Hochschule Kempten
University of Applied Sciences

Bachelorarbeit

Wirtschaftsingenieurwesen

Maschinenbau

Aufbau und Entwicklung eines Konzeptes zur effizienten Datenanbindung einer industriellen Werkzeugmaschine an eine Software zur diskreten Fabriksimulation

Sebastian Schmitt

Matrikelnummer: 353208

Aufgabensteller / Prüfer	Prof. Dr. Ing. Gerald Winz
Arbeit vorgelegt am	25.09.2020
Durchgeführt in der	Fakultät Maschinenbau

Anschrift des Verfassers	Zugspitzstraße 14, 87435 Kempten
	schmitt.basti@icloud.com

Kurzfassung

Im Zuge der Globalisierung und der zunehmenden Digitalisierung, stehen Unternehmen vor der Herausforderung, ihre Produktionsprozesse kontinuierlich zu aktualisieren und zu optimieren. Die dadurch steigende Komplexität der Produktions- und Logistiksysteme erfordert somit den Einsatz von Simulationstechnik, welche dynamische Systemverhalte analysiert, auswertet und verbessert. Durch die digitale Vernetzung rückt die Datenanbindung unterschiedlicher Prozesse an eine Simulationssoftware verstärkt in den Mittelpunkt, um schneller und effizienter auf Veränderungen in der Produktion eingehen zu können.

Das Ziel dieser Bachelorarbeit ist es, ein Konzept für eine effiziente Datenanbindung einer industriellen Werkzeugmaschine der Forschungseinrichtung des Digital Labs an eine Fabriksimulation zu entwickeln und aufzubauen. Für die Datenanbindung werden zwei Methoden mit unterschiedlichen Vorgehensweisen vorgestellt und umgesetzt. Im Anschluss an die Datenanknüpfung, findet eine Integration der Daten in ein geeignetes Simulationsmodell in Plant Simulation statt.

In der ersten Methode werden, da die Werkzeugmaschine kontinuierlich Daten mittels OBDC-Schnittstelle an die Datenbank (MongoDB) überträgt, diese in das Simulationsmodell integriert, um Simulationen anhand der Vergangenheitswerte durchzuführen. Mit einer OPC UA-Schnittstelle wird in der zweiten Methode eine Echtzeitdatenanbindung über einen programmierten OPC UA-Server ermöglicht und schließlich die Daten ebenfalls in ein Simulationsmodell implementiert.

Inhaltsverzeichnis

Abkürzungsverzeichnis

BI	Business Intelligence
DBMS	Datenbank Management System
DBS	Datenbanksystem
DNS	Data Source Name
ERP	Enterprise Resource Planning
IP	Internet Protokoll
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
MQL	MongoDB Query Language
MQTT	Message Queuing Telemetry Transport
NoSQL	Not only SQL
ODBC	Open Database Connectivity
OPC UA	Open Platform Communications United Architecutre
PLM	Product-Lifecycle-Managment
RDBMS	relationales Datenbank Management System
SQL	Structured Query Language
TCP	Transmission Control Protokoll
VDI	Verein Deutscher Ingenieure
XML	Extensible Markup Language
YAML	Yet Another Markup Language

Abbildungsverzeichnis

1 Einstieg, Zielsetzung und Aufbau der Arbeit

Der stetig wachsende internationale Wettbewerb und die zunehmende digitale Vernetzung, stellen in Verbindung mit der dadurch stark schwankenden Nachfrage an Produkten Unternehmen kontinuierlich vor neue Herausforderungen. Für viele Unternehmen hat sich das Handlungsfeld dynamischer, turbulenter und unvorhersehbarer gestaltet. Erhöhte Variantenvielfalt, aktive Mitgestaltung der Kunden an Produkten und verkürzte Produktlebenszyklen zwingen Unternehmen zudem dazu, laufend ihre Produktion flexibler und effektiver zu gestalten. Die dadurch permanent wachsende Komplexität der Produktions- und Logistiksysteme sorgt dafür, dass die Einführung von Simulationstechnik in den letzten Jahrzehnten kontinuierlich an Bedeutung gewonnen hat. Dabei werden Simulationen in der Produktion und Logistik meist vor der Installation und Inbetriebnahme neuwertiger Produktionslinien oder bei bestehenden Materialflussanlagen eingesetzt, um das Systemverhalten mithilfe eines Simulationsmodells gezielt untersuchen zu können (vgl. Birgit et al. 2017, S. 3 f.) (vgl. Gutenschwager et al. 2017, S. 1).

Die fortschreitende Digitalisierung sorgt dafür, dass Produktionsanlagen heutzutage laufend Daten an Datenbanken oder Server senden. Um schneller und effizienter auf Veränderungen der Produktion reagieren zu können, gerät die Datenanbindung an eine Simulation und der Begriff des *digitalen Zwillings* verstärkt in den Mittelpunkt.

Im Rahmen dieser Bachelorarbeit soll untersucht werden, inwiefern eine real operierende Werkzeugmaschine der Forschungseinrichtung des Digital Labs digital abgebildet und an eine diskrete Simulation angebunden werden kann. Ziel der Arbeit ist es, die Datenspeicherung der Werkzeugmaschine zu untersuchen und anschließend die Daten mithilfe geeigneter Datenschnittstellen in ein erzeugtes Simulationsmodell zu integrieren. Anhand einer OPC UA-Schnittstelle und einer ODBC-Schnittstelle werden zwei Methoden vorgestellt, die auf unterschiedliche Art die Daten der Werkzeugmaschine anbinden und in eine Simulation implementieren. Dabei liegt der Fokus beider Methoden darauf, durch einen simplen Simulationsaufbau eine grundsätzliche Datenanbindung zu ermöglichen.

Der grundlegende Aufbau der Arbeit wird in Abbildung 1 grafisch dargestellt. Beginnend mit dem Einstieg und einer Einführung in die diskrete Simulation werden im Anschluss die Grundlagen von Datenbanken behandelt. Danach findet eine detaillierte Beschreibung der eingesetzten Datenschnittstellen statt. Mithilfe der Informationen werden anschließend zwei Datenanbindungsmethoden vorgestellt, umgesetzt und anschließend in ein Simulationsmodell integriert. Das Fazit beinhaltet eine Zusammenfassung der Ergebnisse beider Schnittstellen und einen Ausblick über zukünftige Entwicklungen im Bereich der Datenanbindung.

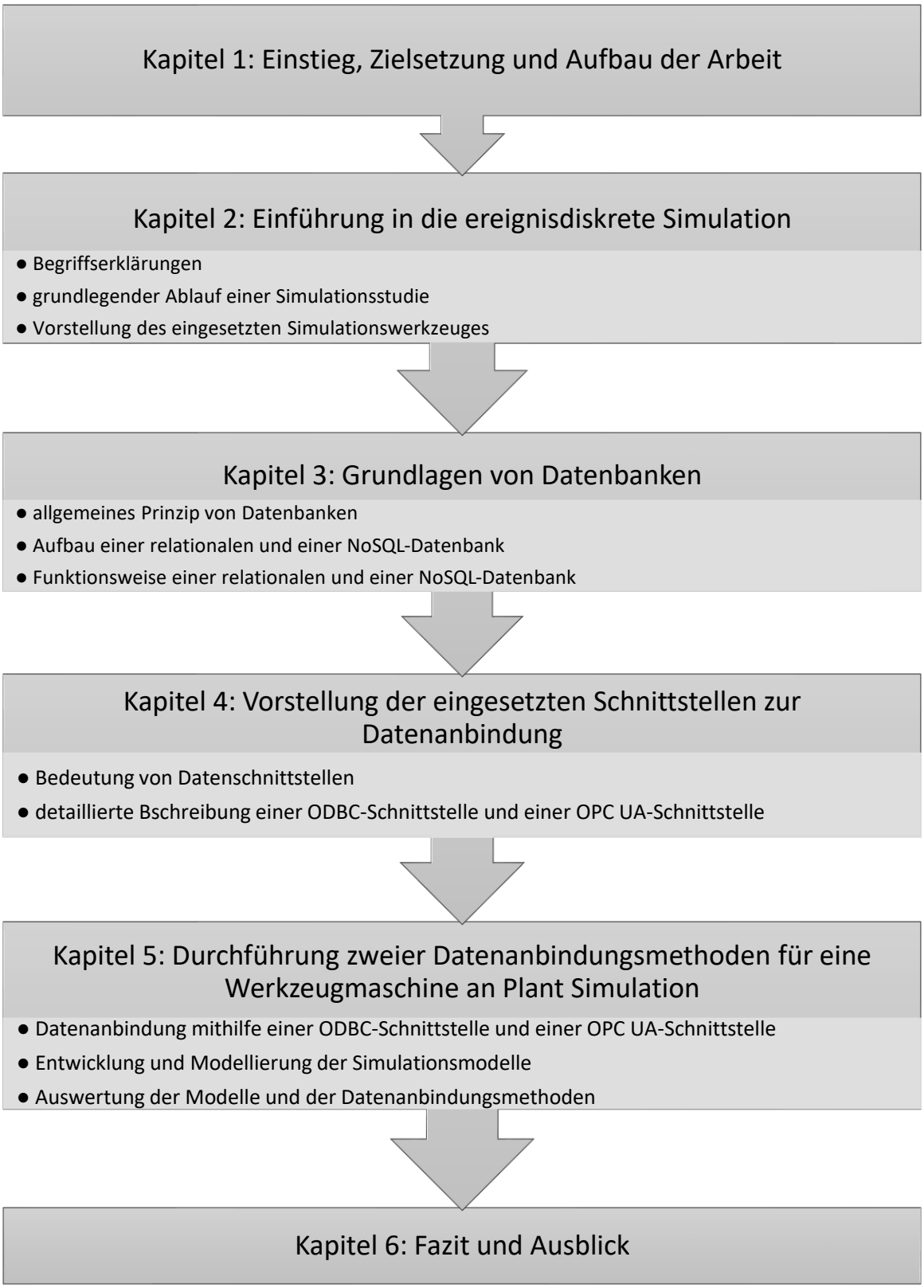


Abbildung 1: grundlegender Aufbau der Bachelorarbeit

2 Einführung in die ereignisdiskrete Simulation

Der Einsatz von Simulationen ist für die Lösung zahlreicher hochkomplexer Optimierungs- und Planungsprobleme prädestiniert. Sie kommen meist dann zum Einsatz, wenn mathematische Optimierungsverfahren oder Tabellenkalkulationen an ihre Grenzen stoßen. Mithilfe einer Simulation gibt es die Möglichkeit, Auswirkungen auf das Gesamtsystem betrachten zu können, ohne es in Teilsysteme untergliedern zu müssen. Dadurch können unterschiedliche Systemverhalten effektiver, detaillierter und unter Berücksichtigung von stochastischen Einflüssen untersucht werden. (vgl. Gutenschwager et al. 2017, S. 37 f.) (vgl. Eley 2012, S. 4).

Vor allem im Bereich der Produktionsplanung und -optimierung ist es aufgrund der stetig wachsenden Komplexität der zu untersuchenden Prozesse sinnvoll, auf die Anwendung von Simulationen zurückzugreifen. Diese ermöglichen es, Produktionsanlagen, Prozesse oder einzelne Maschinen zu analysieren und zu verbessern. Dabei wird mithilfe einer Abstraktion des realen oder imaginären Systems versucht, ein Verständnis für die Problemstellung und deren Zusammenhänge herzustellen. Anschließend können durch eine gezielte Veränderung von Stellgrößen die Auswirkungen auf das System untersucht werden.

In den folgenden Abschnitten werden zunächst wichtige Definitionen erklärt, die häufig im Zusammenhang mit der ereignisdiskreten Simulationen aufkommen. Ziel ist es, ein fundamentales Verständnis für die Bedeutung der Simulation herzustellen. Dabei orientieren sich die Definitionen stets an der VDI-Richtlinie 3633. Die Richtlinie dient als Basis und Orientierung für Anwender, Simulationsstudien effizienter durchzuführen, indem Begriffe eindeutig definiert und Leitsätze zur Verfügung gestellt werden. Die Vorgaben sollen gezielt Missverständnisse zwischen den beteiligten Parteien während der Durchführung einer Simulationsstudie vermeiden (vgl. VDI 2014, S. 2).

Im Anschluss daran, wird der Ablauf einer Simulationsstudie erklärt und das eingesetzte Simulationswerkzeug vorgestellt. Abschließend findet eine Betrachtung des digitalen Zwillings im Zusammenhang mit der ereignisdiskreten Simulation statt.

2.1 Die Simulation

Allgemein ist die Simulation ein Verfahren, bei dem reale oder imaginäre Systeme mithilfe eines Modells abstrahiert werden. Anschließend finden am erzeugten Modell unterschiedliche Experimente statt, um neue Erkenntnisse über das System zu erlangen. Somit kann die Simulation in der industriellen Praxis als eine Entscheidungsunterstützung angesehen werden, um Fehlentscheidungen vorzubeugen und Risiken zu minimieren (vgl. Gutenschwager et al. 2017, S. 22).

Wird bei der Durchführung einer Simulation auf den Einsatz von Computern zurückgegriffen, spricht man hierbei von einer digitalen- oder einer Computersimulation. Digitale Simulationen zeichnen sich dadurch aus, dass das zu integrierende Modell in einer systematischen Struktur vorliegen und in eine geeignete Software implementiert werden muss (vgl. Eley 2012, S. 4). Die genutzte Software ist dabei das Simulationswerkzeug.

Der Begriff Simulation wird laut VDI wie folgt konkretisiert: Die Simulation ist das

„Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierbaren Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind; insbesondere werden die Prozesse über die Zeit entwickelt" (VDI 2014, S. 3).

Die Berücksichtigung des Zeitverlaufes einer Simulation, ist entscheidend für die Ergebnisse und die Auswirkungen auf ein System und ferner der ausschlaggebende Grund für die Abgrenzung von rein statistischen Analysen, wie beispielsweise Tabellenkalkulationsanwendungen. Je nach Betrachtung des Zeitverhaltens in einem Simulationsmodell wird zwischen der kontinuierlichen und der diskreten Simulation unterschieden. In Abbildung 2 ist das Verhalten unterschiedlicher Zustandsmengen grafisch dargestellt, wobei die kontinuierliche Simulation kontinuierliche Zeitmengen, sowie kontinuierliche Zustandsmengen beinhaltet. Daraus resultierend, entsteht eine Abbildung mit Systemzuständen, die sich in einem stetigen Verlauf in Abhängigkeit von der Zeit verändern (vgl. Gutenschwager et al. 2017, S. 24 f.).

Bei der ereignisdiskreten Simulation (auch Ablaufsimulation genannt) verändert sich die Zeit in Abhängigkeit eines eintretenden Ereignisses. Das heißt, die Zeit setzt sich in einem sprunghaften Intervall auf den Zeitpunkt des nächsten Ereignisses. Die dadurch hervorgerufenen Zustandsänderungen sind somit ebenfalls diskret. (vgl. Gutenschwager et al. 2017, S. 53 f.).

Daraus resultiert ein wesentlicher Vorteil der ereignisdiskreten Simulation, denn somit ist die Rechenzeit im Vergleich zu einer kontinuierlichen Simulation deutlich geringer, da die Rechenzeit nur von der Anzahl der eintretenden Ereignisse abhängt und nicht von der Länge der Zeitintervalle (vgl. Gutenschwager et al. 2017, S. 55).

Ein weiterer Vorteil der ereignisdiskreten Simulation ist es, mit hoher Genauigkeit das dynamische Verhalten eines Systems nachspielen zu können. Ein typisches Beispiel für eine ereignisdiskrete Simulation ist die Betrachtung von Bearbeitungsvorgänge in einer Produktion (vgl. Hedtstück 2013, S. 22).

Im Verlauf dieser Bachelorarbeit wird sich ausschließlich auf die ereignisdiskrete Simulation konzentriert.

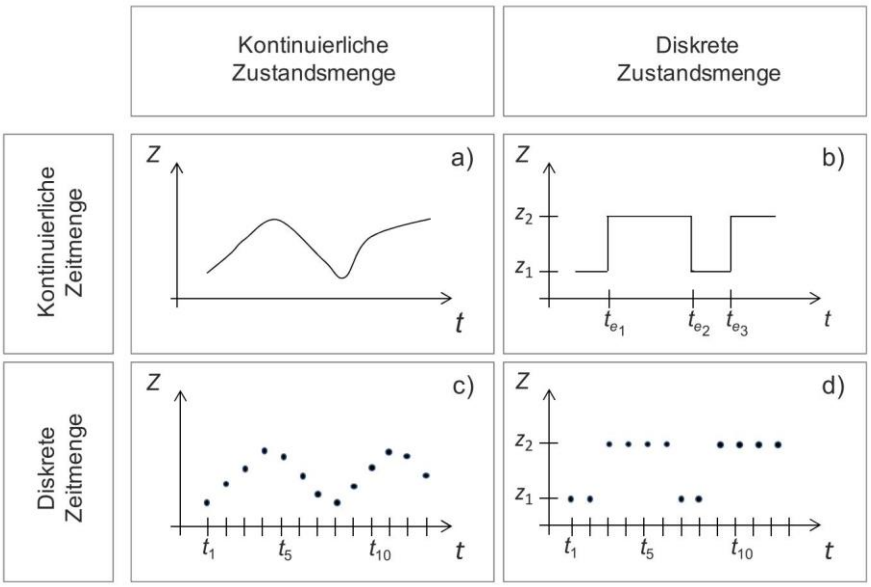


Abbildung 2: Beziehung von Zustands- und Zeitmenge entnommen aus Gutenschwager et al. 2017, S. 16 (in Anlehnung an Ören 1979, S.36)

2.2 Systeme, Modelle und Prozesse

Im Zusammenhang mit der ereignisdiskreten Simulation ist häufig die Rede von Systemen, Modellen und Prozessen. In Anlehnung an die DIN IEC Norm 60050-351 wird ein System als eine Menge von Objekten innerhalb eines abgegrenzten Umfelds bezeichnet, welche Attribute besitzen sowie zueinander in Beziehung stehen und sich gegenseitig beeinflussen (vgl. DIN IEC 60050-351:2013, S. 21).

Dabei können Objekte einerseits zeitweise in einem System vorhanden sein, sogenannte temporäre Objekte wie beispielsweise Aufträge, Werkstücke oder Fahrzeuge. Andererseits können auch dauerhafte Objekte, sogenannte permanente Objekte wie Maschinen, Läger oder Abteilungen in einem System vorkommen (vgl. Hedtstück 2013, S. 11).

In Abbildung 3 werden die Systeme anhand unterschiedlicher Merkmale in verschiedene Typen eingeteilt. Sobald das zeitliche Verhalten berücksichtigt wird, spricht man hierbei von einem dynamischen System. Wird das System nur zu einem Zeitpunkt betrachtet oder spielt die Zeit keine Rolle, handelt es sich um ein statisches System. In der industriellen Praxis werden Systemverhalten meist durch zufällig eintretende Ereignisse beeinflusst und sind somit von stochastischer Natur. Zur Untersuchung solcher Systemverhalten, stehen nur experimentelle Methoden zur Verfügung. Enthält ein System keine zufallsabhängigen Komponenten, bezeichnet man es als deterministisches System, welches das Gegenstück zu einem stochastischen System darstellt (vgl. Hedtstück 2013, S. 10 f.).

Sobald das reale oder geplante System abgebildet wird, spricht man hierbei von einem Modell (vgl. Bangsow 2016, S. 2).

Laut VDI-Richtlinie ist ein Modell eine

„Vereinfachte Nachbildung eines geplanten oder existierenden Systems mit seinen Prozessen in einem anderen begrifflichen oder gegenständlichen System“ (vgl. VDI 2014, S. 3).

Expliziter formuliert, bildet ein Modell die Struktur oder das Verhalten eines zu untersuchenden Systems mit einem geringeren Detaillierungsgrad als beim ursprünglichen System ab. Dabei liegt der Fokus der Abstraktion auf den wesentlichen Komponenten und notwendigen Prozessen, die ausschlaggebend sind, um eine Leistungsbewertung am System durchführen zu können. Details, die für die Untersuchung des Systems nicht relevant sind, werden bei der Bildung des Modells weggelassen (vgl. März et al. 2011, S. 13).

