Werte sein dürfen. Aus diesem Grund muss das NC-Programm ASCII-Kodiert werden (vgl. Abschnitt 4.2.1).
Nach der Initialisierung der SE gelangt sie über den *Path Combiner* in den *Attribute Function* Block (vgl. Abb. 4.8). In diesem Block findet die sukzessive Interpretation des NC-Programms statt. Hierzu werden alle relevanten Attribute von der SE ausgelesen, manipuliert und wieder zurück auf die SE gespeichert. So wird zum Beispiel aus dem NC-Programm, welches bei der Interpretation wieder in eine Zeichenkettenmatrix umgewandelt wird, die entsprechenden Wörter *G1* oder *G0* identifiziert (vgl. Anhang

B) und zum Routen der SE durch den *Output Switch* genutzt. Die Berechnung des Ressourcenverbrauches erfolgt dann in den Attribute Function Blöcken G1 und G0 an Hand der in [Lar12] definierten verfahrenspezifischen Funktionen. Auch hier werden für die Berechnungen notwendigen Daten von der SE ausgelesen und am Ende erneut auf die SE gespeichert. Im *Get Attribute* Block werden die gewünschten Ressourcenverbräuche ausgelesen und an das Subsystem *Signalanalyse* übermittelt. Als nächstes gelangt die SE in den Single Server und Verweilt dort für die entsprechende Zeitdauer. Diese Zeitdauer wird in den G1 und G0 Funktionen über die Gleichung der gleichförmigen Bewegung (Gleichung 4.1) berechnet und entspricht zum Beispiel der Zeit, die das Werkzeug (zum Beispiel Drehmeißel) für das Zurücklegen einer bestimmten Strecke benötigt.

$$v_f = \frac{s_{soll} - s_{ist}}{\triangle t} \tag{4.1}$$

Zur Berechnung der Zeitspanne $\triangle t$ muss die obige Gleichung umgestellt werden. Die Verfahrwege s_{soll}und s_{ist} sind aus dem NC-Programm bekannt und die Vorschubgeschwindigkeit v_f ergibt sich aus dem Vorschub, der als Parameter durch den Nutzer vorzugeben ist.
Nach der Verzögerung der SE gelangt diese in den Output Switch. Dort wird an Hand des SE Attributes *processFinished* geprüft, ob die Abarbeitung des NC-Programms abgeschlossen ist. Ist dies der Fall so werden im nachgelagerten *Get Attribute* Block die wichtigen SE Attribute ausgelesen und an die MFK übergeben. Die *Entity Sink* terminiert die SE und generiert das *en_out* Ereignis, welche auch an die MFK übermittelt wird (vgl. Abb.4.5) . Ist die Abarbeitung des NC-Programms nicht abgeschlossen gelangt die SE über den OUT1 Port des Output Switches in den Attribute Function Block *nc_calc*, wo die nächste NC-Programmzeile eingelesen wird.

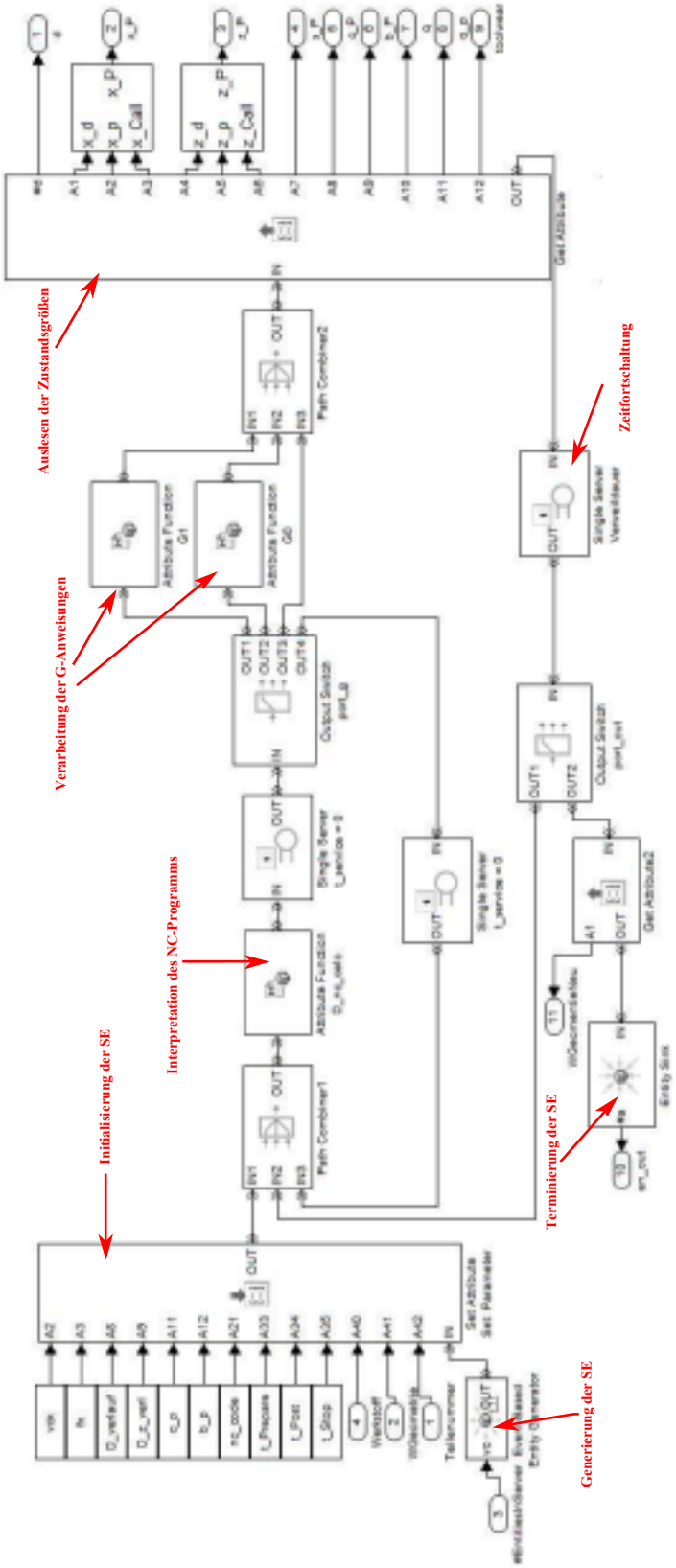


Abbildung 4.8.: Subsystem lokale Steuerung eines CNC-satzbasierten Basismodells.

Listing 4.1: Pseudocode der lokalen Steuerung eines CNC-satzbasierten Basismodells.

```
1  // Wenn das Ereignis #EntitiesInServer auftritt
2  // generiere eine SE
3  if #EntitiesInServer
4          SE=generateSE();
5  endif
6
7  // Initialisiere SE mit den entsprechenden Attributen
8  // einschliesslich der Parameter aus der Parametermaske.
9  // Set Attribute Block in Abbildung 4.8
10 SE.vc = vcx;
11 SE.f = f;
12 ...
13 SE.partNr = Teilenummer;
14
15 // Berechnung der Ressourcenverbraeuche starten
16 while 1
17         // Interpretation des NC-Programms und
18         // Setzen der labels für G1 und G0 Anweisungen
19         // Attribute Function Block nc_calc
20         SE = InterpretationOfNcCode(SE);
21
22         // Steuerung des Output Switches und Verarbeitung
23         // der G Anweisungen
24         if SE.GLabel == 1
25                 SE = ProcessG1Command(SE);
26         elseif SE.GLabel == 2
27                 SE = ProcessG0Command(SE);
28         elseif SE.GLabel == 3
29                 DoNothing
30         elseif SE.GLabel == 4
31                 GoTo Line 24
32         endif
33
34         // Auslesen der Zustandsgrößen Get Attribute Block
35         ...
36         s_P = SE.sP;
37         c_P = SE.cP;
38         ...
39         toolwear = SE.toolwear;
40
41
```

```
42          // Zeitfortschaltung
43          tSim = tSim + SE.time
44
45          // Alle Berechnungen abgeschlossen ?
46          if SE.processFinished
47                  // Auslesen der wesentlichen Attribute
48                  // Get Attribute2 Block und terminiere SE
49                  WGeomentrieNeu = SE.WGeometrieNeu;
50                  terminate(SE);
51                  BREAK
52
53          endif
54  endwhile
```

4.3. Messdatenbasierte Basismodelle

Die interne Struktur der Messdatenbasierten BMs gleichen gemäß Kapitel 3 den CNC-satzbasierten BMs. Analog zu Abschnitt 4.2 wurden auch hier die Hauptanforderungen HA1-4 und HA6 umgesetzt (vgl. Tabelle 3.1). In der Abbildung 4.9 ist die Struktur des Messdatenbasierten BMs zu sehen. Repräsentativ für alle Messdatenbasierten BMs wird ausschließlich das BM Vakuumhärten betrachtet. Die Grundlegende Ablauf entspricht den CNC-satzbasierten BMs aus Abschnitt 4.2.

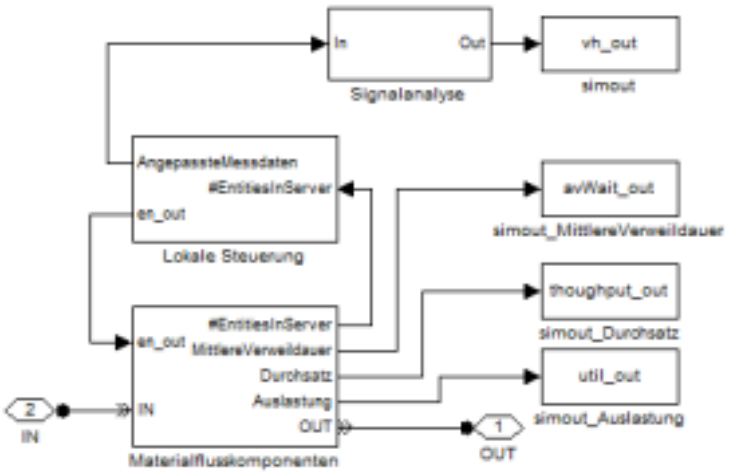


Abbildung 4.9.: Exemplarische Darstellung eines Messdatenbasierten Basismodells.

Die Wesentliche Änderungen beziehen sich ausschließlich auf die Subsysteme *Lokale*

Steuerung und *Materialflusskomponenten*. Auf die BM spezifischen Änderungen wird in den nächsten Abschnitten eingegangen.

4.3.1. Parametermaske

Wie auch die CNC-satzbasierten Basismodelle besitzen auch die Messdatenbasierten Basismodelle zur übersichtlichen Parametrierung. Aufgrund der geringeren Komplexität besitzen die Messdatenbasierten BMs eine stark vereinfachte Parametermaske (vgl. Abb. 4.10). Zur Parametrierung stehen dem Nutzer zwei Tabs: (i) *Haerteofenparamter* und (ii) *Simulationsergebnisse* zur Verfügung. Im Tab *Haerteofenparameter* können Nutzer die Kapazität des Härteofens, Summe der Nebenzeiten vor und nach der Bearbeitung festlegen. Die Kapazität des Härteofens gibt an wie viele Werkstücke der Härteofen aufnehmen kann. In den Feldern *Summe der Nebenzeiten vor und nach der Bearbeitung* kann der Nutzer zum Beispiel Rüstzeiten angeben. Im Aktuellen Fall würde das bedeuten, dass für das beladen und entladen des Härteofens jeweils 600 Sekunden eingeplant werden. In dieser Zeit ist ausschließlich die Grundlast des Ofens aktiv und keine Prozesslast. Da die Messdatenbasierten BMs ausschließlich die am realen Prozess aufgenommenen Messdaten reproduzieren, müssen diese in dem Parametermasken Feld *Messdaten* eingetragen werden. Das Tab Simulationsergebnisse beinhaltet die im Abschnitt 4.2.1 beschrieben Felder.

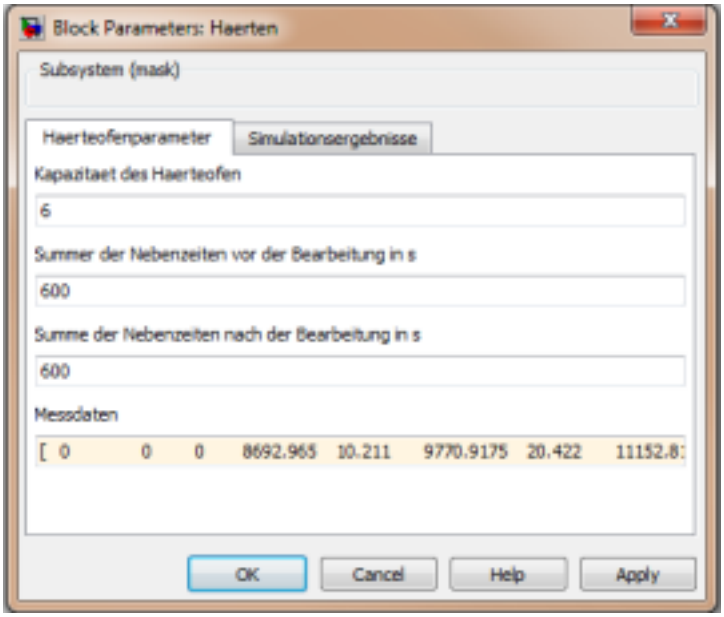


Abbildung 4.10.: Parametermaske des Basismodells Vakuumhärten.

4.3.2. Subsystem Materialflusskomponenten

Abbildung 4.11 zeigt das Subsystem Materialflusskomponenten (MFK) der Messdatenbasierten BMs. Grundlegend sollen sowohl die Subsysteme MFK der CNC-satzbasierten BMs ausch die MFK der Messdatenbasierten BMs einen identischen Aufbau aufweisen (vgl. Kapitel 3). Aus pragmatischen Gründen wurde die Implementierung jedoch unterschiedlich gestaltet. Ein wesentlicher unterschied besteht darin, dass zu Beginn keine Attribute von der Werkstückentität (WE) ausgelesen und auch keine Attribute auf der WE am Ende gespeichert werden (vgl. Abschnitt 4.2.2). Darum sind auch keine *Set Attribute* oder auch *Get Attribute* Blöcke in der Abbildung 4.11 zu sehen.

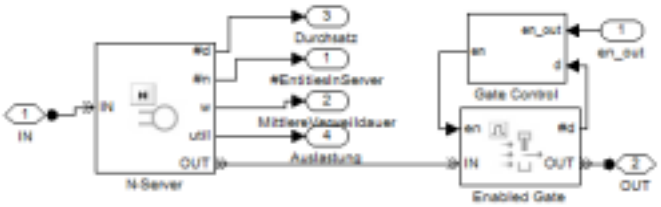


Abbildung 4.11.: Subsystem Materialflusskomponenten eines Messdatenbasierten Basismodells.

Ein weiter unterschied zu der MFK der CNC-satzbasierten BMs besteht darin, dass der *Single Server* duch einen *N-Server* ersetzt wurde. Im Gegensatz zum Single Server kann der N-Server mehrere WEs aufnehmen. Vor der Simulation muss zu nächst festgelegt werden wie viele WEs der N-Server aufnehmen darf. Das geschieht durch die Definition des Parameters *Kapazitaet des Haerteofens* in der Parametermaske (s. Abschnitt 4.3.1). Der letzte Unterschied besteht darin, dass anstelle eines *Releas Gates* ein *Enabled Gate* (EG) verwendet wurde. Das Release Gate lässt beim Auftreten eines Ereignisses genau eine WE durch wobei das EG beim Auftreten eines Ereignissen solange offen bleibt, bis ein erneutes Ereignis das EG wieder schließt. Das dargestellte Gate Control (GC) öffnet beim Auftreten des *en_out* Ereignisses ein *en* Ereignis an das Gate und es wird geöffnet. Die WEs können jetzt den N-Server verlassen. Gleichzeitig übermittelt das EG die Anzahl der WEs (#d Port vom EG), die das EG passiert haben und somit die MFK über den *OUT* Port verlassen haben, an GC (d Port von GC). Wenn die Anzahl der WEs, die das EN passiert haben, der Anzahl der N-Server Kapazität entspricht, wird ein Ereignis von der GC an EG geschickt und das EG geschlossen. Zur Prüfung dieser Bedingung wird der GC analog zum N-Server der Parameter *Kapazitaet des Haerteofens* übergeben.

Nachdem alle WEs die MFK verlassen haben können neue WEs in die MFK eintreten.

4.3.3. Subsystem Lokale Steuerung

Analog den CNC-satzbasierten BMs wird der zeitliche Ressourcenverbrauch in dem Subsystem *Lokale Steuerung* berechnet. Letzteres wird in der nachfolgenden Abbilung 4.12 gezeigt. Weiterhin wird ein Pseudocode 4.2 angegeben.
Immer wenn WEs die MFK betreten wird ein Ereignis #EntitiesInServer generiert. Dieses Ereignis wird in der Lokalen Steuerung dafür verwendet um eine Steuerentität (SE) zu generieren um den Ressourcenverbrauch zu berechnen. Dabei werden im Subsystem *Discrete Event Subsystem* die #EntitiesInServer Ereignisse gezählt. Entspricht die Anzahl der WEs in der MFK der Ofenkapazität wird ein *vc* Ereignis generiert und an das *Event-Based Entity Generator* gesendet. Letzterer generiert die SE, welche im Set Attribute Block mit den entsprechenden Attributen attributiert wird. Im Anschließenden *Attribute Function* Block erfolgt die Anpassung der Messdaten gemäß Abschnitt 3.2.3. Bei der Anpassung der messdaten wird auch die Messzeitdauer ermittelt und als Bedienzeit (im Pseudocode als SE.time angegeben) für den Single Server verwendet. Danach werden die angepassten Messdaten ausgelesen, die SE terminiert und das en_out Ereignis an die MFK übermittelt.

