ulm university universität **u**ulm

**Universität Ulm** | 89069 Ulm | Germany

**Fakultät für Ingenieurwissenschaften und Informatik**
Institut für Datenbanken und Informationssysteme

# Konzeption und Entwicklung einer Modellierungs- und Ausführungsumgebung für Verfahrensprozesse

Masterarbeit an der Universität Ulm

**Vorgelegt von:**
Manuel Fiedler
manuel.fiedler@uni-ulm.de

**Gutachter:**
Prof. Dr. Manfred Reichert
Dr. Matthias Lohrmann

**Betreuer:**
Nicolas Mundbrod

2015

Fassung 11. Dezember 2015



Satz: PDF-LaTeX 2ε

# Kurzfassung

Industrie 4.0 verspricht Industrieunternehmen eine verbesserte Wettbewerbsfähigkeit und eine Optimierung der Produktion, durch die Einführung neuer Softwaresysteme und die Vernetzung aller beteiligten Ressourcen. Eine wichtige Voraussetzung hierfür ist ein System zur Modellierung des gesamten Produktionsprozesses zwischen der Planungsebene und Maschinenebene und die direkte Integration der Produktionsmaschinen. Solche Systeme werden typischerweise als *Manufacturing Execution Systems (MES)* bezeichnet.

In dieser Arbeit wird ein prozess-basiertes Konzept für eine Modellierungs- und Ausführungsumgebung beschrieben, das es erlaubt, den gesamten Verfahrensprozess einer Produktion zu modellieren und damit auch die Aktivitäten der beteiligten Maschinen zu beschreiben. Dazu liefert das Konzept einen Weg die Aktivitäten bei Ausführung des Prozesses an die Maschinen zu senden. Für die Erfassung der Anforderungen und die Beschreibung des Konzepts wird das Maschinenbediensystem SmartControl, der Firma Uhlmann, als Ausgangspunkt herangezogen. Ein weiterer wichtiger Bestandteil dieser Arbeit ist die prototypische Umsetzung des entwickelten Konzepts. Der Prototyp besteht aus einer Client-/Server-Anwendung zur Modellierung, Verwaltung und Instanziierung von Prozessmodellen sowie einem integrierten adaptiven Prozessmanagementsystem, das Prozessinstanzen basierend auf den Prozessmodellen ausführen kann.

# Danksagung

Für die Chance diese Abschlussarbeit sehr praxisnah durchführen zu können und während dieser Zeit interessante, fachübergreifende Einblicke in die Maschinenbau- und Pharmaindustrie zu erhalten, bedanke ich mich bei Hans-Ulrich Seufert der Firma Uhlmann Pac-Systeme GmbH & Co. KG und Prof. Dr. Manfred Reichert der Universität Ulm.

Für die fachliche Betreuung und die Unterstützung bei Uhlmann, in Form von Einweisungen und zahlreicher, langer Besprechungen, bedanke ich mich, neben Hans-Ulrich Seufert, auch bei Markus Armbruster und Ulrich Schmid. Ein großes Dankeschön geht ebenfalls an meinen Betreuer der Universität Ulm, Nicolas Mundbrod, für die vielen Denkanstöße und die produktive Betreuung. Des Weiteren bedanke ich mich bei Hayato Hess und Benjamin Schmitz für die Einführung in das Ausgangsprojekt und bei allen Personen, die am Gegenlesen dieser Arbeit beteiligt waren.

Ich möchte mich außerdem bei meiner Freundin, meinen Eltern und meinen Geschwistern für die moralische Unterstützung bedanken, ohne die mein Master-Studium nicht möglich gewesen wäre.

# Inhaltsverzeichnis

# 1

# Einleitung

In den letzten Jahren sind die traditionelle Industrie und der sich rapide entwickelnde Informationssektor immer stärker zusammengewachsen. Als Auslöser und Treiber für neue Fertigungsmöglichkeiten gelten technische Innovationen, die in den drei vorangegangen großen industriellen Revolutionen zur Erzeugung neuer Produkte führten. Hierzu zählen die Einführung mechanischer Produktionsanlagen, die Fließbandproduktion und der Einsatz von Elektronik und erster IT-Systeme in Unternehmen [KWH13].

Gegenwertig befinden sich Unternehmen bedingt durch die Globalisierung und zunehmende Internationalisierung von Geschäften in einem Konkurrenzumfeld, das längst nicht mehr regional, sondern zunehmend international ist. Darüber hinaus unterliegen Unternehmen vielen verschiedenen unternehmensinternen und -externen Einflussfaktoren wie Unternehmensstrategie, Umweltauflagen und anderen Gesetzesvorschriften. Zur Marktprofilierung und Konkurrenzabgrenzung gilt es neben einer Fokussierung auf die Kernkompetenzen und dem Verfolgen langfristiger Ziele auf Marktanforderungen flexibel durch optimierte Unternehmensprozesse zu reagieren und zukünftige technologische Entwicklungen zu berücksichtigen [WAL08]. Zu den wichtigsten IT-Trends für die produzierende Industrie im Jahr 2015 zählt dabei *Industrie 4.0* [Bit15].

Ein besonderes Merkmal der aufkommenden *vierten industriellen Revolution (Industrie 4.0)* ist die Verzahnung zwischen Produktion und Informations- bzw. Kommunikationstechnologien. Zudem beruht Industrie 4.0 auf intelligenten Fabriken (engl. *Smart Factories*) und nimmt damit Einfluss darauf, wie zukünftig produziert wird. Die technische Basis von Smart Factories sollen digital vernetzte Systeme und eine direkte Kommunikation zwischen Menschen, Maschinen und Produkten darstellen. Hierdurch soll eine weitgehend selbst organisierte Produktion realisierbar sein [Bun15]. Für eine flexible

und effiziente Produktion muss unter anderem die Kommunikation zwischen Menschen, Maschinen und Produkten und deren Aufgaben in einem Prozess verbunden werden, der innerhalb einer *Smart Factory* zentral modelliert, ausgeführt und verwaltet wird. Beispielsweise sollte dieser Prozess dynamisch auf Störungen oder Kundenwünsche reagieren und Anpassungen vornehmen können [KWH13]. Industrie 4.0 fordert daher intelligente und anpassungsfähige Maschinen, um so eine schnellere und flexiblere Produktion zu ermöglichen [KWH13].

## 1.1. Problemstellung

Die Umsetzung einer *Smart Factory* im Sinne von Industrie 4.0 erfordert ein System, das zwischen den Ebenen der Produktionsmaschinen und der Geschäftsplanung liegt. Ein sogenanntes *Manufacturing Execution System* (MES) definiert einen Prozess, der die gesamte Produktion eines Produktes begleitet und diese verwaltet [MFT09]. Es integriert idealerweise alle am Prozess beteiligten Maschinen und stellt eine Schnittstelle zum *Enterprise Ressource Planning* (ERP)-System dar, das die verfügbaren Prozesse im MES ansteuern kann. Ein solches System stellt durch seine Transparenz unter anderem eine große *IT-Compliance* mit zur Verfügung, d.h. es unterstützt Unternehmen, indem es gesetzliche und andere regulatorische Vorgaben erfüllt [SW12]. Besonders in der Pharmaindustrie ist dies von großer Bedeutung, da hier eine Vielzahl gesetzlicher Regulatorien existieren. Beispielsweise müssen alle Arbeitsschritte kontinuierlich dokumentiert werden, sodass im Falle einer Kontrolle oder einer Fehlfunktion der Prozessablauf lückenlos nachvollzogen werden kann [KWH13].

Mittelständische Pharmaunternehmen verwenden meist mehrere kleine Systeme oder SAP-Erweiterungen, um die erforderlichen Richtlinien, wie beispielsweise das *Electronic Batch Recording*[1] (EBR), abzuwickeln. Bei EBR handelt es sich um ein Verfahren, das jeden Arbeitsschritt an jeder *Batch*[2] von Medikamenten genau dokumentiert. In der Pharmaindustrie wird die Implementierung eines MES durch eine Reihe von strengen Richtlinien [Dep97] zusätzlich erschwert.

[1] zu dt. elektronische Batch Dokumentation
[2] zu dt. Serie: Eine große, aber begrenzte Anzahl desselben Produkts.

Zur Einführung eines MES stehen einem Unternehmen drei Möglichkeiten offen, die in Abbildung 1.1 veranschaulicht sind. Die erste Option ist ein neues und umfangreiches MES zu erwerben und an das Unternehmen anzupassen. IT-Unternehmen, wie *Werum*[3] oder *Rockwell Automation*[4] bieten diesbezüglich Softwarelösungen speziell für die Pharmaindustrie an [Vac11]. Die zweite Option ist die direkte Prozesssteuerung durch das ERP-System selbst. Aktuell besitzt SAP einen Markanteil von geschätzt 90% in der deutschen Pharmaindustrie und ca. 24% an Unternehmen weltweit [Col14]. Aber auch eine MES-Lösung von SAP ist je nach Umfang des Einsatzgebietes sehr kostspielig. Zuletzt bietet sich Unternehmen die Option, auf die MES-Lösung eines ihrer Maschinenhersteller zurückzugreifen. Allerdings bieten nur wenige Maschinenhersteller ein System an, das zusätzlich zu seiner standardmäßigen Bediensoftware über eine Prozessplanung auf Linienebene[5] verfügt.

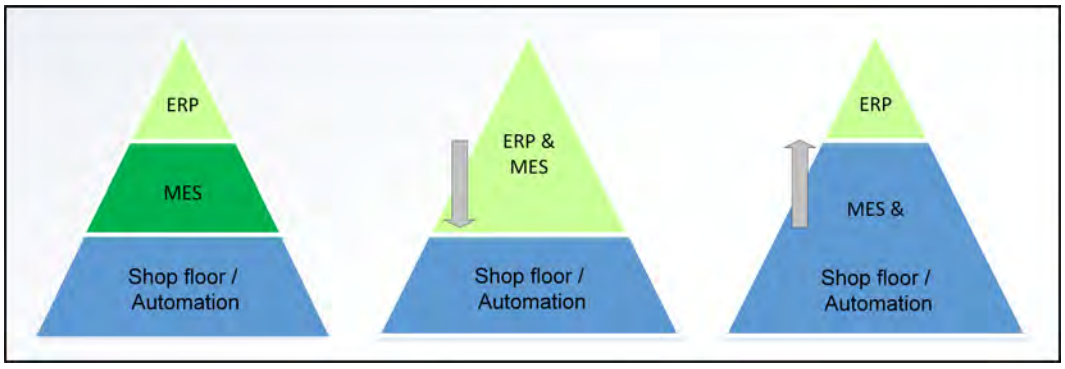


Abbildung 1.1.: Realisierungsmöglichkeiten eines MES

Im Zuge von Industrie 4.0 werden Maschinen intelligenter und enthalten mehr Software. Intelligenter werden sie dabei durch den Einsatz von Sensoren, die jegliche Werte der Maschine auslesen können, und eines umfangreichen Maschinenbediensystems, das die Sensoren ansteuert. Solche intelligenten Maschinen eröffnen eine Vielzahl von Möglichkeiten, um beispielsweise flexibel auf Änderungen im Produktionsablauf oder Fehlermeldungen der Maschine zu reagieren. Um das volle Potential dieser Möglichkeiten in der Produktion auszunutzen, müssen die Maschinen in die Verfahrensprozesse

[3] http://www.werum.com/en/pas-x-software/
[4] http://www.rockwellautomation.com/
[5] Bezeichnet die Produktionslinie, also alle Maschinen, die an der Produktion eines Batches beteiligt sind.

der Pharmaunternehmen integriert werden [KWH13]. Maschinenherstellern bietet sich die Gelegenheit, ihren Kunden eine erweiterte Version ihres Maschinenbediensystems zur Verfügung zu stellen, die es diesen ermöglicht, Verfahrensprozesse unter Einbezug der Maschinen zu modellieren und auszuführen. Diese *Bottom-Up-Integration*[6] der Maschinen bringt den Vorteil einer maschinennahen Prozessunterstützung und die direkte Prozessausführung auf den Maschinen mit sich, da die Maschinenbediensysteme bei der Konzeption miteinbezogen werden. Mittelständischen Pharmaunternehmen wird damit eine attraktive Alternative zu einem MES offeriert, da sie das Bediensystem des Maschinenherstellers bereits verwenden und Fachwissen aufgebaut haben.

Ein MES verfügt in der Regel über eine Reihe von Funktionen im Bereich von Produktion, Qualität und Personal, wie z.B. Betriebsdatenerfassung, Prozessdatenerfassung und Personaleinsatzplanung [MFT09]. Die Implementierung eines Systems, das alle Funktionen eines MES erfüllt, stellt eine komplexe und zeit- und kostenaufwändige Aufgabe dar. Bei der Konzeption eines Systems zur Modellierung und Ausführung von Verfahrensprozessen stellt sich daher zu Beginn das Problem, welche Komponenten eines MES integriert werden müssen, um ein möglichst gutes Kosten-Nutzen-Verhältnis zu erreichen.

Bei der Konzeption einer Modellierungsumgebung muss zu Beginn berücksichtigt werden, welche Rahmenbedingungen an die Verfahrensprozesse gestellt sind. Zum einen muss die Erstellung der Prozesse für den Anwender einfach gestaltet sein, d.h. er soll diese möglichst ohne vorhergehende Schulung modellieren können. Zum anderen sind die Prozesse abstrakt zu halten: bei der Modellierung wird noch nicht festgelegt, an welcher Maschine oder mit welchen Materialien gearbeitet wird, um bei der Produktion eine hohe Flexibilität zu erreichen. Bei der Umsetzung des Konzepts tritt die Problematik auf, dass bei der Prozessmodellierung alle an der Produktion beteiligten Maschinen integriert werden müssen. Dies bedeutet, dass sowohl für die Modellierung als auch für die Ausführung von Prozessen eine universale Schnittstelle des konzipierten Systems zu jeder beteiligten Maschine benötigt wird.

Für die Anwendung in pharmazeutischen Produktionslinien müssen sich Prozesse außerdem an den Richtlinien der *U.S. Food and Drug Administration* (FDA) orientieren,

[6] zu dt. Integration von unten nach oben

die durch die *Code of Federal Regulations Title 21* (CFR21) vorgegeben sind. Im elften Abschnitt der CFR21 ist angegeben, welche Bedingungen ein System im Bezug auf elektronische Daten und Unterschriften erfüllen muss [Dep97].

Es stellt sich folglich die komplexe Problemstellung, die einfache Modellierung von komplexen, maschinennahen Verfahrensprozessen zu ermöglichen, die im Hintergrund von vielen Parametern abhängen und zudem die gesetzlichen Rahmenbedingungen erfüllen müssen. Außerdem sollen Maschinen, bzw. die Maschinenbediensysteme, in den Prozess integriert werden.

## 1.2. Zielsetzung

Ziel dieser Arbeit ist der Entwurf eines Konzepts, das konkret und beispielhaft die Modellierung und Ausführung von Verfahrensprozessen in der pharmazeutischen Verpackungsindustrie ermöglicht. Dafür werden die drei zentralen Komponenten Prozessmanagementsystem, Modellierungs- und Ausführungsumgebung behandelt und integriert. Die Kernfunktion, die im Rahmen dieser Arbeit konzipiert wird, ist die Unterstützung von automatisiert ablaufenden Prozessen durch ein passendes Prozessmanagementsystem. Innerhalb dieser Prozesse werden die an der Produktion beteiligten Maschinen dann integriert. Die Bottom-Up-Integration des entstehenden Systems geht von dem bestehenden Maschinenbediensystem *SmartControl* der Firma Uhlmann (siehe Kapitel 2.3) aus, das in dieser Arbeit beispielhaft für die Realisierung des Konzepts verwendet wird.

Der erste Schritt dieser Arbeit ist daher die Ermittlung der Anforderungen an das entstehende System in Rücksprache mit Fachpersonal der Firma Uhlmann. Es gilt dabei ebenfalls zu ermitteln, welches bestehende Prozessmanagementsystem für die Ausführung der Verfahrensprozesse in Frage kommt. Für die Modellierungs- und die Ausführungsumgebung werden unter anderem Mockups erstellt, die Inhalt und Funktionalität der Komponenten verdeutlichen und gleichzeitig zum Abgleich der Anforderungen dienen.

Ausgehend von dem entworfenen Konzept werden die Modellierungs- und die Ausführungsumgebung in einem *Proof-of-Concept Prototyp* implementiert. Der Prototyp bietet hierbei die Möglichkeit, Verfahrensprozesse, die die Arbeitsabläufe auf den Smart-

Control Maschinen beinhalten, zu modellieren, die resultierenden Prozessmodelle zu speichern und letztlich zu instanziieren. Die dadurch entstehenden Prozessinstanzen werden auf dem zuvor ermittelten Prozessmanagementsystem ausgeführt. Die Modellierung von Arbeitsabläufen für das Maschinenbediensystem SmartControl steht dabei für Uhlmann im Vordergrund, um ihren Kunden eine flexiblere Anpassung ihrer Verpackungsmaschinen zu ermöglichen.

## 1.3. Aufbau der Arbeit

Auf dieses Kapitel folgt ein Grundlagenkapitel (Kapitel 2), das die zentralen Begrifflichkeiten des Prozessmanagements erläutert, MES-Systeme beschreibt und letztlich das System SmartControl der Firma Uhlmann vorstellt.

In Kapitel 3 werden die Anforderungen an das zu entwickelnde Konzept vorgestellt. Hierzu wird zunächst durch ein Szenario der Kontext veranschaulicht. Durch User Stories werden dann die Anforderungen verdeutlicht, bevor die funktionalen und nichtfunktionalen Anforderungen im Detail aufgelistet und nach Priorität bewertet werden.

Das Konzept einer Modelierungs- und Ausführungsumgebung für Verfahrensprozesse ist in Kapitel 4 beschrieben. Dabei wird zuerst ein Überblick über die Zusammenhänge der Komponenten gegeben und anschließend alle Komponenten mit ihren Funktionalitäten beschrieben. Kapitel 4 beinhaltet außerdem eine detaillierte Beschreibung der Anbindung des Bediensystems SmartControl und einige Konzeptentscheidungen im Bezug auf Usability.

Anschließend wird in Kapitel 5 die prototypische Implementierung des Konzepts vorgestellt. Hierbei wird die Ausgangsbasis erläutert und die Implementierung der Client-Anwendung, der Server-Anwendung und der Kommunikationskomponente beschrieben.

Kapitel 6 vergleicht die Anforderungen mit dem entworfenen Konzept und Prototypen und veranschaulicht die Schnittstelle zu SmartControl. Außerdem nennt es verwandte Arbeiten der Bereiche MES, Prozessmodellierung und Prozesskonfiguration und stellt ihre Zusammenhänge zu dieser Arbeit her.

Abschließend erfolgt in Kapitel 7 eine Zusammenfassung der Arbeit und ein detaillierter Ausblick, der Anregungen für Erweiterungen des Konzept, sowie der Implementierung beinhaltet.

# 2

# Grundlagen

In diesem Kapitel werden die grundlegenden Aspekte beschrieben, auf denen diese Arbeit aufbaut. Zunächst folgen für diese Arbeit relevante Begriffsdefinitionen aus dem Bereich des Prozessmanagements. Anschließend folgt eine Definition und Beschreibung von *Manufacturing Execution Systemen*. Außerdem wird das System SmartControl der Firma Uhlmann beschrieben, da dieses bei der Konzeption des hier vorgestellten Systems miteinbezogen und als Beispiel für ein Maschinenbediensystem verwendet wird.

## 2.1. Prozessmanagement

Unter Prozessmanagement werden Konzepte, Methoden und Techniken zur Unterstützung des Designs, der Administration, der Konfiguration und der Analyse von Prozessen verstanden [Wes12]. In Kapitel 1.2 wurde bereits hervorgehoben, dass ein Prozessmanagementsystem (PrMS) ein notwendiger Bestandteil dieser Arbeit ist. Durch ein Prozessmanagement sollen Menschen und Maschinen enger miteinander verbunden werden und durch den Einsatz eines PrMS zur Unterstützung von Verfahrensprozessen die Lücke zwischen Geschäftsplanung und der Produktion geschlossen werden [Wes12]. Dies bringt den Vorteil, dass Prozesse unter Einbezug aller an der Produktion beteiligten Faktoren modelliert und anschließend ausgeführt werden können.

In der Industrie existieren zur Prozessmodellierung und im Prozessmanagement eine Vielzahl unterschiedlicher Bezeichnungen. Um Synonyme zu verhindern orientiert sich diese Arbeit an der Terminologie von [RW12]. Außerdem werden die Begriffe Arbeitsplan,

Auftrags- und Maschinenparameter definiert, wie sie in der Firma Uhlmann verwendet werden.

**Prozess:** *Ein Prozess ist eine Abfolge von Aktivitäten, der mindestens ein Start- und ein Endereignis enthält, von Menschen oder Maschinen ausgeführt wird und ein gesetztes Ziel erreichen soll* [FR10].
Ein Prozess kann beispielsweise einen Arbeitsablauf zur Vorbereitung einer Maschine beschreiben, der von den Maschinenoperatoren oder automatisch ausgeführt wird, bevor eine Maschine zu arbeiten beginnt.

**Prozessmodell:** *Ein Prozessmodell ist die Repräsentation eines Prozesses, durch eine spezifizierte Modellsprache* [RW12].
In dieser Arbeit wird die *Business Process Model Notation* (BPMN 2.0) [FR10] zur Modellierung von Prozessen verwendet. Dabei handelt sich um eine durch die Object Management Group (OMG) standardisierte grafische Prozessnotation.

**Prozessinstanz:** *Die Prozessinstanz stellt den aktuellen Stand der Ausführung eines Prozessmodells dar* [RW12].
In der Terminologie von Uhlmann wird eine Prozessinstanz auch als Auftrag bezeichnet, da eine Prozessinstanz in der Industrie immer einen Produktionsauftrag darstellt. So wird z.B. für die Herstellung von 10.000 Tabletten ein Auftrag erstellt, welcher auf einem Prozessmodell beruht und als Prozessinstanz ausgeführt wird.