Innerhalb eines Modells laufen unterschiedliche Systeme ab und somit auch eine Vielzahl an Prozessen mit verschiedenen Attributen und Abhängigkeiten. Dabei wird ein Prozess allgemein definiert, als eine

„Gesamtheit von aufeinander einwirkenden Vorgängen in einem System, durch die Materie, Energie oder Information umgeformt, transportiert oder gespeichert wird" (DIN IEC 60050-351:2013, S. 32).

Bezogen auf ein dynamisches System, ist ein Prozess die Zusammenfassung aller Objekte eines Systems mit dessen Eigenschaften und Beziehungen. Dabei können Prozesse in einem dynamischen System sequentiell, parallel, nebenläufig und synchronisiert kohärieren. Betrachtet man nur ein einzelnes Objekt, so wird dieses als Teilprozess innerhalb des Gesamtsystems angesehen. Ein diskreter Prozess zeichnet sich dadurch aus, dass die Ereignisse meist den Abschluss einer Aktivität, sowie den Beginn einer neuen Aktivität kennzeichnen. Ausgenommen ist das erste und letzte Ereignis (vgl. Hedtstück 2013, S. 16).

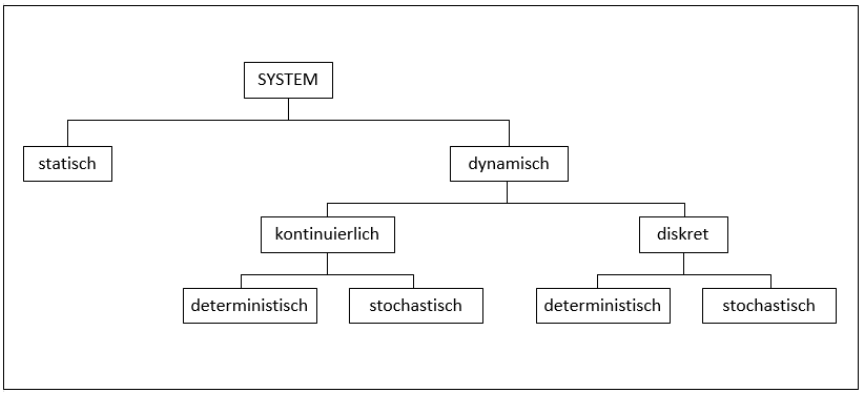


Abbildung 3: Typen von Systemen (in Anlehnung an Hedtstück 2013, S. 11)

2.3 Ablauf einer Simulationsstudie

Mit dem Begriff der Simulationsstudie ist der gesamte Prozess von der Erstellung eines Simulationsmodells über die Durchführung einer oder mehrerer Simulationsläufe bis hin zur Übertragung der Ergebnisse zurück auf das zu untersuchende System gemeint (vgl. Eley 2012, S. 4).

In den meisten Fällen ist in einer Simulation eine umfassende Simulationsstudie impliziert. Das Ziel einer Simulationsstudie ist eine Prozessverbesserung anhand zuvor festgelegter Key Performance Indicators (KPI) durchzuführen, welche die Leistungsfähigkeit eines Systems messen und anschließend optimieren. In diesem Zusammenhang stellt die Optimierung die Verbesserung eines Prozesses dar und nicht die Suche nach einem Optimum. Die Kunst bei der Zielerreichung liegt darin, eine Annäherung der Ziele nur soweit vorzunehmen, bis sich insgesamt eine maximale Zunahme der gewünschten Wertschöpfung ergibt. Eine typische Zielsetzung ist beispielsweise die Reduktion der Produktionszeit. Dabei spielen Kennzahlen wie Durchlaufzeiten, Liegezeiten oder Rüstzeiten eine wichtige Rolle, die in der durchzuführenden Simulationsstudie betrachtet und analysiert werden müssen (vgl. Hedtstück 2013, S. 7).

Um eine Simulationsstudie erfolgreich und effizient durchführen zu können, ist es wichtig sorgfältig und systematisch vorzugehen. Deshalb ist die Anwendung eines Vorgehensmodells unabdingbar, welches die einzelnen Abläufe der Durchführung einer Simulationsstudie definiert. Ein Vorgehensmodell unterstützt den Prozess einer Simulation von der Aufgabenspezifikation bis hin zur Erzeugung von Resultaten. Ziel ist es, den Ablauf bei der Entwicklung eines Modells so zu strukturieren, dass durch ein systematisches Vorgehen Fehler von vorn herein vermieden werden. Wird ein Vorgehensmodell sinnvoll angewendet und somit eine Simulationsstudie optimal durchgeführt, können wertvolle Erkenntnisse über ein betrachtetes System erlangt werden (vgl. Gutenschwager et al. 2017, S. 140 f.).

2.4 Tecnomatix Plant Simulation

In der industriellen Praxis gibt es unterschiedliche Simulationswerkzeuge, die das Vorgehen bei einer Simulationsstudie unterstützen sollen. Die Werkzeuge dienen hauptsächlich dazu, funktionale Simulationsmodelle unter Berücksichtigung aller Einflussfaktoren zu erstellen und auszuwerten (vgl. Gutenschwager et al. 2017, S. 219).

Im Zeitalter der Digitalisierung stehen Simulationen verstärkt bei der Umsetzung der Digitalen Fabrik als wichtige Methode im Fokus. Vor allem Schnittstellenentwicklungen für den Datenaustausch werden somit immer bedeutender (vgl. Bracht et al. 2018, S. 83).

Dadurch entstehen neue individuelle Anforderungen an die Simulationshersteller, die die unterschiedlichen Schnittstellen zur Verfügung stellen müssen, um weiterhin wettbewerbsfähig agieren zu können. Tecnomatix Plant Simulation bietet eine Vielzahl von Schnittstellen an, mit denen Datenbanken, Grafiken oder Daten als Textdateien ausgelesen werden können.

Das Simulationsinstrument gehört zu den ereignisdiskreten Simulationen (vgl. Abschnitt 2.1) und ist eine Standardsoftware, mit dem komplexe Produktionssysteme und -prozesse übersichtlich in Computermodellen abgebildet werden können. Mit Plant Simulation ist es möglich den Materialfluss, die Ressourcenauslastung und die Supply-Chain-Prozesse auf allen Ebenen eines Unternehmens zu optimieren (vgl. Siemens Product Lifecycle Management Software Inc. 2008).

Dabei zeichnet sich das Simulationssystem laut Siemens PLM Software (vgl. Siemens Product Lifecycle Management Software Inc. 2008) unter anderem durch folgende Leistungsmerkmale aus:

- Möglichkeit zur Durchführung von Experimenten, Analysen und „Was-wäre-wenn"-Szenarien
- Objektorientierte, hierarchische Modelle von Fabriken einschließlich Geschäfts-, Logistik- und Produktionsprozessen
- Einfache Anwendung durch Windows-Konformität
- Bausteinbibliotheken für das schnelle und effiziente Modellieren typischer Szenarien
- Offene Systemarchitektur, die Schnittstellen und Integrationsmöglichkeiten unterstützt (ODBC, OPC UA, HTML, SQL, usw.). Grafiken und Diagramme zur Analyse von Durchsatz, Ressourcenauslastung und Engpässen

Vor allem die offene Systemarchitektur, mit den unterschiedlichen Schnittstellen sorgt dafür, dass Plant Simulation ein wichtiges Einstiegswerkzeug in das Thema der Digitalen Fabrik darstellt. Um die im späteren Verlauf durchgeführten Testmodelle besser nachvollziehen zu können, wird im folgenden Abschnitt auf die Grundlagen von Plant Simulation eingegangen.

2.4.1 Grundlegende Funktionen

Die Erzeugung von Simulationsmodellen findet in Plant Simulation in der Bedienoberfläche statt, die auch als TUNE-Fenster bezeichnet wird. Dabei sind die wichtigsten Bereiche dieser Arbeitsoberfläche in Abbildung 4 mit rot umrandeten Feldern markiert.

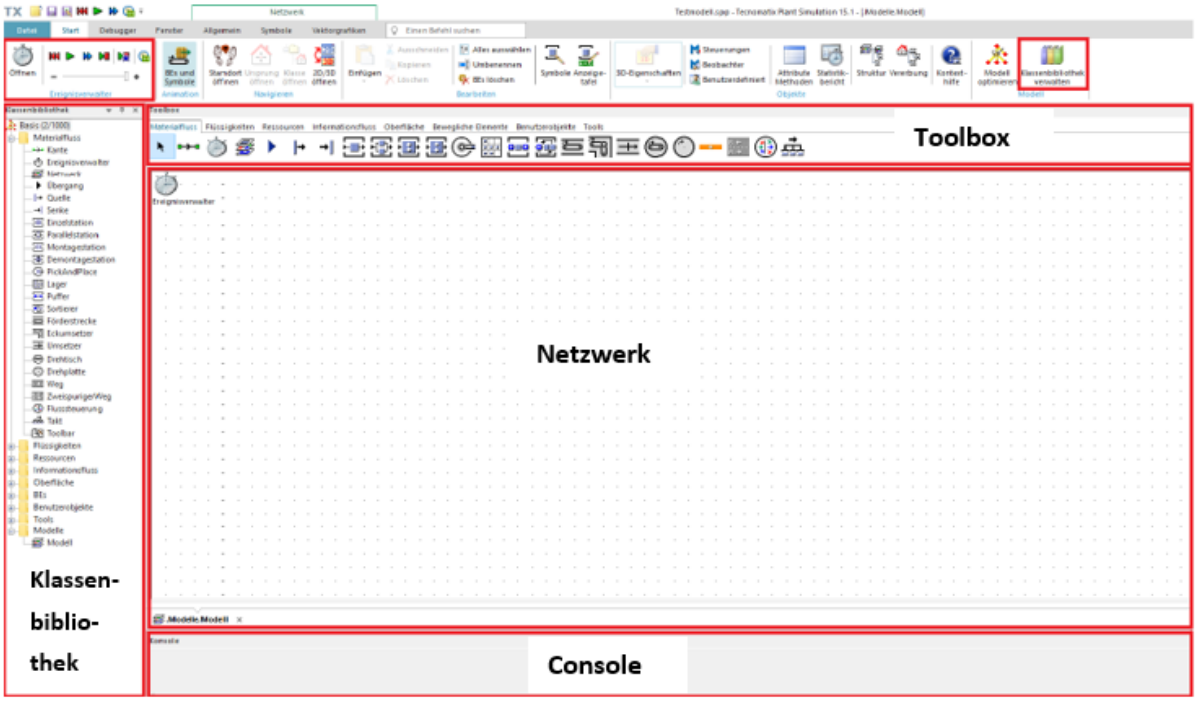


Abbildung 4 Die Arbeitsoberfläche von Tecnomatix Plant Simulation

Am linken Rand des TUNE-Fensters befindet sich die Klassenbibliothek, die alle benötigten Bausteine für eine Erstellung eines Simulationsmodells umfasst. Die Bausteine sind in die Gruppen Materialfluss, Ressourcen, Informationsfluss, Oberfläche, BEs (Bewegliche Einheiten), Tools und Modelle eingeteilt. Je nach Anwendungszweck können die Objekte aus ihren Gruppen ausgewählt und falls nötig individuell angepasst werden. Sollten benötigte Objekte wie beispielsweise Datenbankschnittstellen fehlen, die regulär nicht in der Klassenbibliothek angezeigt werden, können diese nachträglich über das Icon "Klassenbibliothek verwalten" beschafft werden (vgl. Eley 2012, S. 33).

Die Toolbox im oberen Drittel des Programms listet alle verfügbaren Elemente auf und ermöglicht eine schnelle Übertragung der Bausteine in das Modell. Die Toolbox kann auf eigene Bedürfnisse entsprechend umgestaltet werden. Die Konsole im unteren Bereich der Bedienoberfläche liefert nützliche Informationen während eines Simulationsprozesses. So werden z.B. Fehlermeldungen angezeigt, die während eines Simulationslaufes entstanden sind (vgl. Bangsow 2016, S. 9).

Das Netzwerk befindet sich in der Mitte der Arbeitsoberfläche. In das Fenster können je nach Anwendung verschiedene Objekte aus der Klassenbibliothek oder der Toolbox gezogen und miteinander in Verbindung gebracht werden. Während eines Simulationslaufes sind Bewegungen unterschiedlichster Art im Netzwerkfenster sichtbar. Über den Ereignisverwalter, der sich im Netzwerk befindet oder über die Toolbox eingefügt werden kann, wird die Simulation

zeitlich gesteuert. Optional kann der Ereignisverwalter, sobald er im Netzwerk sichtbar ist, unter der Registerkarte „Start“ bedient werden. Der Ereignisverwalter kann die Geschwindigkeit der Simulation sowie der Start- und Endzeitpunkt einstellen (vgl. Eley 2012, S. 34).

2.4.2 Elemente von Plant Simulation

In der Tabelle 1 werden die für die Arbeit benötigten Elemente grafisch dargestellt und kurz erklärt.

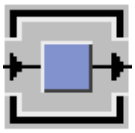
Bezeichnung	Funktion	Symbol
Quelle	Eine Quelle produziert bewegliche Objekte (BEs). In der Realität meist vergleichbar mit dem Wareneingang einer Fabrik	
Senke	Eine Senke vernichtet bewegliche Objekte (BEs). In der Realität meist der Warenausgang oder der Versand einer Fabrik.	
Bewegliche Elemente (BE)	BEs bilden Objekte wie Werkstücke oder Transportbehälter ab. Sie können produziert, transportiert oder bearbeitet werden.	
Einzelstation	Eine Einzelstation nimmt ein BE entgegen und gibt es nach Ablauf der Rüst- und Bearbeitungszeit an das nächste Objekt weiter	
Kante	Die Kante stellt eine Materialflussverbindung zwischen zwei Objekten im gleichen Netzwerk her. Die Flussrichtung wird mittels Pfeil auf der Verbindungslinie dargestellt.	
ODBC	Eine ODBC-Schnittstelle stellt eine Verbindung zu einer beliebigen Datenquelle her.	
OPC UA	Eine OPC UA-Schnittstelle ermöglicht den Zugriff auf Prozessteuerungsgeräte (SPS-Steuerungen)	
Methode	Die Methode erstellt Steuerungen, die andere Elemente beeinflussen können. Methoden werden in der Programmiersprache SimTalk geschrieben	
Tabelle	Eine Tabelle ist eine Liste mit mehreren Spalten. Mittels Methode können Werte erzeugt oder eingelesen werden	

Tabelle 1: Elemente in Plant Simulation (vgl. Eley, 2012 S. 33 ff.)

Da sich die Bachelorarbeit intensiv mit Datenanbindungen auseinandersetzt, werden die einzelnen Schnittstellen separat in Kapitel 4 erläutert.

2.4.3 Die Programmiersprache SimTalk

Das Verhalten der Objekte in Plant Simulation ist in der Praxis oft nicht ausreichend, um realistische Systeme präzise genug darstellen zu können. Aus diesem Grund bietet Plant Simulation eine eigene Programmiersprache namens SimTalk an. Mit dieser ist es möglich die Standardeigenschaften von Objekte individuell durch Funktionen und Prozeduren zu erweitern. Die verschiedenen Anweisungen werden in dem Informationsflussobjekt „Methode“ programmiert und anschließend je nach Anwendung mit den einzelnen Objekten verknüpft.

Eine Methode erstellt Steuerungen, mit denen sie andere Elemente beeinflussen kann. Steuerungen beinhalten wiederum Variablen, die in einer Methode deklariert werden müssen. Eine Variable ist ein definierter Ort im Speicher; das heißt, ein Bereich im Programm indem Informationen gespeichert werden. Deshalb ist es sinnvoll aussagekräftige Namen für eine Variable zu verwenden, um den Überblick in einer Methode zu bewahren. Dabei unterscheidet Plant Simulation zwischen lokalen und globalen Variablen. Eine lokale Variable ist ausschließlich innerhalb einer Methode verfügbar. Im Umkehrschluss sind globale Variablen methodenübergreifend ansprechbar. Jede Variable braucht einen eindeutigen Datentyp, um die Informationen als Zahlen, Zeichen oder Zeichenketten speichern zu können. Dafür stehen in SimTalk elementare Datentypen zur Verfügung, die in Tabelle 2 aufgelistet sind. Die verwendeten Datentypen von SimTalk gleichen den Datentypen der gängigsten Programmiersprachen (vgl. Bangsow 2016, S. 17).

Datentyp	Verwendung	Definitionsbereich
Integer	Ganzzahlige Werte	-2147483648 ≤ integer ≤ 2147483648
String	Zeichenkette	Zeichen (jeder Buchstabe, Zahlen, Sonderzeichen)
Boolean	Bool'sche Variablen	TRUE oder FALSE
Real	Gleitkommazahlen	-8.9 ∗ 10307 ≤ real ≤ 8.9 ∗ 10307
Time	Zeiten	Angabe in Sekunden bzw. im Zeitformat: <hh>:<mm>:<ss.ss>
Date	Datumsangaben	Datum von 1.1.1970 bis 31.12.2038
Datetime	Datums- und Zeitangaben	Datum zwischen 1.1.1970 0:00:00 Uhr und dem 31.12.9999 23:59:59 Uhr
Object	Verweis auf einen Baustein	Verweis auf ein Object (außer Kommentar)
Table	Tabelle mit mehreren Spalten und Zeilen	vielschichtige Datenmengen
List	Tabelle mit nur einer Spalte	vielschichtige Datenmengen

Tabelle 2: Plant Simulation-Datentypen (vgl. Eley, 2012 S. 51)

2.5 Der Begriff „digitaler Zwilling" im Zusammenhang mit der diskreten Simulation

Durch die Fortschreitung der Digitalisierung, gewinnt der Begriff des digitalen Zwillings vermehrt an Bedeutung. Dabei wird der Digitale Zwilling im Informatiklexikon der Gesellschaft für Informatik als *„digitale Repräsentanzen von Dingen aus der realen Welt"* (Kuhn Thomas 2017) definiert.

Diese Dinge können sowohl physische Objekte als auch nicht-physische Dinge sein. Bezogen auf die Produktion und Logistik ist ein digitaler Zwilling die Abbildung einer geplanten beziehungsweise realen Produktionsanlage und kann als ein Gesamtsystem mit unterschiedlichen Teilsystemen angesehen werden. Er erzeugt auf digitaler Ebene eine vollständige Nachbildung der Produktionsprozesse und gibt die dynamischen Abläufe während der Fertigung in Echtzeit wieder. Mit Hilfe von Simulationsmodellen können sowohl die Merkmale der Systeme als auch deren Verhalten beschrieben werden. In Abbildung 5 ist der Digitale Zwilling mit seinen Bestandteilen grafisch dargestellt. So besteht eine kontinuierliche Interaktion zwischen den einzelnen Modellen und der Datenbank. Die Simulationsmodelle greifen auf die Datenbank zu. Die Informationsmodelle sorgen dafür, dass die Daten eine Semantik erhalten und unterstützen das Simulationsmodell, indem sie die benötigten Informationen auffinden und interpretieren. Des Weiteren sind für den durchgehenden Import und Export der Daten aus den Werkzeugen Schnittstellen notwendig, die eine Übertragung dieser an die Datenbank

ermöglichen. Die Visualisierung ist ein wichtiger Faktor des Digitalen Zwillings, welche die hinterlegten Daten anschaulich darstellt und als Grundlage für eine Analyse dient (vgl. Schüller et al. 2019, 71 ff.).

Die Ablaufsimulation ist ein Teilsystem des Digitalen Zwillings, mit der es durch eine regelmäßige Durchführung möglich ist, gezielt auf Veränderungen oder Störungen in der Produktion eingehen zu können. Produktionssituationen können somit kontinuierlich vorausschauend betrachtet werden.

Zukünftig soll es laut dem Frauenhofer-Institut für Produktionsanlagen und Konstruktionstechnik sogar möglich sein, die reale und digitale Produktion komplett zu fusionieren. Dadurch entsteht ein Gesamtsystem, das sich im laufenden Betrieb selbständig überwacht, steuert und korrigiert. Das System erkennt Störungen und entscheidet selbstständig wie das Problem behoben werden kann. Dabei besteht eine kontinuierliche Kommunikation zwischen den Maschinen und der Software. Somit übernimmt der Produktionsleiter nur noch eine überwachende Funktion und muss selbst nicht eingreifen. Über den Datentransfer zwischen der Anlage und dem Digitalen Zwilling lässt sich zudem die Qualität der Werkstücke laufend kontrollieren (vgl. Frauenhofer-Institut für Produktionsanlagen und Konstruktionstechnik IPK, S. 1 ff.).