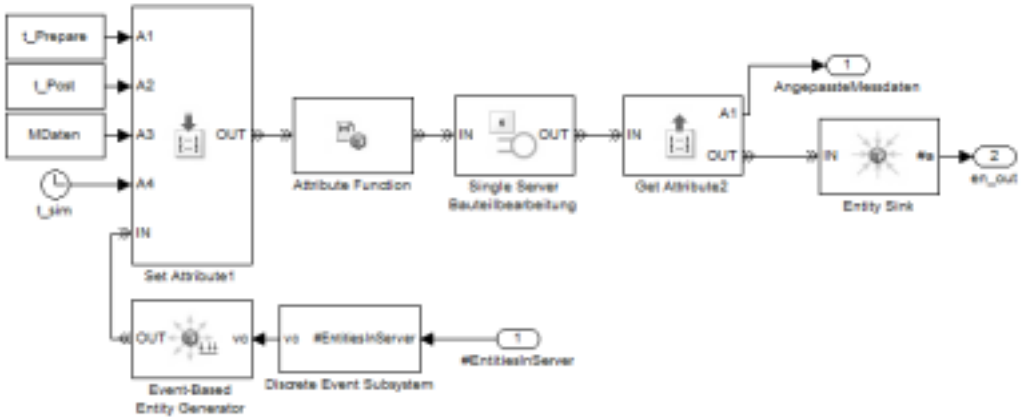


Abbildung 4.12.: Subsystem lokale Steuerung.

Listing 4.2: Pseudocode der lokalen Steuerung eines Messdatenbasierten BMs.

```
1 // Wenn das Ereignis #EntitiesInServer max_capa mal auftritt
2 // (max_capa entspricht der Kapazitaet des Haerteofens)
3 // generiere SE
```

```
4  if #EntitiesInServer == max_capa
5          SE = generateSE();
6  end
7
8  // Initialisiere SE mit den entsprechenden Attributen
9  // einschliesslich der Parameter aus der Parametermaske
10 // Set Attribute Block in Abbildung 4.12
11 SE.tPrepare = t_Prepare;
12 SE.tPost = t_Post;
13 SE.mDaten = MDaten;
14 ...
15 SE.tiSim = t_sim;
16
17 // Verschieben der Messdaten um die Simulationszeit
18 // analog zum Abschnitt 3.2.3 (ti entspricht SE.tiSim)
19 // Attribute Function Block
20 SE = adjustMeasurmData(SE);
21
22 // Zeitfortschaltung (SE.time entspricht tMess in Abschnitt 3.2.3)
23 // Single Server
24 tSim = tSim + SE.time;
25
26 // Lese neue Messdatan von der SE
27 // Get Attribute
28 AngepassteMessdaten = SE.adjustMeasData;
29
30 // Alle Berechnungen abgeschlossen
31 // Terminiere SE
32
33 terminate(SE);
```

5. Validierung der Basismodelle

Die im vierten Kapitel vorgestellten Basismodelle sind an Hand der in [Lar12] entwickelten Basismodellen zu validieren. Mit anderen Worten, es soll nur das BM Verhalten validiert werden. Eine solche Validierung wird in der Literatur [Wei10] *Black-Box-Testing* (BBT) genannt. Beim BBT wird das zu testende Modell als eine Black-Box betrachtet. Der strukturelle Aufbau eine Modells ist nicht von Interesse. Bei dieser Methode werden Modellinputs generiert und das Modell ausgeführt. Während der Modellverarbeitung werden die Modelloutputs beobachtet und gegen *Erwartungswerte (EW)* abgeglichen. Stimmen die Modelloutputs mit den EWs überein so ist die Validierung als erfolgreich einzustufen. Im anderen Fall ist die Validierung als fehlgeschlagen zu bewerten und es muss eine Fehlersuche durchgeführt werden.
Bei der Validierung der CNC-satzbasierten BMs sollen Simulationsergebnisse, wie Gesamtleistungsaufnahme, KSS-Verbrauch, Verschleissäquivalent, Zeitspannungsvolumen und die Ausführungsgeschwindigkeit der Basismodelle betrachtet sowie gegenübergestellt werden. Bei den Messdatenbasierten BMs werden nur die gemessenen Leistungswerte einer Werkzeugmaschine reproduziert. D.h., die Ausgabe dieser BMs muss dem Eingang gleichen.
Es sei darauf hingewiesen, dass die statistischen Größen (s. Kapitel 3 und Kapitel 4) der restrukturierten BMs nicht an Hand der BMs von Larek validiert werden kann, da die BMs von Larek keine Berechnung solcher Größen ermöglichen.

5.1. Validierung der CNC-satzbasierte Basismodelle

Jedes CNC-satzbasierte BM wird in Kombination mit einer Quelle und Senke zu einem ausführbaren MATLAB/SimEvents Modell komponiert. Auch aus den BMs von Larek werden ausführbaren MATLAB/SimEvents Modellen erstellt. Beide Modelle werden mit den selben Inputs ausgeführt und die Modellausgänge beobachtet. Dabei wird das MATLAB/SimEvents Modell nach Larek als „fehlerfrei“ angenommen und somit werden

die Ergebnisse von diesem MATLAB/SimEvents Modell als Erwartungswerte (EW) angenommen. Abbildung 5.1 verdeutlicht diesen Sachverhalt.

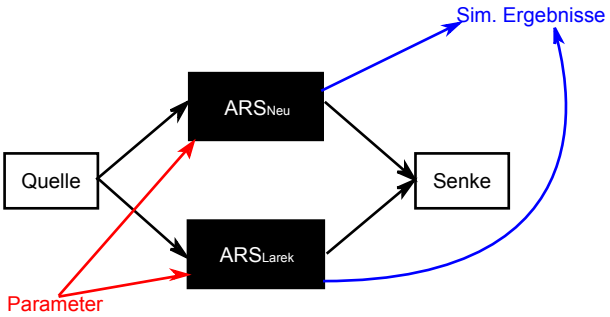


Abbildung 5.1.: Schematische Darstellung zum Testen der BMs.

Nach der Simulation werden die Simulationsergebnisse evaluiert um Aussagen über die Validität der restrukturierten BMs zu treffen.

Nach der Ausführung und Evaluierung der einzelnen MATLAB/SimEvents Modelle mit unterschiedlichen Parametern, wurde festgestellt, dass die Abweichungen der einzelnen Modelloutputs der einzelnen BMs ähnlich sind. Aus diesem Grund werden die Abweichungen nur an Hand des BMs Außenrundschleifen diskutiert.
Für die Darstellung der Abweichungen der Simulationsergebnisse werden folgende Parameter gewählt:

Schnittgeschwindigkeit 50 $[\frac{m}{s}]$
Radiale Vorschubgeschwindigkeit [1.6 0.48 0.16] $[\frac{mm}{min}]$
NC-Code als ASCII Matrix
1. Grundlast der Schleifmaschine 2870 [W]
2. Grundlast der Schleifmaschine 3263 [W]
Förderstrom der KSS-Pumpe beim Abrichten 1.5 $[\frac{l}{min}]$
Förderstrom der KSS-Pumpe beim Schleifen 10 $[\frac{l}{min}]$
Summe der Nebenzeiten vor der Bearbeitung 10 [s]
Summe der Nebenzeiten nach der Bearbeitung 60 [s]

Die Leistungsprofile sind in Abbildung 5.2 zu sehen.

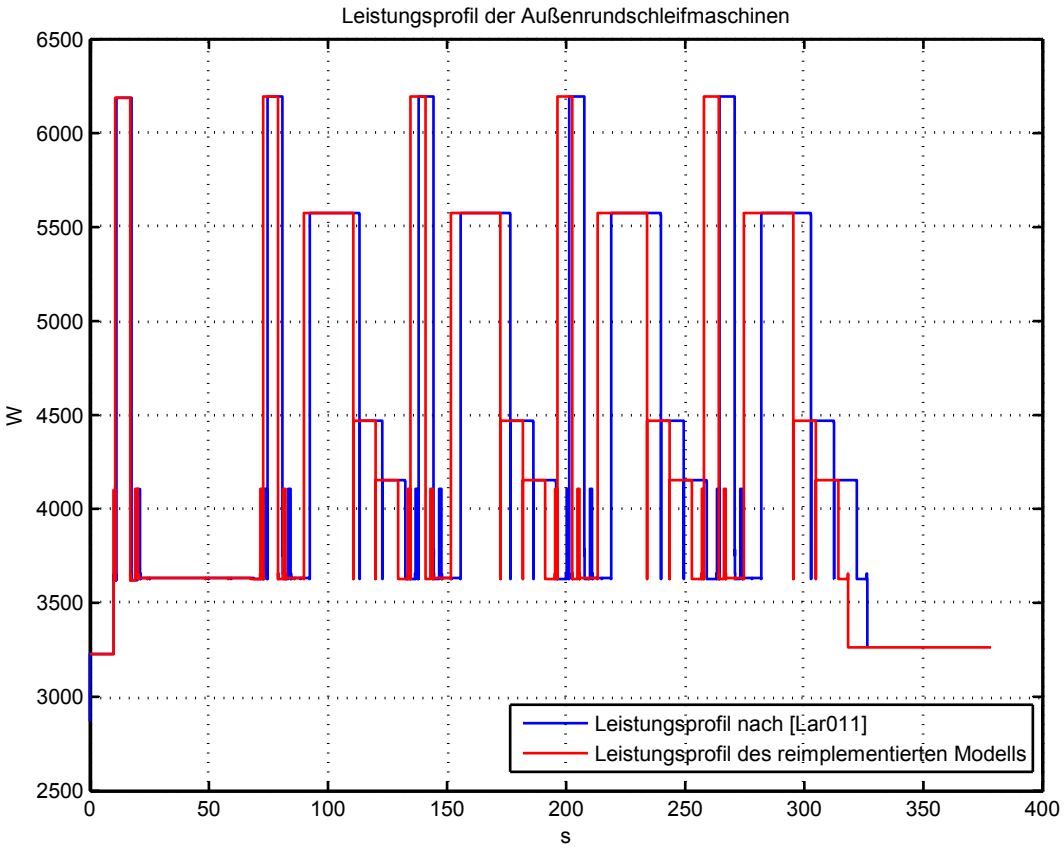


Abbildung 5.2.: Leistungsprofile der beiden Modelle.

Die beiden Lastprofile sind von einander versetzt und besitzen unterschiedliche Bearbeitungsdauern.
Der Versatz der beiden Verläufe kommt durch einen zusätzlichen *Single Server* Block, der sich in der Ereignissteuerung der BMs nach [Lar12] befindet. Dieser Server verzögert die Entität um eine Zeitschrittweite von 0.1s. Diese Verzögerung ist notwendig, da die Komponentenabbilder der BMs bei der Aktivierung weitere Entitäten, die dieselbe Bedienzeit aufweisen wie die Ereignissteuerung, beinhalten.
Bei der Abarbeitung des Modells müssen erst die Entitäten aus den Komponentenabbilder terminiert werden, bevor die Ereignissteuerung die Zustandsberechnung erneut aktivieren kann und diese wieder die Komponentenabbilder, durch das Erzeugen weiterer Entitäten, aktiviert. Wird die Entität in der Ereignissteuerung nicht um eine kleine Zeitschrittweite verzögert, kann es passieren, dass die Zustandsberechnung aufgerufen wird und erneut die Komponentenabbilder aktiviert. Da diese noch die Entitäten aus dem alten Berechnungsschritt beinhalten kommt es zu einer Fehlermeldung und die Simulation bricht ab.
Der durch diese Verzögerung resultierende Versatz wirkt sich auf den KSS-Verbrauch,

Verschleissäquivalent und auf das Zeitspanvolumen aus.

Der letzte waagerechte rote Verlauf entspricht der *Summe der Nebenzeiten nach der Bearbeitung* und dauert wie gefordert 60 Sekunden. Er wird bei dem blauen Lastverlauf nicht dargestellt, da die Signalanalyse des BMs beim Verlassen der WE nicht mehr ausgeführt wird und somit den letzten Schritt nicht aufzeichnen kann.

Zwar sind die Abweichungen beider Verläufe nicht sehr groß, werden jedoch kumuliert und führen so massiven Abweichungen. Dies ist der Fall, wenn zum Beispiel Losgrößen wesentlich größer als 1 zu fertigen sind.

5.2. Validierung der Messdatenbasierten Basismodelle

Bei der Validierung der Messdatenbasierten BMs ist die Vorgehensweise analog der Validierung der CNC-satzbasierten BMs gewählt worden. Es wurden ebenfalls Abweichungen berechnet, die für alle Messdatenbasierten BMs gelten. Somit wird ausschließlich das BM Vakuumhärten (VH) betrachtet. Das Simulationsergebnis des VH BMs ist in der Abbildung 5.3 gezeigt.

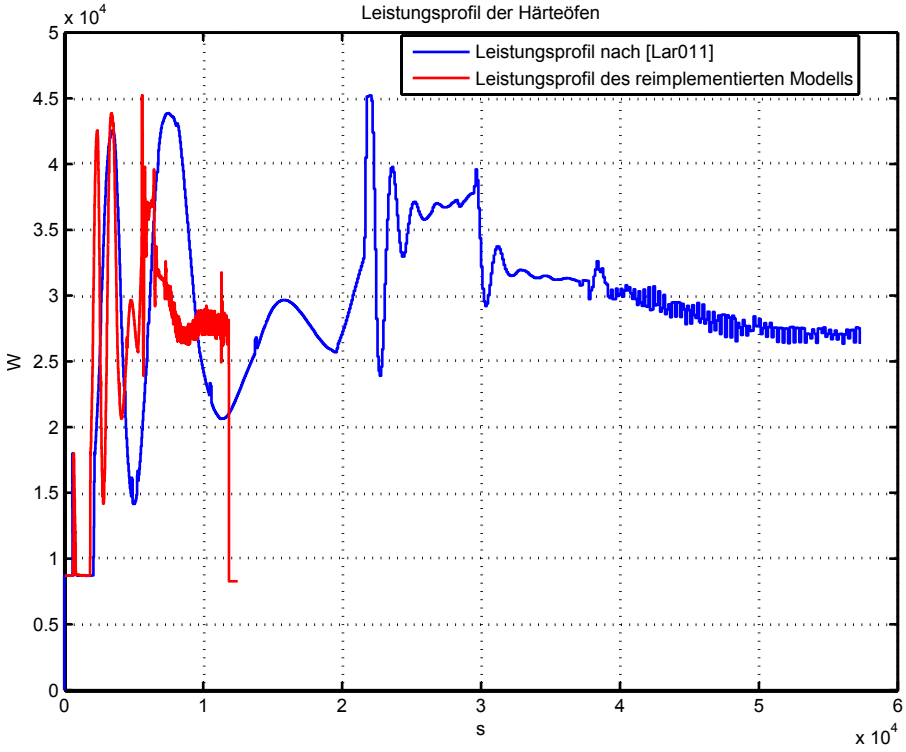


Abbildung 5.3.: Leistungsprofile der Härteöfen.

Es ist deutlich zu sehen wie sich die beiden Lastverläufe von einander unterscheiden. Dies liegt daran, dass das Profil des BMs nach [Lar12] (blau) nicht richtig reproduziert wird, da bei der Reproduzierung immer die kumulierte Zeit als Zeitschrittweite der Ereignissteuerung übergeben wird und nicht die Differenz der einzelnen Messzeiten.

Die gesamte Abweichung der Energieverbräuche liegt bei jedem MB unter 10%, wenn die *Summe der Nebenzeiten nach der Bearbeitung* vernachlässigt werden. Bei der Mitbetrachtung dieses Parameters kann die Abweichung des Energieverbrauches deutlich höher liegen.

5.3. Gesamtübersicht der Validierung

Abschließend wurden die BMs nach [Lar12] und die reimplementierten BMs bezüglich der Modellausführungszeit gegenübergestellt. Hierfür wurde die minimal mögliche Anzahl der WEs in den einzelnen BMs bearbeitet und die zugehörige Simulationszeit aufgezeichnet.

Tabelle 5.1.: Darstellung der Ausführungsgeschwindigkeiten

Basismodelle	Simulationszeiten		Abweichung in %
	nach [Lar12]	reimplementiert	
Außenlängsdrehen	11.3s	9.7s	14.2
Außenrundschleifen	12.7s	11s	13.4
Außenrundschleifhärten	13s	11.1s	15
Vakuumhärten	24.5s	2.4s	90.2
Anlassen	10.7s	2.4s	81.4
Induktionshärten	9s	2.4s	73.3

Durch die Reimplementierung der Modelle konnte die Ausführungszeit der Modelle bis zu 90% gesenkt werden. Zwar liegen die Ausführungszeiten der CNC-satzbasierten BMs bei nur rund 14.2%, dies bringt jedoch viel, wenn die BMs mehrmals hintereinander in Prozessketten ausgeführt werden.
Bei den Messdatenbasierten BMs wurde gezeigt, dass das Reproduzieren der Messdaten innerhalb des BMs nur zur Erhöhung der Modelllaufzeit führt und eigentlich nicht notwendig ist.

6. Modellierung und Simulation von Prozessketten

In diesem Kapitel erfolgt die Demonstration der restrukturierten Basismodelle an Hand von zwei Beispielen. Als erstes wird auf die Modellierung einer rein sequenziellen Prozesskette gemäß [Lar12] eingegangen. Danach erfolgt ein Beispiel zur Modellierung von erweiterten Prozessketten im Kontext einer Fertigungsstruktur.

6.1. Untersuchung einer einfachen Prozesskette

Hier wird an Hand der ersten Prozesskette, die aus den Prozessstufen Außenlängsdrehen (ALD), Vakuumhärten (VH), Anlassen (A) und Außenrundschleifen (ARS) besteht, die Funktionsweise und Modellierung gezeigt.
Im Vorfeld der Simulation müssen die entsprechenden Pfade der einzelnen Modellordner gesetzt werden. Um die verfahrensspezifischen Funktionen sowie die NC-Satze zu laden. Im Anschluss kann das MATLAB/SimEvents Modell wie gewohnt per *Drag and Drop* erstellt werden. Abbildung 6.1 zeigt das erstellte Modell der ersten Prozesskette.