**Aktivität:** *Eine Aktivität kann entweder atomar oder komplex sein* [RW12].

- **Atomare Aktivität:** *Eine atomare Aktivität beschreibt eine menschliche oder programmatische Aufgabe. Sie besteht aus Eingabewerten, die der Benutzer im Zuge der Aufgabe ausfüllen oder bestätigen muss, und aus Ausgabewerten, die durch die Erfüllung der Aufgabe entstehen.*
  Ein Beispiel solch einer Aktivität wäre die Aufgabe *Temperatur prüfen* (Abbildung 2.1): Diese kann sich auf Maschinenparameter (z.B. IST-Temperatur) und Auftragsparameter (z.B. SOLL-Temperatur) beziehen und muss vom Benutzer bestätigt werden, um eine positive Ausgabe weiter zu geben. In der Terminologie von Uhlmann wird eine atomare Aktivität als *Verfahrensteilschritt* bezeichnet.

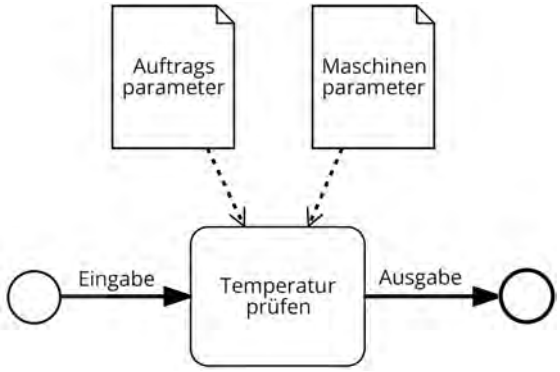


Abbildung 2.1.: Beispiel für eine Aktivität in einem Prozess

- **Komplexe Aktivität:** Bei einer komplexen Aktivität handelt es sich um einen Subprozess, d.h. einen eingebetteten weiteren Prozess. Subprozesse ermöglichen damit die hierarchische Schachtelung von Prozesses und von Aktivitäten.
  In der Terminologie von Uhlmann spricht man hier von einem *Verfahrensschritt*: Diese werden zur Kapselung aller Aktivitäten, die auf einem Arbeitsplatz ausgeführt werden, eingesetzt.

**Prozessmanagementsystem:** *Ein Prozessmanagementsystem (PrMS) ermöglicht die Modellierung, Verwaltung und Ausführung von Prozessen. Zur Laufzeit der Prozessinstanz verteilt es automatisch Aktivitäten an Benutzer sobald diese ausführbar sind und es kann atomare und komplexe Aktivitäten automatisch ausführen* [RW12].
Die speziellen Anforderungen, die diese Arbeit an ein PrMS stellt, werden in Kapitel 3.3 erläutert.

**Arbeitsplan:** Ein Arbeitsplan ist ein spezifisches Prozessmodell. Es enthält eine Abfolge von Subprozessen, die sich jeweils auf eine Maschine der Linie beziehen. In einem Arbeitsplan werden Subprozesse beschrieben, die wiederum aus atomaren Aktivitäten bestehen. Ein Arbeitsplan wird bei der Erstellung parametrisiert und stellt damit eine Vorlage dar, aus der verschiedene Aufträge generiert werden können.
Ein Beispiel für einen Arbeitsplan ist „Ressourcen für Maschine bestellen“. Abhängig von der Art der Maschine, die als Parameter festgelegt wird, kann aus diesem Arbeitsplan für jede vorhandene Maschine ein Auftrag erstellt werden.

**Auftragsparameter:** In einem Prozessmodell werden Auftragsparameter definiert. Diese Parameter werden innerhalb des Prozessmodells für eine abstrakte Beschreibung von Elementen verwendet. Bei der Instanziierung eines Prozesses, bzw. der Generierung eines Auftrags, müssen diese Parameter mit Werten gefüllt werden (z.B. 'Artikelnummer' = '123456').

**Maschinenparameter:** Eine Maschine besitzt eine Vielzahl von Sensoren, die über ein Bediensystem (z.B. SmartControl) ausgelesen werden können. Diese Werte werden als Maschinenparameter bezeichnet.

## 2.2. Manufacturing Execution System

Ein *Manufacturing Execution System* (MES) dient der Unterstützung der Produktion und beinhaltet Funktionen zur Produktionsplanung, -dokumentation und -steuerung [MFT09]. Hierbei stellt ein MES Schnittstellen zu Systemen wie ERP und SCADA[1] her, um Informationen auszutauschen und eine dynamische und flexible Produktion zu ermöglichen.

Mit der Bottom-Up Integration eines Maschinenbediensystems, die in Kapitel 1.1 bereits diskutiert wurde, ist konkret der Ausbau des Systems auf das dritte Level des *ANSI/ISA 95 Standards* gemeint (siehe Abbildung 2.2). Dieser Standard wurde von der *International Society of Automation*[2] erstellt und legt unter anderem detailliert fest, welche Daten zwischen einem MES und ERP-System ausgetauscht werden. Er definiert außerdem, aus welchen Ebenen (Levels) die Produktion besteht [Bra05]. *Ebene 0* ist die eigentliche Produktion auf der Maschine. *Ebene 1* definiert Aktivitäten, die den Produktionsprozess durch Sensoren erfassen und manipulieren können. Auf *Ebene 2* befinden sich die Aktivitäten, die für die Überwachung, Aufsichtskontrolle und Automatisierung des Produktionsprozesses zuständig sind. Hier ist unter anderem das EBR-System eingeordnet, das auch in der Pharmaindustrie eine wichtige Rolle innehat. Auf *Ebene 3* wird der Prozess des Produkts verwaltet und von Anfang bis Ende mitverfolgt. Wichtige Aufgaben hierbei sind die Verwaltung der Dokumentionen und die Prozessoptimierung. *Ebene 4*

[1] Supervisory Control and Data Acquisition: System zur Steuerung und Datensammlung an Maschinen.
[2] https://www.isa.org/

befasst sich mit der Materialplanung, von der Bestellung bis hin zur Lieferung, weswegen auf dieser Ebene auch das ERP-System angesiedelt ist. Folglich werden Aufträge für die Produktion auf dem vierten Level angelegt und starten die auf Level drei ablaufenden Prozesse.

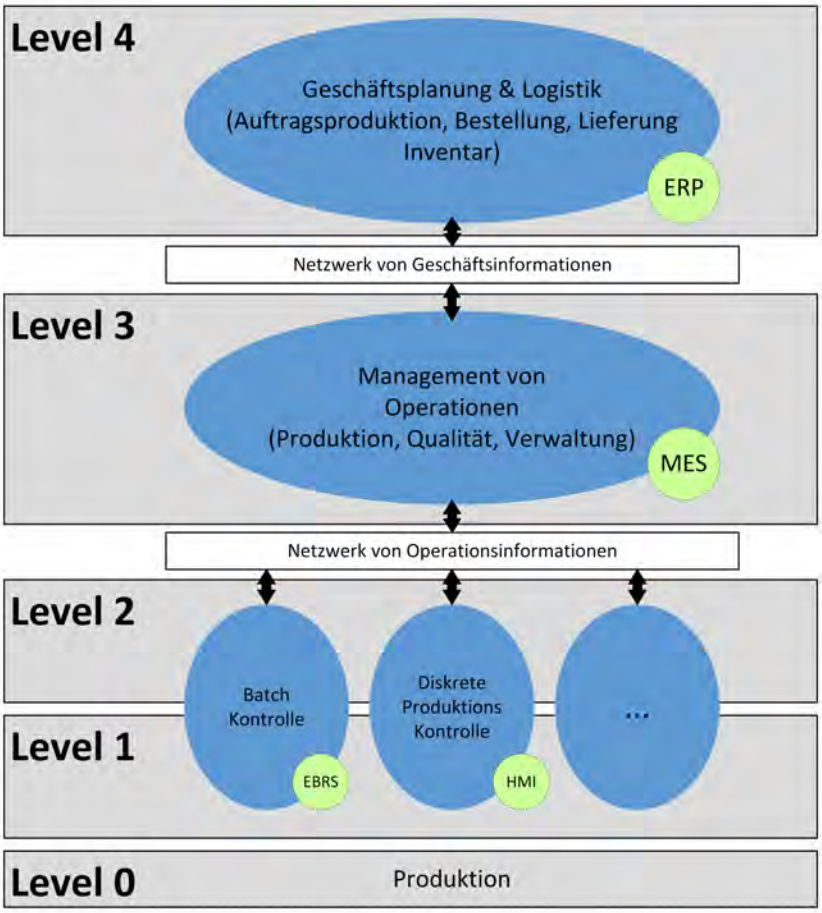


Abbildung 2.2.: ISA 95 Levels nach [Bra05]

Zusätzlich zu seinen Kernfunktionen, muss ein MES eine Material- und Ressourcenplanung sowie ein Echtzeit-Datenmanagement beinhalten, um rechtzeitig auf Maschinenmeldungen reagieren zu können. Es wird ein Informationssystem benötigt, das sich sowohl um die Beschaffung als auch um die Verteilung von Informationen, wie beispielsweise den Status einer Maschine, kümmert. Außerdem muss ein MES Compliance-

Management unterstützt werden, um auf die wachsende Menge an Richtlinien und Regularien zu reagieren [MFT09].

## 2.3. SmartControl

Uhlmann produziert seit 1963 Verpackungsmaschinen für die Pharmaindustrie und kann damit als Maschinenbauunternehmen charakterisiert werden. 2012 führte Uhlmann das Blister Express Center 500 ein, das eine flexible Verpackungslinie darstellt, die bis zu 500 Faltschachteln pro Minute verpackt. Das Bedien- und Steuerungssystem (Human-Machine-Interface, HMI) für diese Verpackungsmaschinen ist SmartControl. Es wird von der Software-Abteilung der Firma Uhlmann eigenständig entwickelt und erschien im Oktober 2012 in der Version 1.0, der derzeitige Stand ist Version 2.2 [Uhl14]. SmartControl ist prozess-orientiert, d.h. es bietet eine systematische Führung durch jeden Bedienschritt und liefert dem Maschinenoperator alle relevanten Informationen der Maschine. Dafür stellt es drei Perspektiven bereit:

1. Die **Maschinen-Perspektive**, bei der es sich um eine grafische und stationsorientierte Darstellung der Maschine handelt.
2. Die **Assistenten-Perspektive**, die den Maschinenoperator durch das Einrichten der Maschinen führt sowie ihn bei der Produktionsausführung unterstützt.
3. Die **Apps-Perspektive**, die administrative Funktionen (z.B. Format Management, Audit Trail, Geräte Setup, etc.) bereitstellt.

Eine wichtige Rolle bei SmartControl spielt eine hierarchische Datenstruktur - der *Maschinenbaum*. Dieser repräsentiert die Struktur der gesamten Verpackungsmaschine und bietet Zugriff auf alle Systemparameter. Die Parameter sind über einen Pfad exakt im Maschinenbaum definiert. So kann beispielsweise die Temperatur einer Heizstation über den Pfad „**.FormingFillZone.HeatingStation.LowerHeatingCircuit1.Temperature*“ bezogen werden.

Für diese Arbeit hat die Assistenten-Perspektive einen hohen Stellenwert, da sie verwendet wird, um Maschinenoperatoren durch Verpackungsprozesse zu leiten und ihnen die

Steuerung der Maschine zu erleichtern. Dabei legt ein Assistent die genaue Ablaufbeschreibung eines Auftrags auf einer bestimmten Maschine fest. Der Maschinenoperator bedient die Maschine, führt die festgelegten Schritte nacheinander aus, füllt Parameter und bestätigt diese. Dabei ist das Assistentensystem sehr statisch, d.h. die vorhandenen Assistenten müssen bereits beim Starten von SmartControl vorhanden sein und können während der Ausführung auch nicht mehr variiert werden. Die Assistenten liegen dabei als XML-Dokumente vor und werden von Fachpersonal der Firma Uhlmann in einem Texteditor angelegt. Abbildung 2.3 zeigt SmartControl in der Assistenten-Perspektive, durch die der Maschinenoperator direkt an der Maschine durch die Produktion geleitet wird. Das Bediensystem ist in drei Bereiche aufgeteilt. Der erste stellt Informationen zur Maschine bereit, an der sich der Maschinenoperator momentan befindet. Der zweite stellt den Assistenten dar, der in diesem Beispiel die Eingabe mehrerer Batch-Parameter verlangt. Im dritten Bereich ist die Navigation des Bediensystems dargestellt und der Maschinenoperator kann hier zwischen der Assistenten-, Maschinen- und Apps-Perspektive wechseln.

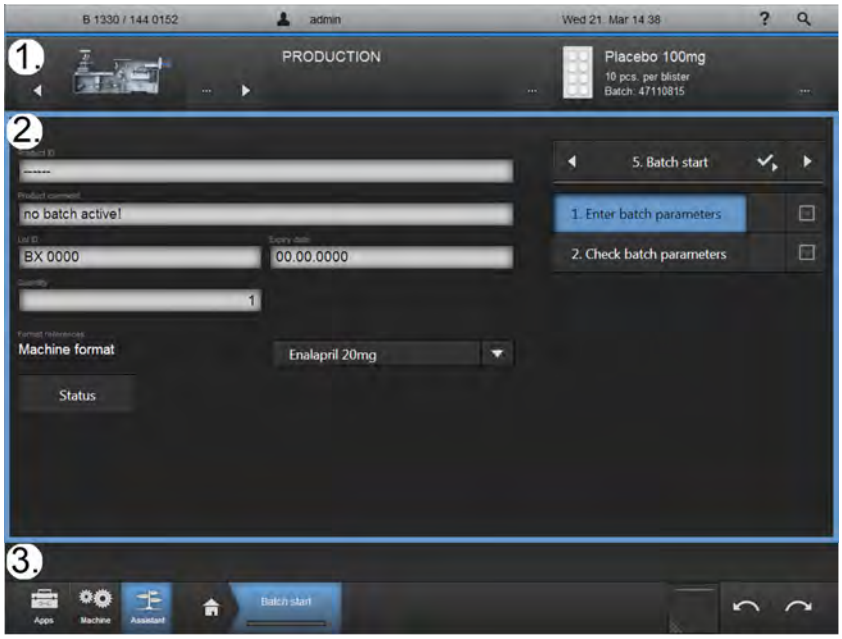


Abbildung 2.3.: SmartControl - Beispiel eines Assistenten

### Architektur von SmartControl

SmartControl ist selbst als webbasierte Anwendung entwickelt worden und seine Architektur ist streng nach dem *Model-View-Controller-Prinzip* (siehe [ACKM04]) aufgebaut, wie in Abbildung 2.4 zu sehen ist. Die Software ist in der Programmiersprache Python [Pyt15] geschrieben und verwendet zur Kommunikation innerhalb des Systems CORBA[3] [ACKM04] als Middleware. Außerdem verfolgt SmartControl eine modellgetriebene Architektur, d.h. es gibt eine Typ-Bibliothek als Domain Specific Language (DSL), die Typ-Systeme für unterschiedliche Schichten und den Code für die Interprozesskommunikation, SPS[4] und Persistierung generiert. Die Implementierung von SmartControl besteht dabei nur aus Open Source Produkten. Neben Python kommen bei der Darstellung im Browser JavaScript und das *Qooxdoo Framework* [Qoo15] zum Einsatz, zur Modellanbindung das *Django-Projekt* (OR-Mapper[5]) und *Omniorb* (CORBA). Assistenten sind außerhalb von SmartControl als XML-Dokumente definiert. Beim Starten von SmartControl werden aus diesen XML-Dokumenten Python-Objekte generiert. Auf diese Weise wird in der Anwendung der Assistent beim Starten des Server statisch aufgebaut und kann zur Laufzeit nicht geändert werden.

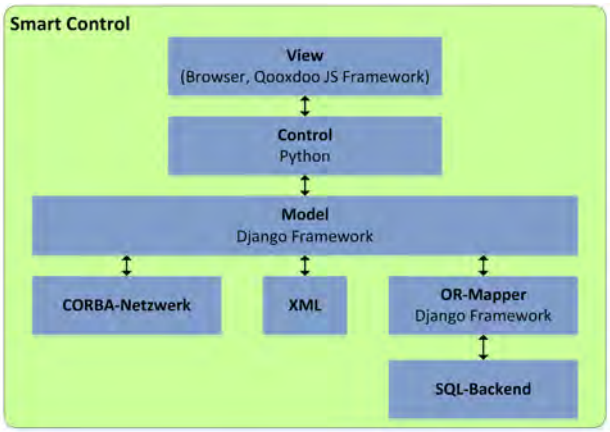


Abbildung 2.4.: SmartControl - Architektur

[3] Common Object Request Broker Architecture: Middleware-Infrastruktur zur Unterstützung der Interfunktionsfähigkeit von Objekten.
[4] Speicherprogrammierbare Steuerung
[5] Objektrelationale-Mapper: Dienen zur Anbindung relationaler Datenbanken an objektorientierte Software.

# 3

# Anforderungen

Dieses Kapitel stellt ein Szenario vor, das die Problemstellung der Arbeit beispielhaft aufzeigt. Das Szenario dient als Basis für die Erhebung der Anforderungen. In Kapitel 3.2 werden dann, in Form von *User Stories*, die Anforderungen an das entstehende System aus Sicht der Anwender beschrieben. Kapitel 3.3 definiert die funktionalen und nicht-funktionalen Anforderungen.

## 3.1. Szenario

In Form eines Szenarios soll aufgezeigt werden, welchen Zweck die Prozessmodelle, die in der Modellierungsumgebung erstellt werden sollen, erfüllen zu haben und welche Angaben hier zwingend zu vorzunehmen sind. Zudem wird gezeigt, welche Angaben ein Prozessmodell benötigt, um instanziiert zu werden.

Das Szenario bezieht sich auf ein abstraktes Beispiel der Verpackungsmaschinenassistenten des Bediensystems SmartControl (siehe Kapitel 2.3) - es bezieht sich speziell auf einen Verpackungsprozess für Tabletten. Dabei beruht es auf Interviews mit Fachpersonal der Firma Uhlmann. Außerdem orientiert es sich an bestehenden Assistenten des Bediensystems SmartControl. Das Szenario wird sowohl aus fachlicher als auch aus technischer Sicht betrachtet. Hierbei wird es in einen übergeordneten Prozess auf Linienebene eingeordnet und zur Verdeutlichung der Prozessmodellierung und Prozessausführung wird anschließend jeweils ein konkretes Fallbeispiel aufgeführt.

### 3.1.1. Fachliche Sicht

Das entstehende System soll zur Modellierung und Ausführung von Verfahrensprozessen dienen. Dabei enthält der Verfahrensprozess jeden Aspekt der Produktion, von der Vorbereitung der Ausgangsmaterialien bis zur Verpackung des Endprodukts. In diesem Szenario werden die Abläufe der Herstellung der Tabletten abstrahiert und es wird im Detail auf die Verpackung von Tabletten (Blisterverpackung) eingegangen. Die Personengruppen *Editoranwender*, *Operator* und *Pharmazeut* stellen die Akteure dar, die mit der Modellierungs- und Ausführungsumgebung in diesem Kontext interagieren. Beim Editoranwender handelt es sich um einen qualifizierter Mitarbeiter, der den Verfahrensprozess kennt und das Prozessmodell für die Herstellung und Verpackung von Tabletten modelliert. Anschließend prüft ein Pharmazeut die Qualität des Prozesses und beurteilt dessen Richtigkeit. Die Operatoren führen die im Prozessmodell definierten Aktivitäten aus. Dabei befinden sich die Operatoren direkt an den Maschinen. Der Verfahrensprozess, den der Editoranwender modelliert, ist in Abbildung 3.1 als übergeordneter Gesamtprozess dargestellt und definiert jede Maschine die an der Produktion beteiligt ist. Die verschiedenen Maschinen, bzw. Arbeitsvorgänge, die der Verpackung vorausgehen, sind hierbei in dem Subprozess „Tablettenherstellung“ aggregiert.

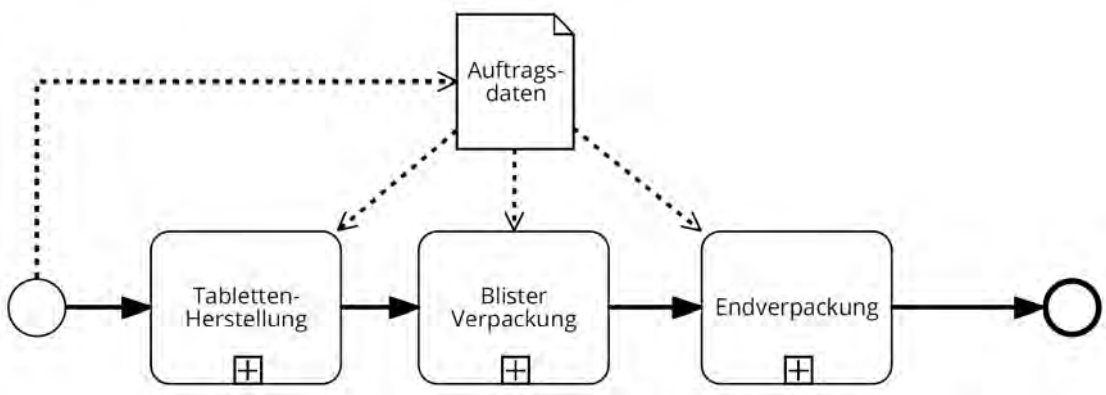


Abbildung 3.1.: Szenario - Gesamtprozess aus Unternehmenssicht

Zusätzlich zu den Subprozessen gibt der Editoranwender im Prozessmodell aus Abbildung 3.1 an, welche Auftragsdaten im Prozess benötigt werden. Die Auftragsdaten

werden bei der Instanziierung eines Prozesses durch das ERP-System zur Verfügung gestellt. Sie beinhalten unter anderem Auftragsnummer, Batchnummer, Vorgangsnummer, Artikelstückliste, Menge, frühester Start, spätestes Ende und Mindesthaltbarkeitsdatum. Es gibt allerdings auch Auftragsdaten, die nicht während der Instanziierung, sondern zur Ausführungszeit der Prozessinstanz gesetzt werden. Als Beispiel können diesbezüglich die Arbeitsplatz-ID des Mitarbeiters, der IST-Start und das IST-Ende genannt werden.