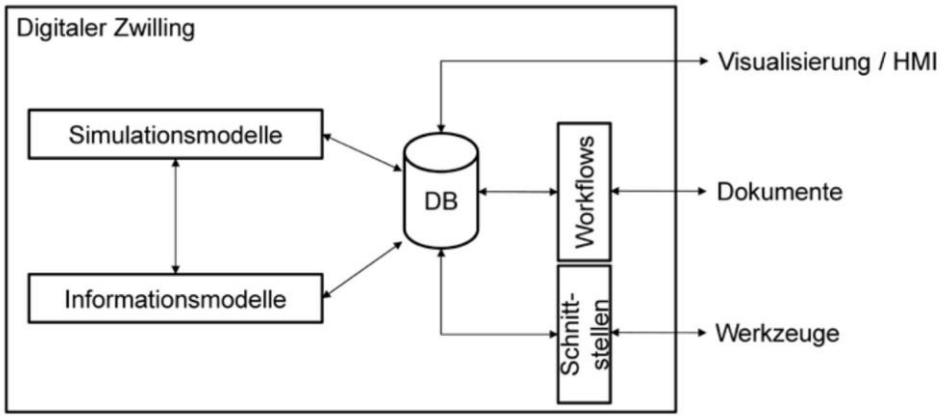


Abbildung 5: Aufbau eines Digitalen Zwillings entnommen aus Schüller et al. 2019, S. 74

Damit das Konzept des Frauenhofer-Instituts in der industriellen Praxis umsetzbar ist, sind Datenbanken notwendig, die die enormen Mengen an Informationen verarbeiten. Die Datenbank ist ein essenzieller Bestandteil, um eine Vielzahl von Daten zu sammeln und zu speichern. Auch in Bezug auf die in dieser Bachelorarbeit erzeugten Simulationsmodelle, spielen Datenbanken eine wesentliche Rolle und werden somit im folgenden Kapitel spezifischer erläutert.

3 Grundlagen von Datenbanken

Die Produktion und Logistik wird immer komplexer und vielfältiger. Dabei entstehen jeden Tag Unmengen an Informationen aus unterschiedlichen Quellen. Um die Produktion wettbewerbsfähig zu halten, müssen immer mehr Informationen analysiert, ausgewertet und in Korrelation zueinander gebracht werden. Für die Herausfilterung der relevanten Informationen sind heutzutage Unternehmen auf computergestützte Systeme wie beispielsweise Datenbanken angewiesen.

Eine Datenbank ermöglicht es, eine Vielzahl an Daten abzuspeichern und verschiedene komplexe Beziehungen zwischen den Daten zu definieren. Nach Geisler ist eine Datenbank ein verteiltes, integriertes Computersystem, das im Wesentlichen aus Nutzdaten, welche die Benutzer anlegen, löschen oder verschieben können, und aus Metadaten, welche dafür zuständig sind, die Nutzdaten zu strukturieren, besteht. In einer Tabelle wären z.B. die Metadaten die Überschriften, die die Inhalte der Tabelle, also die Daten in geordneter Form, darstellen (vgl. Geisler 2014, S. 1).

Damit aus einer Datenbank Informationen gewonnen werden können, müssen die Daten in Zusammenhang gebracht werden. Ein einfaches Szenario aus der Produktion soll den Zusammenhang zwischen Daten und Informationen verdeutlichen. Ein Industrieunternehmen muss für einen Auftrag das Produkt ND1 herstellen. Dabei ist die Produktnummer allein nur ein Datensatz. Sobald die Nummer in ein Computersystem eingegeben wird, erhält man weitere Daten, die im direkten Zusammenhang mit der Produktnummer stehen. Durch die Verknüpfung dieser Daten entstehen Informationen, die Entscheidungen oder Handlungen auslösen können. Da der Computer alle getätigten Aufträge mit der Produktnummer ND1 kennt, kann er weitere Daten über den Verlauf der Produktion beschaffen. Die daraus resultierenden Informationen können genutzt werden, um den Produktionsprozess individuell anzupassen. Das heißt, erst die gewonnen Informationen aus den unterschiedlichen Daten sind das Wichtige und Wertvolle. Setzt man die Daten in einen anderen Zusammenhang, entstehen dadurch auch andere Informationen (vgl. Geisler 2014, S. 1 f.).

Für die Verwaltung einer Datenbank wird ein Datenbankmanagementsystem (DBMS) benötigt. Ein DBMS ist dafür zuständig, eine Datenbank zu erstellen, zu organisieren und zu pflegen. Dabei fungiert das DBMS als eine Schnittstelle zwischen dem Anwender und der Datenbank. Formuliert der Anwender eine Abfrage an die Datenbank, wird diese zum DBMS weitergeleitet. Das DBMS wertet die Abfrage aus, sucht die benötigten Daten aus der Datenbank heraus und leitet sie zurück an den Anwender (siehe Abb. 6). So können Daten leicht zur Datenbank hinzugefügt, gelöscht oder geändert werden. Die leistungsstarke Suchfunktion eines

DBMS ermöglicht es, Daten schnell in einer Datenbank zu finden. Datenbanken arbeiten somit hauptsächlich im Hintergrund. Häufig wird in der Praxis der Begriff Datenbank anstatt DBMS verwendet, obwohl es sich in den meisten Fällen um ein DBMS handelt (z.B. Oracle und Access). Datenbankmanagementsysteme sind heutzutage ein wesentlicher Bestandteil der Informationsgesellschaft und zwingend erforderlich, um aus Daten Informationen zu formen. (vgl. Geisler 2014, S. 4 f.).

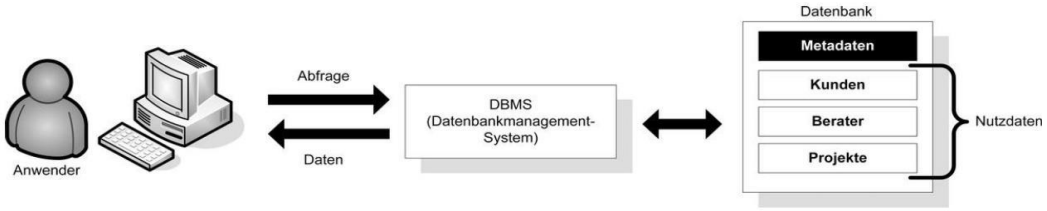


Abbildung 6: Funktion eines DBMS zwischen Anwender und Datenbank (entnommen aus Geisler 2014, S. 4)

Damit ein DBMS optimal eingesetzt werden kann, ist es wichtig bei der Entwicklung einer Datenbank die verwendeten Datenstrukturen an das DBMS anzupassen. Das Design, welches auch als Datenbankmodell bezeichnet wird, muss logisch und stimmig sein, um eine Datenbank effektiv nutzen zu können. Dabei gibt ein Datenbankmodell vor, auf welche Art und Weise die Daten zu speichern sind. Datenbankmodelle können anhand deren grundlegenden Struktur in folgende unterschiedliche Kategorien eingeteilt werden (vgl. Geisler 2014, S. 32 ff.):

- Hierarchische Datenbanken
- Netzwerk-Datenbanken
- Relationale Datenbanken
- NoSQL Datenbanken

Ein übersichtliche Darstellung einiger heute verfügbarer DBMS verschiedener Hersteller und die grundlegenden Datenbankmodelle sind in Tabelle 3 abgebildet.

DBMS	Hersteller	Modell
Casandra	Apache	NoSQL (Column-Family-Systeme)
MongoDB	MongoDB	NoSQL (Document Stores)
MS Access	Microsoft	relational
MySQL	Oracle	relational
SQLite	Entwicklerteam	relational
Visual FoxPro	Microsoft	relational

Tabelle 3: Überblick verschiedener DBMS mit deren Hersteller und Modell (in Anlehnung an: Kudraß und Brinkhoff 2015, S. 26)

Anhand Tabelle 3 ist zu sehen, dass die in der industriellen Praxis am meisten eingesetzten Datenbanken auf einem relationalen DBMS (RDBMS) beruhen. Somit werden diese im folgenden Abschnitt spezifischer betrachtet.

3.1 Relationale Datenbanken

Das Konzept der relationalen Datenbank basiert auf einer zweidimensionalen Wertetabelle, bestehend aus Zeilen und Spalten. Dabei legt die Tabelle die grundlegende Struktur der Daten fest und wird somit auch als Relation bezeichnet. Mehrere Relationen bilden eine Sammlung von Tabellen. Während die Spalten der Tabelle Attribute bzw. Datenfelder repräsentieren, stellen die Zeilen Datensätze dar (siehe Abb. 7). Die auch als Tupel bezeichneten Datensätze, bestehen aus einer Reihe von Attributwerten. Das Relationsschema bestimmt die Anzahl und den Typ der Attribute für eine Relation (vgl. Kudraß und Brinkhoff 2015, S. 67 ff.).

Um Beziehungen zwischen mehreren Tabellen herstellen zu können, werden sogenannte Primär-/Fremdschlüsselbeziehungen benötigt. Mit einem Primärschlüssel, der ein Attribut einer Tabelle A ist, wird jeder Datensatz genau identifiziert. Relationale Datenbanken sind darauf ausgelegt, nur eine einzelne Zeile mit einem bestimmten Primärschlüsselwert in einer Tabelle zuzulassen, um eine Eindeutigkeit zu erzwingen. Ein Fremdschlüssel ist ebenfalls ein Attribut in der Tabelle A, deren Werte den Werten des Primärschlüssels aber in einer Tabelle B entsprechen (vgl. Geisler 2014, S. 39 f.).

In Abbildung 7 ist ein vereinfachtes Szenario dargestellt, welches den Umgang mit eintreffenden Materialen und deren Weiterverarbeitung in einem Unternehmen beschreibt. Ziel ist es den Zusammenhang zwischen Primär- und Fremdschlüssel zu verdeutlichen.

Beim Eintreffen einer Lieferung werden die Materialien in einer Datenbank mit den dazugehörigen Informationen gespeichert. Für die Weiterverarbeitung des Materials soll klar definiert werden, welche Presse die geeigneten Eigenschaften aufweist, um eine optimale Bearbeitung zu erzielen. Die einzelnen Pressen sind in einer weiteren Tabelle genau gekennzeichnet und mit zusätzlichen Attributen beschrieben. Um genau festzulegen, welche Maschine das Material weiterverarbeiten soll, muss eine Beziehung zwischen der Tabelle des Materials und der Tabelle der Maschinen/Pressen hergestellt werden. Dabei wird der Primärschlüssel aus der Tabelle der Maschinen, in die Tabelle des Materials überführt (siehe grünen Pfeil). Der überführte Primärschlüssel ist in der Tabelle des Materials nun ein Fremdschlüssel (siehe grünmarkierte Umrandung). Bei Eintritt einer Lieferung im Unternehmen, muss nun die für die Weiterverarbeitung zuständige Presse angegeben werden. Bei der fehlenden Auswahl der Pressen, erzeugt die Software eine Hinweismeldung, die dem Benutzer signalisiert, dass noch keine Presse ausgewählt wurde.

Die im zuvor beschriebenen Szenario eingesetzte Beziehung nennt man 1:N-Beziehung. Ein Datensatz aus der Tabelle Material steht mit mehreren Datensätzen der Tabelle Maschine in Beziehung. Des Weiteren gibt es noch M:N und 1:1 Beziehungen.

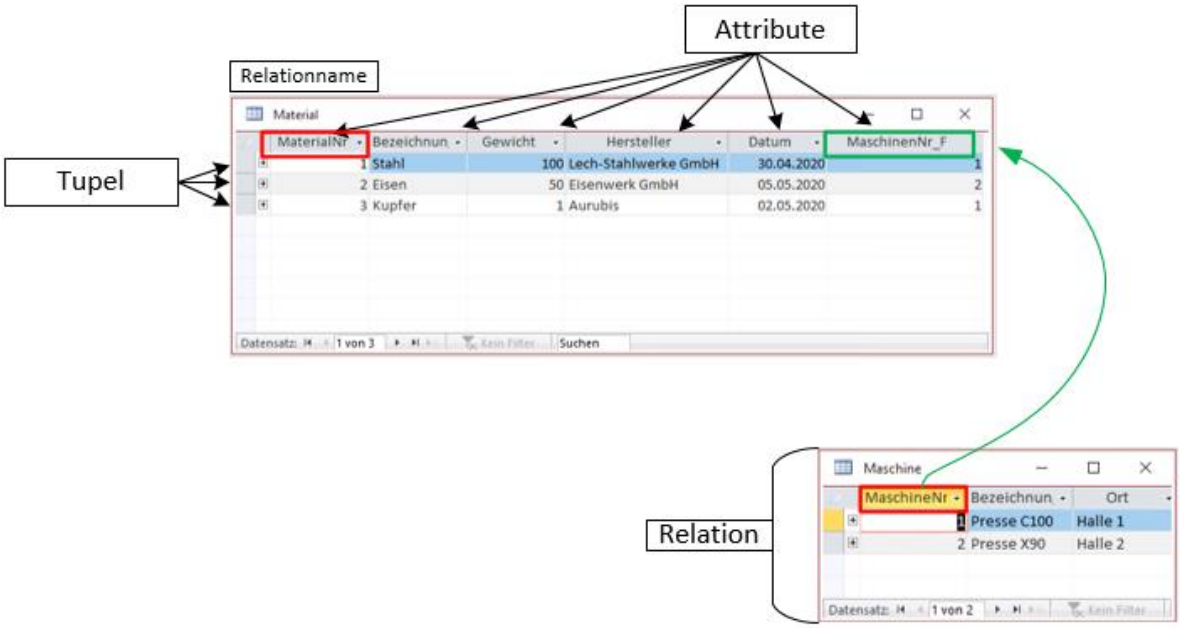


Abbildung 7: Beispielszenario und grundlegendes Konzept eines relationalen Datenmodells

Der wesentliche Vorteil einer relationalen Datenbank ist die strukturelle Unabhängigkeit. Das heißt, es ist ein direkter Datenzugriff möglich, ohne davor durch verzweigte Strukturen navigieren zu müssen. Dadurch können relationale Datenbanken einfach entworfen und verwaltet werden. Durch die Unterstützung von Ad-hoc-Abfragen, die bestimmte Abfragen direkt an das Datenbanksystem weitergeben können, ohne dass der Anwender umfassende Strukturkenntnisse der Datenbank benötigt, wird die Flexibilität einer relationalen Datenbank verdeutlicht. Mit der Abfragesprache Structured Query Language (SQL) ist es unter anderem möglich, die in der Datenbank gespeicherten Daten zu verändern oder umfassende Informationsabfragen durchzuführen. Da das Konzept der relationalen Datenbank gegenüber anderer Datenbanken viel einfacher und übersichtlicher gestaltet ist, hat es im Laufe der Zeit eine dominierende Stellung im Datenbanksektor eingenommen (vgl. Geisler 2014, S. 41 ff.).

Besonders hervorzuheben ist die relationale Datenbank SQLite, welches das verbreitetste und meistverwendete Datenbanksystem der Welt ist. Vor allem in Mobiltelefonbetriebssystemen und in Computern findet SQLite Anwendung. Der wesentliche Unterschied zu herkömmlichen relationalen Datenbanken ist, dass das Datenbanksystem vollständig in eine An-

wendung integrierbar ist und somit auf keine laufenden Datenbankserveranwendungen angewiesen ist. Dabei besteht eine SQLite-Datenbank nur aus einer einzelnen Datei, die alle Tabellen beinhaltet. Das hat zur Folge, dass ein vereinfachter Datenaustausch zwischen unterschiedlichen Systemen möglich ist. Ist man jedoch auf die volle Funktionalität der SQL-Befehle angewiesen, ist von einer SQLite-Datenbank abzuraten, da es sich hier um eine „Lite"-Version für SQL handelt (vgl. sqlite 2020).

Allgemein weisen RDBMS in Bezug auf die stetig wachsenden Datenmengen, die heutzutage von einem Datenbanksystem bewältigt werden müssen, einen großen Nachteil auf. Denn ein RDBMS benötigt bei großen Datenmengen eine wesentlich höhere Prozessleistung als andere DBMS. Somit können mit zunehmenden Datenvolumen die Hardwarekosten rasant steigen (vgl. Kudraß und Brinkhoff 2015, S. 368).

Für die erfolgreiche Bewältigung von großen Datenmengen, die sich teilweise im Bereich von mehreren hundert Petabyte bewegen, eignen sich beispielsweise NoSQL-Datenbanken.

3.2 NoSQL-Datenbanken

Die NoSQL-Datenbank (Not only SQL) ist ein Datenbanksystem der neuen Generation, mit dem Ziel, sehr große Datenmengen beherrschen zu können. Dabei ist die wesentliche Eigenschaft einer NoSQL-Datenbank, dass das zugrundeliegende Datenmodell nicht relational ist, sondern die Systeme auf eine verteilte und horizontale Skalierbarkeit (Scale-out) ausgerichtet sind (siehe Abb. 8). Eine horizontale Skalierbarkeit ermöglicht das Einfügen von zusätzlichen dynamischen Rechenknoten, die einen Teil der Datenlast tragen können und somit zu einer Leistungserhöhung des Systems beitragen. Dagegen wird bei einer klassischen vertikalen Skalierung (Scale-up), die vor allem bei relationalen Datenbanken Anwendung findet, eine Leistungssteigerung durch das Hinzufügen von mehr Ressourcen zu einem System ermöglicht. Das heißt, mit immer größer werdenden Datenmengen nehmen die dadurch entstehenden Hardwarekosten exponentiell zu. Zudem ist das Maß der Skalierbarkeit begrenzt, sodass relationale Datenbanken enorme Probleme haben, mit „BigData"-Anwendungen erfolgreich umgehen zu können (vgl. Kudraß und Brinkhoff 2015, S. 369 f.) (vgl. Edlich 2011, S. 3).

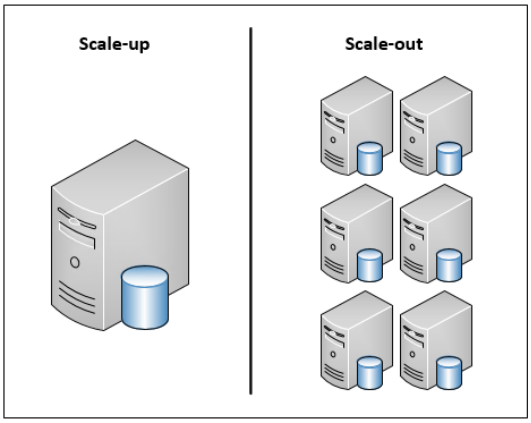


Abbildung 8: Grafische Darstellung der Skalierungsprinzipien

Nicht nur große Firmen wie Yahoo, Amazon oder Facebook verwenden unterschiedliche Arten von NoSQL-Datenbanken, sondern auch vermehrt Industrieunternehmen, die für eine erfolgreiche Gestaltung ihrer Produktionsprozesse Unmengen an Daten analysieren und auswerten müssen. Die heutzutage klassischen NoSQL-Systeme sind vor allem CouchDB, Cassandra, MongoDB, Redis und HBase/Hypertable. Diese können in unterschiedliche Systemkategorien eingeteilt werden (vgl. Edlich 2011, S. 1 ff.).

- Key/Value-Systeme (Redis)
- Column-Family-Systeme (HBase, Hypertable und Cassandra)
- Document Stores (MongoDB, CouchDB)
- Graphdatenbanken (SonseDB)

Bezogen auf die in dieser Bachelorarbeit betrachteten Datenübertragungen, sind die Datenbanksysteme der *Document Stores* von besonderer Relevanz, da die Werkzeugmaschine kontinuierlich Daten an eine speziell eingerichtete MongoDB übermittelt. Deshalb wird im folgenden Abschnitt der Aufbau und das Verhalten von dokumentorientierten Datenbanken näher erläutert.