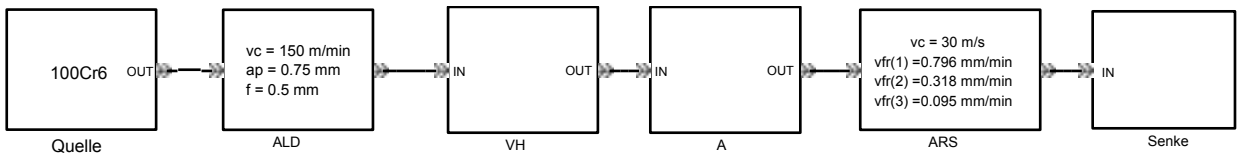


Abbildung 6.1.: Modell der ersten Prozesskette.

Weiterhin ist in der Abbildung der gewählte Werkstoff und die Parameter der CNC-satzbasierten BMs zu sehen.
Für die Simulation ist eine Losgröße von 12 Werkstücken gewählt worden.

Die Simulationsergebnisse - Leistungsprofil aller Modelle über der Zeit- sind in Abbildung 6.2 zu sehen.

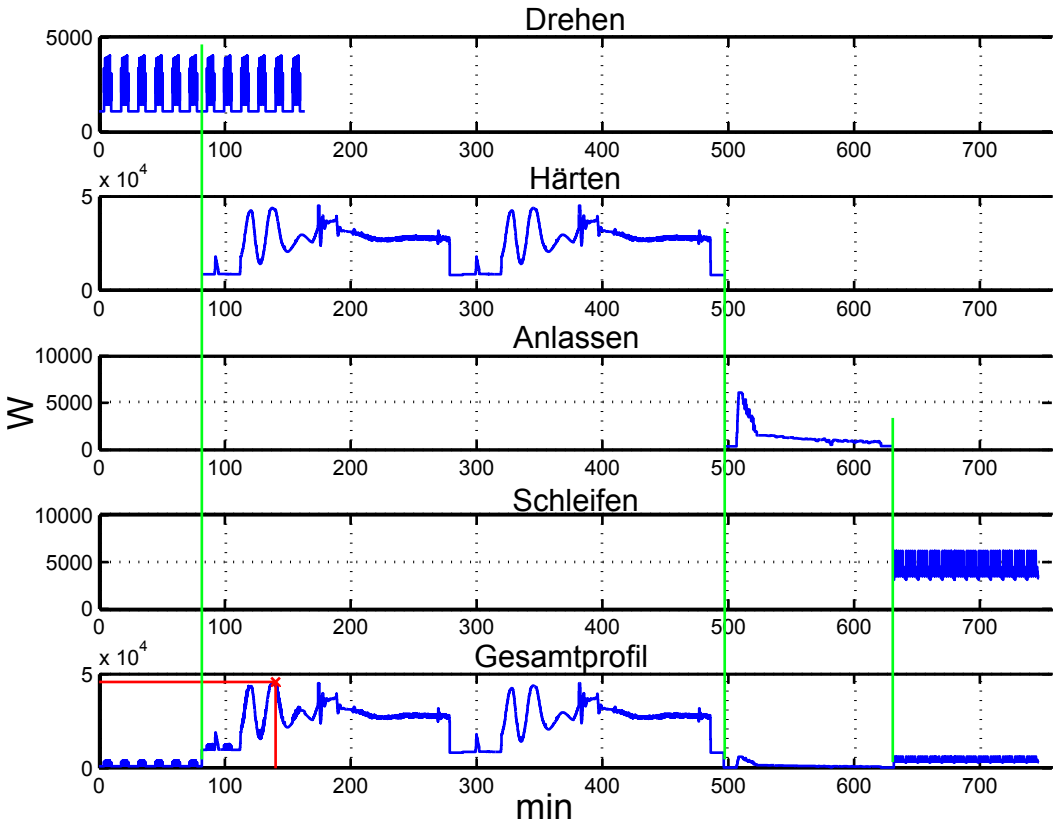


Abbildung 6.2.: Simulationsergebnisse der ersten Prozesskette.

Die Simulationszeit der gesamten Simulation beträgt circa 748 Minuten.
Es ist gut zusehen wie das BM ALD alle zwölf Werkstücke nacheinander bearbeitet. Weiterhin ist zu sehen, dass VH erst die Bearbeitung startet, wenn genau sechs Werkstücke den ALD Block verlassen haben. Dies ist auch an dem ersten grünen Strich zu erkennen. Zwischen dem Ende der Bearbeitung des sechsten Werkstückes und dem Beginn des Härtens ist kein Zeitverzug festzustellen. Weiterhin ist zu erkennen, dass der A Block, erst nachdem VH alle zwölf Werkstücke gehärtet hat, anfängt zu arbeiten. Auch hier ist keine zeitliche Lücke zwischen den beiden Modellen zu sehen.
Abschließend bearbeitet ARS die Werkstücke und gibt diese an die Senke weiter.

Weiterhin ist das Gesamtlastprofil der ersten Prozesskette aufgeführt. Die roten Linien markieren die maximale Lastspitze. Diese befindet sich an der Stelle $t_{spitze} = 140.3$ Minuten und hat einen Wert von $P_{max} = 45.9$ kW.

6.2. Untersuchung einer erweiterten Prozesskette

Mit Hilfe der reimplementierten BMs können nicht nur einfache sondern auch erweiterte Prozessketten modelliert werden. Dies kann beispielsweise eine komplexe Fertigungsstruktur mit mehreren unterschiedlichen Werkzeugmaschinen sein.
Hierfür ist ein Beispiel einer erweiterten Prozesskette erstellt worden, die folgende Prozessstufen beinhaltet: 4x ALD, 2x VH, 2x I, 2x A und 2x ARS. Weiterhin sind zwei Werkstückquellen, eine Senken, diverse FIFO-Puffer und Output Switches zu Zuteilung der Werkstücke gewählt. Die erweiterte Prozesskette ist in Abbildung 6.3 auf der nächsten Seite gezeigt.

Ziel der erweiterten Prozesskettensimulation ist die Ermittlung des Lastverlaufes sowie der Gesamtauslastung der erweiterten Prozesskette über der Zeit, der maximalen Lastspitze sowie die Durchlaufzeit des Loses von 96 Werkstücken. Hierfür sollen 48 Werkstücke aus dem Werkstoff 100Cr6 und 48 weitere aus 42CrMo4 hergestellt werden. Zur Verteilung der Werkstücke auf die entsprechenden *Output Switches* mit unterschiedlichen Strategien eingesetzt. Den ALDs werden abwechselnd jeweils ein Werkstück zugeführt (*Round Robin*). Die zwei Verteilungsstrategie lautet *First port that is not blocked* und bedeutet, dass jeweils dem freien BM die Werkstücke zugeführt werden. Ist beispielsweise VH belegt so werden die Werkstücke automatisch VH2 zugeführt. Zur Ermittlung der gesamt Durchlaufzeit des Loses (96 Werkstücke) wird ein zustandsbasierter Simulationsabbruch gewählt.

Abbildung 6.4 auf Seite 52 zeigt das Gesamtleistungsprofil der erweiterten Prozesskette und der maximalen Lastspitze. Die Gesamtauslastung der erweiterten Prozesskette ist in Abbildung 6.5 auf Seite 52 zu sehen.

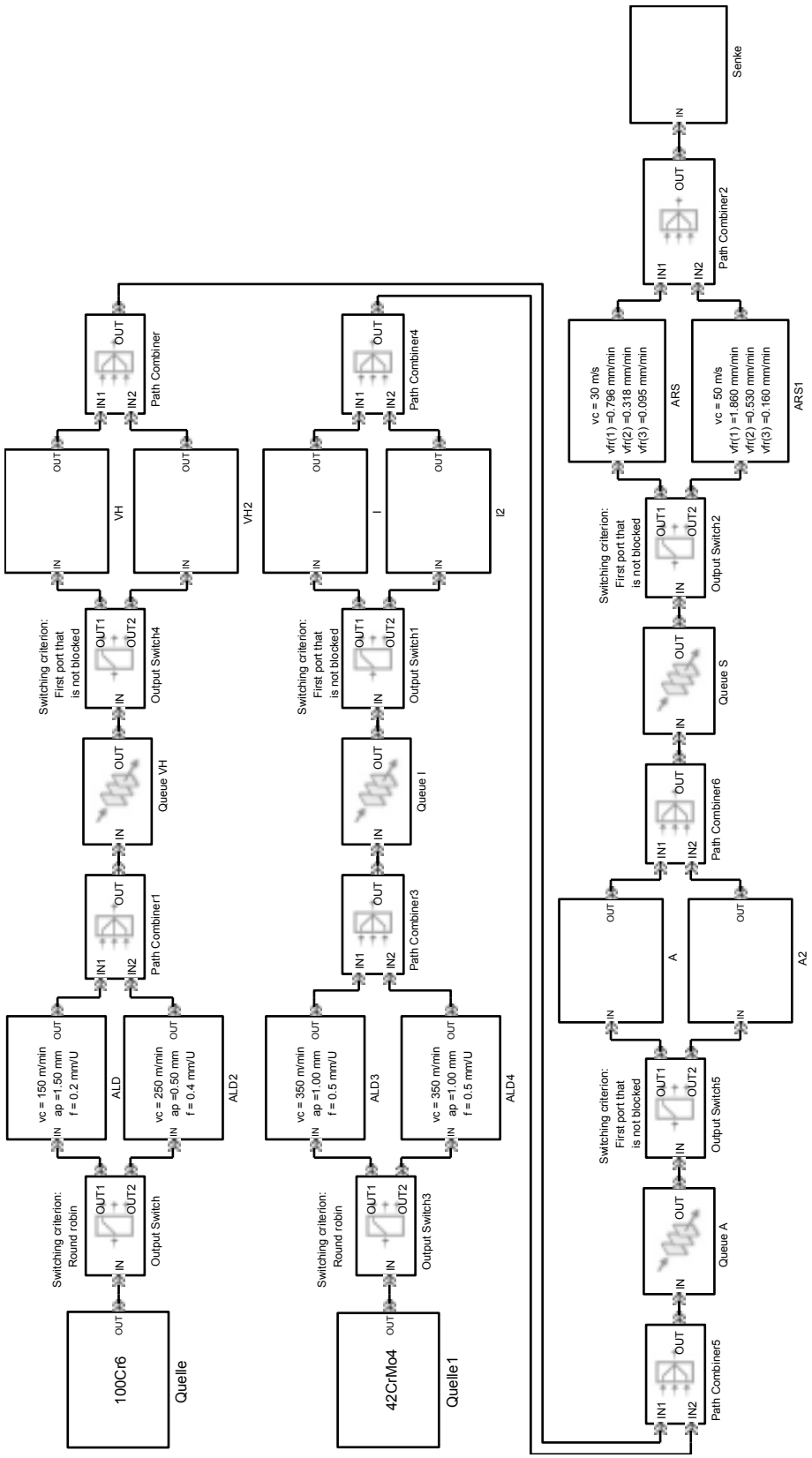


Abbildung 6.3.: Erweiterte Prozesskette.

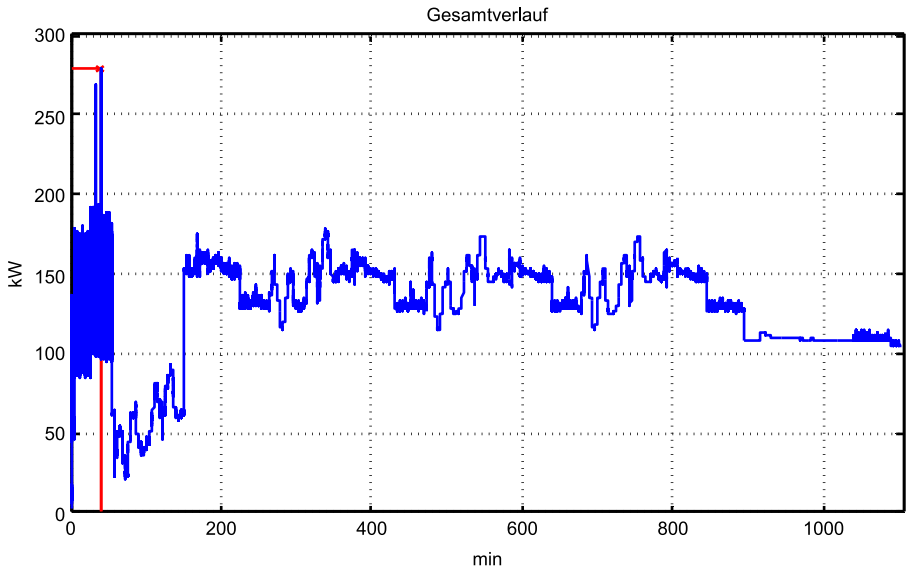


Abbildung 6.4.: Gesamtlastprofil der erweiterten Prozesskette.

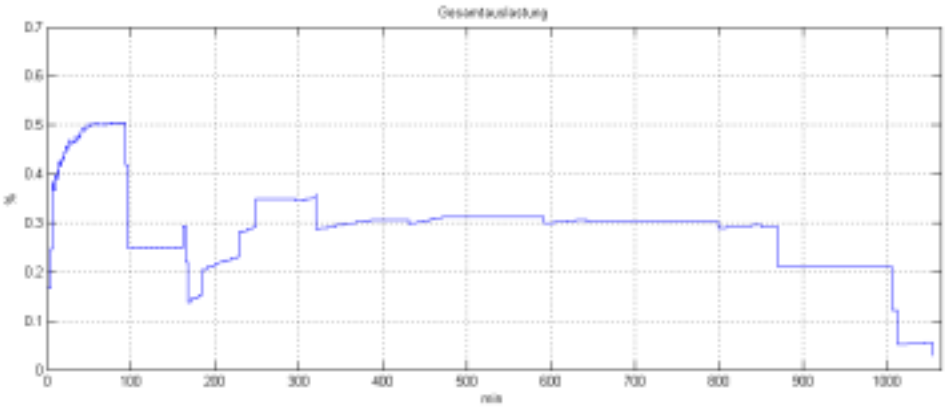


Abbildung 6.5.: Gesamtauslastung der erweiterten Prozesskette.

Die beiden Abbildungen sind Bestandteil der anfangs gestellten Fragestellungen über den Gesamtlastverlauf und der Gesamtauslastung der erweiterten Prozesskette. Weiterhin kann die maximale Lastspitze zum Zeitpunkt t = ca. 40 Minuten und einem Wert von 280kW ermittelt werden. Die Gesamtdurchlaufzeit zur Fertigung von 96 Werkstücken beläuft sich auf ca. 1102 Minuten.

Weiterhin ist zu erkennen, dass der gesamte Lastverlauf der erweiterten Prozesskette starken Schwankungen unterliegt. Darüber hinaus ist die Prozesskette nicht voll ausgelastet. Die Gesamtauslastung beträgt im Mittel ca. 33%. Mit diesen Informationen

können jetzt entsprechende Strategien zur Glättung des Lastprofils und der Erhöhung der Gesamtauslastung der Prozesskette erarbeitet werden.

7. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit wurde die in [Lar12] erstellte Modellbibliothek unter den Gesichtspunkten: (i) Modellkomplexität, (ii) Laufzeitverhalten und (iii) Mehrfachverwendung typengleicher BMs analysiert. Weiterhin wurden unter den drei Gesichtspunkten umfangreiche Simulationsexperimente mit der Modellbibliothek durchgeführt. Als Ergebnis wurden drei wesentliche Punkte identifiziert, die sich negativ auf die eingangs genannten Gesichtspunkte auswirken. Zum einen werden sehr viele globale sowie persistente Variablen mit einem unkontrollierten Zugriffen in den einzelnen BMs verwendet. Bei einer Mehrfachverwendung typengleicher BMs in einem Prozesskettenmodell greifen die BMs unkontrolliert auf dieselben globalen Variablen zu. Es kommt somit zu einer *race condition* Problematik.
Ein weiterer Punkt ist die Vermengung von Materialfluss und Steuerung in den einzelnen BMs. Hierbei werden Werkstückentitäten (WEs), die im Modell erzeugt werden, zum Steuern des Gesamtablaufes verwendet. Somit wird die Modellkomplexität erhöht, was sich negativ auf das Laufzeitverhalten auswirkt. Weiterhin wird durch ein solches Vorgehen die Möglichkeit genommen, statistische Größen wie mittlere Verweildauer der Werkstücke in einer Maschine, Maschinenauslastung etc., die durch die *built-in* MATLAB/SimEvents Blöcke berechnet werden, auszuwerten. Darüber hinaus wurde festgestellt, das einige BMs nötig kompliziert implementiert wurden, was die Anzahl der Blöcke in einem BM und die Rechenzeit erhöht. Die genannten Defizite erforderten eine grundlegende Restrukturierung der Modellbibliothek.

Im dritten Kapitel wurden konzeptionelle Lösungen bezüglich der identifizierten Defizite erarbeitet. Zunächst wurde vorgeschlagen jedes BM als eine parametrierbare Black-Box mit definierten Ein-/Ausgabeschnittstellen zu implementieren. Hierbei wurden alle notwendigen Systemparameter der jeweiligen BMs identifiziert und in einer Parametermaske zusammengefasst. Der Anwender kann ausschließlich die Parameter der Parametermaske manipulieren. Außerdem wurde zur Trennung vom Materialfluss und Steuerung eingeführt. Somit können auch die automatisch berechneten statistischen Größen der *built-in* MATLAB/SimEvents Blöcken genutzt werden. Ferner wurden die globalen und persistenten

Variablen als Attribute der Entitäten abgebildet. Demzufolge werden alle Variablen entitätenbezogen in dem jeweiligen BM mit einem kontrollierten Lese und Schreiben Zugriff gespeichert. Als Ergebnis steht eine restrukturierte und verbesserte Modellbibliothek zur Verfügung.