Abbildung 3.2 zeigt das Prozessmodell des Subprozesses „Blisterverpackung“. Hier definiert der Editoranwender die Aktivitäten, die der Operator an einer Blistermaschine der Firma Uhlmann zur Verpackung von Tabletten ausführen muss. Zuerst muss der Operator das Format der Maschine im Subprozess „Formatwechsel“ einstellen und die Produktion starten, woraufhin die Maschine mit der Verpackung der Tabletten beginnt. Ist die Verpackung abgeschlossen, deaktiviert der Operator die Maschine und beginnt mit dem Subprozess „Reinigung“.

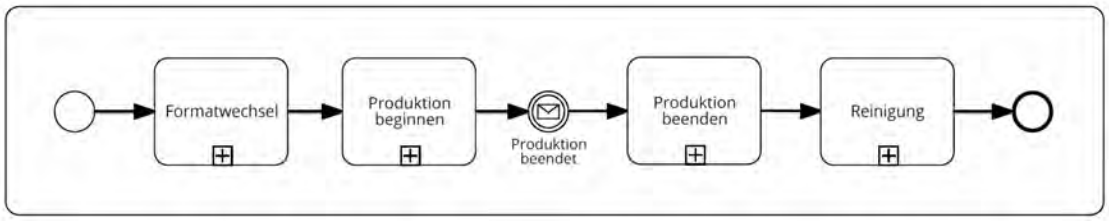


Abbildung 3.2.: Szenario - Subprozess „Blisterverpackung“ aus Steuerungssicht

Der Subprozess „Formatwechsel“ wird in Abbildung 3.3a genauer dargestellt. Das Prozessmodell beginnt mit der Aktivität zur Vorbereitung der Station, an der sich die Maschine befindet, sowie der Bereitstellung der Faltschachteln und Prospekte. Danach werden die Faltschachteln in die Maschine eingefahren und Vorabtests durchgeführt, die den ordnungsgemäßen Ablauf garantieren sollen. Anschließend wird ein weiterer Test anhand einer gültigen *Standard Operating Procedure*[1] (SOP) durchgeführt, um die Qualität der verpackten Blister zu überprüfen. Der Operator bekommt zu jeder Aktivität eine Aufgabenbeschreibung am Maschinenbediensystem angezeigt und bestätigt diese durch eine elektronische Unterschrift. Bei der Befüllung der Maschine mit den Packmit-

[1] zu dt. Standardvorgehensweise

teln vergleicht er mithilfe eines Barcodescanners, ob die richtigen Faltschachteln und Prospekte eingelegt wurden.

Der Subprozess „Reinigung“ beinhaltet das Leerlaufen der Maschine, das Entfernen der Packmittel und das Leeren der Abfallbehälter (siehe Abbildung 3.3b). Dabei bezieht sich der Operator wieder auf eine gültige SOP, die in SmartControl als verlinktes Dokument angegeben ist.

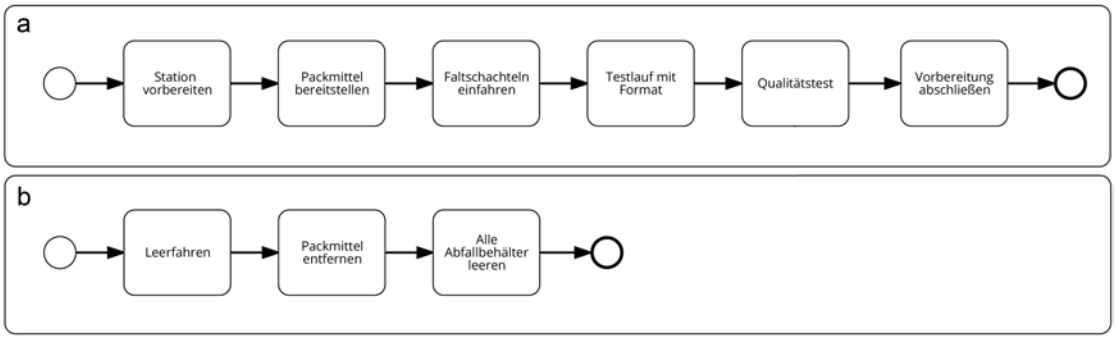


Abbildung 3.3.: Szenario - Subprozesse Formatwechsel und Reinigung aus Steuerungssicht

Um zu verdeutlichen, wie genau das Szenario mit dem zu entwickelnden System in Verbindung steht, werden im Folgenden zwei Fallbeispiele angegeben. Das erste Fallbeispiel beschreibt nochmals im Detail das Vorgehen des Editoranwenders bei der Modellierung des Verfahrensprozesses. Das zweite beschreibt die Erstellung einer Prozessinstanz, beruhend auf den Prozessmodellen des Szenarios, und deren Ausführung.

**Fallbeispiel: Modellierung des Verfahrensprozess zur Tablettenherstellung**

Um den Verfahrensprozess zur Tablettenherstellung zu modellieren, erstellt der Editoranwender zunächst den Gesamtprozess aus Unternehmenssicht. Hierbei modelliert er ein Prozessmodell wie es in Abbildung 3.1 dargestellt ist. Die Auftragsdaten modelliert er dabei nicht als Datenobjekt im Prozessmodell, sondern als Auftragsparameter in einer Tabelle, die bei der Instanziierung des Prozesses gefüllt werden. Dabei markiert er einige Auftragsparameter, wie z.B. Auftragsnummer und Artikelstückliste als Pflichtfelder, während Auftragsparameter wie frühester Start optional sein können. Nachdem er den

Gesamtprozess modelliert hat, spezifiziert er die Subprozesse. Die Modellierung der Subprozesse findet aus der Steuerungssicht statt, d.h. der Editoranwender gibt genau an, welche Arbeitsschritte durch den Operator ausgeführt werden müssen. Zu diesem Zweck erstellt er für jede Aktivität ein Formular, welches in SmartControl als Assistent bearbeitet wird. Im konkreten Beispiel des Subprozesses „Formatwechsel“ hinterlegt er die Aktivitäten aus Abbildung 3.3 als Anweisungen in einer Formularansicht, wie es in Abbildung 3.4 dargestellt ist. Der Operator soll die Aktivitäten „Station vorbereiten“, „Packmittel bereitstellen“, „Faltschachteln einfahren“, „Testlauf mit Format“, „Qualitätstest“ und „Vorbereitung abschließen“ ausführen und bestätigen. Der Editoranwender legt dabei außerdem fest, dass zur Aktivität „Packmittel bereitstellen“ die Artikelnummern der Faltschachteln und Prospekte aus der Artikelstückliste benötigt werden. Zusätzlich soll der Subprozess „Formatwechsel“ mit einer elektronischen Unterschrift abgeschlossen werden, damit der Operator den ordnungsgemäßen Abschluss seiner Arbeit bestätigt und dies aus Gründen der Rechtssicherheit dokumentiert ist.

Hat der Editoranwender alle Aktivitäten modelliert, wird das Prozessmodell von einem Pharmazeuten der Qualitätssicherung geprüft. Findet dieser Fehler im Prozessmodell, weist er dieses zurück und sendet einen Prüfbericht an den Editoranwender, der daraufhin das Prozessmodell mithilfe des Prüfberichts korrigiert. Der Pharmazeut prüft das Prozessmodell erneut und gibt es schließlich zur Instanziierung frei.

**Fallbeispiel: Instanziierung eines Prozessmodells zur Tablettenherstellung**

Als Beispiel zur Instanziierung des durch den Editoranwender modellierten Prozessmodells sollen in einem Pharmaunternehmen innerhalb eines Monats 40.000 Packungen Kopfschmerztabletten hergestellt werden. Dies wird in einem ERP-System geplant, das daraufhin dem Prozessmanagementsystem meldet, dass es eine Prozessinstanz erstellen soll. Die Prozessinstanz beruht auf den Prozessmodellen, die im Szenario angeben wurden (siehe Abbildung 3.1 - 3.3). Die Auftragsdaten, die zur Instanziierung verwendet werden, sind in Tabelle 3.1 vereinfacht dargestellt. Die Prozessinstanz steuert mit dessen Hilfe zuerst die Herstellung der einzelnen Tabletten. Der Subprozess „Blisterverpackung“ wird gestartet, sobald 840.000 Tabletten produziert wurden, da eine

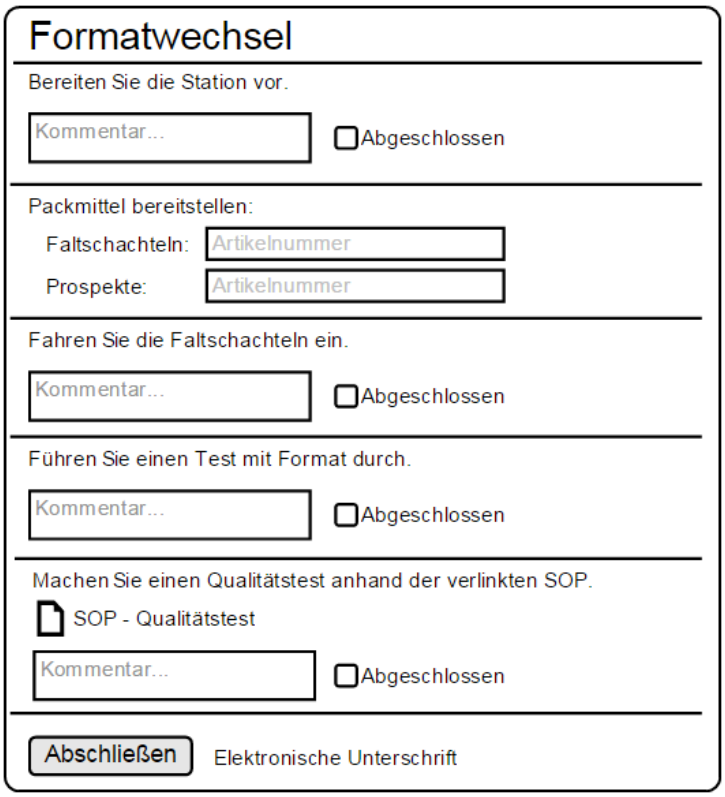


Abbildung 3.4.: Szenario - Formularansicht des Subprozess „Formatwechsel“

Tabelle 3.1.: Beispielhafte Auftragsdaten zur Instanziierung eines Prozessmodells

| | | | | |
|---|---|---|---|---|
| **Auftragsnummer** | 100100 | | | |
| **Batchnummer** | 1 | | | |
| **Vorgangsnummer** | 123456 | | | |
| **Menge** | 40.000 | | | |
| **Frühester Start** | 21.07.2015 | | | |
| **Spätestes Ende** | 22.08.2015 | | | |
| **Mindesthaltbarkeitsdatum** | 01.01.2017 | | | |
| **Artikelstückliste** | **Pos-Nr.** | **Artikelnr** | **Wert** | **Menge** |
| | 010 | 4711 | Ibuprofen | 600mg |
| | 010 | 4801 | Hypromellose | ∼ |
| | 010 | 4802 | Macrogol 400 | ∼ |
| | 010 | 4803 | Macrogol 6000 | ∼ |
| | 010 | 4804 | Magnesium stearat | ∼ |
| | 010 | 4805 | Maisstärke | ∼ |
| | 020 | 111456 | Faltschachteln | 40.000 |
| | 030 | 111478 | Prospekte | 40.000 |

Packung 20 Tabletten enthält und mit einem Ausschuss von ca. 5% gerechnet wird. Ist der Subprozess gestartet, wird die Aktivität „Formatwechsel“ als SmartControl-Assistent an eine passende Blistermaschine gesendet und der Operator wird darüber informiert, dass er die Aktivität per Assistenten zu bearbeiten hat.

Der Operator wechselt entsprechend dann in die Assistentenansicht seiner Blistermaschine. Dort bereitet er die Stationen vor und bestätigt dies abschließend. Sofern er von der Standardvorgehensweise abweicht oder etwas Ungewöhnliches auftritt, kann er dies in einem Kommentarfeld dokumentieren. Im nächsten Arbeitsschritt stellt er die Faltschachteln und Prospekte bereit (siehe Abbildung 3.4). Dazu wählt er zunächst das Feld für die Faltschachteln aus, scannt dann den Barcode der bereitgestellten Faltschachteln ein und macht anschließend dasselbe für die Prospekte. Auf diese Weise kann über die Auftragsdaten bestätigt werden, dass es sich um die richtigen Artikel handelt. Der Operator führt alle Arbeitsschritte in seinem Assistenten aus und unterschreibt das Formular elektronisch. Die Aktivität ist beendet und die Prozessinstanz stellt die nächste Aktivität bereit, die den Operator anweist, die Verpackung an der Blistermaschine einzuleiten.

### 3.1.2. Technische Sicht

In Anbetracht der hier vorliegenden Aufgabenstellung - der Modellierung und Ausführung von Verfahrensprozessen - stellen sich einige technische Bedingungen an das Szenario. Diese betreffen die Komponenten, Funktionen und Schnittstellen der Modellierungs- und Ausführungsumgebung, die in Kapitel 3.1.1 aus fachlicher Sicht betrachtet wurden. Hierbei muss besonders auf das Maschinenbediensystem SmartControl (vgl. Kapitel 2.3) geachtet werden, da es in die Prozessausführung integriert werden muss. Im Folgenden sind einige Bedingungen aufgelistet, die aus technischen Sicht des Szenarios beachtet werden müssen:

- **Die Erstellung der Prozessmodelle in der Ausführungsumgebung** muss mit dem PrMS kompatibel sein. Dies bedeutet, dass das verwendete PrMS die Prozessmodelle interpretieren und ausführen können muss.
- **Für die Kontrolle und Freigabe von Prozessmodellen** durch Pharmazeuten muss die Modellierungsumgebung diesem eine einfache Möglichkeit geben, Prozessmodelle zu prüfen, zu kommentieren und freizugeben. Dazu wird letztendlich ein Freigabe-Prozess benötigt. Dieser beinhaltet, dass jede Änderung am Prozess von einer Person kontrolliert werden muss, die nicht an der Änderung beteiligt war.
- **Die Bereitstellung von Prozessmodellvorlagen (Templates)** soll dem Editoranwender die Erstellung der Prozessmodelle erleichtern. Beispielsweise benötigt er ein Template zur Tablettenproduktion, das er als Vorlage zur Produktion jeglicher Tablettenarten verwenden kann.
- **Jedes Prozessmodell benötigt Auftragsdaten**, welche der Editoranwender als Parameter definiert. Diese beschreiben bei der Instanziierung die Konfiguration des Prozesses. Beim Anlegen eines Prozessmodells ausgehend von einem Template sollen bereits die wichtigsten Auftragsdaten gegeben sein. Der Editoranwender kann anschließend weitere Auftragsparameter hinzufügen.
- **Prozessinstanzen müssen an eine passende Maschine gebunden werden.** Damit der Operator Aufträge an seinem Arbeitsplatz bearbeiten kann, muss es eine Schnittstelle zwischen dem Bediensystem SmartControl und der Prozessaus-

führung geben. Diese Schnittstelle muss in der Lage sein, die Aktivitäten an die richtigen Maschinen zu senden und ein Ergebnis entgegenzunehmen.

## 3.2. User Stories

Nachdem das Szenario beschrieben hat, was in dem entstehenden System modelliert und ausgeführt werden soll, wird nun festgelegt, welche Anforderungen die verschiedenen Anwender an die Modellierungs- und Ausführungsumgebung haben. Da sich diese Anforderungen stetig weiterentwickelten, wurden sie nach dem Scrum-Modell der agilen Software Entwicklung (vgl. [Rub12]) erfasst. Dieses sieht eine empirische und inkrementelle Vorgehensweise vor und ermöglicht es dem Anwender Anforderungen auf eine einfache Weise zu erstellen. Hierbei fängt die Aufnahme der Anforderung mit der Erstellung von Benutzergeschichten, sogenannten *User Stories*, an. Dabei werden die einzelnen Funktionalitäten der Software aus Sicht des Anwenders beschrieben. Die User Stories werden anhand des Grundgerüsts „Als (Anwendertyp) möchte ich (folgende Aktion durchführen), um (dieses Ziel zu erreichen)“ erhoben. Bei der Aufnahme der User Stories sollten dabei die sogenannten „INVEST“-Kriterien eingehalten werden [Rub12]:

- **Independet (dt. unabhängig):** User Stories sollen möglichst unabhängig voneinander sein. Dadurch wird die Umsetzung erleichtert, da weniger Querbezüge mit eingeplant werden müssen.
- **Negotiable: (dt. verhandelbar)** User Stories sind keine endgültigen Anforderungen, sondern stehen als Platzhalter für eine verhandelbare Position.
- **Valuable: (dt. wertvoll)** User Stories müssen einen Mehrwert für den Anwender, den sie betreffen, erbringen.
- **Estimatable: (dt. abschätzbar)** Die Dauer der Umsetzung einer User Story sollte für das Entwicklungsteam abschätzbar sein.
- **Sized appropriately (Small) (dt. angemessene Größe):** Die Größe einer User Story sollte möglichst klein gehalten werden, um sie sinnvoll in *Scrum Sprints* bearbeiten zu können. Bei einem Scrum Sprint handelt es sich um eine Zeitspanne

von typischerweise zwei Wochen, in der vorher definierte Anforderungen in den fünf Schritten Analysiere, Entwickle, Baue, Integriere und Teste umgesetzt werden sollen.

- **Testable (dt. testbar:** Eine User Story sollte so getestet werden können, dass sie nach ihrer Umsetzung entweder bestätigt oder abgelehnt werden kann.

Der erste Schritt in der Erhebung der User Stories ist es, die Akteure zu bestimmen. Konkret sollen jene Anwendertypen identifiziert werden, die mit dem entstehenden System in Berührung kommen werden. Folgende Anwendertypen wurden bestimmt:

- **Editoranwender:** Ein qualifizierter Mitarbeiter mit dem notwendigen Fachwissen, um Prozessmodelle zu erstellen (technischer Prozessexperte).
- **Pharmazeut:** Personen aus dieser Gruppe geben Prozessmodelle frei und sind im Normalfall Teil der Qualitätssicherung (fachlicher Prozessexperte).
- **Maschinenoperator:** Anwender an den Maschinen, die dort Aufträge bearbeiten.
- **Support-Gruppe:** Diese Gruppe besitzt das technische Fachwissen, über den Ablauf der Prozessinstanzen und kann den Status aller Prozessinstanzen einsehen. Dadurch können sie die Maschinenoperatoren unterstützten.

Für die Akteure der Anwendertypen wurden die User Strories anhand des in Kapitel 3.1 beschriebenen Szenarios, sowie durch Interviews mit Prozessexperten der Firma Uhlmann erstellt. Die einzelnen User Stories werden in den nachfolgenden Tabellen 3.2 bis 3.7 beschrieben und halten dabei eine logische Reihenfolge ein, die in Anbetracht der Entwicklung des Produktivsystems festgelegt wurde. Jede Tabelle hat eine eindeutige Identifikation, einen Namen, eine Beschreibung und eine Liste von Akzeptanzkriterien. Die Aufgabe der Akzeptanzkriterien ist zum einen, die User Stories detaillierter zu beschreiben und, zum anderen, Anforderungen festzulegen, die zur Erfüllung dieser beitragen.