3.2.1 Dokumentorientierte Datenbanken

Bei dokumentorientierten Datenbanken werden im Vergleich zu relationalen Datenbanken, die Daten nicht in einer Datenbanktabelle abgespeichert, sondern in einzelnen Dokumenten. Das heißt, ein Dokument ist vergleichbar mit einer Zeile in einer Datenbanktabelle. Dabei sind die verwendeten Dokumente für die Speicherung und den Datenaustausch keine unstrukturierten Textdokumente, sondern strukturierte Datenformate wie beispielsweise JSON oder YAML. Hierdurch wird das Sichern von komplexen Datenstrukturen ermöglicht. Da die zu speichernden Dokumente kein spezielles Schema benötigen, können unterschiedliche Arten von

Dokumenten in einer Datenbank aufgenommen werden. Um Datenanfragen effizient durchzuführen, lohnt es sich ähnlich strukturierte Dokumente in einer Kollektion (collection) zu speichern. Eine Kollektion sammelt Dokumente und ist vergleichbar mit einer Tabelle einer relationalen Datenbank. Sie zeichnet sich dadurch aus, dass sie im Gegensatz zu einer Tabelle, völlig unterschiedlich aufgebaute Dokumente aufnehmen kann. Bei wachsenden Datenvolumen lassen sich mithilfe einer automatischen Partitionierung der Datenbank, die Daten in einer Kollektion auf mehrere Systeme verteilen, um eine horizontale Skalierbarkeit (vgl. Abschnitt 3.2) zu erreichen. Somit sind dokumentorientierte Datenbanksysteme darauf ausgelegt, sehr große Mengen an Daten zu verarbeiten und finden häufig Verwendung in den Bereichen des Content Managements, E-Commerce-Anwendungen aber auch im Bereich der Real-Time-Analyse (vgl. Kudraß und Brinkhoff 2015, S. 373 ff.).

3.2.2 MongoDB

Die MongoDB ist eine dokumentorientierte Datenbank, die in der Programmiersprache C++ geschrieben ist und als Open Source zur Verfügung steht. Der Name setzt sich aus dem englischen Wort *humongous* zusammen und bedeutet übersetzt gigantisch.

Ziel dieser Datenbank ist es, möglichst kurze Reaktionszeiten auch bei großen Datenmengen zu generieren. Neben den allgemeinen Features einer dokumentorientierten Datenbank, bietet MongoDB weitere nützliche Funktionen an, wie beispielsweise die automatische Reproduktion aller unbeendeten Transaktionen nach einem Crash des Systems. Der Einsatzbereich von MongoDB erstreckt sich über eine Vielzahl von Plattformen. Je nach Betriebssystem stehen für die Plattformen Linux, Windows, MacOS X sowie Solaris spezielle Versionen für die Anwendung zur Verfügung. Nach Edlich liegen die wesentlichen Vorteile einer MongoDB vor allem in folgenden Punkten (vgl. Edlich 2011, S. 132 ff.):

- Gute Performance durch eine effiziente Nutzung des Arbeitsspeichers
- Durch eine horizontale Skalierung der Daten besonders für große Datenvolumen geeignet
- Formulierungen von dynamischen Abfragen sind möglich
- Sehr viele Anbindungsmöglichkeiten durch die Bereitstellung von Treibern für eine Vielzahl von Sprachen

Für die Verwaltung eines laufenden MongoDB-Servers bietet MongoDB verschiedene Wege an. Zum einen können Benutzer mittels Mongo Shell Lese- sowie Schreiboperationen durchführen. Dabei ist Mongo Shell ein Kommandozeilen-Client, mit dem man in der Eingabeaufforderung Befehle in der Sprache JavaScript ausführen kann. Zum anderen ist MongoDB mit

unterschiedlichen Treibern ausgestattet, sodass eine Verwaltung auch mit den offiziellen Programmiersprachen möglich ist. Möchte man die Daten in einer grafischen Oberfläche bearbeiten, stehen hier ebenfalls einige Programme, wie z.B. MongoDB Compass zur Verfügung.

In Abbildung 9 sind die Datenbanken für die Werkzeugmaschine des Digital Labs grafisch in MongoDB Compass dargestellt. Am linken Rand des Fensters, befinden sich allgemeine Informationen zur Datenbankverbindung und eine Auflistung der vorhandenen Datenbanken. Nachdem eine Datenbank ausgewählt wurde, erscheint im mittleren Bereich des Fensters eine Übersicht aller verfügbaren *collections* mit den dazugehörigen spezifischen Angaben. In diesem Teil des Fensters ist eine Bearbeitung der Datenbank möglich. Wird eine *collection* geöffnet, erzeugt MongoDB Compass alle in dieser *collection* befindlichen Dokumente mit den einzelnen Attributen, die anschließend individuell verändert werden können.

Das Tool bietet die Möglichkeit, Anpassungen einer Datenbank auch von Benutzern ohne spezifisches Fachwissen einer Programmiersprache vornehmen zu können. Es ist zu beachten, dass bei Fehlhandlungen wichtige Daten der Datenbank auf Dauer gelöscht werden können, sodass nur Personen mit der Datenbank arbeiten sollten, die ein gewisses Fachwissen vorweisen können.

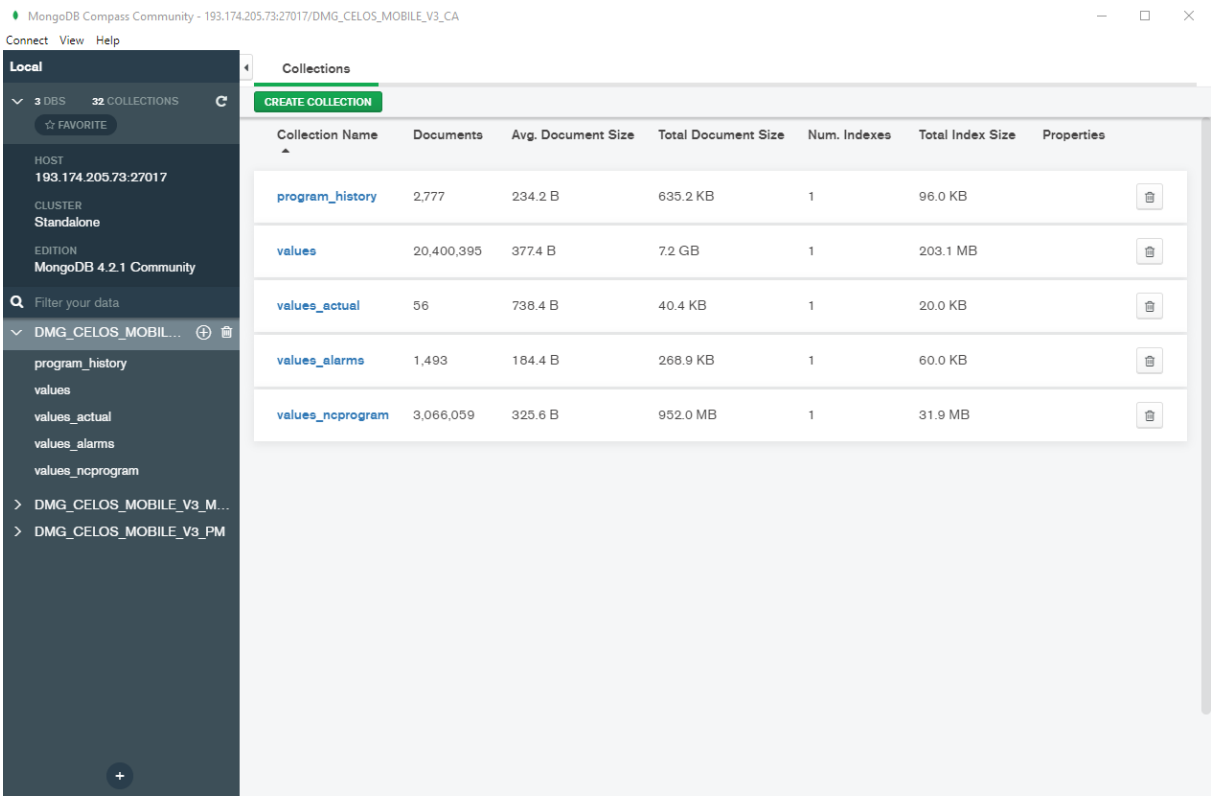


Abbildung 9: Grafische Darstellung des MongoDB Compass Tools mit einer Datenbank für eine Werkzeugmaschine

Damit Business Intelligence-Anwendungen wie beispielsweise Excel, Tableau oder Qlik, die SQL als Abfragesprache benutzen, auch auf Daten einer MongoDB zugreifen können, wurde der MongoDB Connector for BI (Business Intelligence) entwickelt. Dieser ermöglicht es, ein relationales Schema darzustellen und SQL-Abfragen in die speziell von MongoDB genutzte Abfragesprache MongoDB Query Language (MQL) umzuwandeln. Anschließend wird die umgewandelte Abfrage an die Datenbank weitergeleitet und die benötigten Daten an das Anwendungsprogramm zurückgesendet. Ein vollständiger Verbindungsprozess von einem Anwendungsprogramm zu einer Datenbank, ist in Abbildung 10 dargestellt. Dabei beinhaltet ein sogenanntes BI-System folgende Komponenten (vgl. MongoDB 2020, S. 1):

- MongoDB Datenbank: Lagert alle benötigten Daten
- BI Connector: Bietet ein relationales Schema und übersetzt SQL-Abfragen zwischen einem BI-Tool und der MongoDB
- ODBC Data Source Name (DSN): Sorgt für die Authentifizierung und beinhaltet spezielle Verbindungskonfigurationen
- BI-Werkzeug: Datenvisualisierung und -analyse.

Je nachdem ob sich eine Datenbank lokal oder auf einem Netzwerk befindet, sind unterschiedliche Vorgehensweisen nötig, um eine Verbindung zwischen Datenbank und Anwendungsprogramm herzustellen. In Kapitel 5 wird eine spezielle Methode beschrieben, die eine auf einem Netzwerk befindliche MongoDB mit einer Simulationssoftware über einen BI-Connector verbindet und die Daten für eine Simulation verwendet.

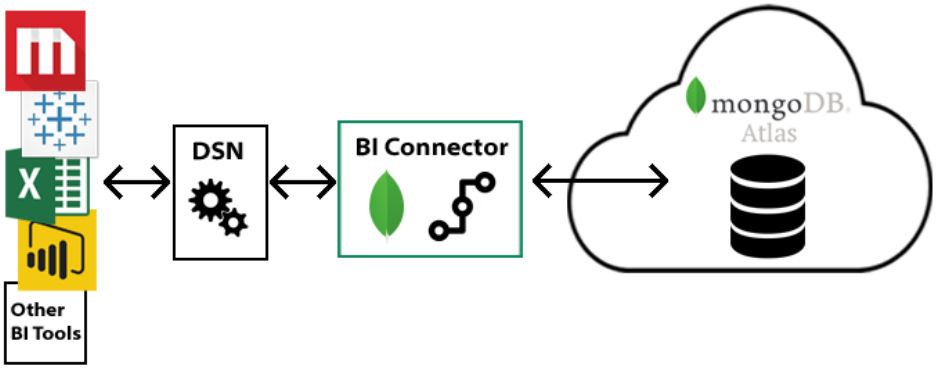


Abbildung 10: Abbildung eines kompletten BI-Systems mit den dazugehörigen Komponenten (entnommen aus MongoDB 2020, S. 1)

4 Vorstellung der eingesetzten Schnittstellen zur Datenanbindung

Für die erfolgreiche Umsetzung eines Digitalisierungsprozesses innerhalb eines Industrieunternehmens ist es essenziell, Ablaufketten zu vernetzen und eine Kommunikation zwischen unterschiedlichen Systemen zu ermöglichen. Dadurch können wertvolle Daten gewonnen, analysiert und anschließend zur Effizienzsteigerung einer Produktion genutzt werden. Um Datenstrukturen eines Systems abzufragen, auszulesen und an weitere Geräte zu übermitteln, sind Schnittstellen zwischen den kommunizierenden Systemen erforderlich. Dabei gibt es je nach Anwendung eine Vielzahl an Schnittstellen. In den folgenden Abschnitten werden zwei Schnittstellen explizit beschrieben, die für einen Datenaustausch zwischen einer Werkzeugmaschine und einer Simulationssoftware in Frage kommen. Dabei handelt es sich zum Einen um eine ODBC-Schnittstelle, die vor allem für Datenbankabfragen geeignet ist, indem sie eine Verbindung zwischen dem Anwendungsprogramm und einer Datenbank herstellt und zum anderen um eine OPC UA-Schnittstelle, die einen Standard für eine plattformunabhängige Kommunikation zwischen mehreren Systemen darstellt und Regelgrößen, Messwerte, Parameter usw. nicht nur transportiert, sondern auch maschinenlesbar semantisch beschreibt.

4.1 Die ODBC-Schnittstelle

Die Open Database Connectivity kurz ODBC ist eine von Microsoft entwickelte einheitliche Programmierschnittstelle, die sich auf datenbankspezifische Treiber stützt, um mit einer Vielzahl verschiedener Datenbanksysteme zu arbeiten. Sie bietet die Möglichkeit unabhängig des Datenbanktyps identische Funktionen für Datenabfragen oder -veränderungen zur Verfügung zu stellen. Dabei fungiert eine Datenbankschnittstelle im Allgemeinen als eine Art Zwischenschicht, die als Kopplung zwischen Datenbank und Anwendungsprogramm dient. Der Vorteil bei dieser von Geisler benannten Middleware liegt darin, dass man in der Anwendung nur über einen logischen Namen auf die Datenbank zugreift und bei der Programmierung einer Anwendung nicht auf die Implementierung einzelner Datenbanksprachen achten muss. Findet eine Veränderung an der eigentlichen Datenbank statt, so muss dies lediglich an der Zwischenschicht angepasst werden. Das eigentliche Anwendungsprogramm bleibt von der Veränderung unberührt. Eine neue Komplementierung des Anwendungsprogramms ist somit nicht erforderlich (vgl. Geisler 2014, S. 53 f.).

Aufbau und Funktionsweise einer ODBC-Schnittstelle werden anhand einer vereinfachten Architektur aus Abbildung 11 im Folgenden kurz erläutert:

ODBC-Anwendung

Die ODBC-Anwendung ist das Programm (z.B. Plant Simulation), das der Benutzer bedient und ODBC-Befehle aufruft, um SQL-Befehle an die Datenbank zu senden und Ergebnisse zurückzubekommen (vgl. Geisler 2014, S. 55).

ODBC-Treiber-Manager

Der Treiber-Manager lokalisiert anhand des zuvor festgelegten Data Source Name (DSN) den zuständigen Treiber und lädt die ODBC-Treiber. Im Anschluss daran, leitet er die Daten an den richtigen ODBC-Treiber weiter (vgl. Luber 2017, S. 1).

ODBC-Treiber

Der ODBC-Treiber selbst führt ODBC-Funktionen aus, übermittelt SQL-Anfragen an die spezifische Datenbank und liefert das Ergebnis an die ODBC-Anwendung zurück (vgl. Geisler 2014, S. 55).

Bezogen auf eine MongoDB übermittelt der ODBC-Treiber die SQL-Anfragen an den MongoDB Connector for BI. Anschließend verarbeitet dieser die SQL-Anfrage weiter (siehe Abschnitt 3.2.2) und sendet die erforderlichen Daten an den ODBC-Treiber zurück.

DBMS

Das DBMS beinhaltet die Daten, auf die der Benutzer der ODBC-Anwendung zugreifen möchte (vgl. Geisler 2014, S. 55).

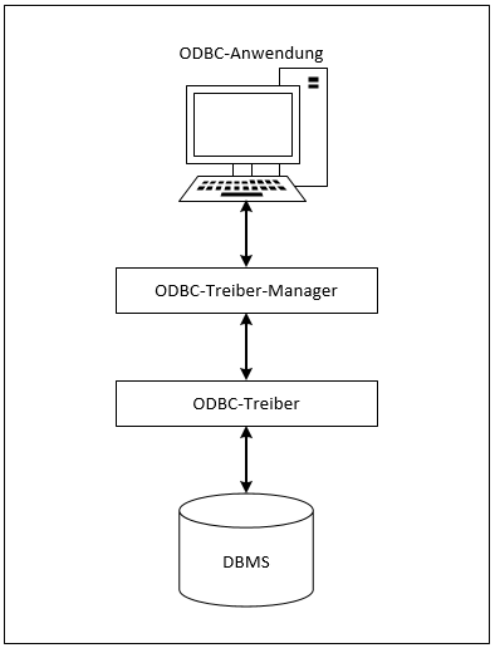


Abbildung 11: Architektur einer ODBC-Schnittstelle

Die Möglichkeit auf eine Vielzahl an unterschiedliche Datenbanktypen zugreifen zu können, bietet nicht nur Vorteile. Denn die Einbindung von Open Database Connectivity mit seinen Komponenten und Treibern kann unter Umständen zu längeren Antwortzeiten bei der Ausführung von Datenbankaktionen führen. Trotzdem ist eine ODBC-Schnittstelle aufgrund ihrer Standardisierung und somit resultierenden Flexibilität ein äußerst nützliches Tool, um Datenbanken verschiedenster Art mit Anwendungen zu verbinden (vgl. Luber 2017, S. 2).

4.2 Die OPC UA-Schnittstelle

Open Platform Communications United Architecutre (OPC UA), ist ein plattformunabhängiger Kommunikationsstandard für den industriellen Datenaustausch zwischen unterschiedlichen Maschinen eines Unternehmens. Ziel von OPC UA ist es, Maschinen und Anlagen auf allen Ebenen von verschiedenen Herstellern zu vernetzen und dadurch einen Standard für die Kommunikation zwischen Maschinen wie auch Herstellern von PLM- und ERP-Systemen zu definieren. Ein zentrales Konzept ist dabei, dass die vom Server zur Verfügung gestellten Daten über eine Art Verzeichnisstruktur aufgelistet werden. OPC UA kann Maschinendaten wie beispielsweise Regelgrößen, Messwerte, Parameter usw. nicht nur transportieren, sondern auch maschinenlesbar semantisch beschreiben (vgl. Plenk 2017, S. 93 f.).

Die OPC UA Infrastruktur basiert auf dem Konzept des Client-Server-Modells. Das Client-Server-Modell beschreibt eine Möglichkeit zur Verteilung von Aufgaben innerhalb eines Netzwerks. Die Aufgaben werden mithilfe von Clients und Servern erledigt. Der Client kann auf Wunsch einen Dienst vom Server anfordern. Der Server beantwortet die Anforderung und stellt dem Client einen Zugang zum angeforderten Dienst zur Verfügung. Dabei ist jeder Server in der Lage gleichzeitig mit mehreren Clients zu interagieren. Jeder Client kann mit mehreren Servern verbunden sein. Sobald der Client eine Verbindung über die spezifische IP-Adresse des OPC UA-Servers hergestellt hat, können gezielt Informationen der Maschine oder Anlage abgefragt werden. Die Hauptkomponente eines OPC UA-Servers ist der *AddressSpace.* Innerhalb des *AddressSpace* werden in einer objektorientierten Struktur verschiedene *Nodes* (Knoten) vom Server bereitgestellt. *Nodes* repräsentieren unter anderem reale Objekte einer Fertigungsumgebung und können beispielsweise verwendet werden, um Sensordaten, Messwerte oder spezifische Informationen einer Anlage abzubilden (vgl. Hoffmann 2019, S. 92 ff.).

In Abbildung 12 ist die Grundstruktur eines OPC UA *AddressSpace* abgebildet. Der *Root* Ordner bildet den Einstiegspunkt in den *AddressSpace*. Der Ordner *Types* beinhaltet alle Definitionen zu bestehenden oder benutzerdefinierten *Nodes*. Innerhalb des Ordners *Objects* sind zuvor definierte *Nodes* abgebildet.

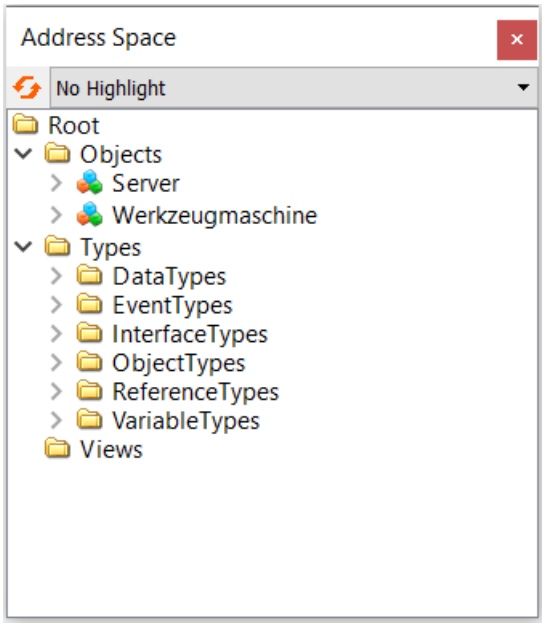


Abbildung 12: AddressSpace eines simulierten OPC UA-Servers

Um *Nodes* eindeutig zu identifizieren, besitzt jede *Node* eine *NodeID*. Eine *NodeID* setzt sich aus dem jeweiligen *NamespaceIndex* und einer zusätzlichen Identifikation in Form eines *Strings* oder eines *Integers* zusammen. Der *Namespace* dient dazu, *Nodes* anhand spezifischer Merkmale zu gruppieren. In Abbildung 13 sind die Attribute einer *Node* grafisch dargestellt. Die *Node* besitzt folgende *NodeID*: ns=2;i=4. Dabei steht *ns* für den Namespace und *i* für den zusätzlichen Identifikator, der in diesem Fall aus einem *Integer* besteht.