Die Validierung der einzelnen BMs erfolgte auf Basis der Ergebnisse in [Lar12] unter Verwendung der Black-Box-Testing Methode. Dabei traten zum Teil erhebliche Abweichungen bei den Ergebnissen zwischen den neu strukturierten BMs und den BMs von Larek auf. Die Ursachen dafür wurde auf eine ungenaue Modellierung in [Lar12] zurückgeführt. Ein weiteres Ergebnis der Validierung zeigte, dass die durchgeführten Restrukturierungsmaßnahmen positiv auf das Laufzeitverhalten der BMs auswirken. Es konnte eine Simulationslaufzeiteinsparrung von bis 90% erzielt werden.

Im sechsten Kapitel wurde die Verwendung der neuen Modellbibliothek im Kontext der erweiterten Prozesskettensimulation aufgezeigt. Dabei wurde auch gezeigt, dass die Mehrfachverwendung typengleiche BMs in Prozessketten ohne weiteres möglich ist.

Ausblickend ist zu erwähnen, dass die BMs um Steuerungsschnittstellen zum Ein- und Abschalten der Grundlast der jeweiligen BMs erweitert werden sollte. Dieser Sachverhalt wurde zwar konzeptionell betrachtet, jedoch nicht implementiert. Des Weiteren sollte die Verwendbarkeit des Modellierungsansatzes zur Modellierung weiterer Fertigungsverfahren in MATLAB/SimEvents untersucht werden.

Literaturverzeichnis

[Jun08] Junge, Mark: *Simulationsgestützte Entwicklung und Optimierung einer energieeffizienten Produktionssteuerung.*, Universität Kassel, Diss., 2008

[KY99] Kredel, Heinz ; Yoshida, Akitoshida: *Thread- und Netzwerk-Programmierung mit Java : Praktikum für die Parallele Programmierung.* Heidelberg: dpunkt-Verl., 1999

[Lar12] Larek, Roland: *Ressourceneffiziente Auslegung von fertigungstechnischen Prozessketten durch Simulation und numerische Optimierung.*, Universität Bremen, Diss., 2012

[LBM+11] Larek, Roland ; Brinksmeier, Ekkard ; Mayer, Daniel ; Pawletta, Thorsten ; Hagendorf, Olaf: A discrete-event simulation approach to predict energy consumption in machining processes. In: *Production Engineering Research and Development.*, Springer, 2011, S. 575–579

[Mat11a] MathWorks: *SimEvents User's Guide*, 2011

[Mat11b] MathWorks: *Simulink Getting Started Guide*, 2011

[MBS06] Mawick, K. ; Bender, W. ; Stranzinger, B.: *Qualitätsverbesserung und Energieeinsparung bei der Wärmebehandlung in Durchlauföfen durch betriebsorientierte Prozeßsimulation: Schlußbericht [für den Zeitraum: 01.07.2004 bis 30.06.2006].* BFI, 2006 (Bericht / Betriebsforschungsinstitut, VDEh - Institut für Angewandte Forschung GmbH)

[NBL07] Nauheimer, Harald ; Bertsche, Bernd ; Lechner, Gisber: *Fahrzeuggetriebe: Grundlagen, Auswahl, Auslegung und Konstruktion. 2. Auflage.* Springer, 2007

[NWK+08] Neugebauer, Reimund ; Westerkämper, Engelbert ; Klocke, Fritz ; Kuhn, Axel ; Schenk, Michael ; Michaelis, Alexander ; Spath, Dieter ; Weidner, Eckhard: *Untersuchung zur Energieffizienz in der Produktion.* Frauenhofer Gesellschaft, 2008 (Abschlussbericht)

[PS11] Pechmann, Agnes ; Schöler, Ilka: Optimizing Energy Costs by Intelligent Production Scheduling. In: *Glocalized Solutions for Sustainability in Manufacturing*, 2011, S. 293–298

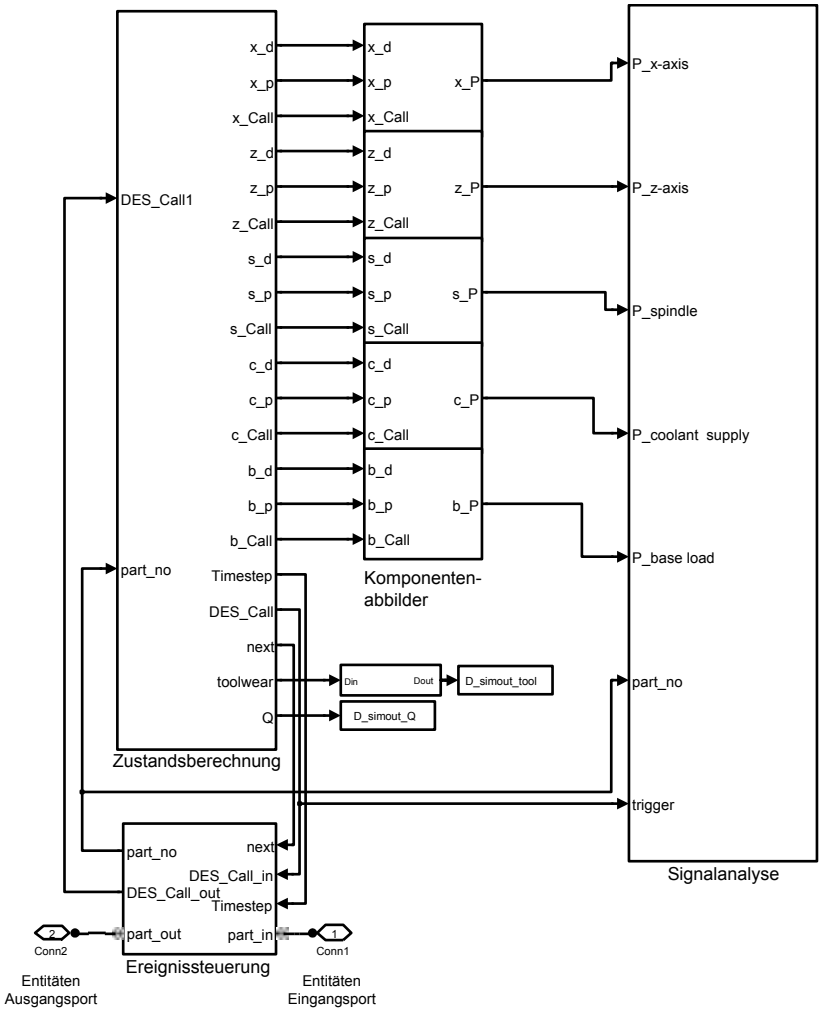
[Wei10] Weissleder, Stephan: *Test Models and Coverage Criteria for Automatic Model-Based Test Generation with UML State Machine*, Humboldt-Universität Berlin, Diss., 2010

[ZPK00] Zeigler, Bernard ; Praehoffer, Herbert ; Kim, Tag G.: *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, 2000

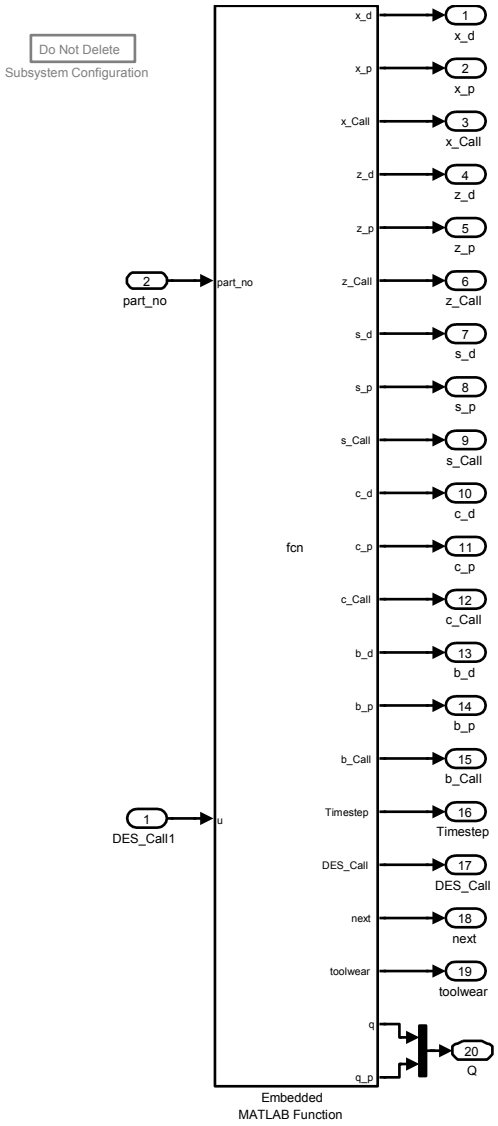
A. Umsetzung der Basismodelle ALD und VH von Larek

Auf den Folgenden Seiten wird die Umsetzung eines Außenlängsdrehen (ALD) Basismodells und des Basismodells Vakuumhärten in MATLAB/SimEvents nach [Lar12] gezeigt.