Tabelle 3.2.: Login am SmartControl (US-01)

| | |
|---|---|
| **Identifikation** | US-01 |
| **Name** | Login am SmartControl |
| **Beschreibung** | Als Editoranwender möchte ich mich mit meinem SmartControl Login an der Modellierungs- und Ausführungsumgebung anmelden können, um dasselbe Benutzerkonto auf allen Systemen zu benutzen. |
| **Akzeptanzkriterien** | • Zur Anmeldung benutze ich denselben Login, den ich auch im SmartControl verwende.<br>• Ich bekomme dieselbe Rolle zugeteilt, wie in SmartControl. |

Tabelle 3.3.: Modellierung eines Prozessmodells (US-02)

| | |
|---|---|
| **Identifikation** | US-02a |
| **Name** | Modellierung eines Prozessmodells |
| **Beschreibung** | Als Editoranwender möchte ich einen Arbeitsablauf als Prozess modellieren können, um ein Prozessmodell, z.B. für die Tablettenherstellung, zu erstellen. |
| **Akzeptanzkriterien** | • Die Modellierung des Prozesses ist dabei strukturiert und ich werde bei der Umsetzung unterstützt.<br>• Bei der Erstellung der Prozesse werde ich durch geeignete Templates unterstützt. |

| | |
|---|---|
| **Identifikation** | US-02b |
| **Name** | Fehlerfreiheit des Prozessmodells |
| **Beschreibung** | Als Editoranwender möchte ich unterstützt werden, um ein fehlerfreies Prozessmodell zu erstellen. |
| **Akzeptanzkriterien** | • Die Modellierungsumgebung verhindert, dass ich Kontrollelemente verwenden kann, die in meinem Prozessmodell nicht erlaubt sind. |

| | |
|---|---|
| **Identifikation** | US-02c |
| **Name** | Konfiguration eines Prozessmodells |
| **Beschreibung** | Als Editoranwender möchte ich einem Prozessmodell Auftragsparameter hinzufügen, die bei der Instanziierung des Modells gesetzt werden, um aus einem Prozessmodell verschiedene Prozessinstanzen konfigurieren zu können. |
| **Akzeptanzkriterien** | • Durch die Angabe von Auftragsparametern ist ein Prozessmodell variabel und kann z.B. für die Herstellung verschiedener Tabletten verwendet werden kann. |

| | |
|---|---|
| **Identifikation** | US-02d |
| **Name** | Maschinenparameter im Prozessmodell |
| **Beschreibung** | Als Editoranwender möchte ich einem Prozessmodell Maschinenparameter hinzufügen, damit der Maschinenoperator bei der Ausführung der Prozessinstanz direkt die notwendigen Informationen der Maschine erhält. |
| **Akzeptanzkriterien** | • Bei der Auswahl der Maschinenparameter unterstützt mich ein benutzerfreundliches Auswahlmenü. |

Tabelle 3.4.: Kontrolle eines Prozessmodells (US-03)

| | |
|---|---|
| **Identifikation** | US-03 |
| **Name** | Kontrolle eines Prozessmodells |
| **Beschreibung** | Als Pharmazeut möchte ich kontrollieren, ob die erstellten Prozessmodelle fachlich korrekt sind, um die Richtigkeit in der späteren Produktion zu gewährleisten. |
| **Akzeptanzkriterien** | • Die Ansicht auf das Prozessmodell soll dabei vereinfacht sein und sich auf fachliche Angaben beschränken.<br>• Wenn ich ein Prozessmodell freigebe, dann unterzeichne ich es elektronisch.<br>• Ich kann das Prozessmodell mit einer Begründung zurückweisen, damit es erneut bearbeitet werden kann. |

Tabelle 3.5.: Generierung einer Prozessinstanz (US-04)

| | |
|---|---|
| **Identifikation** | US-04a |
| **Name** | Generierung einer Prozessinstanz |
| **Beschreibung** | Als Editoranwender möchte ich aus einem freigebenen Prozessmodell Prozessinstanzen generieren. |
| **Akzeptanzkriterien** | • Bei der Instanziierung werden die, im Prozessmodell definierten, Auftragsparameter von Hand eingegeben. |

| | |
|---|---|
| **Identifikation** | US-04b |
| **Name** | Automatische Generierung einer Prozessinstanz |
| **Beschreibung** | Die Generierung einer Prozessinstanz aus einem freigebenen Prozessmodell soll aus dem ERP-System automatisiert möglich sein. |
| **Akzeptanzkriterien** | • Bei der Instanziierung werden die, im Prozessmodell definierten, Auftragsparameter von dem ERP-System bezogen. |

Tabelle 3.6.: Bearbeitung einer Prozessinstanz (US-05)

| | |
|---|---|
| **Identifikation** | US-05a |
| **Name** | Bearbeitung eines Auftrags |
| **Beschreibung** | Als Maschinenoperator möchte ich einen Auftrag an meinem Arbeitsplatz angezeigt bekommen, um ihn einfach zu bearbeiten. |
| **Akzeptanzkriterien** | • Ich werde nacheinander über die Arbeitsschritte, die ich ausführen muss, informiert.<br>• Bei der Bearbeitung werde ich durch einen Assistenten unterstützt.<br>• Der Assistent informiert mich über die relevanten Maschinenwerte in meinem Arbeitsschritt. |

| | |
|---|---|
| **Identifikation** | US-05b |
| **Name** | Unterbrechen der Bearbeitung |
| **Beschreibung** | Als Maschinenoperator möchte ich die Bearbeitung meines Auftrags unterbrochenen können, um flexibler zu sein. |
| **Akzeptanzkriterien** | • Ich kann die Bearbeitung meines unterbrochenen Auftrags zu einem beliebigen Zeitpunkt wieder aufnehmen.<br>• Während ein Auftrag unterbrochen ist kann ich einen anderen Auftrag zur Bearbeitung wählen. |

| | |
|---|---|
| **Identifikation** | US-05c |
| **Name** | Abschluss eines Auftrags |
| **Beschreibung** | Als Maschinenoperator möchte ich einen Auftrag an meinem Arbeitsplatz abschließen können, um die Bearbeitung zu bestätigen. |
| **Akzeptanzkriterien** | • Jeden Arbeitsschritt schließe ich durch eine elektronische Unterschrift ab.<br>• Durch den Abschluss des letzten Arbeitsschrittes kann ich den Auftrag abschließen. |

Tabelle 3.7.: Support eines Prozessmodells (US-06)

| | |
|---|---|
| **Identifikation** | US-06a |
| **Name** | Einsicht einer Prozessinstanz |
| **Beschreibung** | Als Mitglied der Support-Gruppe möchte ich den Status aller Prozessinstanzen einsehen können, um den korrekten Ablauf zu überwachen. |
| **Akzeptanzkriterien** | • Ich kann jede Aktion, die an einer Prozessinstanz vorgenommen wurde, einsehen um Fehler zu erkennen.<br>• Ich kann jede Aktivität der Prozessinstanz detailliert einsehen. |

| | |
|---|---|
| **Identifikation** | US-06b |
| **Name** | Fehlerbenachrichtigung von Prozessinstanzen |
| **Beschreibung** | Als Mitglied der Support-Gruppe möchte ich über Fehler in Prozessinstanzen informiert werden, um schnell auf diese reagieren zu können. |
| **Akzeptanzkriterien** | • Die Überwachung informiert mich automatisch, wenn ein Fehler aufgetreten ist.<br>• Ich bekomme alle relevanten Informationen der Prozessinstanz angezeigt. |

## 3.3. Technische Anforderungen

Da die in Kapitel 3.2 gewonnenen User Stories eine reine Benutzersicht auf die Problemstellung und die zu entwickelnde Modellierungs- und Ausführungsumgebung bieten, werden in diesem Kapitel die erhobenen technischen Anforderungen an das zu entwickelnde Konzept (vgl. Kapitel 4) zusätzlich vorgestellt. Um diese zu ermitteln, wurde auf den User Stories aufgebaut und dokumentiert, welche Anforderungen ein System erfüllen muss, um diese umzusetzen.

### 3.3.1. Funktionale Anforderungen

Im Folgenden werden die funktionalen Anforderungen an das zu entwickelnde Konzept zur Modellierung und Ausführung von Verfahrensprozessen dargestellt. Hierbei ist zu beachten, dass die Modellierungsumgebung die Erstellung von Prozessmodellen, wie sie das Szenario (vgl Kapitel 3.1.1) beschreibt, unterstützen, sowie die Anforderungen, die durch die User Stories aus Kapitel 3.2 erhoben wurden, miteinbeziehen muss. Die Ausführungsumgebung muss die Integration von vorhandenen Maschinen berücksichtigen, um das Prinzip der *SmartFactory* und eine flexible Produktion (vgl. Kapitel 1.1) sicherzustellen. Außerdem muss die Ausführungsumgebung Eigenschaften des Prozessmanagements umsetzen (vgl. Kapitel 2.1) und ein geeignetes PrMS verwenden.

Tabelle 3.8 definiert die funktionalen Anforderungen an die Modellierunsumgebung und Tabelle 3.9 an die Ausführungsumgebung. Die Priorität der Anforderungen ist in der Spalte „Pr.“ der Tabellen wie folgt klassifiziert:

- M = „Muss-Anforderung“
- S = „Soll-Anforderung“
- W = „Wunsch-Anforderung“

Hierbei ist zu beachten, dass sowohl Soll- als auch Wunsch-Anforderungen im Konzept berücksichtigt werden, aber nicht zwingend Bestandteil des Proof-of-Concept Prototypen sein müssen. Die Entwicklung des Proof-of-Concept Prototypen fokussiert sich auf die Realisierung der wichtigsten Anforderungen.

Tabelle 3.8.: Modellierungsumgebung (FA-01)

| Nr. | Bezeichnung | Beschreibung | Pr. |
|---|---|---|---|
| **1** | **Modellierungsumgebung** | | |
| **1.1** | Prozessmodell erstellen | Innerhalb der Modellierungsumgebung soll der Editoranwender ein Prozessmodell anlegen. Dabei definiert er lediglich Knoten und Kanten des Modells. Dies bedeutet, er legt die benötigten Subprozesse fest und verknüpft diese. Hierbei beschränkt sich Kontrollfluss des Prozessmodells, neben dem sequentiellen Ablauf, auf UND- und ODER-Verzweigungen. | M |
| **1.2** | Prozessmodell speichern | Der Editoranwender soll das Prozessmodell speichern können. Der Speicherstand enthält alle Subprozesse und wird dabei versioniert. | M |
| **1.3** | Prozessmodell freigeben | Bevor das Prozessmodell instanziiert werden kann, muss es zuvor von einem Pharmazeuten geprüft und freigegeben werden. | S |
| **1.4** | Subprozess erstellen | Bei der Prozessmodellierung soll der Editoranwender die Möglichkeit haben, einen Subprozess neu zu modellieren oder aus einer Liste vorhandener Subprozesse einen geeigneten auszuwählen. | M |
| **1.5** | Subprozess speichern | Subprozessmodelle sollen zusammen mit dem Prozessmodell automatisch gespeichert werden und können später wieder verwendet werden. | M |
| **1.6** | Aktivitäten erstellen | Innerhalb eines Subprozessmodells kann der Editoranwender atomare Aktivitäten erstellen und diesen Texte sowie typisierte Felder zuordnen. | M |
| **1.7** | Aktivitäten speichern | Die Aktivitäten sollen beim Speichern des Subprozesses automatisch mitgespeichert werden. | M |

| | | | |
|---|---|---|---|
| **1.8** | Auftragsparameter anlegen | Der Editoranwender soll innerhalb der Modellierungsumgebung Auftragsparameter anlegen können. Diesen Auftragsparametern werden bei der Instanziierung des Prozessmodells Werte übergeben, wodurch eine Konfiguration des Prozessmodells ermöglicht wird. Das Format der Auftragsparameter wird durch die Typen-Bibliothek von SmartControl definiert. | S |
| **1.9** | Maschinenparameter referenzieren | Der Editoranwender soll in einer Aktivität Maschinenparameter referenzieren können, so dass später beim Ausführen z.B. IST-Werte der Maschine angezeigt und bearbeitet werden können. Dies erleichtert dem Operator die Steuerung der Maschine. | S |
| **1.10** | Auftragsparameter referenzieren | Der Editoranwender soll in einer Aktivität Auftragsparameter referenzieren können, so dass später beim Ausführen z.B. SOLL-Werte des Auftrags angezeigt und mit den IST-Werten verglichen werden können. | S |
| **1.11** | Steuerelemente einfügen | In eine Aktivität soll der Editoranwender beliebige, typisierte Steuerelemente einfügen können, um dem Maschinenoperator Beschreibungen sowie Eingabefelder zur Verfügung zu stellen. Beispielsweise soll er Textfelder oder Auswahlfelder einfügen können. | M |
| **1.12** | Vorschau zu Prozessmodell anzeigen | Während der Modellierung kann der Editoranwender eine Vorschau des Prozessmodells einsehen. Dies soll es ihm erleichtern, das Prozessmodell so zu modellieren, wie es bei der Ausführung auf den Maschinen dem Operator angezeigt wird. | W |

Tabelle 3.9.: Ausführungsumgebung (FA-02)

| Nr. | Bezeichnung | Beschreibung | Pr. |
|---|---|---|---|
| **2** | **Ausführungsumgebung** | | |
| **2.1** | Prozessinstanz erstellen | Aus einem freigegebenen Prozessmodell kann manuell oder durch ein ERP-System eine Prozessinstanz anhand der Auftragsdaten erstellt werden. | M |
| **2.2** | Prozessinstanz ausführen | Nachdem eine Prozessinstanz erstellt wurde, wird sie automatisch ausgeführt. | M |
| **2.3** | Aktivität bearbeiten | Die Aktivität einer gestarteten Prozessinstanz muss an die richtige Maschine gesendet werden, auf der sie dann dem Maschinenoperator angezeigt wird und er sie bearbeiten kann. | M |
| **2.4** | Aktivität signieren | Nach der Bearbeitung signiert der Maschinenoperator die Aktivität, um deren erfolgreiche Bearbeitung zu bestätigen. | S |
| **2.5** | Prozessinstanz unterbrechen | Die Bearbeitung eines Auftrags kann nach jeder abgeschlossenen Aktivität unterbrochen werden. Die Bearbeitung des Auftrags wird dann anschließend von diesem Punkt wieder aufgenommen. | W |

### 3.3.2. Nichtfunktionale Anforderungen

Zusätzlich zu den funktionalen Anforderungen an die Modellierungs- und Ausführungsumgebung für Verfahrensprozesse werden in diesem Kapitel nichtfunktionale Anforderungen an das zu entwickelnde Konzept erhoben. Die nichtfunktionalen Anforderungen beschreiben Eigenschaften, die dieses bei der Umsetzung der funktionalen Anforderungen einhalten muss.

Die nichtfunktionalen Anforderungen sind in Tabelle 3.10 auflistet und werden sowohl an die Modellierungs- als auch an die Ausführungsumgebung gleichermaßen erhoben. Sie wurden unter Einbezug der User Stories und durch Interviews mit Fachpersonal der Firma Uhlmann festgelegt. Sie sollen dabei helfen, die in Kapitel 1.2 genannten Ziele zu erreichen. Der Aufbau von Tabelle 3.10 ist derselbe, wie der der funktionalen Anforderungen (siehe Kapitel sec:chapter3:techAnforderungen:funk).

Tabelle 3.10.: Nichtfunktionale Anforderungen

| Nr. | Bezeichnung | Beschreibung | Pr. |
|---|---|---|---|
| **1** | **Benutzbarkeit** | | |
| **1.1** | Erlernbarkeit | Der Editoranwender soll ohne eine vorhergehende Schulung Prozessmodelle erstellen können. | M |
| **1.2** | Bedienbarkeit | Durch eine strukturierte Benutzeroberfläche soll der Editoranwender intuitiv Subprozesse und Aktivitäten erstellen und das Prozessmodell speichern können. | M |
| **1.3** | Look & Feel | Das Design der Modellierungsumgebung soll sich am bestehenden SmartControl orientieren. Auf diese Weise soll dem Editoranwender ein Wiedererkennungswert vermittelt werden. | S |
| **1.4** | Internationalisierung | Standardmäßig wird Deutsch als Sprache verwendet. Das entwickelte System soll allerdings internationalisierbar sein und somit das Laden anderer Sprachpakete ermöglichen. | W |

| | | | |
|---|---|---|---|
| **2** | **Entwicklungsspezifisch** | | |
| **2.1** | Systemumgebung | Die Modellierungs- und Ausführungsumgebung sollen in SmartControl integriert werden. Dazu sollen sie als web-basierte Anwendungen konzipiert werden. | S |
| **2.2** | Service-orientierung | Die Kommunikation der Ausführungsumgebung mit den Maschinen soll über Web Services stattfinden. Dadurch soll eine einheitliche Schnittstelle geschaffen werden. | S |
| **2.3** | Integration von Maschinen | Die Konzeption des Prozessmodells soll bewusst abstrakt sein, um auch Maschinen, die nicht SmartControl als Bediensystem verwenden, zu integrieren. | M |
| **2.4** | Struktur der Aktivitäten | Damit Aktivitäten von Prozessinstanzen auf den SmartControl-Maschinen ausgeführt werden können, müssen sie in einer konformen XML-Struktur übergeben werden. | M |
| **3** | **Sicherheit** | | |
| **3.1** | Vertraulichkeit | Die erstellten Prozessmodelle, Subprozesse und Aktivitäten müssen vertraulich behandelt werden. | M |
| **3.2** | Sichere Datenhaltung | Die fertigen Prozessmodelle müssen sicher auf dem Server gespeichert werden. Sie dürfen nur durch die Modellierungsumgebung, d.h. von berechtigten Anwendern, einsehbar sein. | M |
| **3.3** | Korrektheit | Die Ausführungsumgebung darf nur die Instanziierung korrekter Prozessmodelle erlauben. | M |

# 4

# Konzept

Dieses Kapitel stellt das Gesamtkonzept des zu entwickelnden Systems zur Modellierung und Ausführung von Verfahrensprozessen vor. Hierzu wird zunächst das Architekturkonzept des Systems erläutert. Daraufhin wird der Inhalt und der Aufbau der Client- und der Server-Anwendung, die für das entworfene System notwendig sind, beschrieben und es werden Konzeptentscheidungen im Bezug auf die Usability der Client-Anwendung festgelegt. Anschließend wird die Kommunikation des Systems mit den Maschinen vorgestellt, wobei SmartControl als Beispiel für die Integration von Maschinenbediensystemen in den Verfahrensprozess herangezogen wird.

## 4.1. Überblick

Die Funktionsweise eines Systems zur Modellierung und Ausführung von Verfahrensprozessen ist aus Sicht der Akteure, die mit ihm interagieren, in drei Teile gegliedert: Zuerst wird ein Prozessmodell durch den Editoranwender erstellt und von einem Pharmazeuten auf seine Richtigkeit überprüft (1). Daraufhin wird das Prozessmodell instanziiert und ausgeführt. Zur Ausführung sendet die Prozessinstanz zu jedem Subprozess Arbeitsanweisungen, bzw. SmartControl Assistenten an die passenden Maschinen (2). Anschließend führen die Operatoren diese dort aus und ein Ergebnis wird an die Prozessinstanz zurückgesendet, woraufhin diese mit der Ausführung der nächsten Aktivität fortfährt (3). Abbildung 4.1 zeigt diese Funktionsweise und bezieht sich dabei auf das Szenario aus Kapitel 3.1, das die Subprozesse „Tablettenherstellung“, „Blisterverpackung“ und „Endverpackung“ beinhaltet.

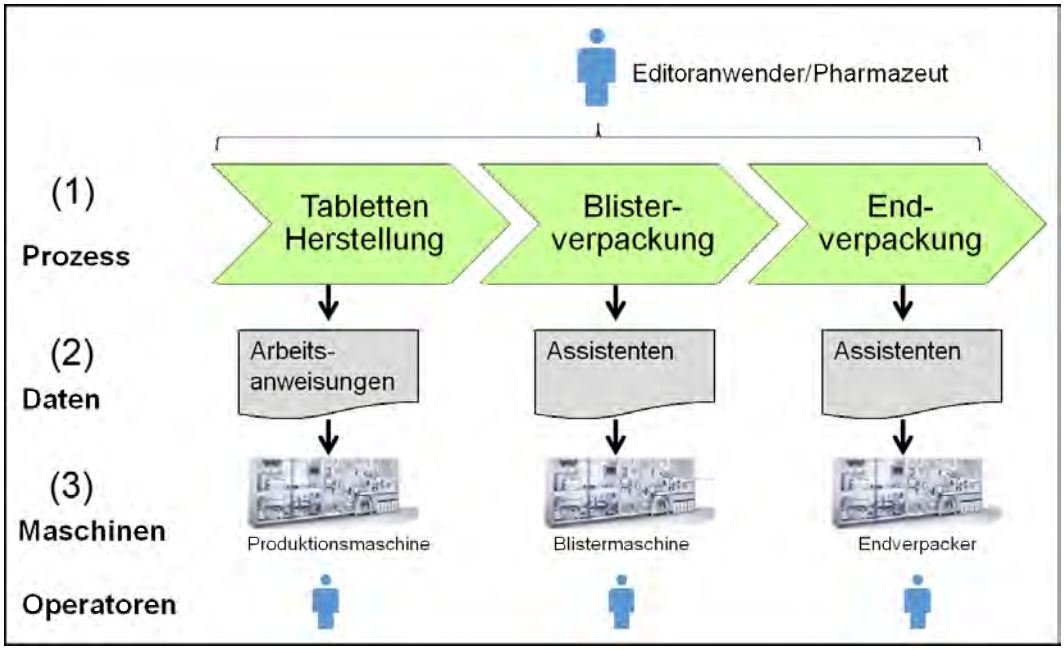


Abbildung 4.1.: Konzept - Anwendersicht

Das Architekturkonzept eines solchen Systems ist jedoch um einiges komplexer, wie aus Abbildung 4.2 ersichtlich ist. Bei der Konzeption der Architektur muss beispielsweise darauf geachtet werden, dass mehrere Personen als Editoranwender agieren und jeweils von ihrem Arbeitsplatz Prozessmodelle anlegen, bearbeiten oder freigeben wollen. Die Speicherung und Verwaltung von Prozessmodellen findet allerdings zentral auf einem Server und die Ausführung von Prozessinstanzen auf einem Prozessmanagementsystem (vgl. Kapitel 2.1) statt. Außerdem erfordert die Integration der Maschinen in den Verfahrensprozess, dass eine Kommunikation zwischen dem PrMS und den vorhandenen Maschinen in der Produktion möglich ist.
In Abbildung 4.2 werden die Schichten der Architektur und deren Komponenten vorgestellt:

In der **Modellierungs-** und **Überwachungsschicht** befindet sich die Client-Anwendung, die für die Modellierungsumgebung einen Modellierungseditor bereit stellt. Über den Modellierungseditor können der Editoranwender und der Pharmazeut Prozessmodelle erstellen und bearbeiten. Zusätzlich enthält die Client-Anwendung eine Ausführungsüber-

wachung, um den Status von Prozessinstanzen verfolgen zu können. Die Kommunikation zwischen Client- und Server-Anwendung findet über eine REST-API statt. Die Server-Anwendung stellt der Modellierungsumgebung Schnittstellen zum Speichern und Laden von Prozessmodellen zur Verfügung. Bei REST handelt es sich um einen Architekturstil für verteilte Systeme. REST steht für *Representational State Transfer*, ist ressourcenorientiert und verwendet HTTP als Kommunikationsprotokoll [RR07]. Clients können Ressourcen, wie beispielsweise Prozessmodelle, auf dem Server über die HTTP Methoden abrufen (GET), anlegen (POST), ändern (PUT) und löschen (DELETE). Die Prozessmodelle werden auf dem Server durch die Prozessmodell-Repository-Komponente in einer Datenbank gespeichert.