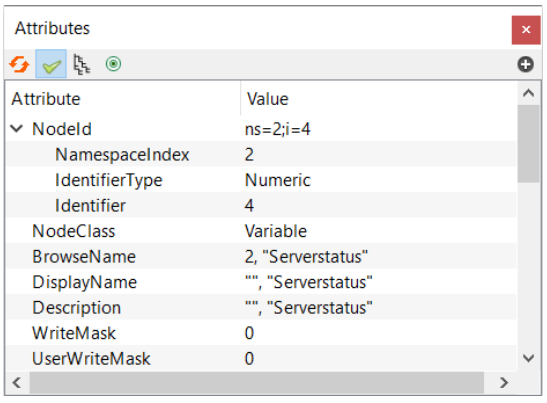


Abbildung 13: Grafische Darstellung der Attribute einer Node

Eine OPC UA-Schnittstelle bietet nicht nur die Möglichkeit Daten in Echtzeit abzurufen, sondern diese auch gezielt zu verändern. Somit können Anwendungen wie beispielsweise Simulationssoftwareprogramme einerseits reale Prozesse digital abbilden und laufend analysieren, andererseits kann bewusst Einfluss auf den echten Prozess genommen werden. Inwiefern sich eine OPC UA-Schnittstelle in Plant Simulation eignet, wird in Abschnitt 5.3 untersucht.

5 Durchführung zweier Datenanbindungsmethoden für eine Werkzeugmaschine an Plant Simulation

Dieses Kapitel beinhaltet den Hauptteil dieser Arbeit und untersucht, inwiefern es möglich ist eine real operierende 5-Achs-Fräsmaschine der Forschungseinrichtung des Digital Labs digital abzubilden und in eine Fabriksimulation zu implementieren. Dazu werden mithilfe der im vorherigen Kapitel beschriebenen Datenschnittstellen zwei unterschiedliche Methoden vorgestellt und angewandt. Die erste Methode greift auf eine Datenbank zu und importiert mithilfe einer ODBC-Schnittstelle die Daten nach Plant Simulation. Im Anschluss daran findet eine Auswertung und Filterung der Daten in Plant Simulation statt, sodass abschließend eine Integration der Daten in ein erzeugtes Simulationsmodell durchgeführt werden kann. Die zweite Methode versucht über einen OPC UA-Server und mit der Verwendung einer OPC UA-Schnittstelle die Werkzeugmaschine in Echtzeit an das Simulationsprogramm anzubinden und eine parallel ablaufende digitale Bearbeitung der Werkstücke zu erzeugen. Innerhalb einer Methode werden zunächst alle Vorarbeiten beschrieben die nötig sind, um eine Datenanbindung an Plant Simulation zu ermöglichen. Anschließend erfolgen die Beschreibung und die Modellierung geeigneter Simulationsmodelle. Zum Abschluss werden die jeweiligen Methoden ausgewertet und beurteilt.

5.1 Datenanbindung mithilfe einer ODBC-Schnittstelle

Die in Abschnitt 4.1 beschriebene ODBC-Schnittstelle eignet sich vor allem für die Erstellung einer Verbindung zwischen einer Datenbank und einem Anwendungsprogramm. Da die betrachtete Werkzeugmaschine kontinuierlich Daten an eine speziell eingerichtete MongoDB sendet, bietet es sich an, die Simulationssoftware Plant Simulation mit dieser Datenbank zu verbinden und anschließend die Daten in einem Simulationsmodell zu verwenden. Um dies zu ermöglichen, muss zunächst eine Verbindung vom Bedienplatz zur Datenbank hergestellt werden. Anschließend ist eine Konfiguration einer ODBC-Schnittstelle notwendig, damit Plant Simulation über die eingerichtete Schnittstelle Datenabfragen an der MongoDB durchführen kann. Sobald eine erfolgreiche Übertragung der Daten stattgefunden hat, kann in einem geeigneten Simulationsmodell untersucht werden, ob eine sinnvolle Verwendung der Daten möglich ist. In den folgenden Abschnitten werden die Schritte von der Herstellung einer Datenbankverbindung bis hin zur Bewertung des Simulationsmodells detailliert beschrieben. Dabei wurden alle Aktivitäten auf einem x64-bit Windows Betriebssystem durchgeführt. Je nach Betriebssystem kann die Vorgehensweise zur Datenanbindung individuell abweichen.

5.1.1 Herstellung der Datenbankverbindung

Damit Plant Simulation Zugriff auf eine MongoDB erhalten kann, muss zunächst eine Verbindung zwischen der Datenbank und dem Bedienplatz hergestellt und anschließend eine ODBC-Datenquelle konfiguriert werden. Dazu sind folgende Komponenten nötig:

- MongoDB
- MongoDB Connector for BI
- ODBC Data Source Name (DSN)

Die eigentliche MongoDB läuft auf einem Computer, der mit der Maschine verbunden ist. Darauf läuft ein Programm, das die Daten per MQTT von der Maschine erhält, verarbeitet und in die Datenbank einfügt. MQTT ist ein offenes Netzwerkprotokoll für die Kommunikation zwischen Maschinen und ermöglicht den Transport von Daten. Zusätzlich läuft eine Kopie der MongoDB auf einem Hochschulserver, die täglich die Daten der originalen Datenbank erhält. Bei der Erzeugung der Datenbankverbindung wird nicht auf die originale Datenbank, sondern auf die Kopie der MongoDB zugegriffen.

Der MongoDB Connector for BI kann direkt von der MongoDB-Internetseite heruntergeladen werden. Nach Abschluss der Installation befindet sich der MongoDB Connector in einem bestimmten Verzeichnis auf dem Computer. Der Zugriff auf die Datenbank erfolgt ausschließlich aus den in diesem Verzeichnis befindlichen Dateien, die per Windows-Eingabeaufforderung angesteuert werden. Falls kein bestimmtes Verzeichnis während der Installationsanleitung ausgewählt wurde, befindet sich der MongoDB Connector for BI in folgendem Verzeichnis:

C:\Program Files\MongoDB\Connector for BI\2.13\bin

Mit Hilfe der Windows-Eingabeaufforderung auch cmd.exe genannt, ist es möglich Kommandozeilenbefehle zu verarbeiten. Die cmd.exe wird in dieser Arbeit hauptsächlich verwendet, um mit der MongoDB zu interagieren (vgl. Abbildung 14). Mit dem Befehl

cd C:\Program Files\MongoDB\Connector for BI\2.13\bin

findet eine Navigation in den Ordner des MongoDB Connectors for BI statt. Um eine Verbindung zur Datenbank herzustellen, ist die folgende Kommandozeile notwendig:

mongosqld --mongo-uri mongodb://193.174.205.73:27017 --sampleNamespaces DMG_CELOS_MOBILE_V3_CA. --auth --mongo-username sebastian-schmitt --mongo-password theprinterhasclearlybeendrinking --mongo-authenticationSource DMG_CELOS_MOBILE_V3_CA*

Die Funktionsweise der Kommandozeile setzt sich aus folgenden Befehlen zusammen:

mongosqld

Die *mongosqld* akzeptiert eintreffende SQL-Anfragen und sendet diese weiter an die MongoDB. Anschließend erzeugt sie abfragemögliche Schemata für die ausgewählte Datenbank.

mongo-uri

Der Befehl *mongo-uri* gibt eine Adresse an, zu der eine Verbindung hergestellt werden soll.

sampleNamespaces

Dieser Befehl wählt bestimmte Collections aus und wandelt sie anschließend in ein Schema um, damit SQL-Abfragen möglich sind. Wird dieser Befehl nicht angewendet, wandelt der MongoDB Connector alle vorhandenen Collection in ein Schema um.

auth

Der Befehl *auth* authentifiziert eingehende Clientanforderungen und benötigt zusätzlich die Befehle *mongo-username* und *mongo-password*

mongo-authenticationSource

Der Befehl gibt die Datenbank an, die die Anmeldeinformationen für den Benutzer enthält.

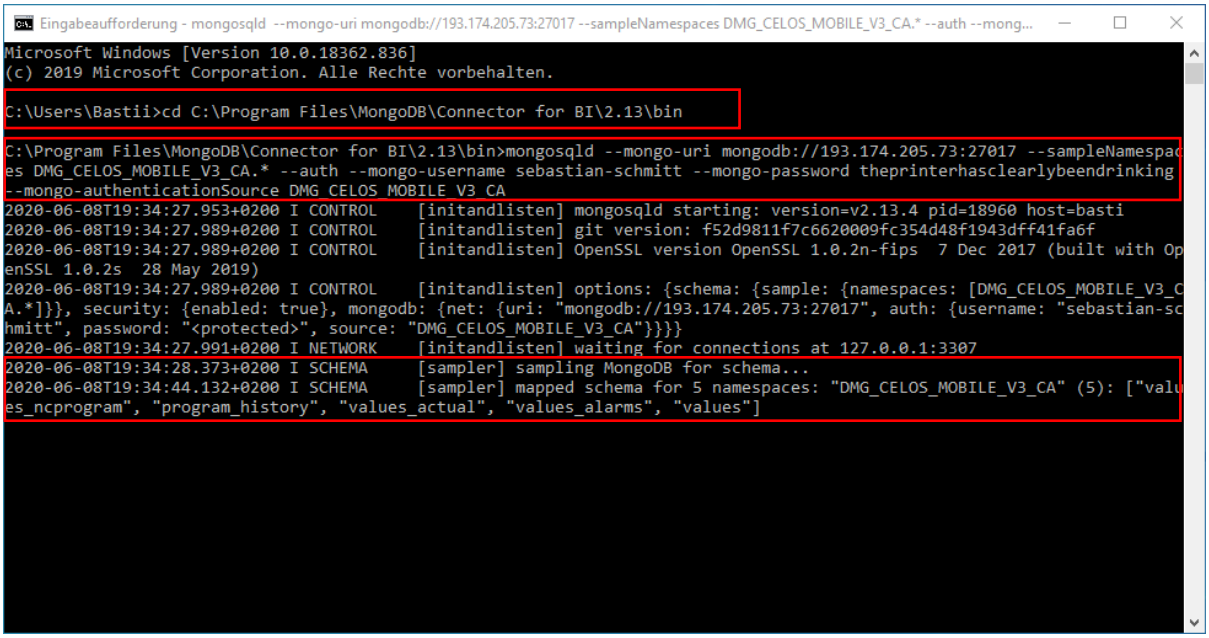


Abbildung 14: Grafische Darstellung eines Verbindungsaufbaus zu einer MongoDB

Nach erfolgreicher Bearbeitung der Befehle durch die Eingabeaufforderung ist eine aktive Verbindung zur MongoDB hergestellt. Bevor Plant Simulation eine SQL-Abfrage an die Datenbank senden kann, ist es notwendig eine ODBC-Datenquelle zu konfigurieren. Unter Windows gibt es jeweils abhängig des Systemtyps eine ODBC-Datenquellen-Administration. Innerhalb dieses Systemtools können ODBC Konfigurationen durchgeführt werden. Im Reiter System-DSN ist es möglich, neue Systemdatenquellen hinzuzufügen (siehe Abbildung 15).

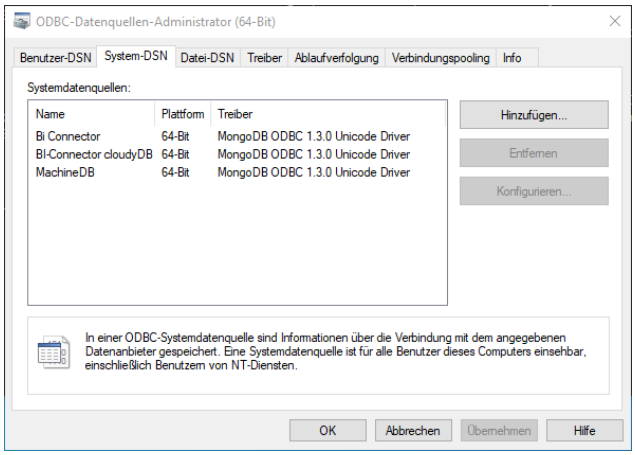


Abbildung 15: ODBC-Datenquellen-Administrator

Innerhalb des Fensters der spezifischen Konfiguration (siehe Abbildung 16), ist die Eingabe unterschiedlicher Informationen erforderlich. Dabei ist auf folgende Punkte zu achten:

1. Die TCP/IP Server-Adresse ist nicht die der MongoDB, sondern die des MongoDB Connectors, da über den MongoDB Connector eine Verbindung zur Datenbank hergestellt wird
2. Der User ist mit der Authentifizierungsdatenbank in einem speziellen Format anzugeben, ansonsten wird die Verbindung blockiert. In diesem Fall besteht der User aus folgender Zusammensetzung*: benutzername?source=Authentifizierungsdatenbank*
3. Mithilfe der Test-Schaltfläche findet eine Überprüfung der Verbindung statt. Sobald die Meldung „Connection Successful“ erscheint, ist die Verbindung ordnungsgemäß eingerichtet

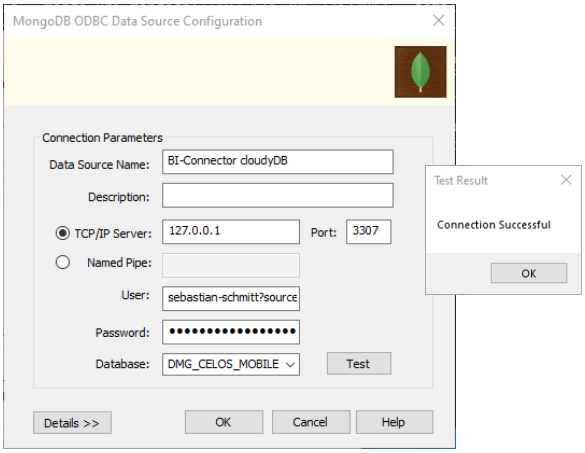


Abbildung 16: MongoDB ODBC Data Source Configuration

Bevor die konfigurierte ODBC-Datenquelle mit Plant Simulation angesteuert und auf die MongoDB zugegriffen werden kann, müssen die Daten in der Datenbank auf Eignung und Kompatibilität untersucht werden. Anschließend findet die Modellierung des Simulationsmodells statt.

5.1.2 Untersuchung der Daten hinsichtlich Eignung und Kompatibilität

Damit gezielte Datenbankabfragen stattfinden können, muss zunächst überlegt werden, welche Daten allgemein für eine Integration in eine Simulation in Frage kommen und wie diese in der Datenbank abgespeichert werden können. Bei der Untersuchung des Verhaltens einer Werkzeugmaschine in einem Fertigungsprozess sind vor allem Bearbeitungszeiten, Rüstzeiten und Störzeiten von entscheidender Bedeutung.

In der MongoDB sind folgende *collections* abgespeichert:

- program_history
- values
- values_actual
- values_alarms
- values_ncprogram

Innerhalb der *collection program_history* befinden sich alle Programme mit einer Startzeit und einer Endzeit, die in der Werkzeugmaschine abgespielt wurden. Unter anderem sind dort unterschiedliche Bearbeitungs-, Einstellungs- und Aufwärmprogramme aufgezeichnet. Alle anderen *collections* zeichnen hauptsächlich Werte auf, welche Kräfte, Positionen und Statusmeldungen der Werkzeugmaschine wiedergeben. Da die Werte der anderen *collections* nur den aktuellen Zeitpunkt aufzeichnen, zu der je ein Ereignis aufgetreten ist, ist es nicht möglich die exakte Dauer der Ereignisse zu bestimmen. Zudem sendet die Werkzeugmaschine keine

Statusmeldungen an die Datenbank, die die Beendigung von Ereignissen beschreiben. Somit können unter anderem Alarmmeldungen nicht genau einem Vorgang zugeordnet werden. Zwar verfügt die Werkzeugmaschine über ein Andonsystem, welches den aktuellen Betriebszustand an die Datenbank sendet, dieses findet aber nach Auswertung aufgrund der erheblichen Inkonsistenz keine Verwendung. Folglich sind nicht alle Daten für die Integration der Werkzeugmaschine in die Simulation verwendbar und geeignet.

5.1.3 Modellierung eines Simulationsmodells in Plant Simulation

In diesem Abschnitt wird ein Simulationsmodell erzeugt, das Werkstücke auf Basis realer Daten der Werkzeugmaschine bearbeitet und anschließend auf einer Arbeitsstation fertigstellt. Ziel ist es, die Daten der Werkzeugmaschine sinnvoll in eine Simulation zu integrieren. Dazu muss zunächst ein geeigneter Datenbereich in der Datenbank der Werkzeugmaschine festgelegt werden, der zu einer fest definierten Anzahl an Werkstücken nützliche Informationen zu Bearbeitungs- und Rüstvorgängen liefert. Bei Betrachtung der Datenbank, bietet sich vor allem der Zeitraum zwischen April und Mai an, in dem fünfzig Werkstücke auf der Werkzeugmaschine bearbeitet wurden. Dabei bestand die Bearbeitung eines Werkstückes aus acht individuellen Bearbeitungsschritten. Diese bilden die Grundlage für die Erstellung des Simulationsmodells.

Das Simulationsmodell aus Abbildung 17 setzt sich aus folgenden Bausteinen zusammen:

- Quelle
- Einzelstation (Werkzeugmaschine)
- Einzelstation (Arbeitsstation)
- Senke

Die Quelle ist für die Produktion der fünfzig Werkstücke verantwortlich und sendet diese nach einander an die Werkzeugmaschine. Auf der Werkzeugmaschine finden mithilfe der Daten aus der Datenbank zum einen unterschiedliche Rüstvorgänge statt und zum anderen individuelle Bearbeitungen der Werkstücke. Nachdem die Bearbeitung der Werkstücke abgeschlossen ist, werden die Werkstücke an die Arbeitsstation übergeben. Die Arbeitsstation ist für die Fertigstellung der Werkstücke verantwortlich und übermittelt die Werkstücke abschließend an die Senke. Die Senke bildet die Endkomponente der Simulation und vernichtet die fertiggestellten Werkstücke wieder.