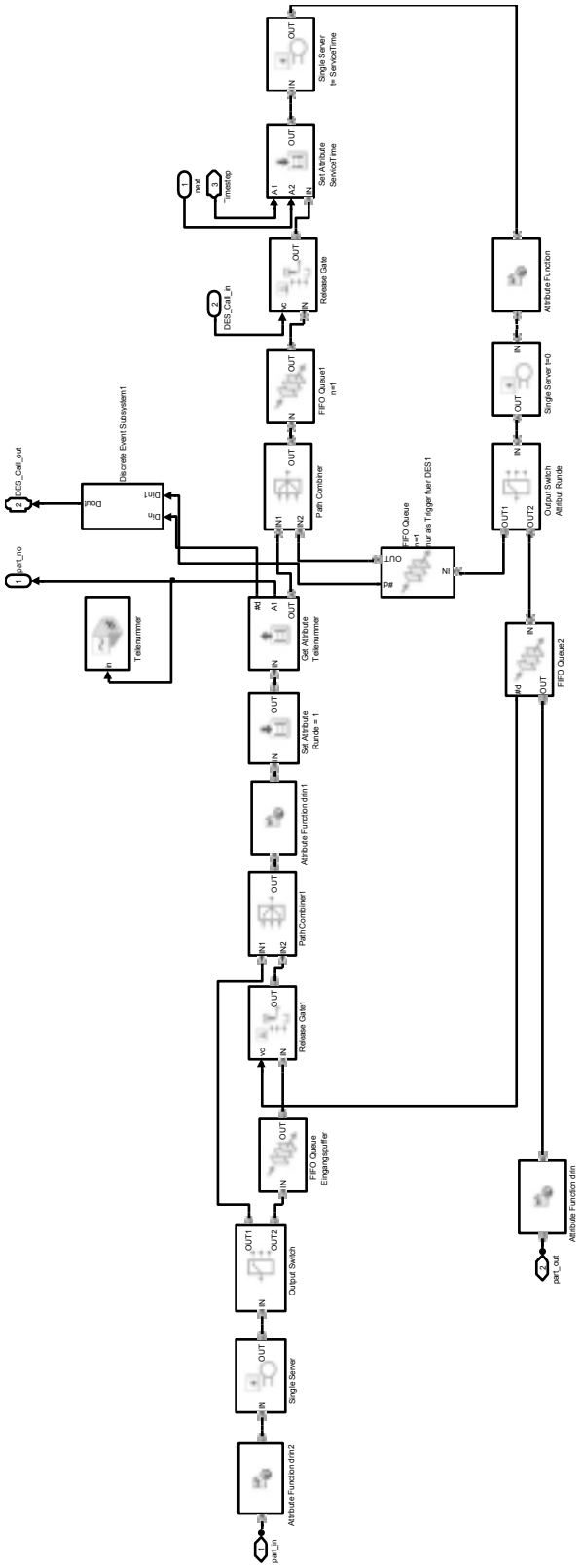
A.1. Außenlängsdrehen



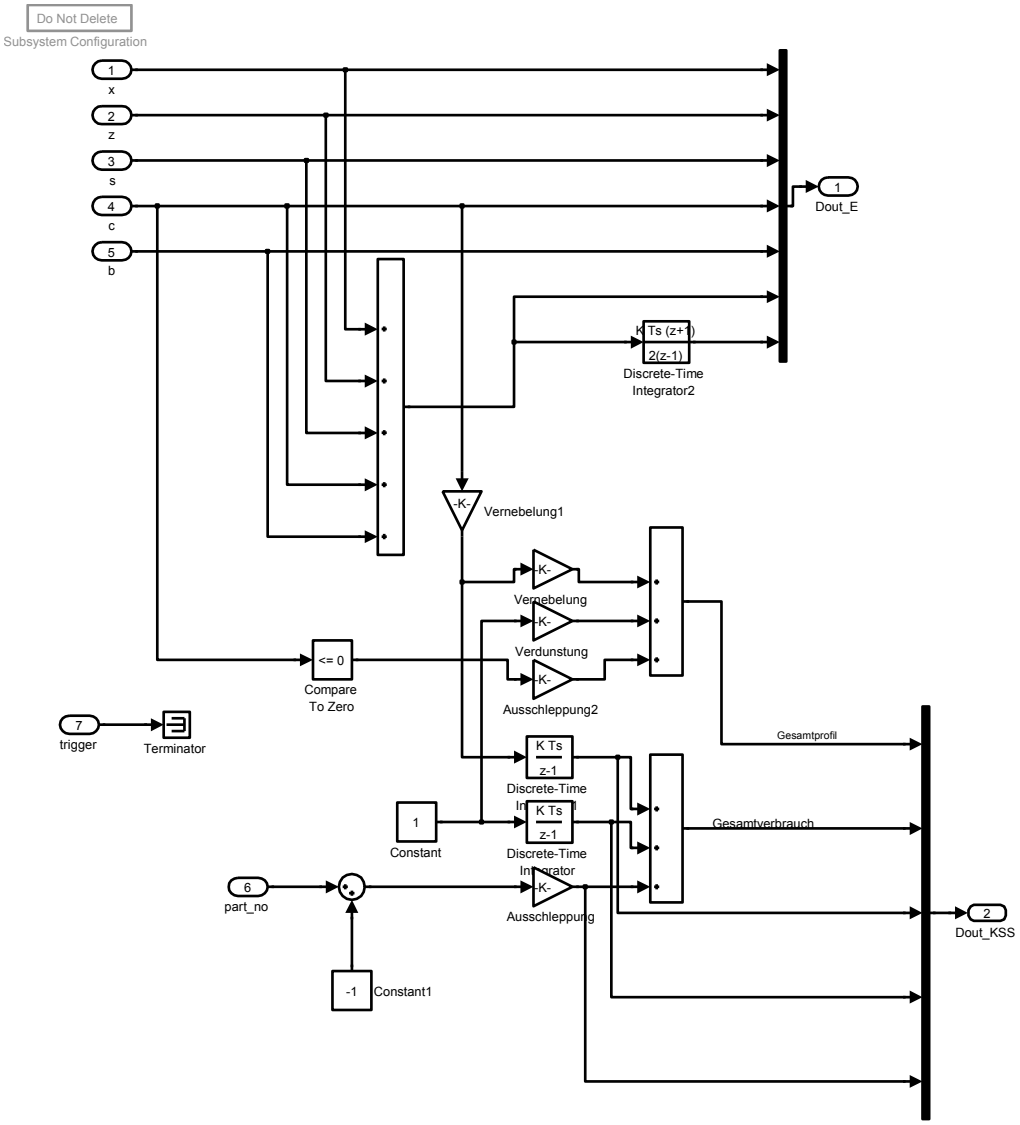
A.1.1. Subsystem Zustandsberechnung



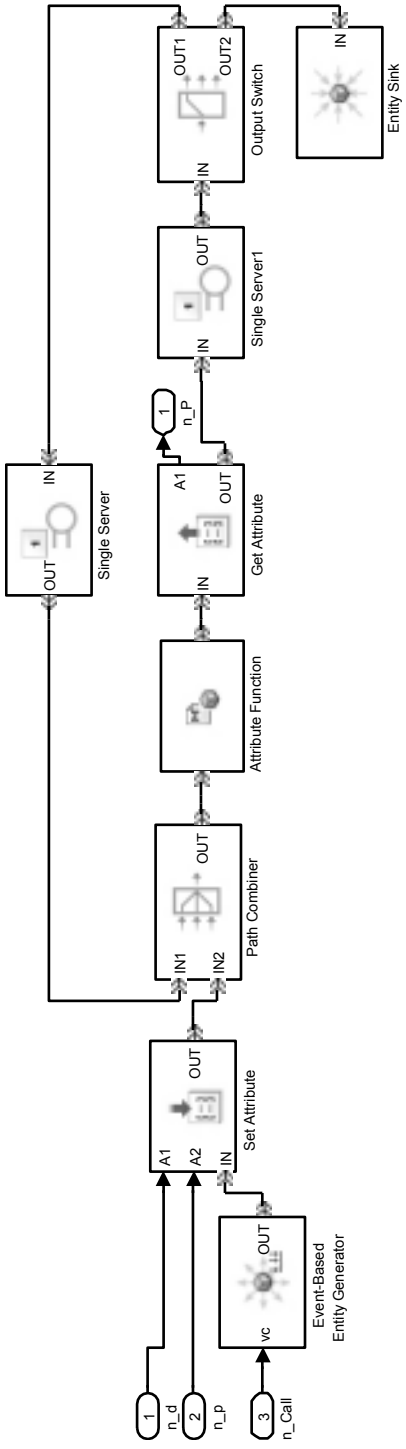
A.1.2. Subsystem Ereignissteuerung



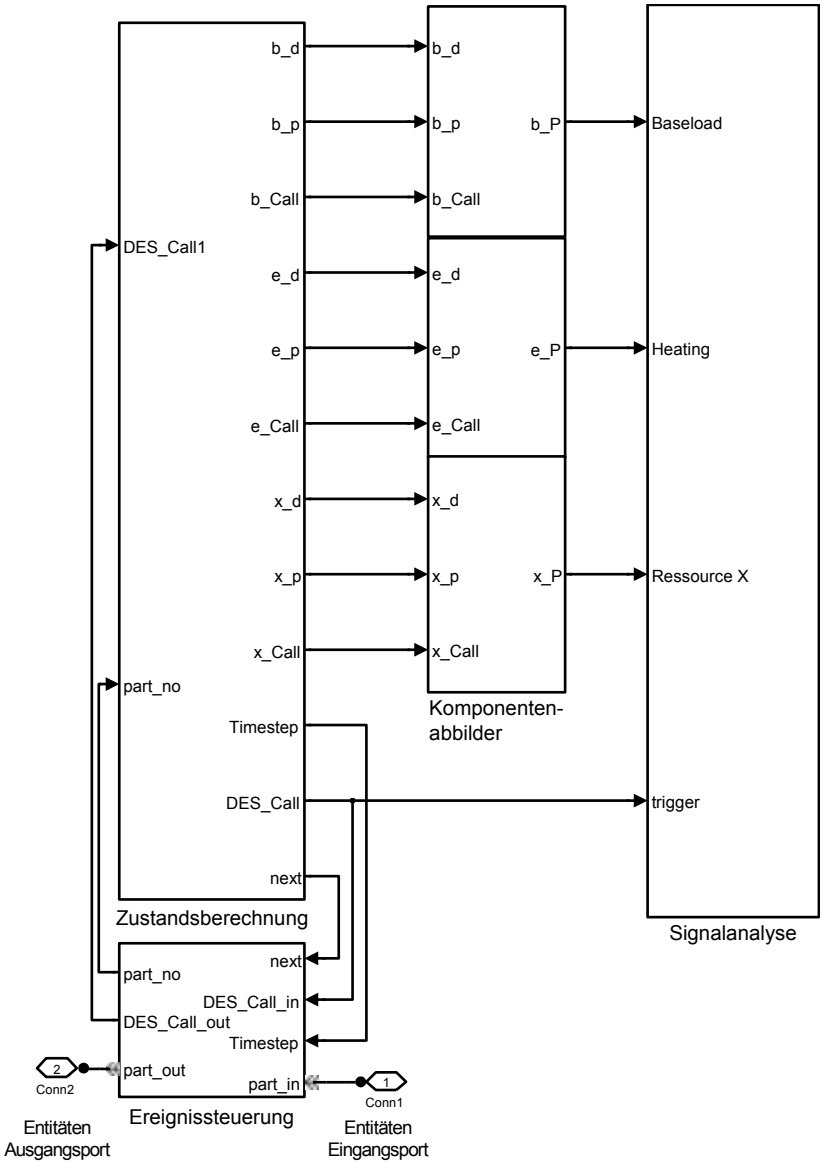
A.1.3. Subsystem Signalanalyse



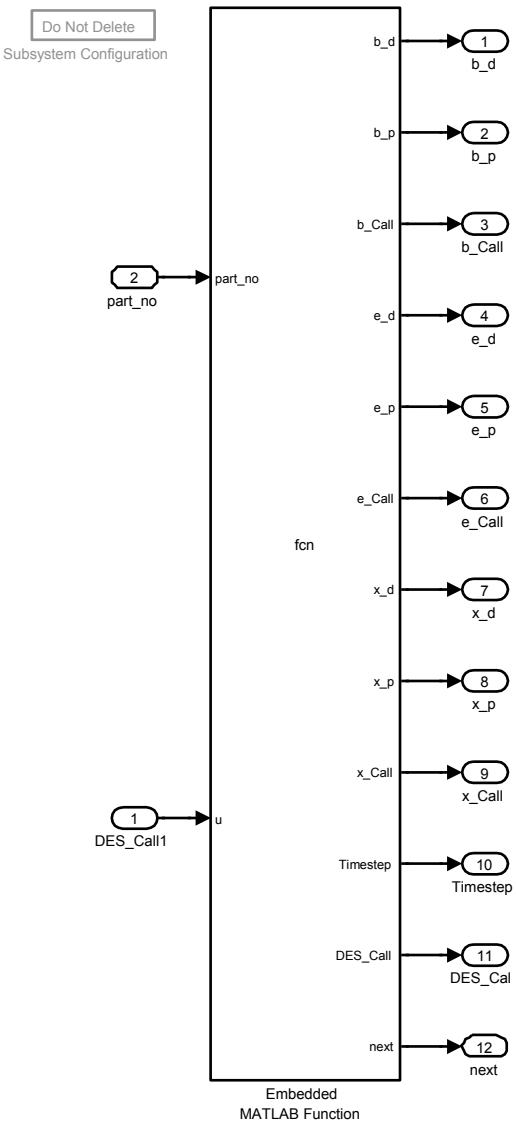
A.1.4. Subsystem Komponentenabbilder



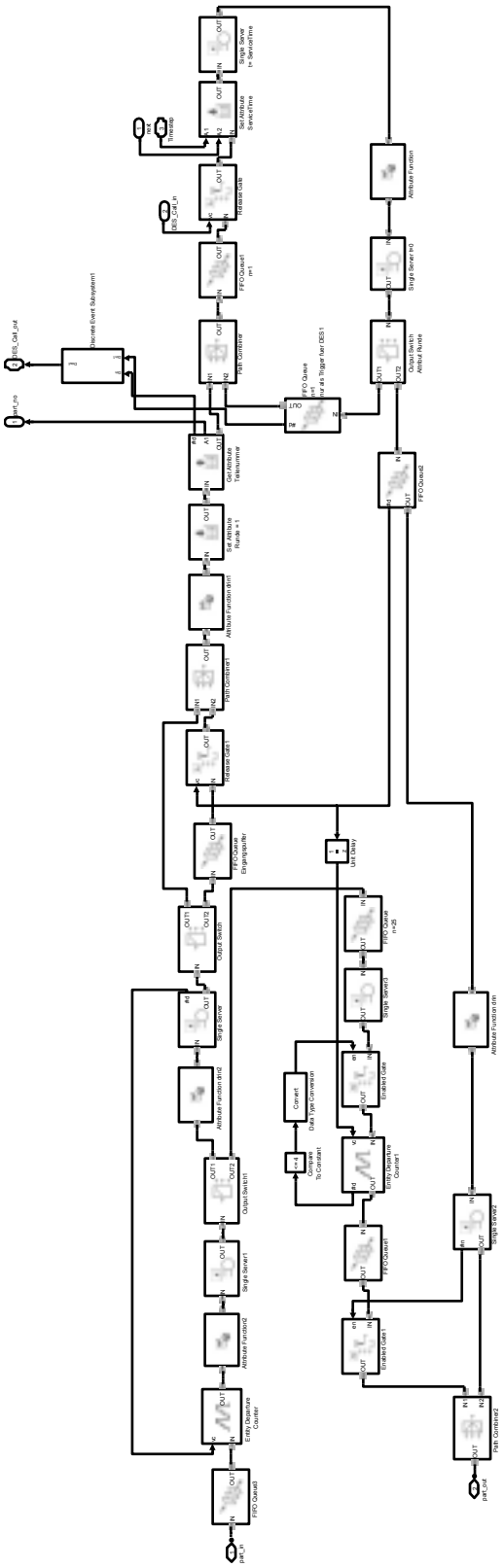
A.2. Vakuumhärten



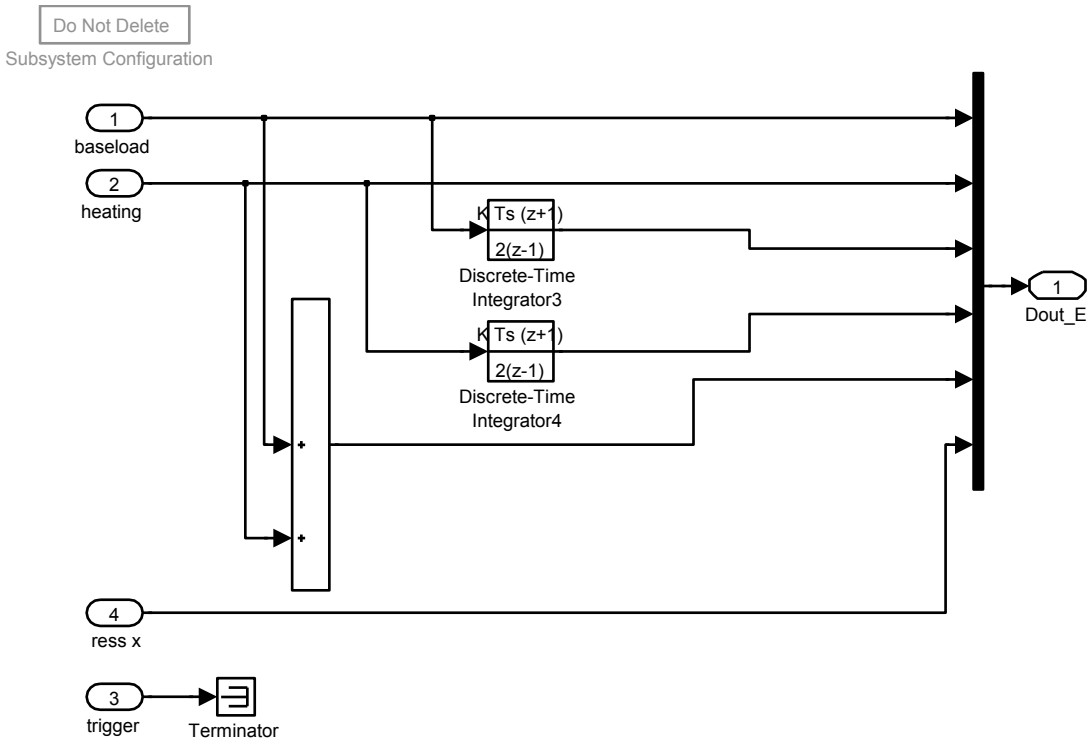
A.2.1. Subsystem Zustandsberechnung



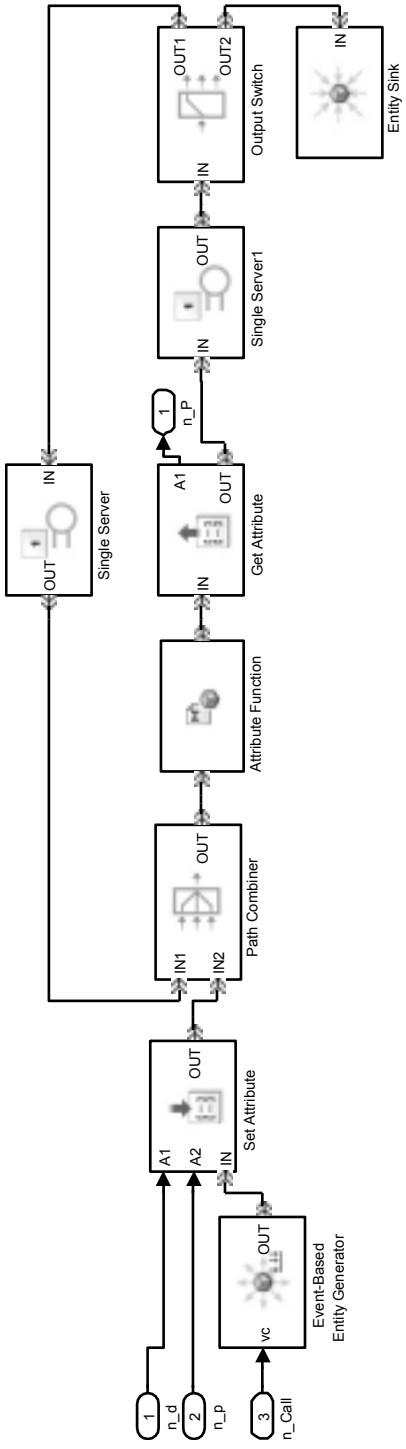
A.2.2. Subsystem Ereignissteuerung



A.2.3. Subsystem Signalanalyse

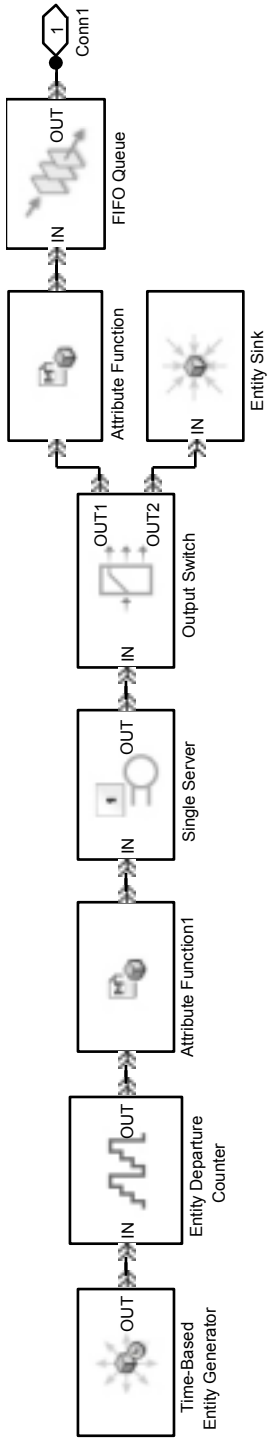


A.2.4. Subsystem Komponentenabbild



A.3. Hilfsmodelle

A.3.1. Quelle



A.3.2. Senke

Gewöhnlicher Senke Block der MATLAB/SimEvents Bibliothek.

B. Beispiel eines CNC-Programms

Im diesem Kapitel wird kurz auf die CNC-Programme und deren Verarbeitung eingegangen.

B.1. Verarbeitung der CNC-Sätze

Ein CNC-Satz besteht aus mehreren Wörtern, welche aus Adressbuchstaben und einem darauffolgenden Wert bestehen. Die entsprechende Unterteilung wird in Abbildung B.1 gezeigt.

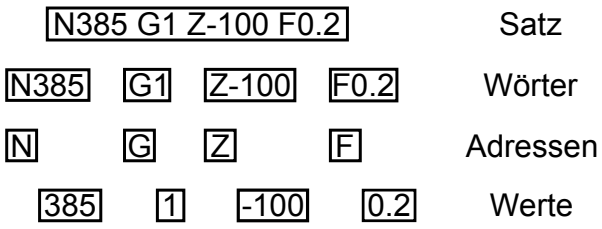


Abbildung B.1.: Detaillierte Darstellung eines CNC-Satzes

Der Abbildung kann entnommen werden, dass sich der Satz in der 385. Zeile befindet und das Werkzeug in linearer Interpolation mit einem Vorschub von $0.2\frac{mm}{U}$ im Eingriff auf die Z-Koordinate -100 bewegen soll. Im Anhang B befindet sich eine Übersicht zur CNC-Programmierung, eine Erläuterung aller Adressen sowie ein nach [Lar12] zur Verfügung gestelltes CNC-Programm.

Anhand dieser CNC-Sätze kann die Abfolge der Lastzustände der Achsen und Spindel sowie die dazugehörigen Eingriffsgrößen berechnet werden. Innerhalb der Sätze werden für die Parameter Schnitt- und Vorschubgeschwindigkeit sowie Drehzahl Platzhaltewerte (888 oder 999) eingesetzt. Sie können zur Laufzeit eindeutig erkannt und durch die Vorgaben eines Simulationsskriptes ersetzt werden.

B.2. Beispiel CNC-Programm nach [Lar12] mit ap = 1.5

N310 MSG("AUSSEN VORDREHEN")
N315 M8
N320 G95 S1079
N330 G0 Z20
N340 X65
N350 Z1
N360 X59
N370 G96 S999 % 999 entspricht einem Platzhalter für die Schnittgeschwindigkeit
N385 G1 Z-100 F0.2 % Egalisierungsschnitt
N390 X60
N400 G0 X60.4 Z-99.8
N410 Z20 M00
N470 X56
N480 G1 Z-100 F888 % 888 entspricht einem Platzhalter für den Vorschub
N490 X58
N500 G0 Z-99.8
N510 Z1
N570 X53
N580 G1 Z-100 F888 % 888 entspricht einem Platzhalter für den Vorschub
N590 X55
N600 G0 Z-99.8
N610 Z1
N670 X50
N680 G1 Z-100 F888 % 888 entspricht einem Platzhalter für den Vorschub
N690 X52
N700 G0 Z-99.8
N710 Z1
N770 X47
N780 G1 Z-100 F888 % 888 entspricht einem Platzhalter für den Vorschub
N790 X49
N800 G0 Z-99.8
N810 Z1
N870 X44

N880 G1 Z-100 F888 % 888 entspricht einem Platzhalter für den Vorschub
N890 X46
N900 G0 Z-99.8
N910 Z1
N970 X41
N980 G1 Z-100 F888 % 888 entspricht einem Platzhalter für den Vorschub
N1290 X43
N1300 G0 Z-99.8
N1310 Z1
N1320 X40
N1340 G1 Z-100 F0.1 % Schlichtschnitt
N1350 X41
N1355 M9
N1360 G0 Z-99.8
N1370 X65
N1380 Z1
N1390 Z20
N1410 G0 D0 G53 X310

C. Spezifische MATLAB Funktionen und das numerische Werkstückmodell

In diesem Kapitel wird auf die in [Lar12] entwickelten verfahrensspezifischen Funktionen und das numerische Werkstückmodell von Larek eingegangen.