Die Server-Anwendung befindet sich in der **Logik-** und **Prozessschicht**. Sie enthält neben dem Prozessmodell-Repository noch eine Ausführungssteuerung in der Ausführungsumgebung und ermöglicht dem Anwender dadurch das Verwalten von Prozessinstanzen, indem eine direkte Verbindung mit dem PrMS hergestellt wird. Das Prozessmanagementsystem wird in die Server-Anwendung integriert. Hier werden Prozessinstanzen ausgehend von der Ausführungssteuerung angelegt, ausgeführt und gesteuert. Wird eine Prozessinstanz ausgeführt, so legt das PrMS die betroffenen Aktivitäten über eine REST-Schnittstelle an der Kommunikations-Anwendung an. Der gesamte Kommunikationsablauf zwischen dem PrMS, der Kommunikations-Anwendung und den Maschinenbediensystemen wird in Kapitel 4.4 detailliert erläutert.

Die **Integrations-Schicht** hat die Aufgabe Aktivitäten, die es durch das PrMS erhalten hat, an die richtigen Maschinen, bzw. deren Maschinenbediensysteme zu senden. Dazu beinhaltet die Kommunikations-Anwendung für jede vorhandene Maschine einen Wrapper. Bei einem Wrapper handelt es sich um einen anwendungsspezifischen Adapter, der ein spezifisches Maschinenbediensystem, wie z.B. SmartControl (vgl. Kapitel 2.3), anbindet [ACKM04]. Die Kommunikations-Anwendung steht damit zwischen dem PrMS und den Maschinen der Produktionslinie. Der Wrapper ist außerdem dafür verantwortlich, abgeschlossene Aktivitäten über die REST-Schnittstelle wieder an das PrMS zu melden.

Auf der **Interaktions-Schicht** stehen die an dem Verfahrensprozess beteiligten Maschinen. Sie empfangen Aktivitäten und visualisieren diese für die Maschinenoperatoren, die

die Aktivitäten bearbeiten und abschließen. Die Interaktions-Schicht von SmartControl besteht aus der Komponente „Assistentenausführung“ und wird in Kapitel 4.5 näher beschrieben.

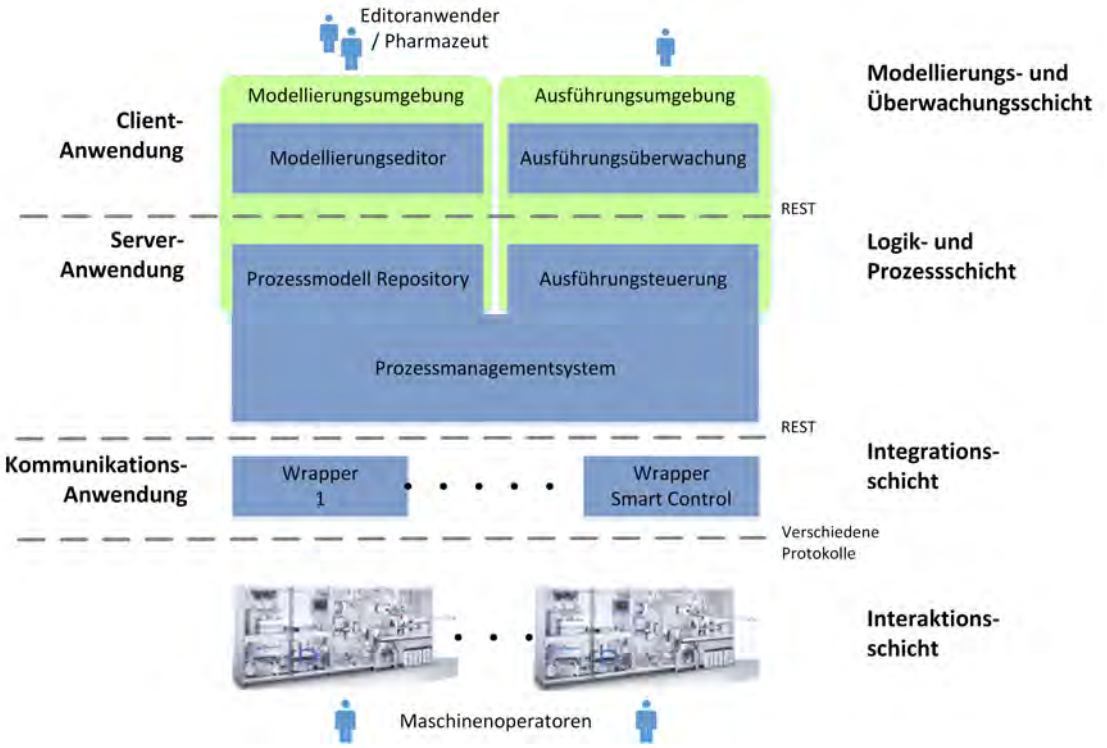


Abbildung 4.2.: Konzept - Schichtenmodell

## 4.2. Client-Anwendung

Aus den nichtfunktionalen Anforderungen (vgl. Kapitel 3.3.2) geht hervor, dass die Client-Anwendung als web-basierte Anwendung konzipiert werden soll. Der Vorteil hierbei ist, dass der Client nicht auf jedem Arbeitsplatz installiert und bei Änderungen aktualisiert werden muss. Jeder Anwender der Client-Anwendung kann diese über den Web-Browser erreichen und sie sogar auf einem mobilen Gerät verwenden. Durch die Anbindung der Server-Anwendung kann der Client auf eine Reihe von Komponenten, wie beispielsweise das Prozessmodell-Repository und das PrMS, zugreifen. Auch muss der Anwender sich

über die web-basierte Anwendung anmelden, wodurch zentral nachvollzogen werden kann, welcher Anwender welche Aktionen durchgeführt hat.

Die Client-Anwendung stellt dem Anwender folgende Funktionalitäten zur Verfügung:

1. Navigation durch die Modellierungs- und Ausführungsumgebung.
2. Anlegen oder Auswählen eines Prozessmodells.
3. Modellieren des Prozessmodells auf Linienebene, in der Modellierungsumgebung.
4. Festlegen von Auftragsparametern, für das Prozessmodell.
5. Modellieren der Subprozesse auf Maschinenebene, z.B. für SmartControl, in der Modellierungsumgebung.
6. Instanziieren des Prozessmodells.

Im ersten Schritt besteht die Client-Anwendung aus einer Auflistung aller verfügbaren Prozessmodelle, die sie von der Server-Anwendung erhält. Der Anwender kann die aufgelisteten Prozessmodelle einsehen, editieren und instanziieren, sofern er dazu berechtigt ist. Außerdem kann er neue Prozessmodelle hinzufügen. Das Bearbeiten eines Prozessmodells erfolgt im Modellierungseditor. Die letzte Komponente der Client-Anwendung ist die Ausführungsüberwachung, in der alle laufenden Prozessinstanzen aufgelistet werden. Die Ausführungsüberwachung soll zum einen darüber informieren, auf welchen Maschinen die Aktivitäten der Prozessinstanzen momentan ausgeführt werden. Zum anderen soll genau nachvollziehbar sein, welche Aktivitäten bereits abgeschlossen wurden und woran aktuell gearbeitet wird.

Im Folgenden wird der Aufbau der Komponenten und das Navigationskonzept der Client-Anwendung mithilfe von Mockups erklärt. Anhand der Mockups wird außerdem aufgezeigt, wie die oben aufgezählten Funktionalitäten durch den Editoranwender innerhalb der Client-Anwendung verwendet werden können.

#### Navigationskonzept

Das Startmenü der Client-Anwendung beinhaltet jeweils eine Ansicht für die Auflistung von Prozessmodellen und Prozessinstanzen. Durch die Auswahl eines Prozessmodells

gelangt der Anwender in die Modellierungsumgebung und durch die Auswahl einer Prozessinstanz entsprechend in die Ausführungsumgebung. Zum Wechseln zwischen den Ansichten im Startmenü verfügt die Client-Anwendung über eine Navigationsleiste, die im oberen Bildschirmbereich platziert ist. Abbildung 4.3 zeigt ein Mockup des Startmenüs mit der Navigationsleiste (1) und der Auflistung der Prozessmodelle (2) - hierbei handelt es sich um das Auswahl-Portal der Modellierungsumgebung. Das Mockup gibt dabei das Prozessmodell „Tablettenherstellung“ aus dem Szenario an, das in Kapitel 3.1 vorgestellt wurde.

Da Prozessmodelle vor der Verwendung freigegeben werden müssen, werden in der Ansicht zunächst alle freigegebenen Prozessmodelle gruppiert angezeigt. Darunter folgen alle weiteren Prozessmodelle, die wiederum gruppiert werden in Prozessmodelle, die der angemeldete Anwender erstellt hat, und diejenigen, die von anderen Anwendern erstellt wurden. Die einzelnen Prozessmodelle werden durch eine graphische Vorschau dargestellt und verfügen über eine „Bearbeiten“-Schaltfläche, die es dem Anwender ermöglicht das gewünschte Prozessmodell in der Modellierungsumgebung einzusehen und zu bearbeiten. Die Client-Anwendung ist "rollen-basiert", d.h. die Funktionen, die der angemeldete Anwender in der Client-Anwendung erhält, hängen von seiner Rolle ab. Hat der Anwender die Rolle eines Pharmazeuten, wird diesem zusätzlich eine Schaltfläche zur Freigabe des Prozessmodells angezeigt. Ist das Prozessmodell bereits freigegeben, verfügt es außerdem über eine „Ausführen“-Schaltfläche, die die direkte Instanziierung desselben erlaubt. Um ein neues Prozessmodell anzulegen, wird die „Hinzufügen“-Schaltfläche im unteren rechten Bildschirmbereich verwendet, diese ist in Abbildung 4.3 durch ein Plus-Zeichen dargestellt.

**Modellierungseditor**

Die Modellierung eines Verfahrensprozesses wird im Modellierungseditor in mehrere Bereiche unterteilt. Der Editoranwender erstellt zunächst den Gesamtprozess aus Unternehmenssicht und definiert dazu alle am Prozess beteiligten Maschinen als maschinennahe Subprozesse, wie im ersten Fallbeispiel in Kapitel 3.1.1 vorgestellt. Für die Modellierung des Gesamtprozesses bekommt der Editoranwender einen graphischen

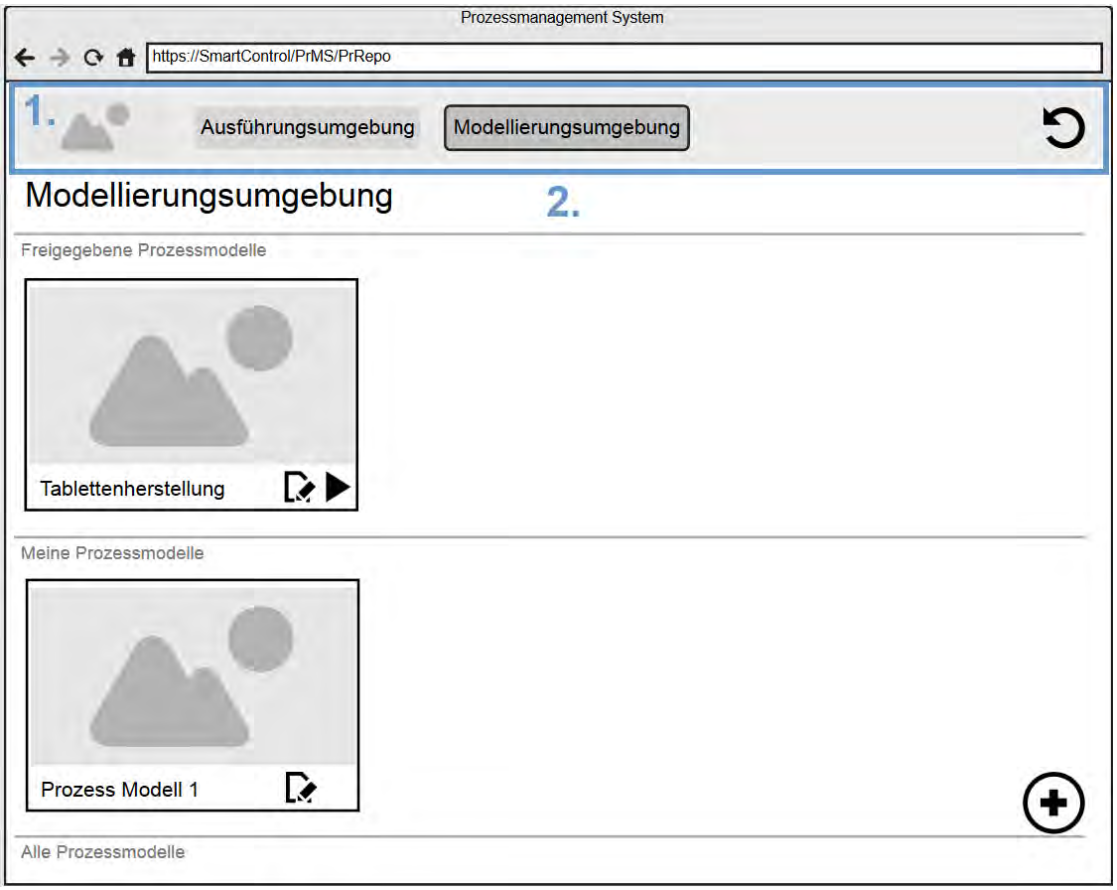


Abbildung 4.3.: Mockup - Prozessmodellauswahl

Modellierungseditor zur Verfügung gestellt, der in Abbildung 4.4 als Mockup veranschaulicht ist. Wie im Startmenü wird die Navigation innerhalb des Modellierungseditors durch eine Navigationsleiste im oberen Bildschirmbereich dargestellt (1). Der linke Bereich zeigt alle verfügbaren Steuerelemente, die in das Prozessmodell per Drag&Drop, also durch das Ziehen mit der Maus, einfügt werden können (2). Die hierbei bereitgestellten Steuerelemente sind in den Anforderungen in Kapitel 3.3 definiert. Neben maschinennahen Subprozessen sollen demnach UND- bzw. ODER-Gatter zur Steuerung des Kontrollfluss angeboten werden. Das Prozessmodell modelliert der Editoranwender in Bereich (3) als Graph in der standardisierten Prozessmodellierungssprache BPMN [FR10]. Das in Abbildung 4.4 gezeigte Prozessmodell besteht dabei lediglich aus drei Subprozessen und bezieht sich auf das Szenario aus Kapitel 3.1. Der Modellierungseditor beinhaltet außerdem einen Bereich (4), in dem der Editoranwender die Bezeichnung des Prozessmodells anpassen und das Prozessmodell jederzeit speichern und löschen kann.

Um einen der maschinennahen Subprozesse modellieren zu können, muss der Editoranwender den gewünschten Subprozess in der Navigationsleiste auswählen. Dadurch gelangt er in eine weitere Ansicht. Die Trennung der Modellierung von Gesamtprozess und Subprozessen erhöht die Übersichtlichkeit des Modellierungseditors.
Da maschinennahe Subprozesse jene Aktivitäten enthalten, die auf den Maschinenbediensystemen ausgeführt werden, benötigen sie eine individuelle Struktur. Im Falle von SmartControl ist dies das Assistenten-Format, dessen Modellierung in Kapitel 4.5 beschrieben ist. Für das Konzept bedeutet dies, dass es möglich sein muss, den Modellierungseditor um Plugins[1] zu erweitern. Dadurch wird gewährleistet, dass eine Vielzahl von Maschinen unterstützt werden kann. Beim Hinzufügen eines Subprozesses muss dazu die Art des Maschinenbediensystems, wie beispielsweise SmartControl, ausgewählt werden, damit für die Modellierung dieses Subprozesses das richtige Plugin verwendet werden kann.

[1] Bei einem Plugin handelt es sich um Erweiterung des Modellierungseditors. Für jedes Maschinenbediensystem, zu dem Prozessmodelle modelliert werden sollen, wird ein Plugin benötigt. Plugins sind austauschbar und werden je nach Bedarf geladen.

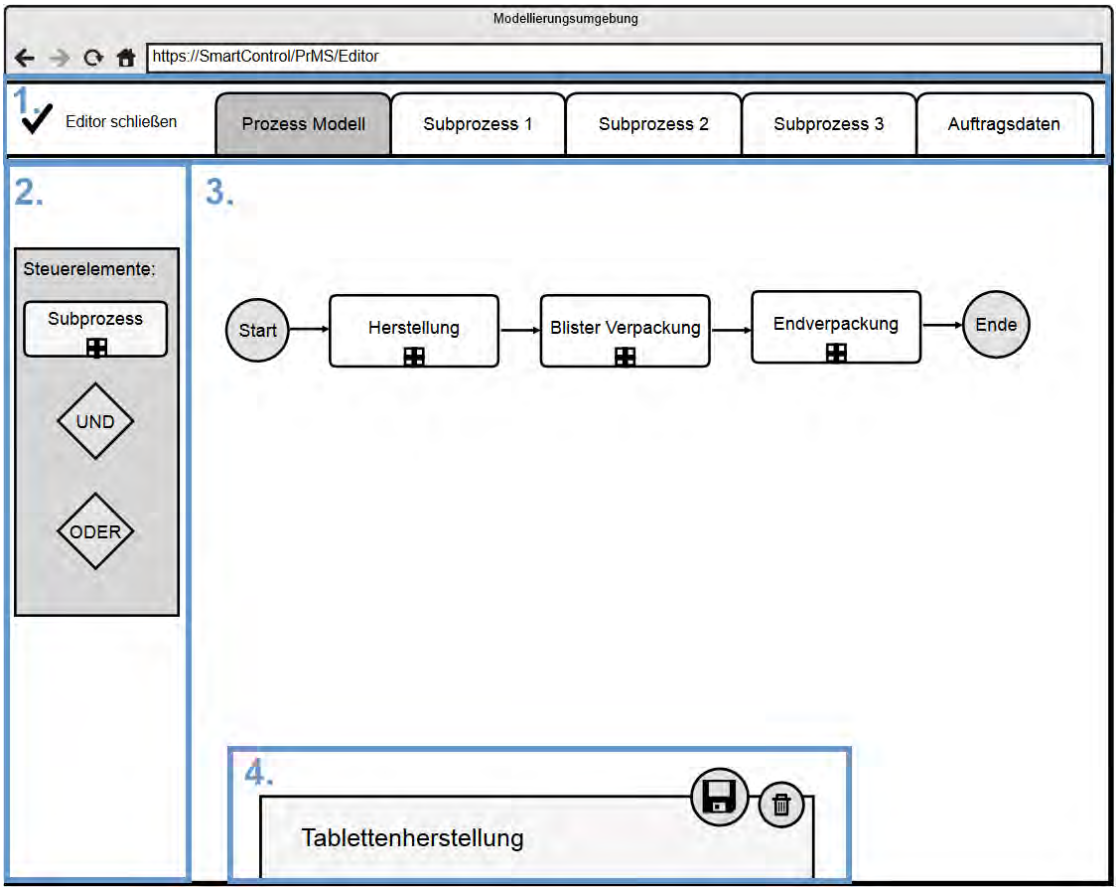


Abbildung 4.4.: Mockup - Modellierung des Gesamtprozesses

Ein weiterer Aspekt des Modellierungseditors ist die Definition von Auftragsparametern. Dazu wird eine Tabelle oder ein Formular zur Verfügung gestellt, in dem alle wichtigen Auftragsparameter festgelegt werden können, die zur Instanziierung des Prozessmodells benötigt werden. Abbildung 4.5 stellt ein Mockup vor, das es dem Editoranwender ermöglicht, alle benötigten Auftragsparameter anzulegen, die in Tabelle 3.1 des zweiten Fallbeispiels angegeben wurden (vgl. Kapitel 3.1.1). Dazu gibt er für jeden Auftragsparameter einen Namen, einen Datentyp und einen Wahrheitswert an. Letztgenannter beschreibt, ob der Auftragsparameter zur Instanziierung benötigt wird. Bei dem Datentyp handelt es sich in der Regel um eine Zahl (Integer), einen Text (String) oder ein Datum (Date). Eine Ausnahme bildet hier der Datentyp *Tabelle*, der für die Stückliste benötigt wird und im Allgemeinen aus sogenannten Key-Value-Paaren[2] besteht.

Die Auftragsdaten, die in Abbildung 4.5 zu sehen sind, beziehen sich dabei auf das gesamte Prozessmodell und können innerhalb des Modellierungseditors in Aktivitäten und ODER-Entscheidungen des Prozessmodells durch den Editoranwender referenziert werden. Zur Instanziierung werden die Auftragsparameter gesetzt und die referenzierten Stellen mit Werten gefüllt. Beispielsweise kann vor den letzten Subprozess des Szenarios („Endverpackung“) aus Kapitel 3.1, ein ODER-Gatter gesetzt werden, das für den Pfad zu diesem Subprozess eine Bedingung definiert. Der Subprozess wird in diesem Fall nur ausgeführt, wenn beispielsweise der Auftragsparameter „Artikelnummer“ einem bestimmten Medikament entspricht.

#### Ausführungsüberwachung

Wurde ein Prozessmodell fertig modelliert und anschließend erfolgreich freigegeben, so kann ausgehend davon eine Prozessinstanz erstellt werden. Die Instanziierung und Ausführung läuft im PrMS ab und ist in Kapitel 4.3 im Genauen beschrieben. In der Client-Anwendung wird im Bereich Ausführungsüberwachung jede Aktivität der Ausführung genau angezeigt und es wird der aktuelle Status der Prozessinstanz vermerkt. Dazu wird zunächst eine Auflistung aller laufenden Prozessinstanzen gegeben. Bei der Auswahl einer Prozessinstanz wird diese graphisch dargestellt und alle bekannten

[2] zu dt. Schlüssel-Wert: Jedem Schlüssel ist ein Wert zugeordnet.