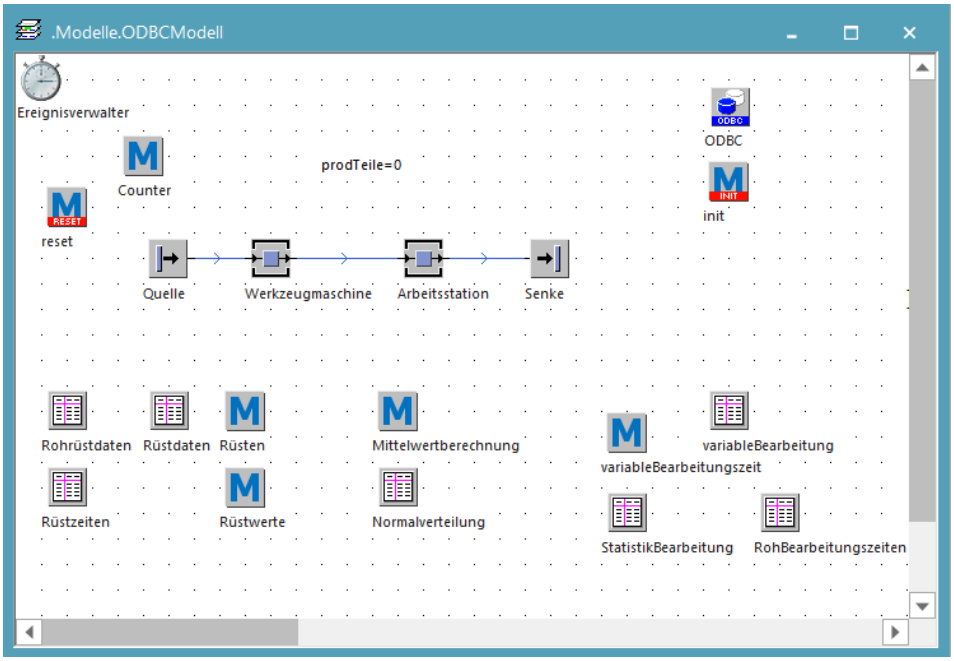


Abbildung 17: Das ODBC-Simulationsmodell

Um die Funktionsweise der Rüst- und Bearbeitungsvorgänge auf der Werkzeugmaschine detailliert beschreiben zu können, wird sie in die folgenden Teilbereiche untergliedert:

- Datenimport
- Erzeugung der spezifischen Bearbeitungszeiten
- Simulation von individuellen Rüstvorgänge

Datenimport

Mithilfe der in Abschnitt 5.1.1 erstellten Datenbankkonfiguration und der *ODBC*-Schnittstelle in Plant Simulation ist es möglich einen grundsätzlichen Datenimport durchzuführen. Dazu stellt die *ODBC*-Schnittstelle eine Verbindung über die konfigurierte ODBC-Datenquelle zum MongoDB-Connector her. Der MongoDB-Connector führt gezielte Abfragen an der MongoDB aus, die ihm mittels einer Methode übergeben werden. Innerhalb dieser Methode (vgl. Abbildung 18) erfolgt zunächst die Ansteuerung des ODBC-Bausteins. Anschließend werden die in Zeile 6-7 ausformulierten SQL-Befehle an den MongoDB-Connector übermittelt. Die Befehle beinhalten eine Anfrage zu allen Bearbeitungs- sowie Rüstdaten, die zwischen April und Mai an der Werkzeugmaschine aufgezeichnet wurden. Der MongoDB-Connector wandelt die Anfrage in MQL um und sucht die gewünschten Daten in der Datenbank heraus. Mit den Anweisungen in Zeile 8-9 werden die Daten abschließend in zwei separate Tabellen namens *RohBearbeitungszeiten* (vgl. Abbildung 19) und *RohRüstzeiten* (vgl. Abbildung 21) gespeichert. Um

zu vermeiden, dass der Simulationslauf mit unvollständigen oder fehlerhaften Daten ausgeführt wird, wird die Methode zu einer Init-Methode umgewandelt, die vor jedem Simulationslauf automatisch startet und eine Datenabfrage durchführt.

Abbildung 18: Das Eingabefenster der Init-Methode

Erzeugung der spezifischen Bearbeitungszeiten

Für die Erzeugung der individuellen Bearbeitungszeiten, müssen die Daten der Tabelle *Roh-Bearbeitungszeiten* (vgl. Abbildung 20) ausgewertet und zusammengefasst werden. Dies erfolgt mithilfe der Methode *variableBearbeitungszeit* (vgl. Abbildung 21). Innerhalb der Methode laufen unterschiedliche Schleifen ab, die anhand der Tabellenspalten: *name, start* und *end* nach jeweils acht Bearbeitungsschritten die Bearbeitungszeit für ein Werkstück berechnen und in einer neuen Tabelle namens *variableBearbeitung* (vgl. Abbildung 19) speichern. Zusätzlich zu den acht Bearbeitungsschritten an einem Werkstück kann es vorkommen, dass einzelne Bearbeitungsvorgänge desselben Typs, aufgrund einer nicht vollständigen oder einer fehlerhaften Bearbeitung in der Werkzeugmaschine, wiederholt wurden. Dies führt zu einem individuellen Anstieg der Bearbeitungszeit und muss ebenfalls in der Methode berücksichtigt werden. Nachdem die Methode vollständig durchlaufen ist, sind fünfzig Werkstücke mit spezifischen Bearbeitungszeiten in der Tabelle *variableBearbeitung* gespeichert. Sobald ein Werkstück auf der Einzelstation eintrifft, wird die benötigte Bearbeitungszeit aus der Tabelle *variableBearbeitung* ausgelesen und an die Einzelstation übergeben.

.Modelle.ODBCModell.variableBearbeitung

Werkstück1

	string 1	time 2
string	BE	Bearbeitungszeit
1	Werkstück1	43:55.6090
2	Werkstück2	43:31.0910
3	Werkstück3	39:18.5470
4	Werkstück4	43:41.8400
5	Werkstück5	49:38.4120
6	Werkstück6	43:21.2480
7	Werkstück7	43:16.5090
8	Werkstück8	58:22.0320
9	Werkstück9	43:21.6530
10	Werkstück10	44:12.7950
11	Werkstück11	43:57.3100

Abbildung 19: Auszug aus der Tabelle variableBearbeitung

.Modelle.ODBCModell.RohBearbeitungszeiten

5e9feb248e15b328ac99003f

	string 1	string 2	datetime 3	string 4	string 5	datetime 6	string 7
string	_id	channelId	end	machineId	name	start	
1	5e9feb248e15b328ac99003f	Channel1	22.04.2020 06:57:31.2880	12430012063	Lage1.mpf	22.04.2020 06:56:46.3000	22.04.20
2	5e9febd18e15b328ac990950	Channel1	22.04.2020 07:04:22.1720	12430012063	Lage1.mpf	22.04.2020 06:59:39.1230	22.04.20
3	5e9fecf78e15b328ac995c6c	Channel1	22.04.2020 07:09:09.0490	12430012063	Lage2.mpf	22.04.2020 07:04:22.1720	22.04.20
4	5e9fee178e15b328ac99aeaa	Channel1	22.04.2020 07:13:58.7650	12430012063	Lage3.mpf	22.04.2020 07:09:10.9870	22.04.20
5	5e9fef2d8e15b328ac99fff8	Channel1	22.04.2020 07:14:49.3770	12430012063	Lage4.mpf	22.04.2020 07:13:58.7650	22.04.20
6	5e9fef658e15b328ac9a0ae9	Channel1	22.04.2020 07:18:47.0730	12430012063	Lage4.mpf	22.04.2020 07:14:49.8580	22.04.20
7	5e9ff0588e15b328ac9a52b4	Channel1	22.04.2020 07:23:34.5850	12430012063	Lage5.mpf	22.04.2020 07:18:47.0730	22.04.20
8	5e9ff1798e15b328ac9aa533	Channel1	22.04.2020 07:28:24.6280	12430012063	Lage6.mpf	22.04.2020 07:23:34.5850	22.04.20
9	5e9ff29a8e15b328ac9af77c	Channel1	22.04.2020 07:33:13.6660	12430012063	Lage7.mpf	22.04.2020 07:28:25.2170	22.04.20
10	5e9ff3ba8e15b328ac9b4a64	Channel1	22.04.2020 07:38:09.5880	12430012063	Lage8.mpf	22.04.2020 07:33:13.6660	22.04.20
11	5e9ff4d78e15b328ac9b9d05	Channel1	22.04.2020 07:42:52.7520	12430012063	Lage1.mpf	22.04.2020 07:38:09.5880	22.04.20
12	5e9ff8be8e15b328ac9c00dc	Channel1	22.04.2020 07:55:09.3170	12430012063	Lage1.mpf	22.04.2020 07:54:48.9780	22.04.20
13	5e9ff8e88e15b328ac9c0530	Channel1	22.04.2020 08:00:12.0990	12430012063	Lage1.mpf	22.04.2020 07:55:28.9500	22.04.20
14	5e9ffa0d8e15b328ac9c586a	Channel1	22.04.2020 08:05:00.3810	12430012063	Lage2.mpf	22.04.2020 08:00:12.0990	22.04.20
15	5e9ffb228e15b328ac9caacc	Channel1	22.04.2020 08:09:48.5920	12430012063	Lage3.mpf	22.04.2020 08:05:00.3810	22.04.20
16	5e9ffc4e8e15b328ac9cfe6b	Channel1	22.04.2020 08:14:38.5360	12430012063	Lage4.mpf	22.04.2020 08:09:48.5920	22.04.20
17	5e9ffd708e15b328ac9d5059	Channel1	22.04.2020 08:19:42.7100	12430012063	Lage5.mpf	22.04.2020 08:14:38.5360	22.04.20
18	5e9ffea18e15b328ac9d8edb	Channel1	22.04.2020 08:24:32.5780	12430012063	Lage6.mpf	22.04.2020 08:19:42.7100	22.04.20
19	5e9fffc38e15b328ac9de11f	Channel1	22.04.2020 08:29:22.1300	12430012063	Lage7.mpf	22.04.2020 08:24:32.5780	22.04.20
20	5ea000e48e15b328ac9e3452	Channel1	22.04.2020 08:34:13.7080	12430012063	Lage8.mpf	22.04.2020 08:29:22.5600	22.04.20
21	5ea001fb8e15b328ac9e872c	Channel1	22.04.2020 08:39:00.4710	12430012063	Lage1.mpf	22.04.2020 08:34:13.7080	22.04.20
22	5ea004968e15b328ac9ee5ac	Channel1	22.04.2020 08:46:16.5630	12430012063	Lage1.mpf	22.04.2020 08:45:19.3800	22.04.20
23	5ea004e08e15b328ac9eea25	Channel1	22.04.2020 08:51:16.2630	12430012063	Lage1.mpf	22.04.2020 08:46:34.0580	22.04.20

Abbildung 20: Auszug aus der Tabelle RohBearbeitungszeiten

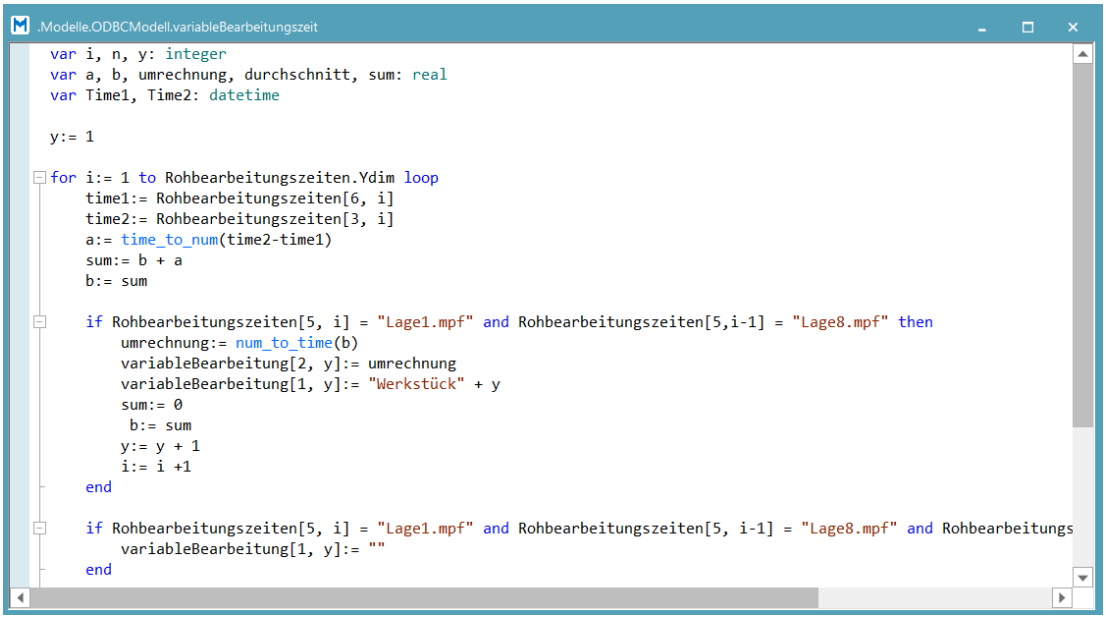


Abbildung 21: Auszug aus der Methode variableBearbeitungszeit

Erzeugung von individuellen Rüstvorgängen

Die Simulation der Rüstvorgänge wird mithilfe der Aufwärmprogramme realisiert, die zu Beginn eines jeden Produktionstages, vor der Bearbeitung der Werkstücke, an der Werkzeugmaschine durchgeführt wurden. Die Aufwärmprogramme dienen dazu, die Bestandteile der Werkzeugmaschine - wie beispielsweise Sensoriken - hinsichtlich Funktionalität zu überprüfen. Um die Rüstzeiten aus den Aufwärmprogrammen zu generieren, wird die Methode *Rüstwerte* (vgl. Abbildung 23) erstellt, die zunächst die Gesamtaufwärmzeit, sowie die dazugehörigen bearbeiteten Werkstücke pro Tag aus den Tabellen *Rohrüstdaten* und *RohBearbeitungszeiten* berechnet und in die Tabelle *Rüstdaten* speichert. Eine zusätzliche Methode namens *Rüsten* liest die einzelnen Werte in der Tabelle *Rüstdaten* (vgl. Abbildung 22) aus und erzeugt immer dann einen Rüstvorgang an der Einzelstation, wenn das erste zu bearbeitende Werkstück auf der Einzelstation eintrifft. Ein weiterer Rüstvorgang wird ausgelöst, sobald die Anzahl der behandelten Werkstücke mit der Anzahl der Werkstücke aus der Tabelle *Rüstdaten* übereinstimmt. Durch diese Technik finden eine Reihe von Rüstvorgängen während der Simulation statt und die Daten der Aufwärmprogramme werden sinnvoll in das Simulationsmodell integriert.

.Modelle.ODBCModell.Rohrüstdaten

5e9fe2cd8e15b328ac97d0d2

	string 1	string 2	datetime 3	string 4	string 5	datetime 6	string 7
string	_id	channelId	end	machineId	name	start	
1	5e9fe2cd8e15b328ac97d0d2	Channel1	22.04.2020 06:21:20.5850	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:21:07.9770	22.04.20
2	5e9fe2d78e15b328ac97d234	Channel1	22.04.2020 06:22:39.8580	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:21:20.8560	22.04.20
3	5e9fe3268e15b328ac97e256	Channel1	22.04.2020 06:23:12.8020	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:22:40.1680	22.04.20
4	5e9fe3488e15b328ac97ea5b	Channel1	22.04.2020 06:23:45.4820	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:23:12.9210	22.04.20
5	5e9fe3688e15b328ac97f1bd	Channel1	22.04.2020 06:25:56.5480	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:23:45.6220	22.04.20
6	5e9fe3eb8e15b328ac9810c9	Channel1	22.04.2020 06:26:29.1050	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:25:56.8010	22.04.20
7	5e9fe40c8e15b328ac9818a8	Channel1	22.04.2020 06:27:53.2930	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:26:29.3080	22.04.20
8	5e9fe4608e15b328ac982a94	Channel1	22.04.2020 06:29:32.4200	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:27:53.6850	22.04.20
9	5e9fe4c38e15b328ac98428a	Channel1	22.04.2020 06:30:05.1140	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:29:32.5780	22.04.20
10	5e9fe4e48e15b328ac984acb	Channel1	22.04.2020 06:30:37.8100	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:30:05.3090	22.04.20
11	5e9fe5048e15b328ac9852f8	Channel1	22.04.2020 06:31:43.1540	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:30:37.9760	22.04.20
12	5e9fe5468e15b328ac986317	Channel1	22.04.2020 06:32:16.2090	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:31:43.4920	22.04.20
13	5e9fe5678e15b328ac986b05	Channel1	22.04.2020 06:32:48.7510	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:32:16.2730	22.04.20
14	5e9fe5888e15b328ac987382	Channel1	22.04.2020 06:33:21.5530	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:32:48.9510	22.04.20
15	5e9fe5a88e15b328ac987be9	Channel1	22.04.2020 06:33:54.3710	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:33:21.8050	22.04.20
16	5e9fe5c98e15b328ac98844a	Channel1	22.04.2020 06:34:26.9810	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:33:54.5510	22.04.20
17	5e9fe5e98e15b328ac988c84	Channel1	22.04.2020 06:34:59.7920	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:34:27.1820	22.04.20
18	5e9fe60a8e15b328ac98950f	Channel1	22.04.2020 06:36:38.1830	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:35:00.0700	22.04.20
19	5e9fe66d8e15b328ac98ae4d	Channel1	22.04.2020 06:37:43.5920	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:36:38.3940	22.04.20
20	5e9fe6af8e15b328ac98bf30	Channel1	22.04.2020 06:38:16.5340	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:37:43.8860	22.04.20
21	5e9fe6cf8e15b328ac98c760	Channel1	22.04.2020 06:38:48.9930	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:38:16.5990	22.04.20
22	5e9fe6ef8e15b328ac98cfae	Channel1	22.04.2020 06:50:31.5350	12430012063	AIR_DMU65_WARMUP.MPF	22.04.2020 06:38:49.2070	22.04.20
23	5ea170ed8e15b328aca7e074	Channel1	23.04.2020 10:39:56.2570	12430012063	AIR_DMU65_WARMUP.MPF	23.04.2020 10:39:45.0820	23.04.20
24	5ea170f68e15b328aca7e192	Channel1	23.04.2020 10:40:56.8640	12430012063	AIR_DMU65_WARMUP.MPF	23.04.2020 10:39:56.4330	23.04.20
25	5ea171348e15b328aca7eff3	Channel1	23.04.2020 10:41:29.6340	12430012063	AIR_DMU65_WARMUP.MPF	23.04.2020 10:40:57.1850	23.04.20

Abbildung 22: Auszug aus der Tabelle Rohrüstdaten

.Modelle.ODBCModell.Rüstwerte

```
for i:= 1 to Rohrüstdaten.Ydim loop
    time2:= Rohrüstdaten[6, i]
    time3:= Rohrüstdaten[3, i]
    a:= time_to_num(time3-time2)
    sum:= b + a
    b:= sum

    if rohrüstdaten[7, i] /= rohrüstdaten[7, i +1]  then
        umrechnung:= num_to_time(b)
        Rüstzeiten[3, y]:=  Rohrüstdaten[7, i]
        Rüstzeiten[2, y]:= umrechnung
        Rüstzeiten[1, y]:= "Rüstzeit" + y
        sum:= 0
        b:= sum
        y:= y + 1

    end
next
Rüstzeiten.setzeDatenTyp(2, "time")

RohBearbeitungszeiten.MaxXDim:= 7
RohBearbeitungszeiten.einfügeSpalte(7)
```

Abbildung 23: Auszug aus der Methode Rüstwerte

.Modelle.ODBCModell.Rüstdaten

29:19.0540

	time 1	integer 2
string		
1	29:19.0540	5
2	17:55.1370	3
3	25:49.0190	4
4	18:48.1220	6
5	22:04.4670	6
6	21:12.0000	4
7	19:53.0550	8
8	38:34.4240	2
9	19:00.9160	5
10	9:56.8480	7

Abbildung 24: Die Tabelle Rüstdaten

.Modelle.ODBCModell.Rüsten *

```
if prodTeile = Rüstdaten[2, 1] + 1    then
@.Name:= "Rüstteil" + prodTeile
Werkzeugmaschine.Rüstzeit:= Rüstdaten[1, 2]
    Rüstdaten[2, 2]:= prodTeile + Rüstdaten[2, 2] - 1
    Rüstdaten.entferneZeile(1)
elseif  prodTeile = 1 then
Werkzeugmaschine.Rüstzeit:= Rüstdaten[1, 1]
else
    Werkzeugmaschine.Rüstzeit:= 0
end

Werkzeugmaschine.Bearbeitungszeit:= variableBearbeitung[2, 1]
variableBearbeitung.entfernezeile(1)
```

Abbildung 25: Die Methode Rüsten

Abschließend erfolgt die Fertigstellung der Werkstücke auf einer Arbeitsstation. Dabei basieren die Bearbeitungszeiten der Arbeitsstation auf rein hypothetischen Werten, um das Zusammenspiel zwischen realen und fiktiven Daten innerhalb der Simulation darstellen zu können.

Nachdem ein Simulationslauf durchgeführt wurde, können die gesammelten Simulationsdaten mit geeigneten Diagrammen analysiert werden. In Abbildung 26 sind die Bearbeitungszeiten der Werkstücke auf der Werkzeugmaschine in einem XY-Diagramm dargestellt. Während die X-Achse die Werkstücknummer abbildet, zeigt die Y-Achse die Bearbeitungsdauer in Minuten an. Mithilfe des Diagramms können die einzelnen Bearbeitungszeiten ausgewertet und gegebenenfalls optimiert werden.