C.1. Das numerische Werkstückmodell

Das numerische Werkstückmodell ist eine 4x5 Matrix und beinhaltet die Geometrie des Rohlings, den flächenbezogenen Energieverbrauch sowie die zugehörige Fläche des Ausgangsteils. Dieses Modell ist vom Benutzer im Vorfeld einer Simulation manuell nach dem realen Ausgangswerkstück zu definieren. Dabei müssen die ersten beiden Zeilen der Z- und X- Koordinate den Werkstückeckpunkten im Werkstückkoordinatensystem entsprechen. Die Anzahl der Spalten berechnen sich aus der Summe der Werkstückecken plus eins, da für die Konturbeschreibung ein geschlossener Kurvenzug benötigt wird. Der erste Punkt muss demzufolge gleich dem letzten Punkt der Werkstückkontur sein. Das Werkstückmodell wird zur Laufzeit durch jeden Eingriff des Werkzeugs aktualisiert. Auf diese Weise werden nicht mehr vorhandene Eckpunkt durch die neu entstandenen Eckpunkte ersetzt.

Weiterhin werden in der dritten Zeile und vierten Zeile der flächenbezogene Energieverbrauch in $\frac{Ws}{mm^2}$ und die dazugehörige Bezugsfläche in mm^2 gespeichert. Dadurch ist es möglich den Energieverbrauch aller vorgehenden Prozesse mit zu berücksichtigen. Ist zum Beispiel bekannt wie viel flächenbezogener Energie bei der Herstellung des Ausgangsteil verbraucht wurde, kann der Wert ohne Probleme an der entsprechenden Stelle im Werkstückmodell eingetragen werden. Abbildung C.1 zeigt das vollständige Werkstückmodell vor und nach dem Längsdrehen einer Welle von 60 auf 40 Millimeter.

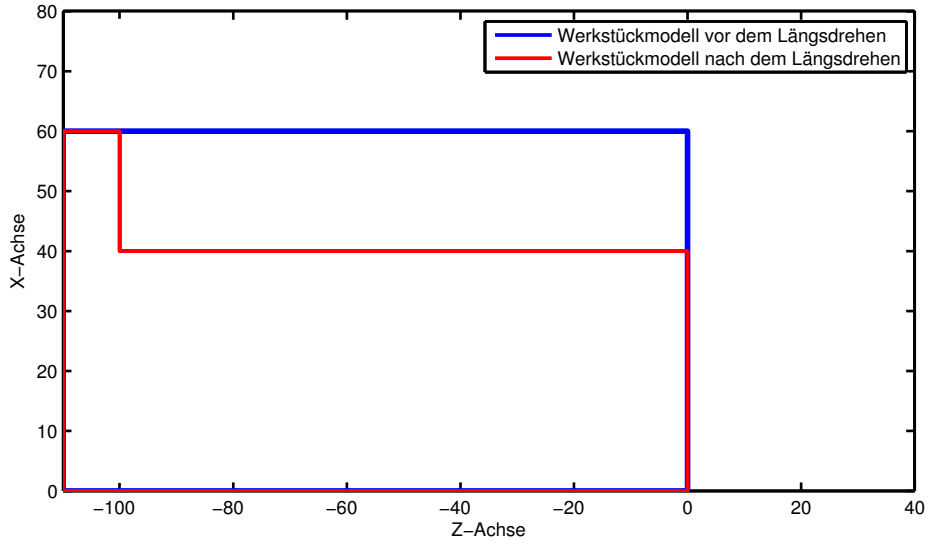


Abbildung C.1.: Graphische Darstellung des Werkstückmodells in der X-Z-Ebene vor und nach der Bearbeitung

Werkstückmodell vor dem Längsdrehen:

$$\begin{matrix} Z-Koordinate \\ X-Koordinate \\ flächenbezogener\,Energieverbrauch \\ bezogene\,Mantelfläche \end{matrix} \begin{bmatrix} -110 & -110 & 0 & 0 & -110 \\ 0 & 60 & 60 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Werkstückmodell nach dem Längsdrehen:

$$\begin{matrix} Z-Koordinate \\ X-Koordinate \\ flächenbezogener\,Energieverbrauch \\ bezogene\,Mantelfläche \end{matrix} \begin{bmatrix} -110 & -110 & -100 & -100 & 0 & 0 & -100 \\ 0 & 60 & 60 & 40 & 40 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12566 & 0 & 0 & 0 \end{bmatrix}$$

Das am Ende resultierende numerische Werkstückmodell kann von den nachfolgenden CNC-satzbasierten Modellen eingelesen und erneut abgeändert werden.

C.2. MATLAB Funktionen der CNC-Satzbasierte BMs

C.2.1. nc_calc.m

Diese Funktion ist *die* Hauptfunktion, welche für jedes Modell verfahrensspezifische erstellt wurde. Hier werden CNC-Sätze Zeilenweise eingelesen, interpretiert und eine der Bewegungsfunktionen G1.m oder G0.m aufgerufen. Auf diese Weise können alle Statusänderungen der einzelnen Komponentenabbilder erkannt und an diese weitergeleitet werden. Die nc_calc.m wird immer dann aufgerufen, wenn die Ereignissteuerung die Zustandsberechnung aktiviert.

C.2.2. G1.m

CNC-Bewegungsanweisung *G1* und entspricht einer linearen Interpolationsbewegung des Werkzeugs im Eingriff mit dem geforderten Vorschub. Für die Simulation ist die Bewegung in zwei Teilbewegungen unterteilt worden. Ein Teil der Bewegung wird ohne Eingriff und der Rest der Bewegung mit Eingriff realisiert.
Die Funktion wird von der Hauptfunktion aufgerufen, falls eine G-Adresse mit dem Wert *1* im aktuellen CNC-Satz identifiziert wird.
Sie beinhaltet weiterhin die Berechnung der Spindelleistung, X/Z-Achsleistung, des Zeitspanungsvolumen und des Werkzeugverschleißes für die *G1* Anweisung. Zusätzlich findet hier die Berechnung der Zeit für die Ereignissteuerung statt.

C.2.3. G0.m

Entspricht der CNC-Bewegungsanweisung *G0* und sieht eine lineare Interpolationsbewegung des Werkzeugs im Eilgang - es erfolgt kein Eingriff in das Werkstück -. Die *G0* Funktion wird, nach dem Lokalisieren der G-Adresse mit dem Wert *0,* von *nc_calc* aufgerufen. Weiterhin findet hier die Berechnung der Zeit für die Ereignissteuerung statt.

C.2.4. part_shape.m

Verwaltet das numerische Werkstückmodell.

C.3. MATLAB Funktion der Messdatenbasierten BMs

Die nc_calc.m Funktion ist die Hauptfunktion der Messdatenbasierten BMs und ist bei jedem BM identisch. Sie liest Messdaten aus der Simulationsumgebung ein und entnimmt ihnen Informationen über den entsprechenden Ressourcenverbrauch. Wie auch bei den CNC-satzbasierten BMs wird diese Funktion immer dann aufgerufen, wenn die Ereignissteuerung die Zustandsberechnung aktiviert.

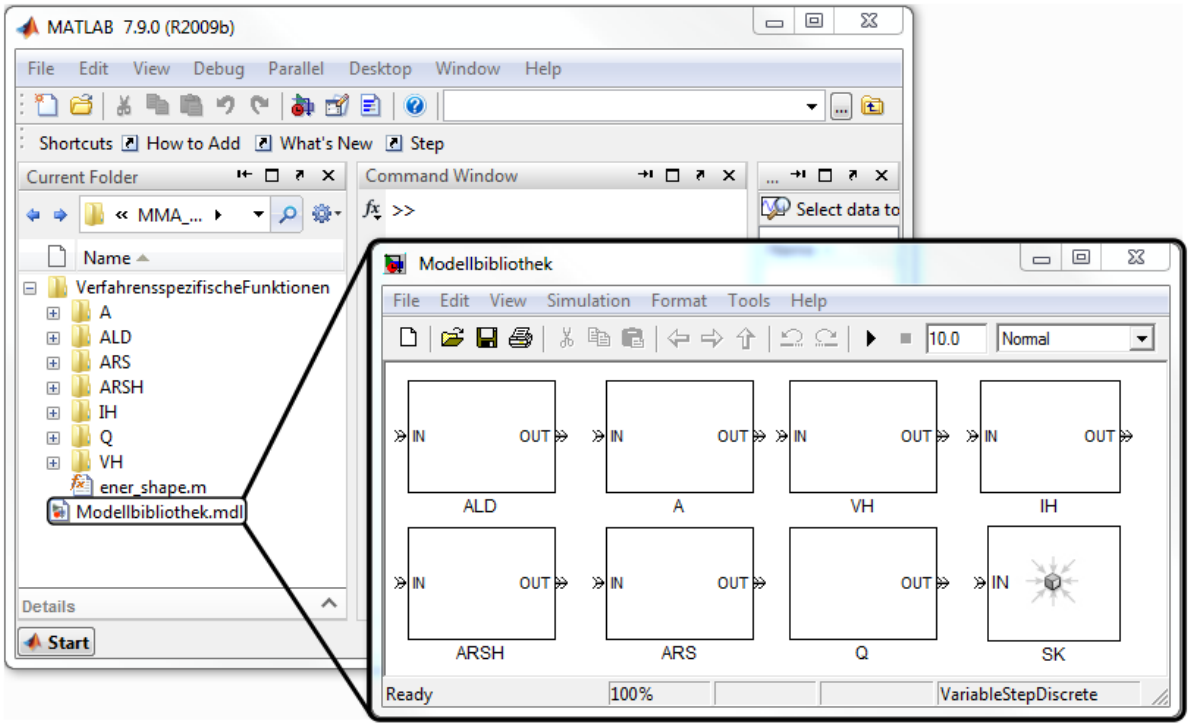
C.4. MATLAB Funktion der Hilfsmodelle

Die Werkstückquelle besitzt die Funktion parts_def.m. Innerhalb dieser Funktion ist eine Matrix mit der Geometrie des Ausgangswerkstückes. Diese Matrix entspricht dem nummerischen Werkstückmodell auf Seite XVII.

D. Ergänzungen zu den Simulationsbeispielen

Hier befinden sich einige Ergänzungen zum Kapitel 2.3.

D.1. Arbeitsverzeichnis mit verfahrensspezifischen Funktionen und der Modellbibliothek



D.2. Experiment-File zum einfachen Simulationsbeispiel

Listing D.1: EF für das ALD_Modell.

```
%% ExperimentFile_ALD_Modell

% Pfade der verfahrenspezifischen Funktionen setzen
clc, p = genpath('VerfahrensspezifischeFunktionen');
addpath(p);

% Randbedingungen
global D_vcx D_fx D_apx
global parts % Nummerische Werkstueckmodell
D_vcx = 350;  % Schnittgeschwindigkeit
D_apx = 1.5;  % Schnitttiefe
D_fx = 0.5;   % Vorschub
tsim = 50000; % Simulationszeit

%% Modellausfuehrung
sim('ALD_Modell',tsim)

%% Anzeigemethoden
% Visualisierung der Simulationsergebnisse
figure('Name','kumulierter Verbrauch insgesamt')
subplot(4,1,1);
t = D_simout_E.time(:,1);
t_sim = t;
P = D_simout_E.signals.values(:,7);
stairs(t,P);
set(gca,'XLim',[0 t_sim(end)],'xtick',[]);
title('kum. Energieverbrauch');
ylabel('Ws')
subplot(4,1,2);
t = D_simout_KSS.time(:,1);
KSS = D_simout_KSS.signals.values(:,4);
stairs(t,KSS);
```

```
33 title('kum. KSS-Verbrauch');
34 set(gca,'XLim',[0 t_sim(end)],'xtick',[])
35 ylabel('l')
36 subplot(4,1,3);
37 t = D_simout_tool.time(:,1);
38 tool = D_simout_tool.signals.values(:,1);
39 stairs(t,tool);
40 title('kum. Verschleissäquivalent');
41 set(gca,'XLim',[0 t_sim(end)],'xtick',[])
42 ylabel('m')
43 subplot(4,1,4);
44 t = D_simout_Q.time(:,1)
45 Q = cumsum(D_simout_Q.signals.values(:,1))
46 stairs(t,Q);
47 title('kum. Spanungsvolumen')
48 ylabel('mm^3'), xlabel('s')
49
50 figure('Name','Übersicht aller Verbrauchsprofile','
      NumberTitle','off')
51 subplot(4,1,1);
52 t = D_simout_E.time(:,1);
53 P = D_simout_E.signals.values(:,6);
54 stairs(t,P);
55 title('Gesamtleistungsaufnahme');
56 set(gca,'XLim',[0 t_sim(end)],'xtick',[])
57 ylabel('W')
58 subplot(4,1,2);
59 t = D_simout_KSS.time(:,1);
60 KSS = D_simout_KSS.signals.values(:,1);
61 stairs(t,KSS);
62 title('KSS-Verbrauchsprofil');
63 set(gca,'XLim',[0 t_sim(end)],'xtick',[])
64 ylabel('l/s')
65 subplot(4,1,3);
66 t = D_simout_tool.time(:,1)
```

```
67 tool = D_simout_tool.signals.values(:,2)
68 stairs(t,tool);
69 title('Verschleissäquivalent');
70 set(gca,'XLim',[0 t_sim(end)],'xtick',[])
71 ylabel('m')
72 subplot(4,1,4);
73 t = D_simout_Q.time(:,1)
74 Q = D_simout_Q.signals.values(:,2)
75 stairs(t,Q);
76 title('Zeitspanungsvolumen Q')
77 ylabel('mm^3/s'), xlabel('s')
```

D.3. Experiment-File zum komplexen Simulationsbeispiel

Listing D.2: EF für das ErsteProzesskette_Modell.

```
1  %% ExperiementFile_ErsteProzesskette
2
3  % Pfade auf die verfahrenspezifischen Funktionen setzen
4  clc, p = genpath('VerfahrensspezifischeFunktionen');
5  addpath(p);
6
7  % Randbedingungen
8  global D_vcx D_fx D_apx S_vcx S_vfrx  A_mess H_mess I_mess
9  global parts % Nummerische Werkstueckmodell
10
11 load('A_mess');           % Messdaten  A
12 load('H_mess');           % Messdaten  VH
13 D_vcx=350;                % Schnittgeschwindigkeit ALD
14 D_apx=1.5;                % Schnitttiefe ALD
15 D_fx=0.5;                 % Vorschub ALD
16 S_vcx=30;                 % Schnittgeschwindigkeit ARS
17 S_vfrx=[1.86 0.53 0.16]; % Radiale Vorschubgeschwindigkeit
      ARS
```

```
18 tsim=230000;              % Simulationszeit
19
20 %% Modellausfuehrung
21 sim('ErsteProzesskette_Modell',tsim)
22
23 %% Anzeigemethoden
24 % Visualisierung der Simulationsergebnisse
25 figure('Name','Energieverbrauchsprofile');
26 subplot(4,1,1);
27 t = D_simout_E.time(:,1)/60;
28 P = D_simout_E.signals.values(:,6);
29 stairs(t,P);
30 title('Gesamtleistungsaufnahme ALD');
31 ylabel('W')
32 subplot(4,1,2);
33 t = H_simout_E.time(:,1)/60;
34 P = H_simout_E.signals.values(:,5);
35 stairs(t,P);
36 title('Gesamtleistungsaufnahme VH');
37 ylabel('W')
38 subplot(4,1,3);
39 t = A_simout_E.time(:,1)/60;
40 P = A_simout_E.signals.values(:,5);
41 stairs(t,P);
42 title('Gesamtleistungsaufnahme A');
43 ylabel('W')
44 subplot(4,1,4);
45 t = S_simout_E.time(:,1)/60;
46 P = S_simout_E.signals.values(:,8);
47 stairs(t,P);
48 title('Gesamtleistungsaufnahme ARS');
49 ylabel('W'), xlabel('min')
```

E. Detaillierte Übersicht zu den Basismodellen

In diesem Kapitel wird detailliert auf die Parametrierung und die Ausgaben der Basismodelle nach [Lar12] eingegangen.

E.1. CNC-satzbasierte Modelle

E.1.1. Basismodell Außenlängsdrehen

Globale Eingabeparameter:

Schnittgeschwindigkeit
$D\_vcx \in R^{+} | 150 \leq D\_vcx \leq 350$ in $\left[\frac{m}{min}\right]$

Vorschub
$D\_fx \in R^{+} | 0.1 \leq D\_fx \leq 0.5$ in $\left[\frac{mm}{U}\right]$

Schnitttiefe
$D\_apx \in \{0.5, 0.75, ...1.5\}$ in $\left[\frac{mm}{U}\right]$

parts
Numerisches Werkstückmodell des Rohlings

Ausgabeparameter:

parts Werkstückgeometrie des bearbeiteten Werkstücks

D_simout_E:

D_simout_E.signals.values(:,1)

Energieverbrauch der X-Achse über der Zeit in $[W]$

D_simout_E.signals.values(:,2)

Energieverbrauch der Z-Achse über der Zeit in $[W]$

D_simout_E.signals.values(:,3)

Energieverbrauch der Spindel über der Zeit in $[W]$

D_simout_E.signals.values(:,4)

Energieverbrauch der Kühlschmierstoffpumpe über der Zeit in $[W]$

D_simout_E.signals.values(:,5)

Energieverbrauch der Grundlast über der Zeit in $[W]$

D_simout_E.signals.values(:,6)

Energieverbrauch aller Verbraucher über der Zeit in $[W]$

D_simout_E.signals.values(:,7)

Kumulierter Energieverbrauch aller Verbraucher in $[Ws]$

D_simout_Q:

cumsum(D_simout_Q.signals.values(:,1))

Kumulierter Werkstoffverbrauch $[mm^3]$

D_simout_Q.signals.values(:,2)

Werkstoffverbrauch über der Zeit in $\left[\frac{mm^3}{s}\right]$

D_simout_KSS:

D_simout_KSS.signals.values(:,1)

Kühlschmierstoffverbrauch über der Zeit in $\left[\frac{l}{s}\right]$

D_simout_KSS.signals.values(:,2)

Kumulierter Kühlschmierstoffverbrauch in $[l]$

D_simout_tool:

D_simout_tool.signals.values(:,1)

Kumulierter Werkzeugverschleiß in $[m]$

D_simout_tool.signals.values(:,2)

Werkzeugverschleiß über der Zeit in $[m]$

Funktionen:

D_nc_calc.m Interpretation der NC-Sätze.