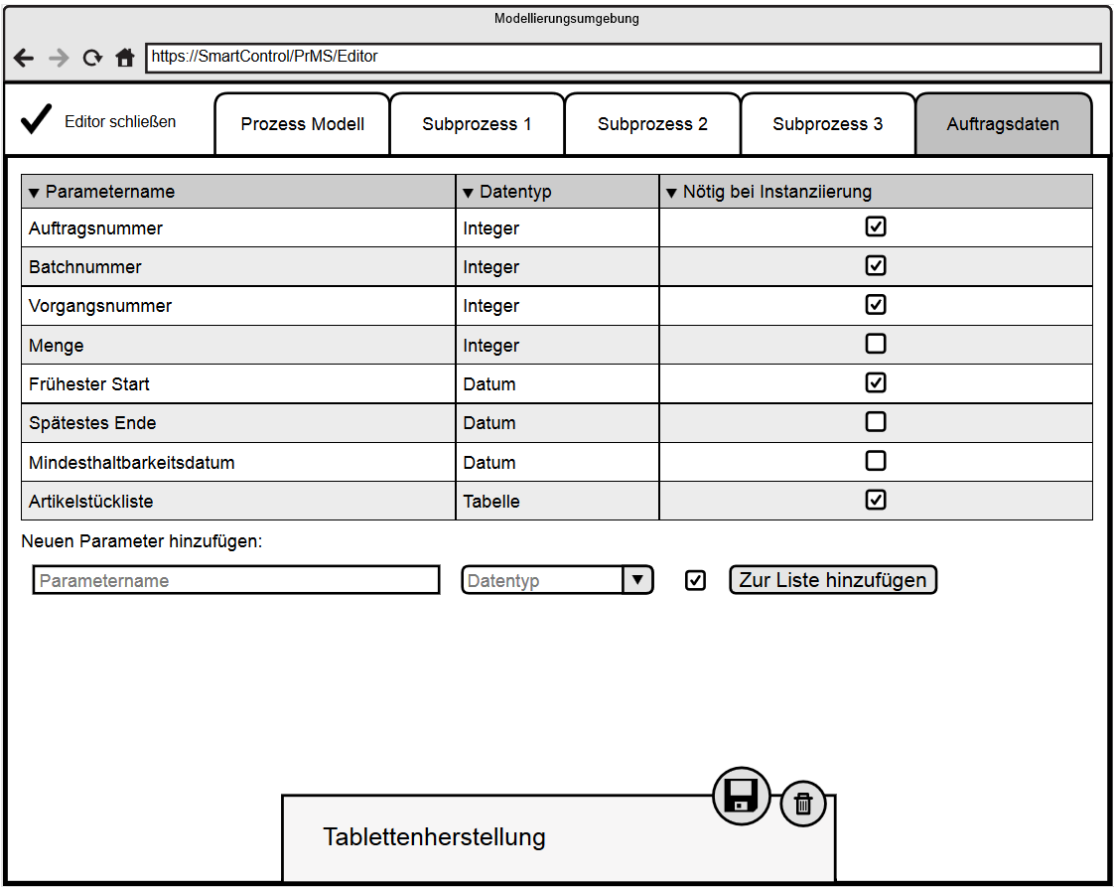


Abbildung 4.5.: Mockup - Festlegen der Auftragsparameter

Informationen werden angezeigt. Das Mockup aus Abbildung 4.6 zeigt die Ausführung einer Prozessinstanz, die auf dem Szenario aus Kapitel 3.1 beruht. Die Symbole in den maschinennahen Subprozessen zeigen, dass „Herstellung“ bereits abgeschlossen wurde und „Blister Verpackung“ momentan in Bearbeitung ist. Außerdem ist der Subprozess „Blister Verpackung“ ausgewählt und im unteren Bereich des Mockups werden tabellarisch alle Inforationen, wie beispielsweise der Status und die Maschine, auf der der Subprozess ausgeführt wird, angezeigt. Dies ermöglicht einen transparenten Ablauf der Prozessinstanzen und eine schnelle Fehlerfindung.

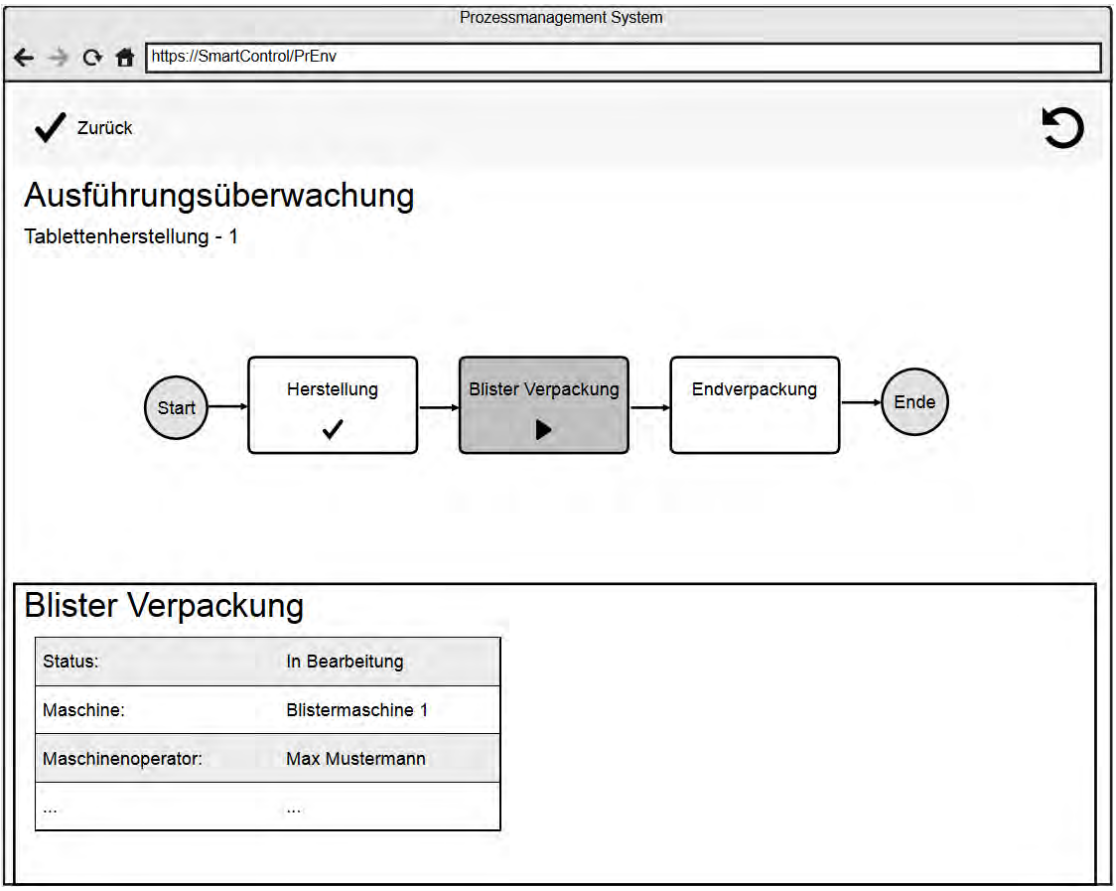


Abbildung 4.6.: Mockup - Ausführungsumgebung

## 4.3. Server-Anwendung

Die Server-Anwendung beinhaltet eine Komponente zur Datenhaltung von Prozessmodellen (Prozess-Repository) und über die REST-Schnittstelle der Server-Anwendung kann die Client-Anwendung die Prozessmodelle an dieser Komponente verwalten. Über die Ausführungssteuerung kapselt die Server-Anwendung das eingebettete Prozessmanagementsystem - diese Komponente ist daher für die zentrale Steuerung von Prozessinstanzen zuständig. Prozesse können entweder durch ein ERP-System automatisch oder manuell über die Client-Anwendung an der Ausführungsumgebung instanziiert werden. Wobei darauf geachtet werden muss, dass die definierten Auftragsparameter korrekt gesetzt werden. In beiden Fällen geschieht die Kommunikation mit der Ausführungsumgebung durch eine REST-Schnittstelle (siehe Abbildung 4.2).

Wird eine Prozessinstanz gestartet, führt das PrMS diese entsprechend der Eingaben Schritt für Schritt aus, wobei jede Aktivität einen Web Service[3] nach dem Request-/Response-Pattern darstellt. Dabei wartet der Sender des Web Service Aufrufs auf eine Antwort des Servers [ACKM04]. Auf diese Weise wird im Prozess nur dann fortgefahren, wenn eine Aktivität abgeschlossen ist und auf das Ergebnis der Aktivität reagiert werden kann. Diese Web Service Aufrufe gehen an der Kommunikations-Anwendung (vgl. Kapitel 4.4) ein, die für die korrekte Weiterleitung der Instruktionen und Daten an die Maschinen sorgt.

## 4.4. Kommunikations-Anwendung

Die Kommunikations-Anwendung stellt einen wichtigen Teil des Gesamtkonzepts zur Unterstützung von Verfahrensprozessen dar. Sie steht als eigenständige Komponente zwischen dem PrMS und den zu Verfügung stehenden Maschinen (siehe Abbildung 4.2). Die Kommunikations-Anwendung übernimmt dabei die Rolle, die Aktivitäten des PrMS zu empfangen, zu interpretieren und an die richtige Maschinen weiterzuleiten, damit

[3] Ein Web Service ist eine Software Anwendung, die über eine Netzwerkschnittstelle entfernt aufgerufen wird, um z.B. Daten auf einem Server zu bearbeiten. Ein Web Service ist über eine URI und eine Schnittstellenbeschreibung genau definiert [ACKM04].

sie dort ausgeführt werden. Für diese Kommunikation werden RESTful Web Services verwendet und die Aktivitäten als Ressourcen in Kommunikations-Anwendung angelegt.

Das in Abbildung 4.7 gezeigte Sequenzdiagramm stellt den Ablauf der Kommunikation zwischen den beteiligten Komponenten dar. Beginnend mit dem Start einer Aktivität im Prozessmanagementsystem zeigt es deren Bearbeitung bis hin zum Abschluss. Die Verwendung der Wrapper-Komponente verbirgt die Heterogenität der Maschinenbediensysteme und ist damit der Schlüssel zu deren Integration im Gesamtsystem [ACKM04]. Startet das PrMS eine neue Aktivität, sendet die Ausführungssteuerung die Aktivität über eine REST-Schnittstelle an die Kommunikations-Anwendung. Dort wird die Aktivität angenommen und vom entsprechenden Wrapper, im Falle von Abbildung 4.7 dem SmartContol-Wrapper, interpretiert. Der SmartControl-Wrapper ermittelt den SmartControl-Assistenten (vgl. Kapitel 2.3) zur entgegengenommenen Aktivität und schickt diesen als XML-Dokument an die betreffende SmartControl Maschine, an der der Assistent Schritt für Schritt abgearbeitet wird. Der Wrapper prüft solange den Status der Abarbeitung auf der Maschine über einen Listener, bis der Assistent abgeschlossen ist, und aktualisiert den Status der Aktivität auf der Kommunikations-Anwendung über die REST-Schnittstelle. Anschließend bezieht die Ausführungssteuerung die bearbeitete Aktivität von der Kommunikations-Anwendung und löscht sie. Auf diese Weise wird jede Aktivität der Prozessinstanz auf der Server-Anwendung abgearbeitet.

Das Sequenzdiagramm (vgl. Abbildung 4.7) bezieht sich hier zwar direkt auf das Maschinenbediensystem SmartControl, jedoch kann das Prinzip auch auf andere System übertragen werden. Die verschiedenen Wrapper achten jeweils auf die für sie spezifischen Ressourcen und verfügen über die Funktionen, die notwendig sind um die Aktivitäten auf den jeweiligen Maschinen abzuarbeiten. Die Wrapper müssen systemnah mit den Bediensystemen der Maschinen entwickelt werden, um eine problemlose Integration jeglicher Maschinen in das Gesamtsystem zu gewährleisten. Im folgenden Kapitel wird genauer darauf eingegangen, wie die Kommunikation zu SmartControl funktioniert.

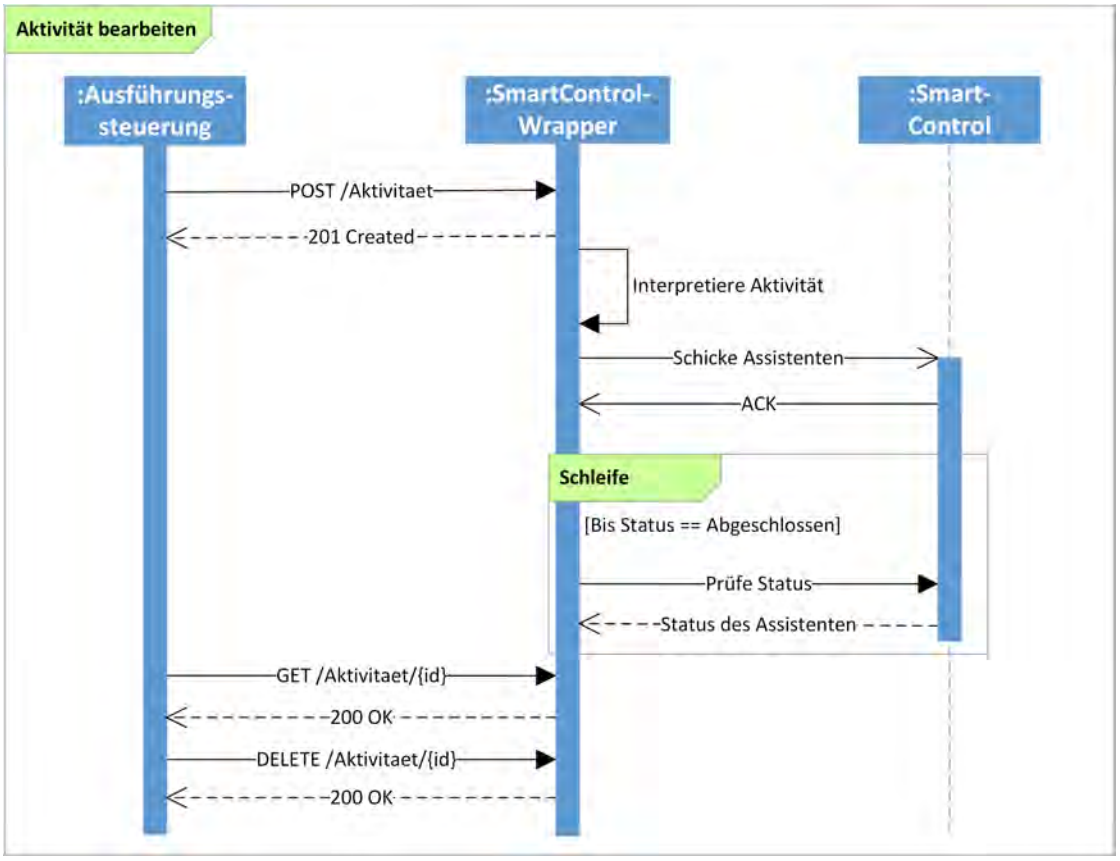


Abbildung 4.7.: Sequenzdiagramm - Kommunikationsmodul

## 4.5. Integration von SmartControl

Im Folgenden wird Integration des Maschinenbediensystems SmartControl (vgl. Kapitel 2.3) im Modellierungseditor und der Interaktions-Schicht der Konzeptarchitektur (vgl. Kapitel 4.1) vorgestellt. Hierbei wird zunächst erläutert, wie ein maschinennaher Subprozess auf SmartControl ausgeführt wird und wie dieser dafür modelliert werden muss.

### 4.5.1. Assistentenmodellierung

Die Subprozesse des Prozessmodells auf Linienebene, das im Modellierungseditor erstellt wird (siehe Abbildung 4.4), repräsentieren die eigentlichen Abläufe auf den Maschinen. Sie beinhalten Aktivitäten, die an den Maschinenbediensystemen der Maschinen ausgeführt werden sollen. Da der Modellierungseditor dem Design der Ablaufumgebung ähneln soll und jedes Maschinenbediensystem einen unterschiedlichen Aufbau der abzuarbeitenden Aktivitäten hat, benötigt die Modellierungsumgebung für jedes Maschinenbediensystem eine individuelle Ansicht. In Kapitel 4.2 wurde deshalb die Verwendung von Plugins zur Modellierung von Subprozessen festgelegt. Im Folgenden wird das Konzept zur Assistentenmodellierung für SmartControl erläutert.

Der Aufbau eines Assistenten, wie er in Kapitel 2.3 vorgestellt wurde, ist in Abbildung 4.8 detaillierter dargestellt. Die Visualisierung des Assistenten teilt sich in zwei Bereiche: Die Aktivitäten auf der rechten Seite (1) und die Inhalte der Aktivitäten entsprechend auf der linken (2). Der Maschinenoperator geht bei der Ausführung des Assistenten alle Aktivitäten (1) sequentiell durch und führt die Arbeitsanweisungen, die sie enthalten, aus. Anschließend bestätigt er die abgeschlossene Aktivität.

Um einen Assistenten für SmartControl, wie er in Abbildung 4.8 dargestellt ist, korrekt modellieren zu können, benötigt der Modellierungseditor unter anderem die Funktion Aktivitäten in eindeutiger Reihenfolge innerhalb eines Subprozesses hinzuzufügen. In die Aktivitäten sollen daraufhin verschiedene Steuerelemente eingefügt werden können. Hierzu wurde ein formularbasierter Ansatz zur Modellierung der Aktivitäten gewählt, da dieser Ähnlichkeit zu den SmartControl Assistenten hat und dies in den Anforde-

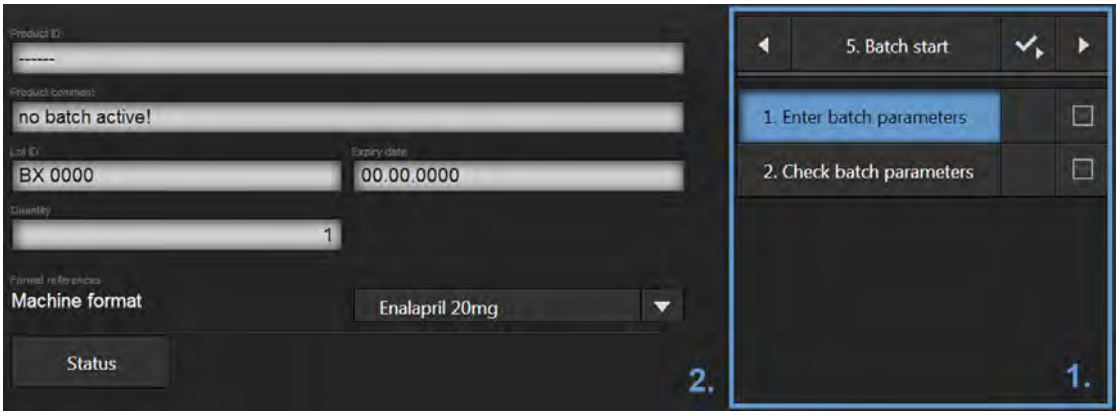


Abbildung 4.8.: SmartControl - Aufbau eines Assistenten

rungen (vgl. Kapitel 3.3) genannt ist. Zu den Steuerelementen gehören Textfelder, die als Beschreibungen in den Aktivitäten verwendet werden, und Maschinenfelder. Maschinenfelder sind typisierte Felder für Maschinenparameter, wie beispielsweise für die Maximal-Temperatur einer Heizplatte oder für den Schalter, der diese Heizplatte aktiviert. Hierbei wird unterschieden zwischen „bekannten“ Maschinenfeldern, die aus dem Maschinenbaum von SmartControl ausgewählt werden können, und „neuen“ Maschinenfeldern, die es ermöglichen neue Daten in den Maschinenbaum einzufügen. Für neue Maschinenfelder wird ein Typ (zunächst Text, Zahl, Auflistung) und ein Name angegeben, damit sie später in den Maschinenbaum eingefügt werden können. Zur Strukturierung der Aktivitäten werden die Text- und Maschinenfelder in Arbeitsschritte eingefügt. Diese Schachtelung von Elementen geht aus den SmartControl Assistenten hervor. Zudem ist es möglich, Unterschriften-Verweise hinzuzufügen, um Stellen zu markieren, an denen der Maschinenoperator mit einer Signatur die abgeschlossene Aufgabe bestätigen muss.

Abbildung 4.9 zeigt das Mockup für die Modellierung der genannten Assistenten. Das Mockup beschreibt damit ein Plugin für den Modellierungseditor. Es unterscheidet sich in den markierten Bereichen zum Mockup aus Abbildung 4.4, verwendet allerdings dieselbe Navigationsleiste und den Bereich zum Speichern und Löschen des Prozessmodells. Es bezieht sich dabei auf das Szenario aus Kapitel 3.1 und zeigt die Modellierung der Aktivität *Formatwechsel*. Der Editoranwender fügt zunächst die Aktivitäten über

die „Schritt hinzufügen“-Schaltfläche im ersten Bereich des Mockups ein. Wählt er dort eine Aktivität aus, so kann er im zweiten Bereich des Mockups den Inhalt der Aktivität bearbeiten. Hierzu fügt er die Steuerelemente aus dem dritten Bereich des Mockups per Drag&Drop in das zentrale Formular ein. Innerhalb des Formulars kann er dann die Steuerelemente bearbeiten, indem er Textfeldern eine Beschreibung gibt und für Maschinenfelder passende Maschinenparameter auswählt.