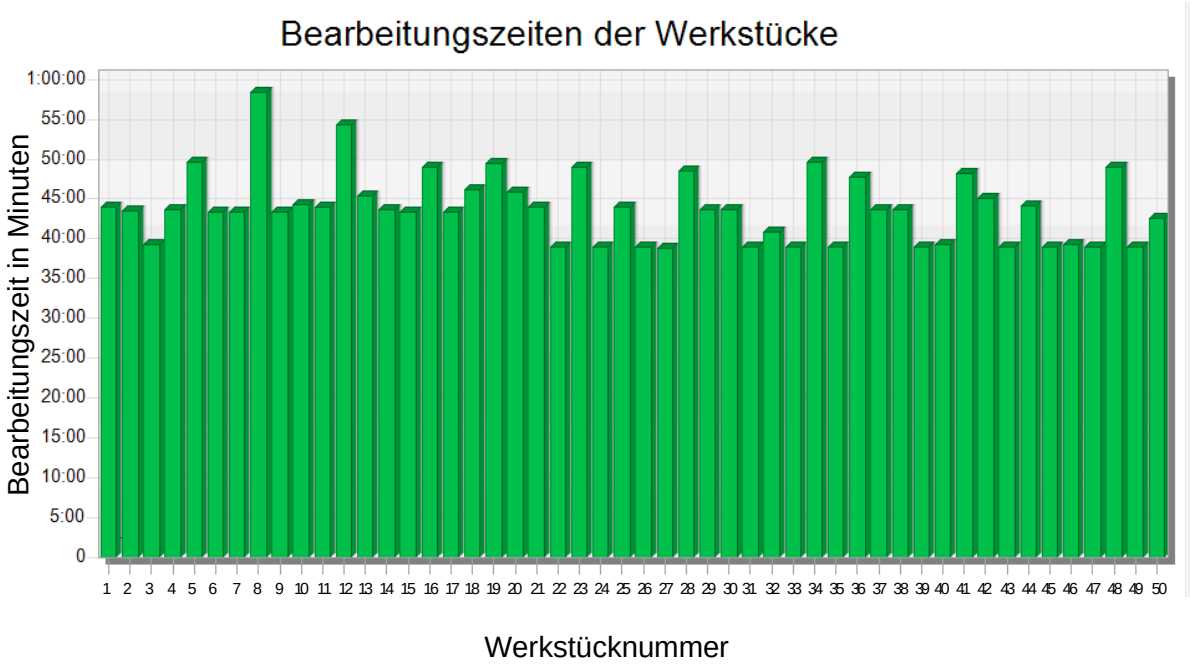


Abbildung 26: XY-Diagramm der Bearbeitungszeiten pro Werkstück

Alternativ zur klassischen Anbindung aller Daten, kann über den Mittelwert und einer anschließenden Normalverteilung ebenfalls eine Simulation durchgeführt werden. Dies ist vor allem dann empfehlenswert, wenn nicht alle Daten zur Verfügung stehen oder eine Simulation auch nach Verwendung aller Daten weiterlaufen soll. Für die Umsetzung in Plant Simulation wird zunächst die Methode *Mittelwertberechnung* erstellt (vgl. Abbildung 27). Innerhalb dieser Methode wird aus allen Daten der Mittelwert berechnet und anschließend in die Tabelle *Normalverteilung* übertragen. In der Tabelle *Normalverteilung* befinden sich der Mittelwert, die Oberschranke, die Unterschranke, sowie das festgelegte Sigma. Die Oberschranke definiert sich aus der Addition des Mittelwerts und des Sigmas, die Unterschranke aus der Division des Mittelwerts und des Sigmas. Das Sigma legt die Standardabweichung der Verteilung fest. Bei Betrachtung der vorhandenen Daten wird ein Sigma von drei Minuten angenommen. Die Methode *Mittelwertberechnung* erzeugt vor der Bearbeitung eines Werkstücks mithilfe der Formel für die Normalverteilung eine Bearbeitungszeit und übergibt sie an die Werkzeugmaschine. Daraufhin bearbeitet die Werkzeugmaschine die Werkstücke mit den vorgegebenen Bearbeitungszeiten und übergibt sie anschließend an die Arbeitsstation. Somit kann das Verhalten der Werkzeugmaschine auf unterschiedliche Art in einer Simulation untersucht werden.

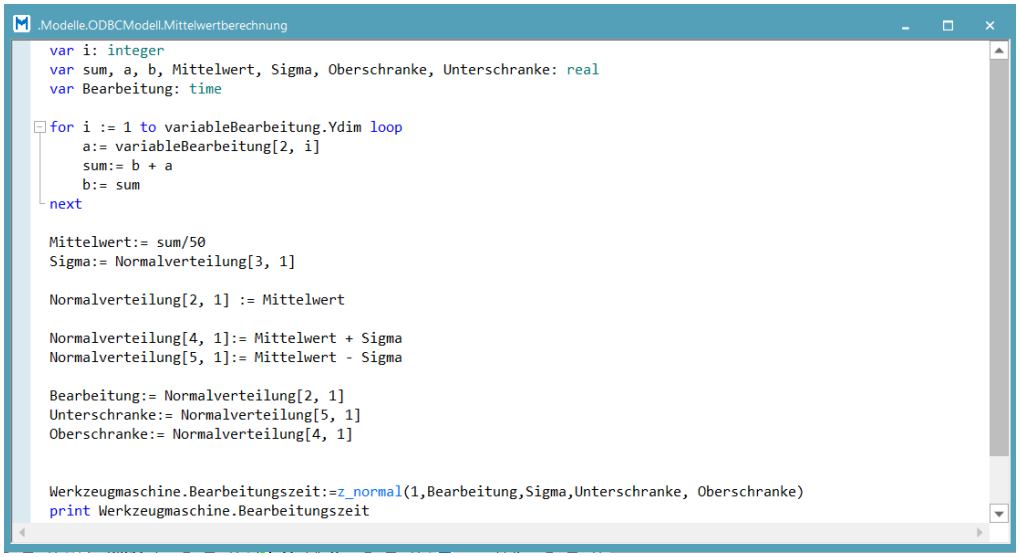


Abbildung 27: Die Methode Mittelwertberechnung

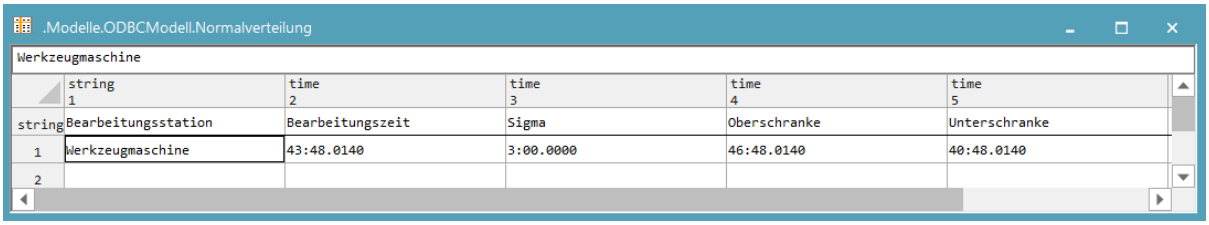


Abbildung 28: Die Tabelle Normalverteilung

5.1.4 Bewertung des Simulationsmodells und der Datenanbindung mittels einer ODBC-Schnittstelle

Das erzeugte Simulationsmodell in Abschnitt 5.1.3 basiert auf den Daten einer real operierenden Werkzeugmaschine der Forschungseinrichtung des Digital Labs. Die Werkzeugmaschine wird hauptsächlich für Forschungszwecke verwendet. Somit findet keine Bearbeitung der Werkstücke unter realen Produktionsbedingungen statt. Dies hat zur Folge, dass sich die Simulation hauptsächlich auf Bearbeitungs- und Rüstvorgänge beschränkt, da unter anderem einzelne Datensätze zu Störmeldungen nicht eindeutig definiert werden können. Für die Auswertung und Filterung der Daten sind komplexe Methoden und Abfragen nötig, zumal die Daten der Datenbank in einem unstrukturierten Format abgespeichert werden. Dennoch zeigt das Simulationsmodell anhand eines einfachen Aufbaus, dass es grundsätzlich möglich ist, eine digitale Abbildung einer Werkzeugmaschine in Form einer Einzelstation mithilfe einer ODBC-Schnittstelle zu erzeugen und in Plant Simulation zu integrieren.

Die Verwendung einer ODBC-Schnittstelle in Plant Simulationen bietet sich an, um das Verhalten eines realen Prozesses, mithilfe aufgezeichneter Daten, in einer Simulation zu untersuchen. Des Weiteren können reale Prozesse in verschiedene Simulationsszenarien digital abgebildet und analysiert werden. Dazu ist eine konsistente Aufzeichnung der Daten in einer

Datenbank erforderlich. Ansonsten können Fehler und somit falsche Aussagen aus dem betrachteten System resultieren. Eine Echtzeitdatenanbindung ist mit einer ODBC-Schnittstelle nicht möglich, da das Ereignis in Form eines Datensatzes erst an die Datenbank übermittelt wird, sobald es abgeschlossen ist.

5.2 Datenanbindung mithilfe einer OPC UA-Schnittstelle

Für die Anbindung der Daten über eine OPC UA-Schnittstelle ist zunächst ein OPC UA-Server erforderlich, der sämtliche Daten des zu untersuchenden Objekts zur Verfügung stellt. Die betrachtete Werkzeugmaschine des Digital Labs verfügt über einen konfigurierten OPC UA-Server, der dauerhaft mit der Werkzeugmaschine verbunden ist. Dieser OPC UA-Server ist mit einer Sicherheitskonfiguration ausgestattet, da eine Clientanwendung unmittelbar Einfluss auf die Werkzeugmaschine nehmen kann. Die Sicherheitskonfiguration des OPC UA-Servers führt zu einem entscheidenden Problem für die Datenanbindung an Plant Simulation. Die aktuellen Versionen von Plant Simulation lassen nur einen anonymen Zugriff auf einen OPC UA-Server zu, sodass jeder Zugriffsversuch auf den OPC UA-Server der Werkzeugmaschine abgeblockt wird. Einerseits ist der benötigte Zeitaufwand für die Umprogrammierung des OPC UA-Servers zu groß. Andererseits entsteht durch die offene Konfiguration ein Sicherheitsrisiko für die Anlage. Aus diesen Gründen wird ein OPC UA-Server programmiert, der das Verhalten der Werkzeugmaschine simuliert.

Im folgenden Abschnitt wird zunächst die Erstellung des OPC UA-Servers detailliert beschrieben. Anschließend erfolgt die Anbindung des erzeugten OPC UA-Servers an ein Simulationsmodell. Zum Schluss findet eine Beurteilung und Auswertung der Methode statt.

5.2.1 Programmierung eines OPC UA-Servers für die Herstellung einer Datenanbindung

Dieser Abschnitt behandelt die Programmierung eines OPC UA-Servers, der Bearbeitungs- sowie Störprogramme simuliert. Für die Realisierung des Servers sind folgende Komponenten notwendig:

- Python 3.8.5
- Eclipse IDE
- UA-Expert

Mit Python, einer universell anwendbaren Programmiersprache, wird der OPC UA-Server erstellt. Dabei findet die eigentliche Programmierung mithilfe des Entwicklerwerkzeuges Eclipse

IDE statt. Eclipse IDE ist ein weitverbreitetes Tool für die Entwicklung von Softwareanwendungen. Die Clientanwendung UA-Expert dient dazu, anschließend die Funktionalität des OPC UA-Servers zu überprüfen.

Sobald Python installiert ist, muss zunächst ein Basispaket für einen OPC UA-Server in Python heruntergeladen werden. Dieses Basispaket ermöglicht es, einen vorprogrammierten Grundaufbau eines OPC UA-Servers zu verwenden und vereinfacht die Programmierung eines OPC UA-Servers. Über die Windows Eingabeaufforderung ist in das Verzeichnis, in dem sich Python befindet, zu navigieren. Anschließend findet durch die Eingabe des Befehls

pip install opcua

der Import des Basispakets statt. Danach kann die Programmierung des Servers in Eclipse IDE beginnen.

In der Benutzeroberfläche des Entwicklerwerkzeuges wird über die Registerkarte *File* ein neues Python-Projekt und eine dazugehörige Datei namens server.py erstellt (vgl. Abbildung 29). Innerhalb der server.py ist der Programmcode für den OPC UA-Server zu definieren. In den ersten Kommandozeilen lädt das Modul alle benötigten Pakete herunter, die im Laufe des Programms Anwendung finden. Um den OPC UA-Server mithilfe eine Clientanwendung ansteuern zu können, wird eine URL festgelegt, über die der Server erreichbar ist. Im Anschluss daran, wird eine objektorientierte *Node* erzeugt, die die Werkzeugmaschine darstellt. Für die Simulation von verschiedenen Daten, sind zusätzlich variable *Nodes* notwendig. Diese sind in Zeile 19-22 programmiert.

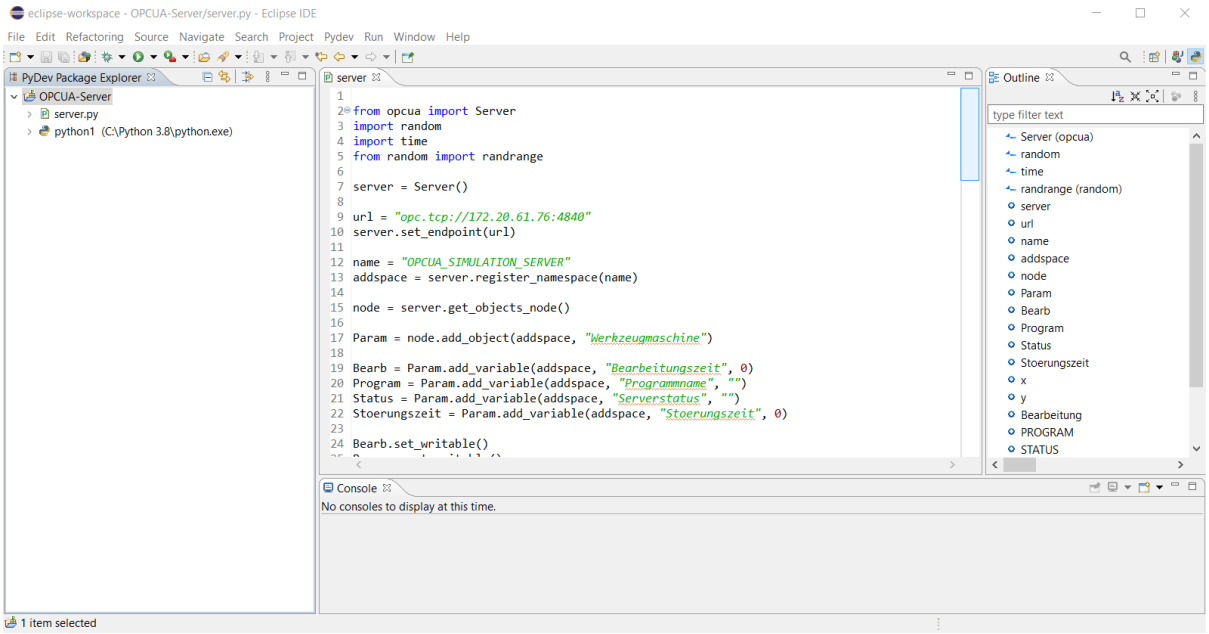


Abbildung 29: Die Eclipse IDE Benutzeroberfläche mit einem Auszug aus dem Programmiercode des OPC UA-Servers

Die Hauptaufgabe des Servers ist es, Bearbeitungen und zufällig eintretende Störungen zu simulieren. Dabei sind die Bearbeitungs- sowie Störzeiten variabel definiert. Die Verfügbarkeit der Werkzeugmaschine ist mit einer geeigneten Verteilungsmethode auf 80% gesetzt. In zufälligen Zeitabständen generiert der Server Störungen, die unterschiedlich lang andauern. Sobald der Fehler behoben ist, startet die Bearbeitung der Werkstücke erneut (vgl. Abbildung 30). Die beschriebene Programmierung ist bei der Verwendung eines realen OPC UA-Servers nicht notwendig, da die Werkzeugmaschine automatisch die Daten an den OPC UA-Server senden würde.

server

```
print("Server started at {}".format(url))

time.sleep(10)

while True:

    x = [["True", "False"], [8, 2,]]
    y = sum(([t] * w for t, w in zip(*x)), [])


    Bearbeitung = (0)
    PROGRAM = ("Mainprogram")
    STATUS = random.choice(y)
    STOERUNG = (0)



    Program.set_value(PROGRAM)
    Status.set_value(STATUS)
    Stoerungszeit.set_value(STOERUNG)
    Bearb.set_value(Bearbeitung)

    if STATUS == "False":
        Program.set_value("Malfunction")
        z = randrange(10, 100)
        Stoerungszeit.set_value(z)
        print(Bearbeitung,PROGRAM, STATUS, STOERUNG)
        time.sleep(z)
    if STATUS == "True":
        Bearbeitung = randrange(40, 50)
        Bearb.set_value(Bearbeitung)
        print(Bearbeitung,PROGRAM, STATUS, STOERUNG)
        time.sleep(Bearbeitung)
```

Abbildung 30: Auszug aus dem Programmiercode des OPC UA-Servers

Der Server startet durch das Öffnen der server.py-Datei, die sich im Eclipse-Verzeichnis befindet. Um die Funktionalität des Servers und der einzelnen Variablen überprüfen zu können, ist die Clientanwendung UA Expert notwendig. Innerhalb der Benutzeroberfläche ist über die Toolbox ein neuer Server hinzuzufügen. Nachdem eine Verbindung zum Server hergestellt ist, zeigt das mittlere Fenster den Status aller Variablen an (vgl. Abbildung 31). Sobald der Statuscode aller Variablen: *Good* ist, kann die Anbindung des Servers an Plant Simulation beginnen.

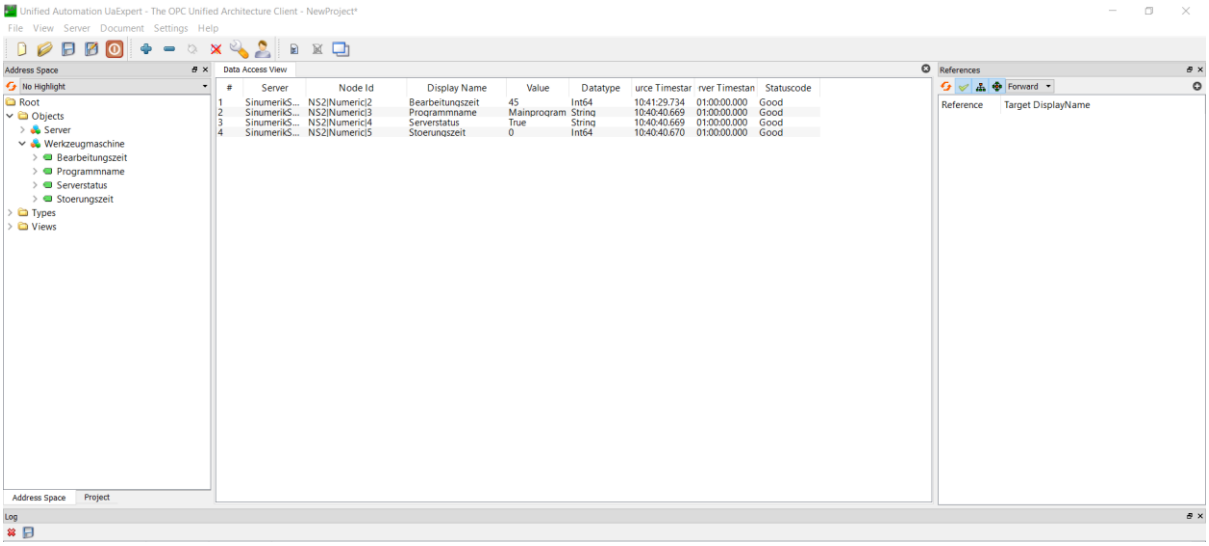


Abbildung 31: Die Benutzeroberfläche von UA-Expert

5.2.2 Modellierung eines Simulationsmodells und Ansteuerung des konfigurierten OPC UA-Servers mit Plant Simulation

Dieser Abschnitt behandelt die Modellierung des Simulationsmodells mit einer OPC UA-Schnittstelle und einer anschließenden Ansteuerung des konfigurierten OPC UA-Servers aus Abschnitt 5.2.1. Dabei ist der Grundaufbau des Simulationsmodells identisch zu dem Modell aus Abschnitt 5.1.3 (vgl. Abbildung 32) und besteht ausfolgenden Komponenten:

- Quelle
- Werkzeugmaschine (Einzelstation)
- Arbeitsstation (Einzelstation)
- Senke

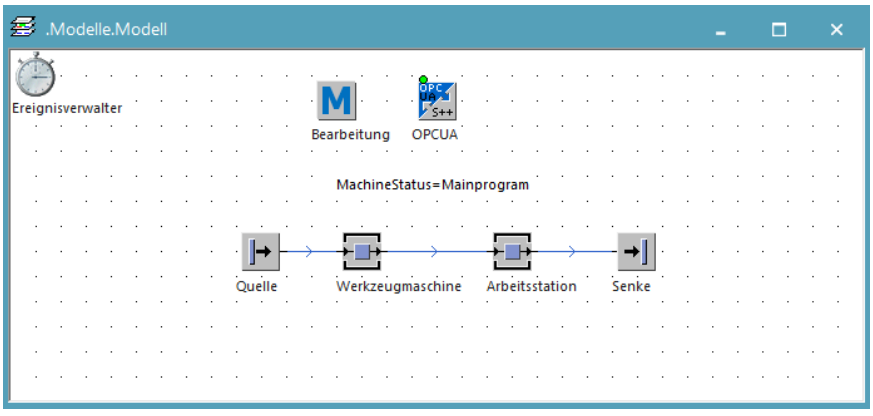


Abbildung 32: Das OPC UA-Simulationsmodell

Für die Anbindung des OPC UA-Servers an Plant Simulation, ist der Schnittstellenbaustein *OP-CUA* notwendig. Innerhalb des Eingabefensters ist zunächst die IP-Adresse des OPC UA-Servers anzugeben. Anschließend sind über die Schaltfläche *Elemente* der Gruppenname, der

Namensraum sowie die Lese- und Schreibintervalle einzutragen (vgl. Abbildung 31). Der Gruppenname ist frei definierbar. Der Namensraum ist fest durch den Server festgelegt. Die Lese- und Schreibintervalle sind frei wählbar und in Millisekunden anzugeben. Ein Standardwert hierfür ist 50.