D_G0.m Berechnung der Dauer und Höhe der Lastzustände für die G0 Anweisung.

D_G1.m Berechnung der Dauer und Höhe der Lastzustände für die G1 Anweisung.

D_drin.m Verwaltung der Blockbelegung.

parts_shape.m Verwaltung des numerischen Werkstückmodells.

Verbraucher/Komponentenabbilder:

X-Achse

Z-Achse

Spindel

Kühlschmierstoffpumpe (kurz KSS-Pumpe)

Grundlast

E.1.2. Basismodell Außenrundschleifen

Globale Eingabeparameter:

Schnittgeschwindigkeit
$S\_vcx \in R^+ | \, 30 \leq S\_vcx \leq 50$ in$\left[\frac{m}{s}\right]$

Radiale Vorschubgeschwindigkeit
$S\_vfrx \in \{(X, Y, Z)\}$
$X \in R^+ | \, 0.48 \leq X \leq 1.86$
$Y \in R^+ | \, 0.19 \leq Y \leq 0.53$
$Z \in R^+ | \, 0.03 \leq Z \leq 0.16$
$X \hat{=} Schruppen$; $Y \hat{=} Schlichten$; $Z \hat{=} Feinschlichten$

parts
Numerisches Werkstückmodell des durch das Drehen bearbeiteten Werkstücks

Ausgabeparameter:

parts Werkstückgeometrie des bearbeiteten Werkstücks

S_simout_E:

S_simout_E.signals.values(:,1)

Energieverbrauch der X-Achse über der zeit in $[W]$

S_simout_E.signals.values(:,2)

Energieverbrauch der Z-Achse über der zeit in $[W]$

S_simout_E.signals.values(:,3)

Energieverbrauch der Werkzeugspindel über der zeit in $[W]$

S_simout_E.signals.values(:,4)

Energieverbrauch der Werkstückspindel über der zeit in $[W]$

S_simout_E.signals.values(:,5)

Energieverbrauch der Abrichtspindel über der zeit in $[W]$

S_simout_E.signals.values(:,6)

Energieverbrauch der Grundlast über der zeit in $[W]$

S_simout_E.signals.values(:,7)

Kumulierter Energieverbrauch aller Verbraucher in $[Ws]$

S_simout_E.signals.values(:,8)

Energieverbrauch aller Verbraucher über der Zeit in $[W]$

S_simout_Q:

S_simout_Q.signals.values(:,1)

Kumulierter Werkstoffverbrauch $[mm^3]$

S_simout_Q.signals.values(:,2)

Werkstoffverbrauch über der Zeit in $\left[\frac{mm^3}{s}\right]$

S_simout_KSS:

S_simout_KSS.signals.values(:,1)

Kühlschmierstoffverbrauch über der Zeit in $\left[\frac{l}{s}\right]$

S_simout_KSS.signals.values(:,2)

Kumulierter Kühlschmierstoffverbrauch in $[l]$

S_simout_tool:

S_simout_tool.signals.values(:,1)

Kumulierter Werkzeugverschleiß in $[mm]$

S_simout_tool.signals.values(:,2)

Werkzeugverschleiß über der Zeit in $[mm]$

Funktionen:

S_nc_calc.m Interpretation der NC-Sätze.

S_G0.m Berechnung der Dauer und Höhe der Lastzustände für die G0 Anweisung.

S_G1.m Berechnung der Dauer und Höhe der Lastzustände für die G1 Anweisung.

S_drin.m Verwaltung der Blockbelegung.

s_parts_shape.m Verwaltung des numerischen Werkstückmodells.

Verbraucher/Komponentenabbilder:

X-Achse

Z-Achse

Werkzeugspindel

Werkstückspindel

Abrichtspindel

KSS-Pumpe

Grundlast

E.1.3. Basismodell Außenrundschleifhärten

Globale Eingabeparameter:

Schnittgeschwindigkeit
$SH\_vcx = 45 \left[\frac{m}{s}\right]$

Radiale Vorschubgeschwindigkeit
$SH\_vfrx = 2 \left[\frac{mm}{min}\right]$

parts
Numerisches Werkstückmodell des durch das Drehen bearbeiteten Werkstücks

Globale Ausgabeparameter:

parts Werkstückgeometrie des bearbeiteten Werkstücks

SH_simout_E:

SH_simout_E.signals.values(:,1)

Energieverbrauch der X-Achse über der zeit in $[W]$

SH_simout_E.signals.values(:,2)

Energieverbrauch der Z-Achse über der zeit in $[W]$

SH_simout_E.signals.values(:,3)

Energieverbrauch der Werkzeugspindel über der zeit in $[W]$

SH_simout_E.signals.values(:,4)

Energieverbrauch der Werkstückspindel über der zeit in $[W]$

SH_simout_E.signals.values(:,5)

Energieverbrauch der Abrichtspindel über der zeit in $[W]$

SH_simout_E.signals.values(:,6)

Energieverbrauch der Grundlast über der zeit in $[W]$

SH_simout_E.signals.values(:,7)

Kumulierter Energieverbrauch aller Verbraucher in $[Ws]$

SH_simout_E.signals.values(:,8)

Energieverbrauch aller Verbraucher über der Zeit in $[W]$

SH_simout_Q:

SH_simout_Q.signals.values(:,1)

Kumulierter Werkstoffverbrauch $[mm^3]$

SH_simout_Q.signals.values(:,2)

Werkstoffverbrauch über der Zeit in $\left[\frac{mm^3}{s}\right]$

SH_simout_KSS:

SH_simout_KSS.signals.values(:,1)

Kühlschmierstoffverbrauch über der Zeit in $\left[\frac{l}{s}\right]$

SH_simout_KSS.signals.values(:,2)

Kumulierter Kühlschmierstoffverbrauch in $[l]$

SH_simout_tool:

SH_simout_tool.signals.values(:,1)

Kumulierter Werkzeugverschleiß in $[mm]$

SH_simout_tool.signals.values(:,2)

Werkzeugverschleiß über der Zeit in $[mm]$

Funktionen:

SH_nc_calc.m Interpretation der NC-Sätze.

SH_G0.m Berechnung der Dauer und Höhe der Lastzustände für die G0 Anweisung.

SH_G1.m Berechnung der Dauer und Höhe der Lastzustände für die G1 Anweisung.

SH_drin.m Verwaltung der Blockbelegung.

s_parts_shape.m Verwaltung des numerischen Werkstückmodells.

Verbraucher/Komponentenabbilder:

X-Achse

Z-Achse

Werkzeugspindel

Werkstückspindel

Abrichtspindel

KSS-Pumpe

Grundlast

E.2. Messdatenbasierte Modelle

E.2.1. Basismodell Vakuumhärten

Globale Eingabeparameter:

H_mess 1102 x 2 Messdaten $P(t)$

Ausgabeparameter:

H_simout_E:

H_simout_E.signals.values(:,1)

Energieverbrauch der Grundlast über der Zeit in $[W]$

H_simout_E.signals.values(:,2)

Energieverbrauch der Prozesslast über der Zeit in $[W]$

H_simout_E.signals.values(:,3)

Energieverbrauch aller Komponentenabbilder über der Zeit in $[W]$

H_simout_E.signals.values(:,4)

Kumulierter Energieverbrauch aller Verbraucher in $[Ws]$

Funktionen:

H_nc_calc.m Verarbeiten der Messdaten.

H_drin.m Verwaltung der Blockbelegung.

Verbraucher/Komponentenabbilder:

Heizleistung

Grundlast

E.2.2. Basismodell Anlassen

Globale Eingabeparameter:

A_mess 720 x 2 Messdaten $P(t)$

Ausgabeparameter:

A_simout_E:

A_simout_E.signals.values(:,1)

Energieverbrauch der Grundlast über der Zeit in $[W]$

A_simout_E.signals.values(:,2)

Energieverbrauch der Prozesslast über der Zeit in $[W]$

A_simout_E.signals.values(:,3)

Energieverbrauch aller Komponentenabbilder über der Zeit in $[W]$

A_simout_E.signals.values(:,4)

Kumulierter Energieverbrauch aller Verbraucher in $[Ws]$

Funktionen:

A_nc_calc.m Verarbeiten der Messdaten.

A_drin.m Verwaltung der Blockbelegung.

Verbraucher/Komponentenabbilder:

Heizleistung

Grundlast

E.2.3. Basismodell Induktionshärten

Globale Eingabeparameter:

I_mess 2 x 2 Messdaten $P(t)$

Ausgabeparameter:

I_simout_E:

I_simout_E.signals.values(:,1)

Energieverbrauch der Grundlast über der Zeit in $[W]$

I_simout_E.signals.values(:,2)

Energieverbrauch der Prozesslast über der Zeit in $[W]$

I_simout_E.signals.values(:,3)

Energieverbrauch aller Komponentenabbilder über der Zeit in $[W]$

I_simout_E.signals.values(:,4)

Kumulierter Energieverbrauch aller Verbraucher in $[Ws]$

Funktionen:

I_nc_calc.m Verarbeiten der Messdaten.

I_drin.m Verwaltung der Blockbelegung.

Verbraucher/Komponentenabbilder:

Heizleistung

Grundlast

E.3. Hilfsmodelle

Nutzer können

- die Anzahl der zu generierenden Werkstück-Entitäten und
- die Periode mit der die Werkstück-Entitäten erzeugt werden in [s]

manuell beeinflussen.

F. Ein und Ausgabe der Basismodelle

In diesem Kapitel wird auf die Parametrierung und die Ausgaben der restrukturierten Basismodellen aus Kapitel 4 eingegangen.

F.1. Basismodell Außenlängsdrehen

Eingabeparameter der Parametermaske:

Tab:	Maschinenparameter	
	Schnittgeschwindigkeit $vc \in R^+ \vert 150 \leq vc \leq 350$ in $\left[\frac{m}{min}\right]$	
	Vorschub $f \in R^+ \vert 0.1 \leq f \leq 0.5$ in $\left[\frac{mm}{U}\right]$	
	NC-Code	als ASCII Matrix für $ap \in \{0.5, 0.75, ...1.5\}$ in $\left[\frac{mm}{U}\right]$
	Grundlast	der Kühlschmierstoffpumpe in [W]
	Grundlast	der Drehmaschine in [W]
	Summe	der Nebenzeiten vor der Bearbeitung in [s]
	Summe	der Nebenzeiten nach der Bearbeitung in [s]
	Wartezeit	zwischen zwei Bewegungen in [s]
Tab:	Werkzeugverlauf	
	2 x n	Zeros Matrix für den Werkzeugverlauf
	Zähler	für den Werkzeugverlauf beginnend bei 2
Tab:	Simulationsergebnisse	
	Name	der P(t) Struktur
	Name	der KSS(t) Struktur

Name der Q(t) Struktur

Name der WKZv(t) Struktur

Ausgabeparameter:

Energieverbrauch über der Zeit in [W]

Werkstoffverbrauch über der Zeit in $\left[\frac{mm^3}{s}\right]$

Kühlschmierstoffverbrauch über der Zeit in $\left[\frac{l}{s}\right]$

Werkzeugverschleiß über der Zeit in $[m]$

Funktionen:

D_nc_calc.m Interpretation der NC-Sätze.

D_G0.m Berechnung der Dauer und Höhe der Lastzustände für die G0 Anweisung.

D_G1.m Berechnung der Dauer und Höhe der Lastzustände für die G1 Anweisung.

parts_shape.m Verwaltung des numerischen Werkstückmodells.

Beispiel:

Für die Simulation sind folgende Eingabeparameter gewählt worden:
Schnittgeschwindigkeit 350
Vorschub 0.5
NC-Code für die Schnitttiefe 1.5 als ASCII Matrix
Grundlast der Kühlschmierstoffpumpe 183
Grundlast der Drehmaschine 1067
Summe der Nebenzeiten vor der Bearbeitung 60
Summe der Nebenzeiten nach der Bearbeitung 60
Wartezeit zwischen zwei Bewegungen 0.1

Abbildung F.1 zeigt eine grafische Darstellung der genanten Ausgabeparameter über der Zeit.

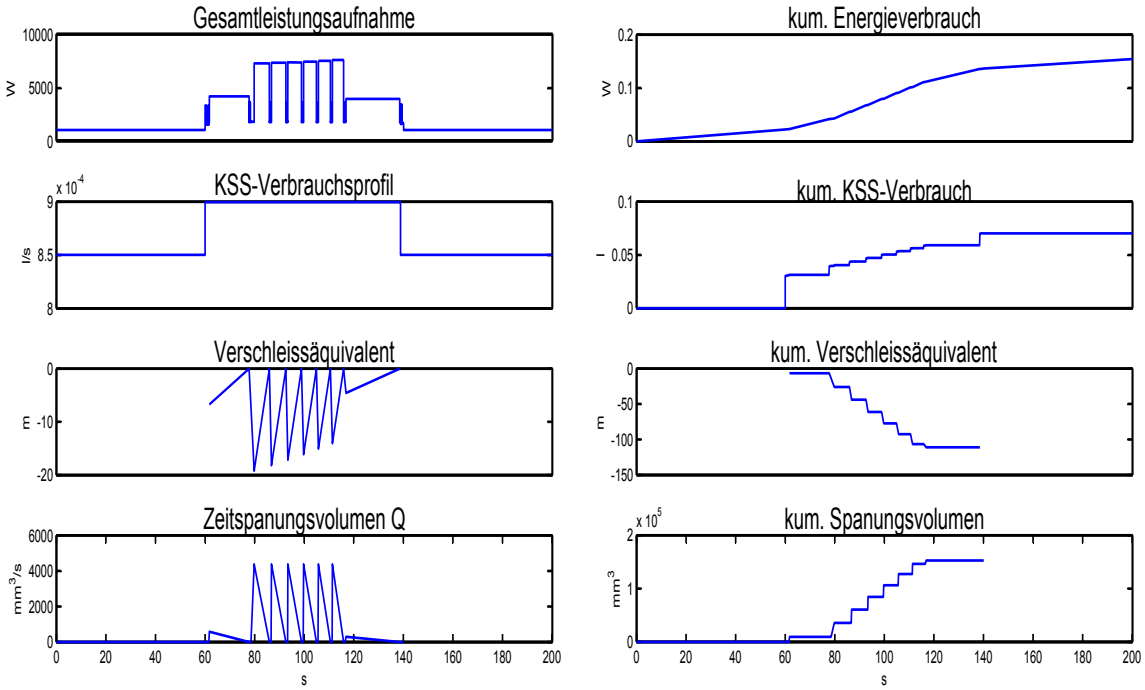


Abbildung F.1.: Darstellung der Ausgabeparameter über der Zeit

F.2. Basismodell Außenrundschleifen

Eingabeparameter der Parametermaske:

Tab: Maschinenparameter

Schnittgeschwindigkeit $vc \in R^+ | \, 30 \leq vc \leq 50$ in $\left[\frac{m}{s}\right]$

Radiale Vorschubgeschwindigkeit $vfr \in \{(X, Y, Z)\}$
$X \in R^+ | \, 0.48 \leq X \leq 1.86$
$Y \in R^+ | \, 0.19 \leq Y \leq 0.53$
$Z \in R^+ | \, 0.03 \leq Z \leq 0.16$
$X \hat{=} Schruppen$; $Y \hat{=} Schlichten$; $Z \hat{=} Feinschlichten$

NC-Code als ASCII Matrix

1. Grundlast der Schleifmaschine in [W]

2. Grundlast der Schleifmaschine in [W]

Förderstrom der KSS beim Abrichten $[\frac{l}{min}]$

Förderstrom der KSS beim Schleifen $[\frac{l}{min}]$

	Summe	der Nebenzeiten vor der Bearbeitung in [s]
	Summe	der Nebenzeiten nach der Bearbeitung in [s]
	Wartezeit	zwischen zwei Bewegungen in [s]

Tab: Werkzeugverlauf

2 x n	Zeros Matrix für den Werkzeugverlauf
Zähler	für den Werkzeugverlauf beginnend bei 2

Tab: Simulationsergebnisse

Name	der P(t) Struktur
Name	der KSS(t) Struktur
Name	der Q(t) Struktur
Name	der WKZv(t) Struktur

Ausgabeparameter:

Die Ausgabe geschieht über das Erzeugen der Variablen im Matlabworkspace. Die Variablen besitzen die Namen, die in dem Tab Simulationsergebnisse eingetragen wurden

Energieverbrauch aller über der Zeit in [W]

Die Leistung der KSS-Pumpe wird beim Schleifen nicht betrachtet.

Werkstoffverbrauch über der Zeit in $\left[\frac{mm^3}{s}\right]$

Kühlschmierstoffverbrauch über der Zeit in $\left[\frac{l}{s}\right]$

Werkzeugverschleiß über der Zeit in $[mm]$

Funktionen:

S_nc_calc.m Interpretation der NC-Sätze.

S_G0.m Berechnung der Dauer und Höhe der Lastzustände für die G0 Anweisung.

S_G1.m Berechnung der Dauer und Höhe der Lastzustände für die G1 Anweisung.

s_parts_shape.m Verwaltung des numerischen Werkstückmodells.

Beispiel:

Für die Simulation sind folgende Eingabeparameter gewählt worden:

Schnittgeschwindigkeit 50
Radiale Vorschubgeschwindigkeit [1.6 0.48 0.16]
NC-Code als ASCII Matrix
1. Grundlast der Schleifmaschine 2870
2. Grundlast der Schleifmaschine 3263
Förderstrom der KSS-Pumpe beim Abrichten 1.5
Förderstrom der KSS-Pumpe beim Schleifen 10
Summe der Nebenzeiten vor der Bearbeitung 10
Summe der Nebenzeiten nach der Bearbeitung 60

Abbildung F.2 zeigt eine grafische Darstellung der genanten Ausgabeparameter über der Zeit.