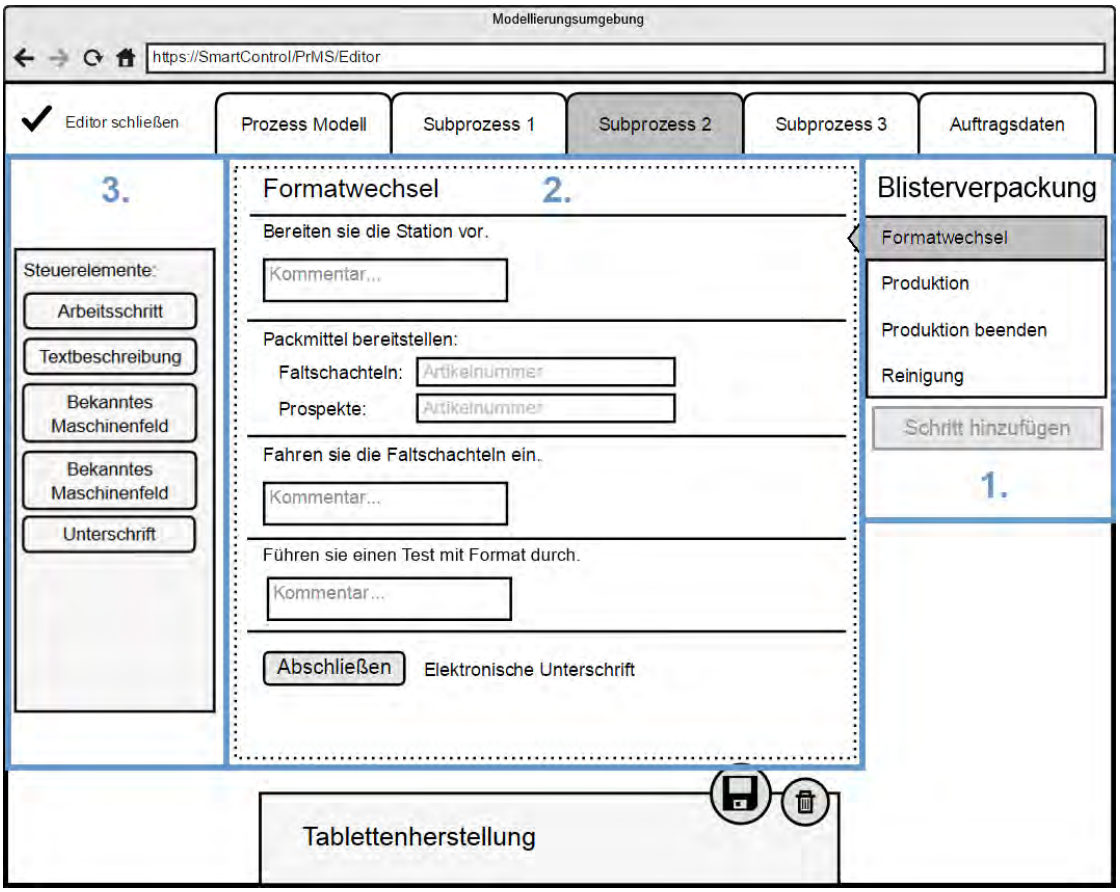


Abbildung 4.9.: Mockup - Editorplugin für SmartControl

### 4.5.2. Dynamische Assistentenausführung in SmartControl

Zum derzeitigen Stand müssen die Assistenten, die in SmartControl verwendet werden, zum Start des Systems vorhanden sein, um sie zu laden. Für das entworfene Konzept ist

dies jedoch nicht ausreichend, da ein Maschinenoperator jedes mal einen Neustart von SmartControl vornehmen müsste, wenn er einen Arbeitsauftrag erfüllen möchte. Es ist deshalb von Seiten SmartControl notwendig, während der Ausführungszeit Assistenten entgegenzunehmen, zu laden und dem Maschinenoperator darzustellen. Zu diesem Zweck ist es vorgesehen, dass der SmartControl Wrapper eine Python-Session auf dem System öffnet und den Assistenten direkt in Python übergibt. Dies erfordert jedoch eine Anpassung der SmartControl Assistenten-Komponente und ist nicht Bestandteil der Prototypen-Entwicklung, sondern wird von der Firma Uhlmann durchgeführt werden.

## 4.6. Usability-Aspekte

Die in den Abbildungen 4.3, 4.4 und 4.9 gezeigten Mockups zielen auf eine klar strukturierte, abstrakte Oberfläche ab - sie bietet dem Anwender die Funktionen, die er in einer bestimmten Situation benötigt, um den Prozess korrekt zu modellieren. Die Navigationsleiste und der Bereich, der das Benennen, Speichern und Löschen des Prozessmodells ermöglicht, sind aus Sicht der Usability frei von weiteren Inhalten zu halten und müssen in jeder Ansicht der Modellierungsumgebung gleich dargestellt werden, um dem Anwender die Navigation zu erleichtern [Nie06].

Der Anwender soll sowohl durch den Aufbau der Seiten, als auch durch deren Gestaltung an SmartControl erinnert werden, um einen Bezug zu diesem herzustellen. Dazu sollen sich die Steuerelemente der Modellierungsumgebung an denen von SmartControl orientieren und zur Gestaltung dieselbe Farbpalette verwendet werden (siehe Abbildung 4.10). Diese Aspekte sind in den nichtfunktionalen Anforderungen (vgl. Kapitel 3.3.2) definiert.


Abbildung 4.10.: Design - SmartControl Farbpalette

Der Großteil der Aktionen, die der Anwender in der Modellierungsumgebung durchführt, sind das Einfügen von Steuerelementen in das Prozessmodell, in die Subprozesse oder in das SmartControl Plugin. Um dem Anwender eine einfache und intuitive Art des Modellierens zu ermöglichen, wurde ein Drag&Drop Ansatz gewählt, der für das Einfügen jeglicher Steuerelementes verwendet wird. Wählt der Anwender ein Steuerelement durch das Klicken und Gedrückt-Halten der Maustaste aus, verändert sich die Grafik des Mauszeigers und er kann das Steuerelement über den Bildschirm ziehen. Nach der Auswahl werden die Bereiche, in die das Steuerelement eingefügt wird angezeigt und hervorgehoben, sobald sich der Mauszeiger über einem Bereich befindet. Das Steuerelement wird eingefügt, nachdem der Anwender die Maustaste loslässt.

# 5

# Systemimplementierung

Inhalt dieses Kapitel ist die prototypische Implementierung des in Kapitel 4 entworfenen Konzepts zur Unterstützung der Modellierung und Ausführung von Verfahrensprozessen. Dazu wird zunächst die Ausgangsbasis der Implementierung vorgestellt, das Datenmodell von Prozessmodellen festgelegt und anschließend die Umsetzung der Client- und der Server-Anwendung detailliert, unter anderem mit Code-Beispielen, erläutert. Außerdem wird der Proof-of-Concept-Ansatz der Kommunikations-Anwendung erklärt.

## 5.1. Ausgangsbasis

Als Ausgangsbasis für die Implementierung des in Kapitel 4 entwickelten Konzepts, dient ein bestehender Prototyp der Universität Ulm [MGKR15]. Der Prototyp, der angepasst und erweitert wurde, eignet sich als Grundlage, da er bereits aus einer Client- und einer Server-Anwendung besteht und das Prozessmanagementsystem *AristaFlow* einbettet. Zusätzlich bietet das Konzept der *Context-aware Process Injection*[1], das in [MGKR15] vorgestellt wurde, eine vielversprechende Grundlage zur erweiterten, kontexteinbeziehenden Unterstützung von Verfahrensprozessen in der Zukunft (vgl. Kapitel 7.2.2).

Die Benutzeroberfläche der Client-Anwendung nutzt HTML, JavaScript und CSS, wobei die Programmierlogik in CoffeeScript [Cof15] geschrieben ist und in JavaScript-Dateien übersetzt wird. Zur Erweiterung von HTML wird die Polymer-Bibliothek [Pol15] verwendet, die eine Vielzahl geeigneter Komponenten zur Verfügung stellt. Der Client kommuniziert über eine REST-Schnittstelle mit dem Server und kann so Prozessmodelle

[1] zu dt. Kontext-orientierte Prozess Injektion

vom Server abrufen, bearbeiten und instanziieren. Die Hauptfunktion der bestehenden Client-Anwendung bietet dem Anwender die Möglichkeit eine Kontext-orientierte Prozessfamilie (engl. Contextaware Process Family - CPF) anzulegen. Ein CPF besteht aus einem Basis-Prozessmodell mit Erweiterungsgebieten, Kontext-Situationen (die auf Prozessparametern beruhen), Prozess-Fragmenten (die zur Ausführungszeit in Erweiterungsgebiete eingefügt werden können) und Injektions-Spezifikationen [MGKR15]. Mithilfe dieser Komponenten können Prozessinstanzen zur Auführungszeit dynamisch auf Kontext-Faktoren reagieren, um die Flexibilität der Prozessinstanz zu erhöhen. In Abbildung 5.1 wird dies anhand des Szenarios aus Kapitel 3.1 gezeigt. Das Prozess-Fragment mit der Aktivität „Endverpackung“ wird hier nur dann eingefügt, wenn die *Kontext-Situation 1* zum dem Zeitpunkt erfüllt ist, an dem die Prozessausführung an *Erweiterungsgebiet 1* gelangt.

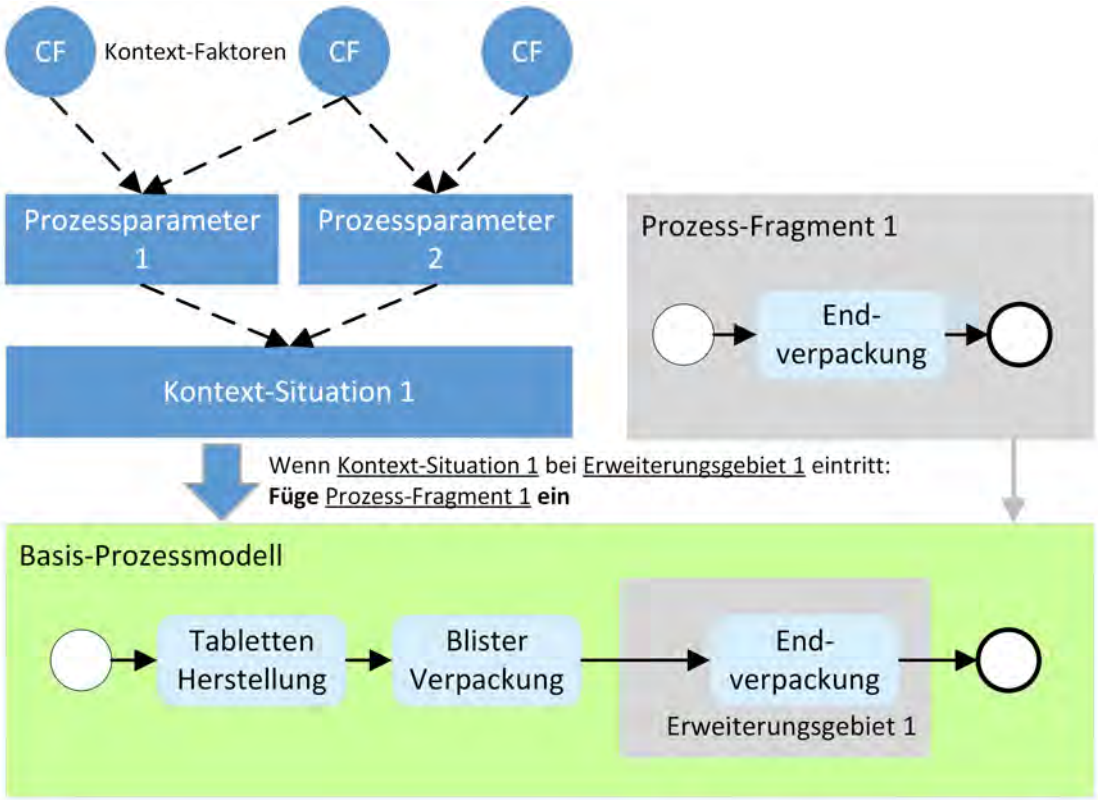


Abbildung 5.1.: Context-aware Process Injection nach [MGKR15]

Die Server-Anwendung ist in JavaEE (Java Enterprise Editon) [Ora15] entwickelt und kann auf einem Anwendungsserver (z.B. Tomcat) ausgeführt werden. Mittelpunkt der Server-Anwendung ist ein Servlet. Dieses Servlet kapselt die Anwendungslogik und kommuniziert mit dem PrMS und der Datenbank über Injektionen[2]. Die Server-Anwendung realisiert zusätzlich ein *JavaBean*-Datenmodell. Auf diese Weise werden Prozessmodelle inklusive der Spezifikationen für Verfahrensprozesse im Java Code als Objekte dargestellt und im Servlet über eine objektorientierte Datenbank-Schnittstelle gespeichert. Die Server-Anwendung kapselt außerdem das PrMS AristaFlow, das eingebettet in die Server-Anwendung gestartet und verwendet wird. Bei AristaFlow handelt es ich um ein prozessgestütztes Informationssystem (PAIS), das aus einem Projekt der Universität Ulm entstanden ist [DRRM$^+$09]. AristaFlow legt sein Hauptaugenmerk auf dynamische Prozesse: Es erlaubt Ad-hoc-Änderungen während der Ausführung von Prozessinstanzen und ermöglicht eine kontrollierte Prozessevolution [RW12]. Eine weitere Stärke von AristaFlow liegt in seiner Integrationsfähigkeit durch eine wohl-definierte API.

## 5.2. Datenmodell

Der Prototyp aus [MGKR15] bezieht sich bei Prozessmodellen immer auf bestehende Prozessmodelle aus AristaFlow, die dort als *Templates* bezeichnet werden. Da die Prozessmodelle in dieser Arbeit aber in der Client-Anwendung selbst erstellt werden, wird in der Server-Anwendung ein neues Datenmodell benötigt, um die Prozessmodelle zu speichern. Dazu wurden sechs neue Klassen für die Server-Anwendung erstellt. Zuerst wird eine Klasse für das Prozessmodell (PrM) selbst benötigt, die eine Liste aller Elemente des Graphen, den der Client zur Darstellung erstellt (vgl. Kapitel 5.3.2), enthält. Für diese Elemente wurde ebenfalls eine Klasse angelegt („GraphField“). Um maschinennahe Subprozesse abzubilden werden außerdem die Klassen „Subprocess“ (Subprozess), „Step“ (Schritt), „Content“ (Inhaltselement) und „Control“ (Steuerelement) benötigt. Diese Klassen bauen aufeinander auf. So enthält ein Prozessmodell eine Liste von Subprozessen, ein Subprozess enthält eine Liste von Schritten, ein Schritt

[2] Wird ein Objekt in einer Java-Klasse injiziert, kann es dort verwendet werden, ohne im Java-Code instanziiert zu werden, und die Server-Anwendung kümmert sich im Hintergrund automatisch um dessen Verwaltung.

enthält eine Liste von Inhaltselementen und ein Inhaltselement enthält eine Liste von Steuerelementen. Im Klassendiagramm (Abbildung 5.2) sind diese Klassen mit ihren Attributen dargestellt, wobei die Beziehungen unter den Klassen verdeutlicht wird.


Abbildung 5.2.: Klassendiagramm - Datenmodell

Speichert der Client ein Prozessmodell auf dem Server über die REST-Schnittstelle, so übergibt er dieses als JSON-Objekt. Die Server-Anwendung nimmt dieses im Java-Code als *LinkedHashMap*-Objekt, das aus *Key-Value-Paaren* besteht, entgegen. Jede der genannten Klassen enthält deshalb einen Konstruktor, der aus den passenden *Key-Value-Paaren* ein Objekt erstellen kann. Listing 5.1 zeigt beispielhaft den Konstruktor der Subprozess-Klasse. Der Subprozess aus dem JSON-Objekt wird als *LinkedHashMap*-Objekt übergeben und über die *Key-Namen* können dessen Eigenschaften abgerufen und gesetzt werden (Zeile 2). Da es sich bei der Eigenschaft „steplist“ um ein Array von Schritten handelt, muss dieses in einer Schleife durchlaufen werden (Zeile 4), wobei die Objekte des Arrays direkt in den Konstruktor der Step-Klasse übergeben werden können (Zeile 5).

Listing 5.1: Subprozess-Konstruktor

```
1  public Subprocess( LinkedHashMap values ) {
2    setName( (String) values.get("name"));
3    ...
```

```
4       for (Object step : (ArrayList) values.get("steplist"))
5         addStep(new Step( (LinkedHashMap) step));
6     }
```

## 5.3. Client-Anwendung Prototyp

Im Folgenden wird der entwickelte Prototyp der Client-Anwendung, der in Kapitel 4.2 konzipiert ist, beschrieben. Hierzu werden die implementierten Komponenten, die das Konzept beschreibt, einzeln vorgestellt und erklärt.

### 5.3.1. Startmenü

Die Web-Anwendung des Clients besteht aus zwei Hauptkomponenten, die zur Einsicht und Steuerung von Prozessinstanzen sowie zur Erstellung von Prozessmodellen dienen. Die erste Komponente wurde aus [MGKR15] übernommen, da sie die Visualisierung und Ausführung von Prozessinstanzen beinhaltet. Diese Komponente stellt eine Verbindung mit dem PrMS her, um Prozessinstanzen zu erstellen und zu steuern. Für den zu entwickelnden Proof-of-Concept Prototypen wird diese Komponente größtenteils unberührt übernommen und als Ausführungsumgebung der Prozessinstanzen verwendet.

Ein wichtiger Aspekt ist die Navigation, die wie in Abbildung 5.3 durch eine Navigationsleiste dargestellt ist und damit dem Konzepts (vgl. Kapitel 4.2) entspricht. Sie dient dem Anwender als Orientierung und zum Wechseln zwischen den Bereichen, z.B. von der Ausführungsumgebung zur Modellierungsumgebung. Abbildung 5.3 zeigt das Startmenü der Client-Anwendung, das aus der Navigationsleiste, der Ausführungsumgebung und der Modellierungsumgebung besteht. Die Modellierungsumgebung wird im Startmenü als eine Auflistung von existierenden Prozessmodellen angezeigt. Die Prozessmodelle werden in der Auswahl dabei durch eine graphisch visualisierte Vorschau dargestellt.

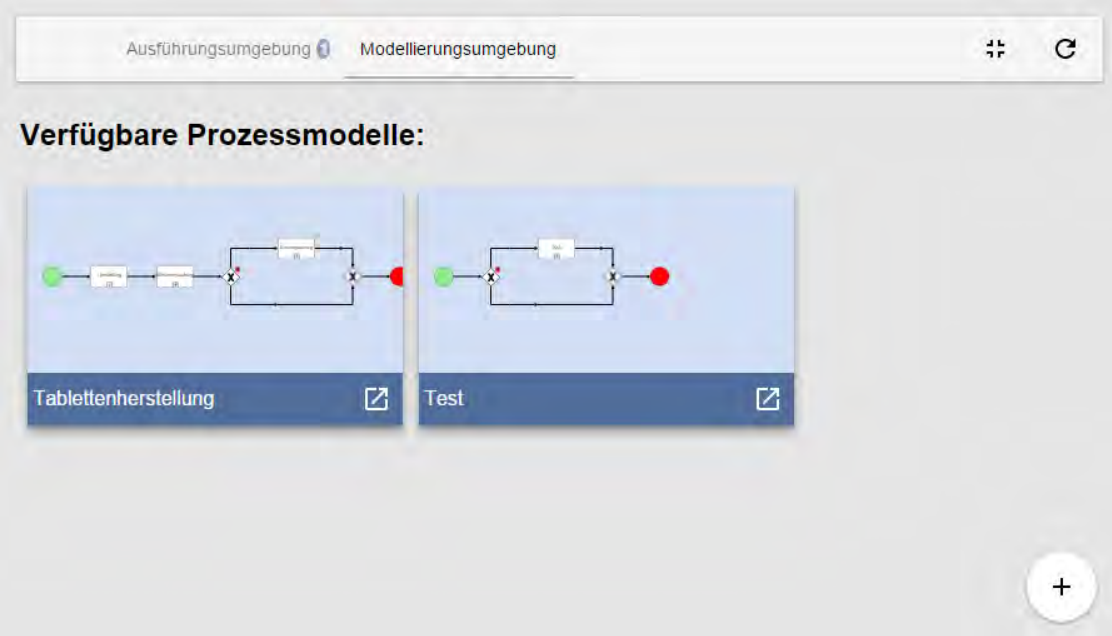


Abbildung 5.3.: Implementierung - Prozessmodell Verwaltung

### 5.3.2. Modellierungseditor

Die Client-Anwendung stellt durch Polymer eine klare Trennung der Komponenten her, indem die Komponenten als benutzerdefinierte Elemente (beginnend mit *Polymer-Element*) in getrennten HTML-Dokumenten gekapselt werden. Dies ermöglicht den Einsatz der Komponenten in übergeordneten Elementen. Jedes Polymer-Element hat eine eigene CoffeeScript-Datei, die dessen Logik enthält. Die Kapselung von Komponenten in Polymer-Elemente macht die Strukturierung der Client-Anwendung einfacher und übersichtlicher. Die Webseiten, aus denen die Client-Anwendungen besteht, werden in der Implementierung als Polymer-Aktivitäten bezeichnet. Dazu gehören eine Hauptaktivität, als Startmenü der Client-Anwendung, eine Polymer-Aktivität zur Darstellung von Prozessinstanzen und eine Polymer-Aktivität zur Darstellung von Prozessmodellen.

Der Modellierungseditor ist in der Polymer-Aktivität *activity-process-editor* entwickelt. Diese Polymer-Aktivität bindet die Komponente *editor-process* ein, um das Prozessmodell zu erstellen, und die Komponente *editor-plugin-smartcontrol*, in der die Subprozesse speziell für SmartControl modelliert werden können. Der Großteil der Entwicklung des

Modellierungseditors stellte die Implementierung dieser beiden Komponenten dar. Listing 5.2 zeigt, wie diese Komponenten in der Aktivität *activity-process-editor* eingebunden sind.

Listing 5.2: Polymer-Element für die Modellierungsumgebung Aktivität

```
1  <polymer-element
2    name="activity-process-editor"
3    attributes="prmID">
4      <template>
5        ...
6        <editor-process tabs="{{tabs}}" ... />
7        ...
8        <editor-plugin-smartcontrol
9            subprocess="{{selectedSubProc}}" />
10       ...
11     </template>
12     <script src="activity-process-editor.js"/>
13 </polymer-element>
```

Bei den Attributen in den HTML-Elementen (z.B. tabs in Zeile 6) handelt es sich um Parameter, die der Komponente übergeben werden. Innerhalb der *editor-plugin-smartcontrol*-Komponente kann so festgestellt werden, welcher Subprozess angezeigt und bearbeitet werden soll. Neben der Möglichkeit, Komponenten zu kapseln, bietet Polymer auch eine Vielzahl von Elementen, die die HTML-Bibliothek erweitern. Eine vollständige Auflistung kann auf der Webseite des Polymer-Projektes eingesehen werden [Pol15].