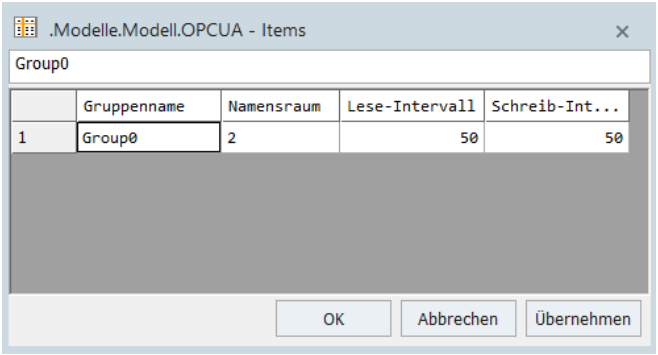


Abbildung 33: Tabelle der Elemente

Die zu einem Namensraum zugehörigen Variablen sind nach Fertigstellung des Elements in einer weiteren Tabelle einzutragen (vgl. Abbildung 32). Der *Bezeichner* beschreibt die eindeutige Identifikation einer Variable und ist somit fest durch den Server vorgegeben. Plant Simulation füllt automatisch den Datentyp der Variablen aus. Der *Alias* ist für die Beschreibung der Variablen zuständig. Sind alle Werte ordnungsgemäß eingetragen und die Kontrollleuchte des Bausteins auf grün, ist eine Verbindung zwischen Plant Simulation und dem OPC UA-Server hergestellt. Die aktuellen Werte der Variablen sind in einer zusätzlichen Tabelle einsehbar (vgl. Abbildung 33).

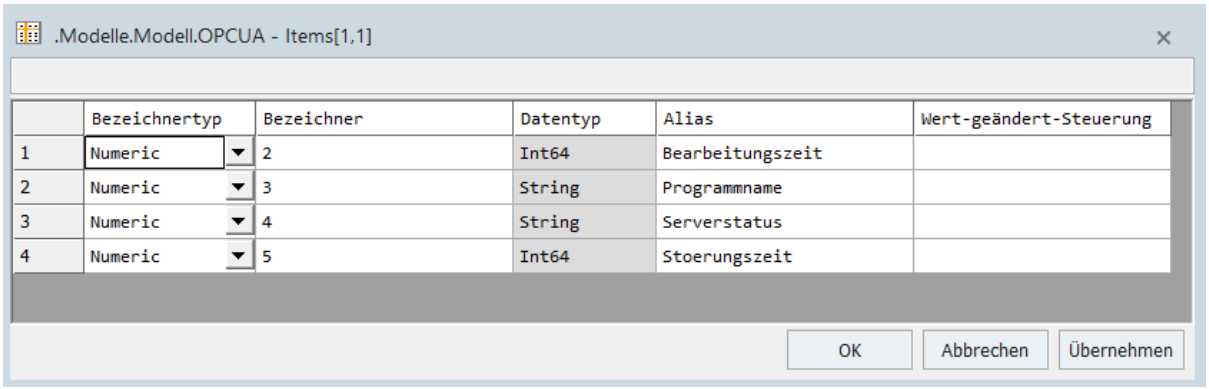


Abbildung 34: Tabelle der einzelnen Variablen

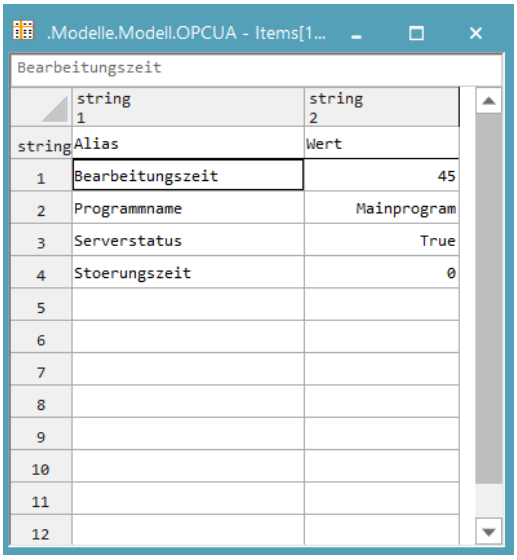


Abbildung 35: Tabelle für den Status aller Variablen des OPC UA-Servers

Der OPC UA-Server ist in Echtzeit mit Plant Simulation verbunden und sendet den Status aller Variablen im Millisekunden-Bereich an die Simulationssoftware. Die Methode *Bearbeitung* greift auf alle Werte zu und übergibt sie an die Einzelstation (Werkzeugmaschine) in Plant Simulation (vgl. Abbildung 34).

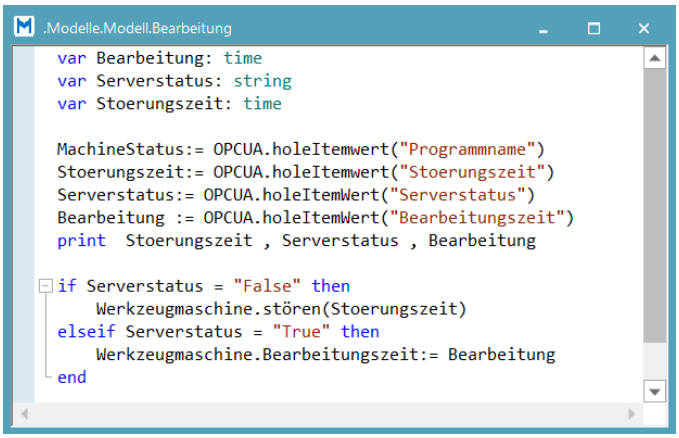


Abbildung 36: Auszug aus der Methode Bearbeitung

Sobald der OPC UA-Server startet, muss die Simulation ebenfalls innerhalb von 10 Sekunden beginnen, um einen synchronen Verlauf der Bearbeitung zu gewährleisten. Mit einem OPC UA-Server, der mit einer realen Maschine verbunden ist, ist dieser Schritt nicht notwendig, da der OPC UA-Server in diesem Fall keine Bearbeitungszeiten simulieren muss. Während eines Simulationslaufes bearbeitet das Simulationsmodell Werkstücke in Abhängigkeit der Werte des OPC UA-Servers. Wechselt die Werkzeugmaschine in den Störungsbetrieb, ist die Einzelstation der Simulation ebenfalls gestört. Nachdem die Bearbeitung der Werkstücke auf der Werkzeugmaschine fertiggestellt ist, findet eine abschließende Behandlung der Werkstü-

cke auf einer Arbeitsstation statt. Die Produktivität der Werkzeugmaschine kann laufend mithilfe eines geeigneten Diagramms untersucht werden (siehe Abbildung 35). Somit ist eine digitale Abbildung der Werkzeugmaschine in Plant Simulation über einen OPC UA-Server und einer OPC UA-Schnittstelle erzeugt und in ein Simulationsmodell integriert.

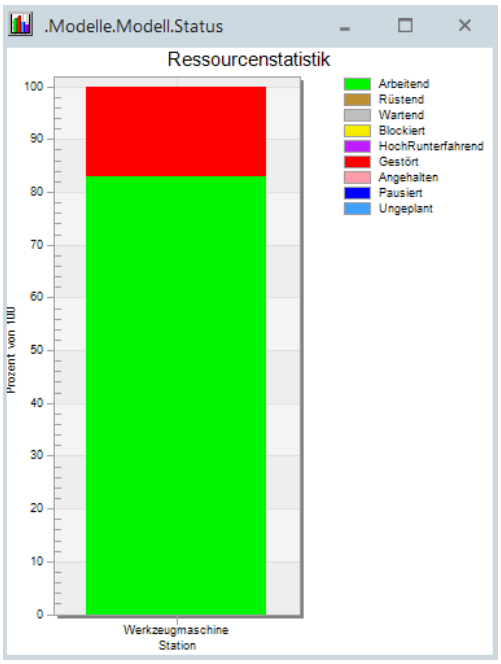


Abbildung 37: Ressourcenstatistik der Werkzeugmaschine

5.2.3 Bewertung des Simulationsmodells und der Datenanbindung mittels einer OPC UA-Schnittstelle

Das Simulationsmodell aus Abschnitt 5.2.2 bearbeitet Werkstücke in Abhängigkeit von Daten eines programmierten OPC UA-Servers und zeigt das Plant Simulation grundsätzlich die Daten eines OPC UA-Servers verwenden und in eine Simulation integrieren kann. Obwohl die Werkzeugmaschine des Digital Labs über einen OPC UA-Server verfügt, kann dieser Server aufgrund der eingestellten Sicherheitskonfiguration nicht verwendet werden. In den aktuellen Versionen von Plant Simulation ist nur ein anonymer Zugriff auf einen OPC UA-Server möglich. Der programmierte OPC UA-Server und die generierten Bearbeitungs- und Störvorgänge dienen dazu, generell eine OPC UA-Schnittstelle in Plant Simulation zu untersuchen.

Eine OPC UA-Schnittstelle bietet die Möglichkeit, reale Prozesse eines Unternehmens in Echtzeit in einer Simulation zu untersuchen. Dazu sind keine komplexen Abfragen und Methoden nötig. Der OPC UA-Server listet die Daten des zu untersuchenden Prozesses automatisch nach einer bestimmten Struktur auf. Plant Simulation importiert die Daten und übergibt sie mit einer geeigneten Methode an die gewünschten Bausteine. Je nachdem wie viele Prozesse mit einem OPC UA-Server verbunden sind, können komplette Produktionslinien digital in einer

Simulation abgebildet und untersucht werden. Die OPC UA-Schnittstelle kann, wie auch die ODBC-Schnittstelle, reale Prozesse in verschiedenen Simulationsszenarien analysieren. Des Weiteren kann eine OPC UA-Schnittstelle unmittelbaren Einfluss auf den verbundenen OPC UA-Server nehmen und somit Parameter des Servers verändern. Dadurch ist nicht nur eine Datenabfrage möglich, sondern auch eine Steuerung des Prozesses aus der Simulationssoftware heraus durchführbar.

6 Fazit und Ausblick

Gegenstand der vorliegenden Arbeit ist es, ein Konzept für die effiziente Datenanbindung einer industriellen Werkzeugmaschine an eine Software zur diskreten Fabriksimulation zu entwickeln und aufzubauen. Da die Werkzeugmaschine Daten an eine Datenbank, sowie an einen konfigurierten OPC UA-Server sendet, gibt es mehrere Möglichkeiten auf die benötigten Daten zuzugreifen. Mithilfe einer ODBC-Schnittstelle wird zunächst die Anbindung der Daten über eine Datenbank betrachtet. Das Simulationsmodell aus Abschnitt 5.1.3 bearbeitet einzelne Werkstücke mit Daten aus der Datenbank und zeigt, dass eine Datenanbindung über eine ODBC-Schnittstelle grundsätzlich möglich ist. Die Anbindung der Daten erfolgt ausschließlich auf Basis von Vergangenheitswerten und bietet die Möglichkeit, durchgeführte Prozesse nach zu simulieren oder innerhalb anderer Simulationsszenarien zu untersuchen. Um präzise Annahmen aus einem System erzielen zu können, ist eine konsistente Datenaufzeichnung erforderlich. Werden einzelne Datensätze nicht berücksichtigt oder falsch interpretiert, können fehlerhafte Ergebnisse und anschließend inkorrekte Aussagen über einen untersuchten Prozess entstehen. Die Filterung und Auswertung der Daten findet in Plant Simulation statt. Aufgrund der großen Datenmengen ist es jedoch zeitaufwendig in Plant Simulation entsprechende Befehle zu programmieren, um einen gezielten Datenbereich zu isolieren und zu nutzen. Zudem können nicht alle Datensätze verwendet werden, da es sich bei der untersuchten Werkzeugmaschine um eine Forschungseinrichtung handelt, die nicht unter realen Produktionsbedingungen Werkstücke bearbeitet. Des Weiteren ist die genaue Zuordnung einzelner Datensätze nicht möglich, sodass Störungsmeldungen im Simulationsmodell nicht ausgewertet werden können. Die Datenanbindung mithilfe einer Datenbank über eine ODBC-Schnittstelle an Plant Simulation ist generell durchführbar. Die Realisierung gestaltet sich als komplex und zeitaufwändig.

Im Gegensatz zu einer ODBC-Schnittstelle greift eine OPC UA-Schnittstelle auf die Echtzeitdaten eines OPC UA-Servers zu. Die Werkzeugmaschine des Digital Labs verfügt über einen konfigurierten OPC UA-Server, welcher sämtliche Maschinendaten empfängt. Dieser OPC UA-Server ist durch eine Sicherheitskonfiguration geschützt, um einen unbefugten Zugriff auf die Werkzeugmaschine zu vermeiden. Die aktuellen Versionen von Plant Simulation bieten ausschließlich einen anonymen Zugriff auf einen OPC UA-Server an, sodass jeder Zugriffsversuch der Werkzeugmaschine abgeblockt wird. Um trotzdem eine OPC UA-Schnittstelle in Plant Simulation untersuchen zu können, wird ein OPC UA-Server programmiert, der Bearbeitungs- sowie Störvorgänge simuliert. Mit dem Simulationsmodell aus Abschnitt 5.2.2 findet eine Echtzeitanbindung der simulierten Daten des programmierten OPC UA-Servers an Plant Simulation statt. Dabei ist der Grundaufbau der Simulation identisch zu dem Aufbau des ODBC-

Modells. Die Werkstücke werden anhand der genierten Bearbeitungszeiten des OPC UA-Servers bearbeitet und anschließend auf einer Arbeitsstation fertiggestellt. Die Simulation reagiert in Millisekunden auf Wertveränderungen des Servers. Mit einem realen OPC UA-Server, der mit einer operierenden Maschine verbunden ist, kann eine digitale Abbildung erzeugt werden, die parallel zur Maschine zeitgleich die Werkstücke bearbeitet. Zusätzlich ist eine OPC UA-Schnittstelle in der Lage, gezielt Parameter des OPC UA-Servers zu verändern. Somit können neben Datenabfragen auch Prozesse aus der Simulationssoftware heraus gesteuert werden. Besonders wegen der Integrierbarkeit von Systemen unterschiedlicher Hersteller in einen OPC UA-Server, eignet sich eine OPC UA-Schnittstelle für die Anbindung von Prozessen an Plant Simulation.

Beide Simulationsmodelle repräsentieren eine grundlegende Datenanbindung mit unterschiedlichen Schnittstellen. In Bezug auf die fortschreitende Vernetzung von Systemen innerhalb eines Unternehmens, wird vor allem OPC UA zukünftig eine entscheidende Rolle im Industriesektor einnehmen, da es die Möglichkeit bietet, verschiedene Systeme von unterschiedlichen Herstellern zu verknüpfen. Sobald die Entwickler von Tecnomatix Plant Simulation einen benutzerdefinierte Zugriff auf einen OPC UA Server in die kommenden Versionen implementieren, können mit einer OPC UA-Schnittstelle sämtliche Prozesse eines Unternehmens digital in Plant Simulation abgebildet und analysiert werden. Allgemein wird für die erfolgreiche Bewältigung der Herausforderungen, die durch die Digitalisierung entstehen, die Simulationstechnik in Verbindung mit der Datenanbindung in den kommenden Jahren für große Unternehmen unabdingbar.

Literaturverzeichnis

Bangsow, S. (2016): Tecnomatix Plant Simulation. Springer International Publishing Switzerland, Cham.

Bracht, U.; Geckler, D.; Wenzel, S. (2018): Digitale Fabrik. Springer Verlag, Berlin.

DIN IEC 60050-351:2013: Internationales Elektrotechnisches Wörterbuch. Beuth Verlag, Berlin.

Edlich, S. (2011): NoSQL. Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken. Hanser Verlag, München.

Eley, M. (2012): Simulation in der Logistik. Springer Verlag, Heidelberg.

Frauenhofer-Institut für Produktionsanlagen und Konstruktionstechnik IPK (Hrsg): Industrie 4.0: Virtueller Zwilling steuert die Produktion [online]. Verfügbar unter: www.fraunhofer.de/de/presse/presseinformationen/2017/februar/industrie-4-0-virtueller-zwilling-steuert-die-produktion.html. [Zugriff am: 20.06.2020].

Geisler, F. (2014): Datenbanken. Grundlagen und Design. Verlagsgruppe Hüthig Jehle Rehm, Heidelberg.

Gutenschwager, K. et al (2017): Simulation in Produktion und Logistik. Springer Verlag, Berlin.

Hedtstück, U. (2013): Simulation diskreter Prozesse. Springer Verlag, Berlin.

Hoffmann, M. (2019): Smart Agents for the Industry 4.0. Springer Verlag, Wiesbaden.

Kudraß, T.; Brinkhoff, T. (2015): Taschenbuch Datenbanken. Hanser Verlag, München.

Kuhn T. (2017): Digitaler Zwilling [online]. Was ist ein digitaler Zwilling? Berlin. [Zugriff am: 25.05.2020]. Verfügbar unter: gi.de/informatiklexikon/digitaler-zwilling/.

Luber, S. (2017): Was ist ODBC? [online]. Augsburg. [Zugriff am: 05.06.2020]. Verfügbar unter: www.bigdata-insider.de/was-ist-odbc-a-626950/.

März, L. et al (2011): Simulation und Optimierung in Produktion und Logistik. Springer Verlag, Berlin.

MongoDB (2020): MongoDB Connector for BI 2.13 [online]. New York City. [Zugriff am: 07.06.2020]. Verfügbar unter: docs.mongodb.com/bi-connector/master/.

Plenk, V. (2017): Angewandte Netzwerktechnik kompakt. Springer Verlag, Wiesbaden.

Schüller, A. et al (2019): "Der Digitale Zwilling in der Prozessindustrie". In: *atp magazin* 1-2, S. 70–74.

Siemens Product Lifecycle Management Software Inc. (2008): Plant Simulation Produktübersicht [online]. Köln. [Zugriff am 20.05.2020]. Verfügbar unter: www.plm.automation.siemens.com/global/de/products/tecnomatix/

sqlite Homepage (2020): About SQLite [online]. [Zugriff am: 12.06.2020]. Verfügbar unter: www.sqlite.org/index.html.

VDI (2014): VDI-Richtlinie 3633. Simulation von Logistik-, Materialfluss- und Produktionssystemen - Grundlagen. Beuth Verlag, Berlin.

Vogel-Heuser, B.; Bauernhansl, T.; Ten Hompel, M. (2017): Handbuch Industrie 4.0. Springer Verlag, Berlin.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt, keine anderen als die angegebenen Hilfsmittel benutzt und wörtliche sowie sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum	Unterschrift