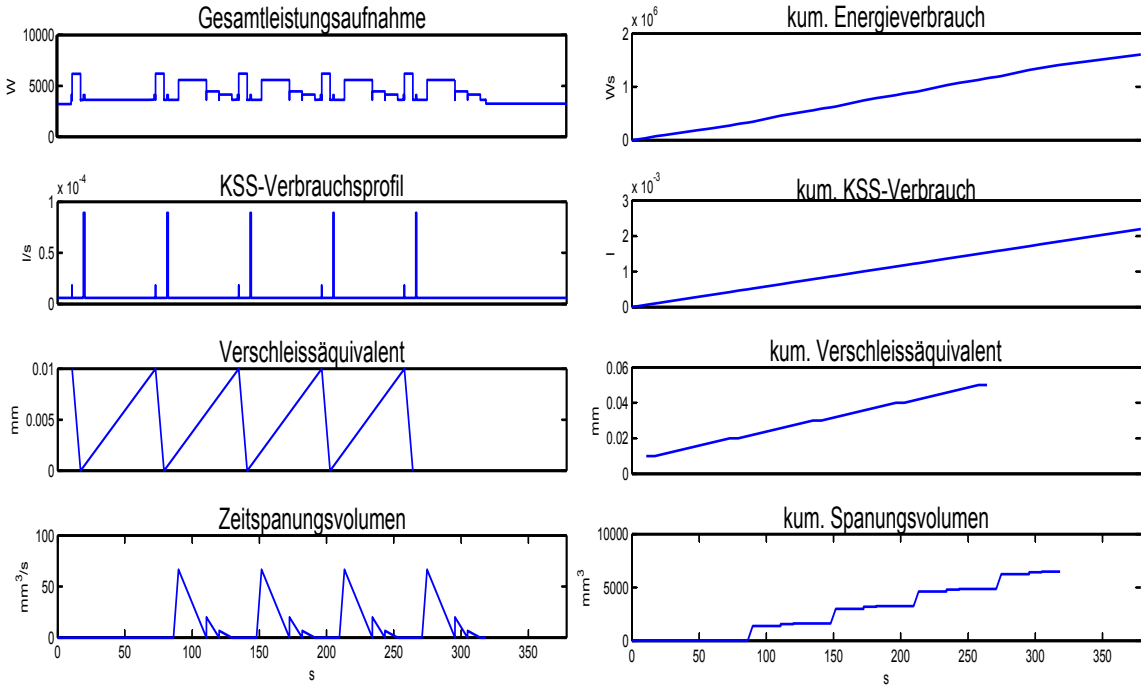


Abbildung F.2.: Darstellung der Ausgabeparameter beim Außenrundschleifen über der Zeit

F.3. Basismodell Außenrundschleifhärten

Eingabeparameter der Parametermaske:

Tab: Maschinenparameter

Schnittgeschwindigkeit $vc = 45 \left[\frac{m}{s}\right]$

Radiale Vorschubgeschwindigkeit $vfr = 2 \left[\frac{mm}{min}\right]$

NC-Code als ASCII Matrix

Grundlast der Schleifmaschine in [W]

Förderstrom der KSS beim Schleifen $[\frac{l}{min}]$

Förderstrom der KSS beim Abrichten $[\frac{l}{min}]$

Summe der Nebenzeiten vor der Bearbeitung in [s]

Summe der Nebenzeiten nach der Bearbeitung in [s]

Wartezeit zwischen zwei Bewegungen in [s]

Tab: Werkzeugverlauf

2 x n Zeros Matrix für den Werkzeugverlauf

Zähler für den Werkzeugverlauf beginnend bei 2

Tab: Simulationsergebnisse

Name der P(t) Struktur

Name der KSS(t) Struktur

Name der Q(t) Struktur

Name der WKZv(t) Struktur

Ausgabeparameter:

Die Ausgabe geschieht über das Erzeugen der Variablen im Matlabworkspace. Die Variablen besitzen die Namen, die in dem Tab Simulationsergebnisse eingetragen wurden

Energieverbrauch aller über der Zeit in [W]

Die Leistung der KSS-Pumpe wird beim Schleifhärten nicht betrachtet.

Werkstoffverbrauch über der Zeit in $\left[\frac{mm^3}{s}\right]$

Kühlschmierstoffverbrauch über der Zeit in $\left[\frac{l}{s}\right]$

Werkzeugverschleiß über der Zeit in $[mm]$

Funktionen:

SH_nc_calc.m Interpretation der NC-Sätze.

SH_G0.m Berechnung der Dauer und Höhe der Lastzustände für die G0 Anweisung.

SH_G1.m Berechnung der Dauer und Höhe der Lastzustände für die G1 Anweisung.

s_parts_shape.m Verwaltung des numerischen Werkstückmodells.

Beispiel:

Schnittgeschwindigkeit 45
Radiale Vorschubgeschwindigkeit 2
NC_Code als ASCII Matrix
Grundlast der Schleifhärtemaschine 2870
Förderstrom der KSS-Pumpe beim Abrichten 1.5
Förderstrom der KSS-Pumpe beim Schleifen 10
Summe der Nebenzeiten vor der Bearbeitung 10
Summe der Nebenzeiten nach der Bearbeitung 60

Abbildung F.3 zeigt eine grafische Darstellung der genanten Ausgabeparameter über der Zeit.

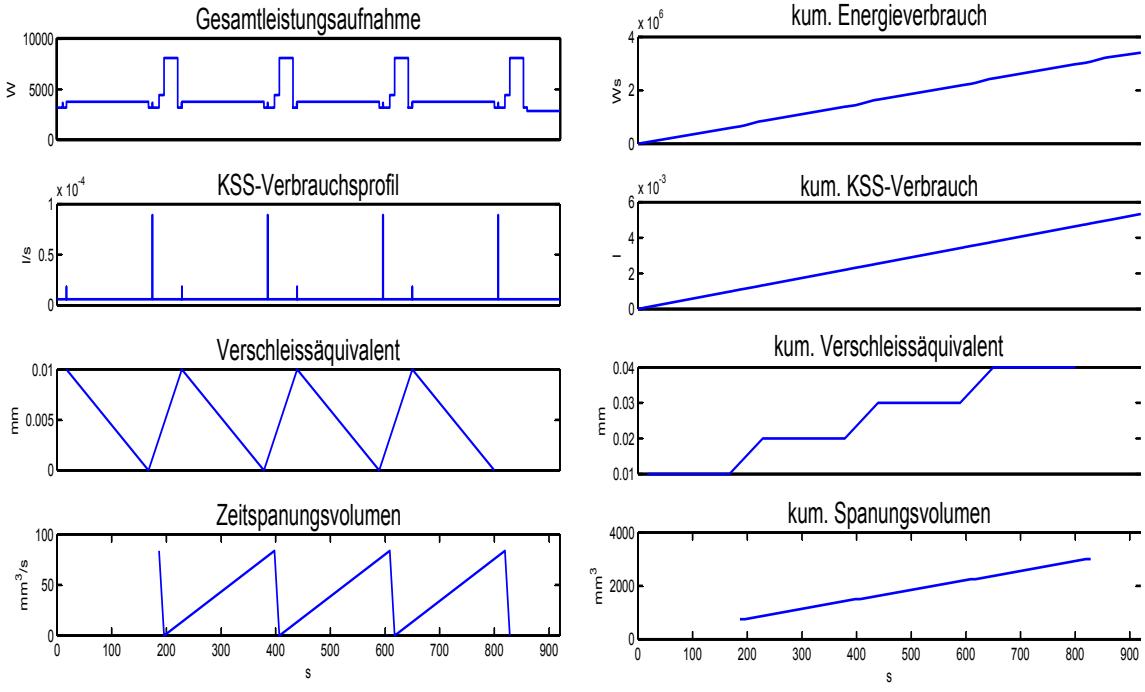


Abbildung F.3.: Darstellung der Ausgabeparameter beim Außenrundschleifhärten über der Zeit

F.4. Basismodell Vakuumhärten

Eingabeparameter der Parametermaske:

Tab:	Härteofenparameter	
	Kapazität	des Härteofens, muss ausschlißlich 6 betragen
	Summe	der Nebenzeiten vor der Bearbeitung in [s]
	Summe	der Nebenzeiten nach der Bearbeitung in [s]
	P(t)	1102 x 2 Messdaten nach Larek
Tab:	Simulationsergebnisse	
	Name	der Ergebnisstruktur

Ausgabeparameter:

Energieverbrauch über der Zeit in [W]

Beispiel:

Kapazität des Härteofens 6
Summe der Nebenzeiten vor der Bearbeitung 600
Summe der Nebenzeiten nach der Bearbeitung 600
Messdaten Matrix

Abbildung F.4 zeigt die Simulationsergebnisse des Härteofens Basismodells.

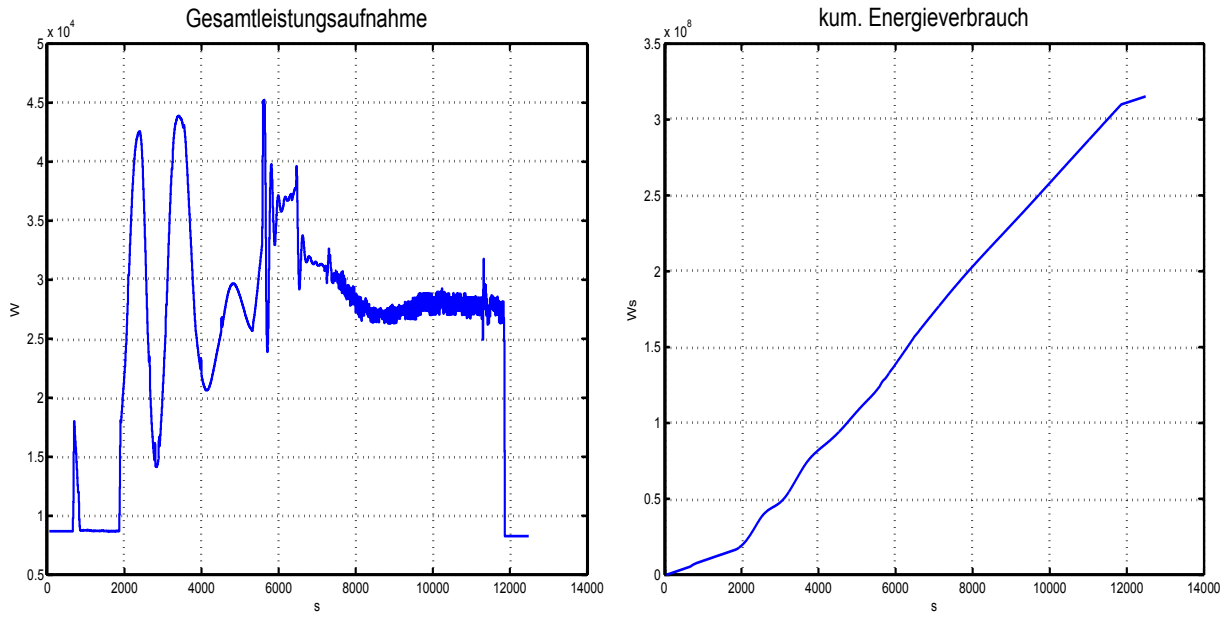


Abbildung F.4.: Simulationsergebnisse des Härteofens

F.5. Basismodell Anlassen

Eingabeparameter der Parametermaske:

Tab: Anlassofenparameter

Kapazität	des Anlassofens, muss ausschließlich 12 betragen
Summe	der Nebenzeiten vor der Bearbeitung in [s]
Summe	der Nebenzeiten nach der Bearbeitung in [s]
P(t)	720 x 2 Messdaten nach Larek

Tab: Simulationsergebnisse

Name	der Ergebnisstruktur

Ausgabeparameter:

Energieverbrauch über der Zeit in [W]

Beispiel:

Kapazität des Anlassofens 12
Summe der Nebenzeiten vor der Bearbeitung 600

Summe der Nebenzeiten nach der Bearbeitung 600
Messdaten Matrix

Abbildung F.5 zeigt die Simulationsergebnisse des Anlassofens Basismodells.

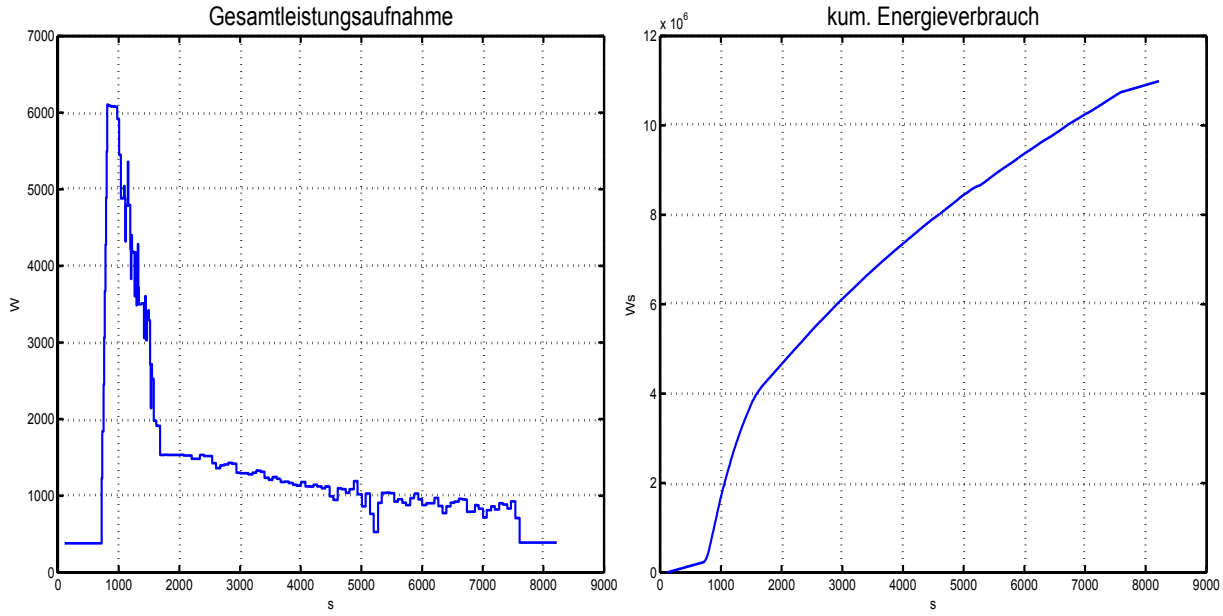


Abbildung F.5.: Simulationsergebnisse des Anlassofens

F.6. Basismodell Induktionshärten

Eingabeparameter der Parametermaske:

Tab: Induktionshärtemaschinenparameter

Kapazität	des Induktionshärtemaschine, muss ausschlißlich 1 betragen
Summe	der Nebenzeiten vor der Bearbeitung in [s]
Summe	der Nebenzeiten nach der Bearbeitung in [s]
P(t)	2 x 2 Messdaten nach Larek

Tab: Simulationsergebnisse

Name	der Ergebnisstruktur

Ausgabeparameter:

Energieverbrauch über der Zeit in [W]

Beispiel:

Kapazität des Induktionsofen 1
Summe der Nebenzeiten vor der Bearbeitung 60
Summe der Nebenzeiten nach der Bearbeitung 60
Messdaten Matrix

Abbildung F.6 zeigt die Simulationsergebnisse des Anlassofens Basismodells.

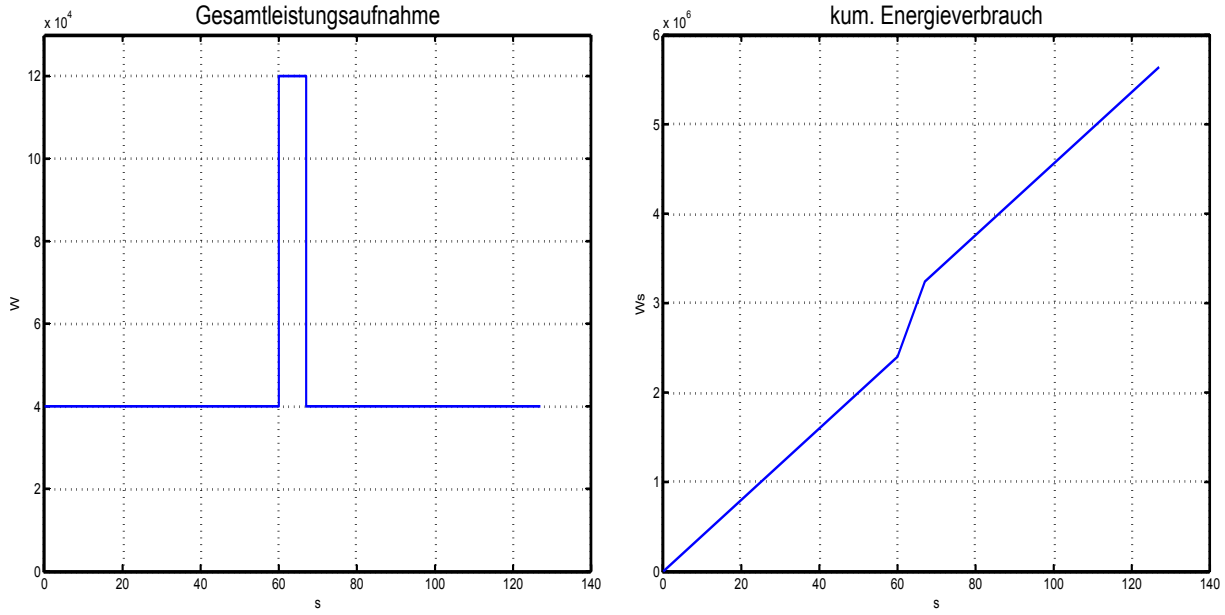


Abbildung F.6.: Simulationsergebnisse des Anlassofens

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die hier vorliegende Arbeit selbstständig, ohne unerlaubte fremde Hilfe und nur unter Verwendung der in der Arbeit aufgeführten Hilfsmittel angefertigt habe.

Wismar, 23. April 2012