Der Modellierungseditor selbst besteht aus einem Bereich zur Modellierung des Gesamtprozesses, speziellen Modellierungsbereichen für die Subprozesse und einem Konfigurationsbereich, in dem Auftragsparameter bearbeitet werden können. Die Implementierung dieser Bereiche werden in den folgenden Abschnitten im Detail beschreiben.

### Prozessmodell erstellen

Beim Öffnen des Modellierungseditors wird das Prozessmodell angezeigt. Dabei orientiert sich die Darstellung an BPMN, um eine standardisierte Prozessmodellierung zu gewährleisten. Wurde das Prozessmodell neu erzeugt, besteht es aus einem Start- und einem Ende-Ereignis, zwischen denen Steuerelemente eingefügt werden können. Abbildung 5.4 zeigt die Umsetzung des Fallbeispiels aus Kapitel 3.1, das aus den drei Subprozessen „Herstellung“, „Bilsterverpackung“ und „Endverpackung“ besteht. In diesem Beispiel wurde jedoch der letzte Subprozess innerhalb eines ODER-Gatters platziert, da die Endverpackung optional ist. Der linke Bereich in Abbildung 5.4 stellt die drei Steuerelemente „Subprozess hinzufügen“, „UND-Gatter hinzufügen“ und „ODER-Gatter hinzufügen“ bereit - diese Steuerelemente können per Drag&Drop in das Prozessmodell gezogen werden. Die Bereiche, in denen die Steuerelemente platziert werden können, werden durch gestrichelte Kästen markiert, die zu Beginn der Drag&Drop-Aktion eingeblendet und nach deren Abschluss ausgeblendet werden, wie es das Konzept in Kapitel 4.6 beschreibt.

Die Visualisierung des Prozessmodells ist durch Scalable Vector Graphic (SVG) aufgebaut. Bei SVG handelt es sich um XML-basierte Vektor-Bilder, die direkt in HTML Seiten eingebunden und auch über Skriptsprachen, wie z.B. CoffeeScript, verändert werden können. Dies bedeutet, das SVG-Element, das in HTML eingebunden ist, ist erweiterbar und besteht aus einer Vielzahl von XML-Elementen, wie Kreisen, Pfeilen und Rechtecken.

Um die Darstellung des Prozessmodells und vor allem die Drag&Drop-Funktion zu ermöglichen, wurde in HTML ein leeres SVG-Element eingefügt und in CoffeeScript eine Klasse für den Graphen angelegt, die das SVG-Element mit den richtigen XML-Elementen füllt. Die Graph-Klasse hat dabei zwei wichtige Objekte: Zum einen ein Objektmodell, das das Prozessmodell in JSON darstellt, und zum anderen eine Referenz zu dem SVG-Element, an das die XML-Elemente angefügt werden. Um den Zugriff auf die XML-Struktur innerhalb des SVG-Elementes zu erleichtern wird die *D3*-Bibliothek verwendet [DDD15]. Diese stellt Funktionen bereit, um schnell einzelne Elemente oder

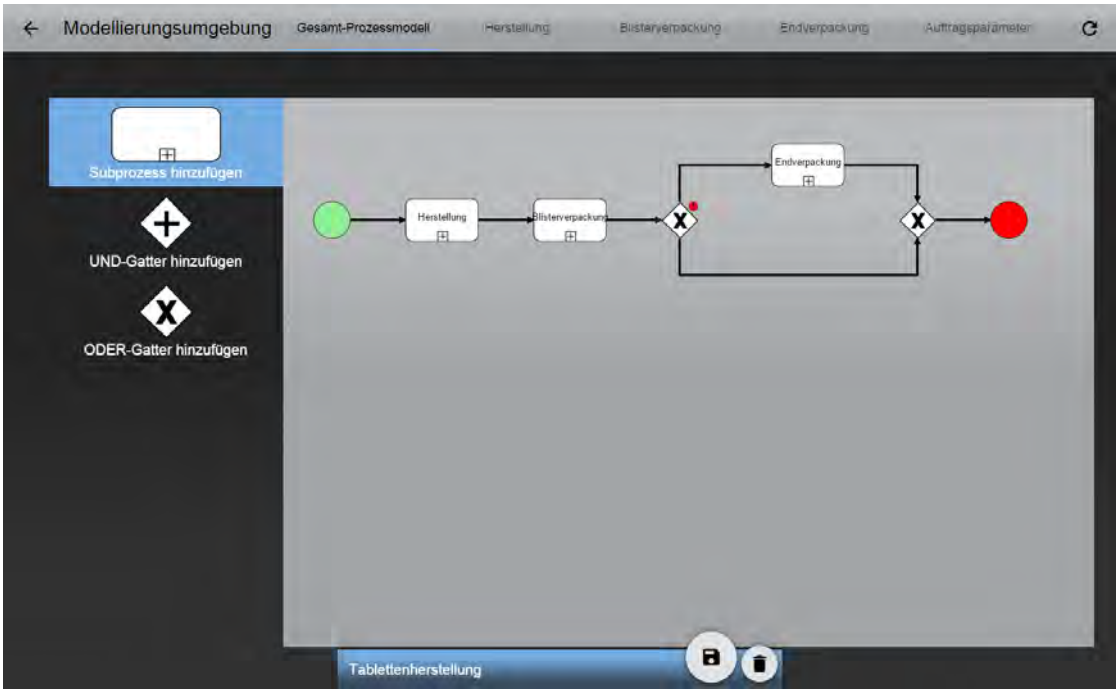


Abbildung 5.4.: Implementierung - Modellierungsumgebung

Elementsammlungen abzurufen. Im Folgenden wird der Ablauf skizziert, der die Vektor-Grafik generiert:

1. Wenn Graph angelegt wird
   Initialisiere Objektmodell mit Start-Ereignis des Prozesses, Drop-Bereich[3], Ende-Ereignis des Prozesses

2. Für jedes Objekt im Objektmodell:

   a) Füge das entsprechende Element in das SVG-Element.

   b) Füge eine Verbindung zum nächsten Objekt in das SVG-Element ein, mit Außnahme des Ende-Ereignisses.

   c) Setzte für die Drop-Bereich Listener, die auf das Maus-Ereignis *onDrop* warten und dadurch das Einfügen von Steuerelemente ermöglichen.

[3] Bereich im Prozessmodell, in den Elemente wie Subprozesse eingefügt werden können. Ein Drop-Bereich befindet sich zwischen jedem der Elemente des Prozessmodells.

3. Wenn ein Steuerelement eingefügt wird:
    a) Ermittle die ID des Drop-Bereichs.
    b) Füge im Objektmodell hinter dem Drop-Bereich mit dieser ID ein neues Objekt vom Typ des Steuerelements ein und dahinter einen neuen Drop-Bereich.
    c) Baue das SVG-Element anhand des Objektmodells erneut auf.
    d) Zeichne das SVG-Element.

Um diese Struktur des Graphen zu verdeutlichen, befindet sich im Anhang das Listing A.1, das das zu Abbildung 5.4 korrespondierende Objektmodell enthält.

Die Drag&Drop-Funktion ist über sogenannte Event-Listener realisiert. Die Event-Listener können für HTML-Elemente im DOM-Baum[4] implementiert werden. Die Steuerelemente implementieren die Funktion *onDragStart(Event)*, welche ausgelöst wird, sobald die Maustaste über dem Element gedrückt wird. Das SVG-Element implementiert die Funktionen *onDragModelEnter(Event)* und *onDragModelLeave(Event)*, um die Drop-Bereiche ein- und auszublenden. Jeder Drop-Bereich implementiert *onDrop(Event)*, das ausgelöst wird, sobald die Maustaste über dem Drop-Bereich und nach dem onDragStart-Event losgelassen wird. Das onDrop-Event ermittelt den Typ des Steuerelements und fügt dieses an die passende Stelle im Objektmodel ein.

### Subprozesse erstellen

Nachdem ein maschinennaher Subprozess in das Prozessmodell eingefügt wurde, müssen ihm Meta-Informationen beigefügt werden. Dazu wird der Subprozess angeklickt und der Dialog aus Abbildung 5.5 erscheint im Modellierungseditor. In diesem Dialog werden Name, Modul und Beschreibung ausgefüllt und es muss ein Plugin ausgewählt werden. Passend zu diesem Plugin wird die Komponente für den gewählten Subprozess geladen. Der Prototyp implementiert ein Plugin zur Erstellung von SmartControl-Assistenten innerhalb eines Subprozesses. Das Plugin zur Modellierung von SmartControl-Assistenten bezieht sich auf das Konzept dargestellt in Kapitel 4.5. Zusätzlich zum Anlegen eines

[4] Das „Document Object Model“ dient als Schnittstelle zwischen JavaScript und HTML und enthält alle Elemente einer Webseite.

neuen Subprozesses besteht die Möglichkeit, einen vorhandenen Subprozess aus einer Liste auszuwählen und diesen als Vorlage zu verwenden.

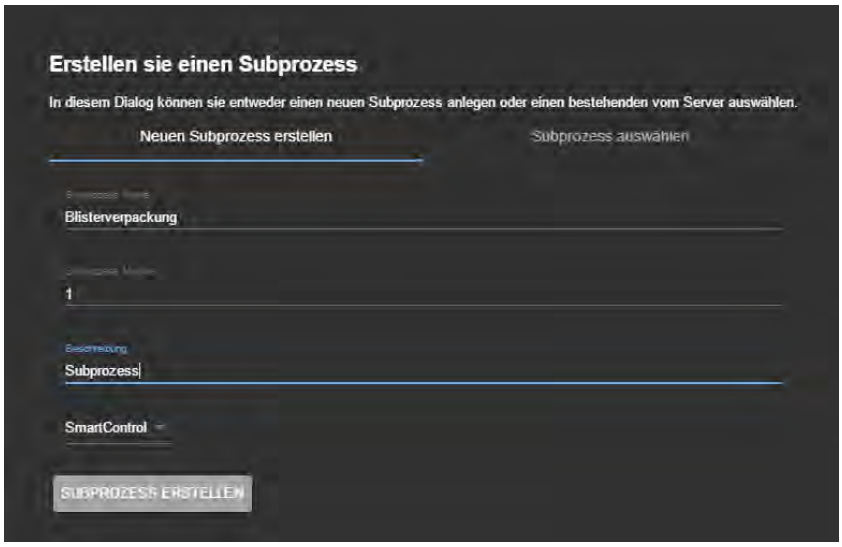


Abbildung 5.5.: Implementierung - Subprozess Dialog

Ein neuer Subprozess wird innerhalb des Modellierungseditors als Objekt in ein Array gespeichert und zusätzlich in die Navigationsleiste eingefügt. Auf diese Weise kann der Subprozess in der Navigationsleiste ausgewählt werden. Ist für den ausgewählten Subprozess das SmartControl-Plugin gesetzt, so wird das Polymer-Element *editor-plugin-smartcontrol* geladen. Die Navigationsleiste besitzt daher auch eine Scroll-Funktion, da sie mit steigender Anzahl an Subprozessen über den Bildschirmrand hinaus gefüllt werden kann.

Das SmartControl-Plugin besteht zunächst aus einer leeren Liste von Aktivitäten, der mit der Schaltfläche „Schritt hinzufügen“ Aktivitäten hinzugefügt werden können. In Abbildung 5.6 ist der Subprozess *Blisterverpackung* ausgewählt, der die Aktivitäten *Formatwechsel*, *Produktion beginnen*, *Produktion beenden* und *Reinigung* enthält. Ist eine Aktivität ausgewählt, wie *Formatwechsel* in Abbildung 5.6, kann diese entsprechend bearbeitet werden. Ein Schritt ist gegliedert in Prozedur-Schritte und diese bestehen wiederum aus Steuerelementen zur Beschreibung und Kennzeichnung von Maschinenfeldern. Der Editoranwender fügt zunächst sequentiell Prozedur-Schritte ein, die durch eine

horizontale Linie voneinander getrennt sind. Anschließend kann er die Steuerelemente an den passenden Stellen durch Drag&Drop platzieren. Die Steuerelementen sind auf der linken Seite aufgereiht und können direkt in die Schritte gezogen werden. Die Logik zum Ein- und Ausblenden der Drop-Bereiche, sowie zum Einfügen von Steuerelementen, besteht aus Event-Listenern und ist dieselbe wie beim Modellieren des Prozessmodells auf Linienebene (vgl. Abbildung 5.4).

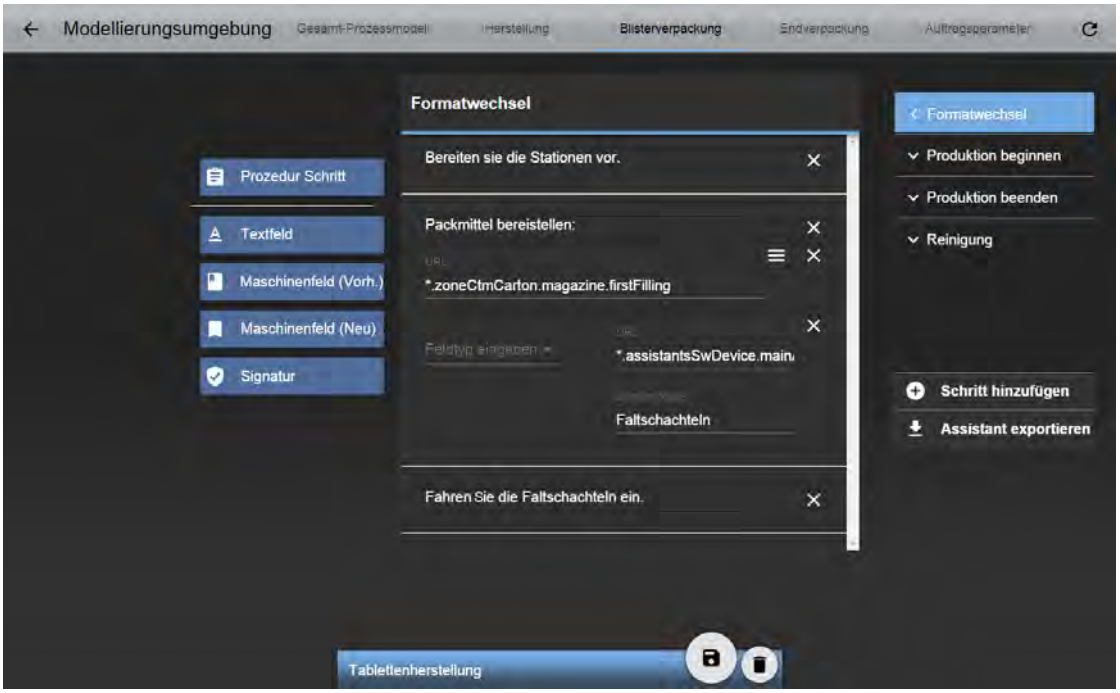


Abbildung 5.6.: Implementierung - SmartControl Subprozess modellieren

Eine Besonderheit, die das Plugin für SmartControl benötigt, sind die Steuerelemente *Maschinenfeld (Neu)* und *Maschinenfeld (Vorhanden)*. Ein Maschinenfeld ist ein Element aus dem SmartControl-Maschinenbaum, über den Maschinenparameter angesprochen werden können. Im Modellierungseditor gibt es zwei Möglichkeiten auf Maschinenparameter zuzugreifen:

1. Die genaue Pfadangabe im Maschinenbaum, wobei der Pfad bekannt sein muss. Beispielhaft ist dies in Abbildung 5.6 im Inhalts-Bereich der Aktivität *Formatwechsel* im dritten Steuerelement zu sehen.

2. Das Anlegen eines neuen Feldes im Maschinenbaum, wie in Abbildung 5.6 im vierten Steuerelement zu sehen ist. Bei einem neuen Maschinenfeld wird der Pfad relativ zum Assistenten angegeben, um das Einfügen des Feldes in den Maschinenbaum zu ermöglichen und der Anwender keine Kenntnis des genauen Pfades besitzen muss. Außerdem benötigt das Feld einen Namen und einen Typ.

Nachdem der maschinennahe Subprozess modelliert wurde, wird daraus ein Assistent erstellt, der in SmartControl eingebunden werden kann. Das Format, in dem diese Funktion den Assistenten erstellt, ist in Kapitel 5.5 geschildert.

#### Auftragsparameter konfigurieren

Neben dem Prozessmodell und den Subprozessen gibt es einen weiteren Bereich in der Navigation - die Auftragsparameter. In diesem Bereich werden alle benötigten Parameter in Tabellenform aufgelistet (siehe Abbildung 5.7). Der Anwender kann neue Auftragsparameter hinzufügen, indem er den Namen des Parameters angibt, einen Typ (Text, Zahl, Datum, Tabelle) auswählt und selektiert, ob der Parameter bei der Initialisierung des Prozessmodells angegeben werden muss. Die Auftragsparameter werden jeweils als JSON-Objekt in ein Array des Modellierungseditors gespeichert.

Die Tabelle ermöglicht eine Abbildung der Auftragsparameter nach „Außen und Innen“. Dies bedeutet, dass bei der Instanziierung eines Prozesses die Auftragsparameter übergeben und vom Anwender oder dem ERP-System mit Werten gefüllt werden müssen. Außerdem kann der Anwender die Auftragsparameter innerhalb des maschinennahen Subprozesses verwenden, indem er sie über eine spezielle Notation in einem Textfeld angibt (vgl. Kapitel 4.2). Möchte der Anwender zum Beispiel die Artikelnummer als Beschreibung angeben, muss er diese in den Auftragsparametern definieren und in einem Textfeld folgende Zeichenkette einbauen: *#param.Artikelnummer*. Beim Instanziieren wird auf diese Zeichenkette dann die korrekte Artikelnummer abgebildet.

Für die Eingabe der Werte bei der Instanziierung ist im Proof-of-Concept Prototypen eine Benutzeroberfläche implementiert, in der die Auftragsparameter eingegeben werden müssen, bevor der Prozess gestartet werden kann. Der Server-Anwendung werden die

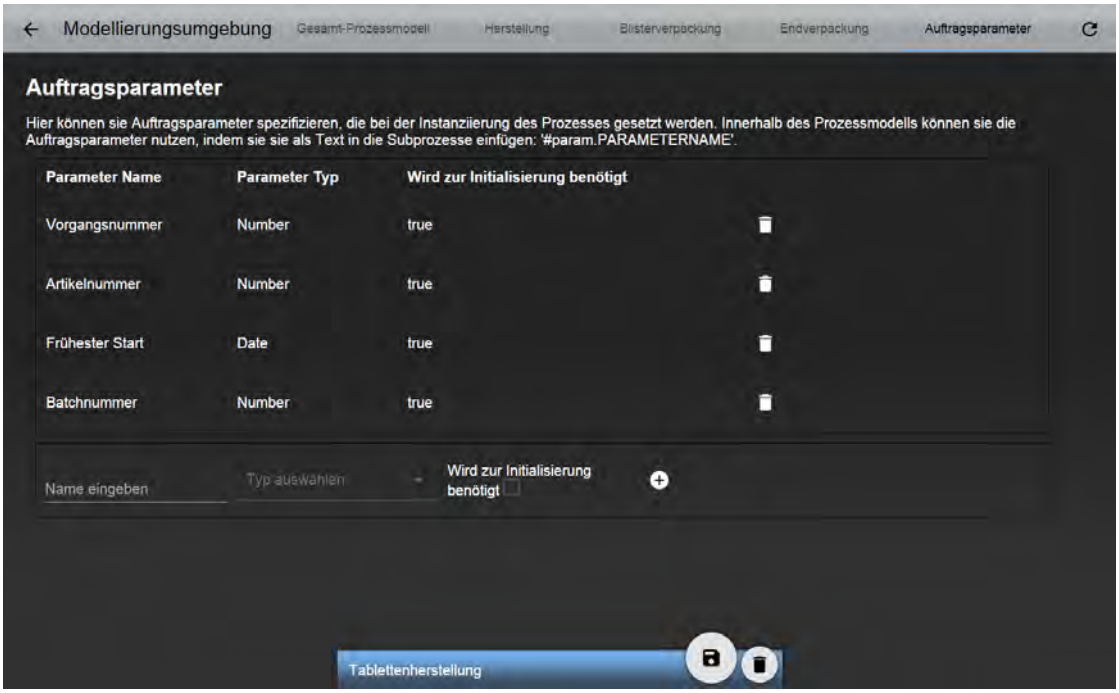


Abbildung 5.7.: Implementierung - Auftragsparameter

Parameterwerte zusätzlich zum Prozessmodellnamen bei der Instanziierung übergeben (vgl. Kapitel 5.3.3).

### Prozessmodell speichern

Abschließend erfolgt im Modellierungseditor das Speichern des Prozessmodells. Die dafür relevanten Elemente, die auf dem Server gespeichert werden müssen, sind:

- **Der Name des Prozessmodells**, als Zeichenkette
- **Das SVG-Element**, als Zeichenkette, um im Navigationsbereich eine Vorschau des Prozessmodells anzuzeigen
- **Das Objektmodell des Graphen**, als JSON-Objekt
- **Die Subprozesse**, als Array von JSON-Objekten
- **Die Auftragsparameter**, als Array von JSON-Objekten

Diese werden in einem JSON-Objekt gebündelt, das mit der Funktion *updatePrM()* auf dem Server unter dem Namen des Prozessmodells gespeichert wird. Näheres zur REST-Schnittstelle wird in Kapitel 5.4.1 und zur Datenhaltung in Kapitel 5.2 genannt.

### 5.3.3. Ausführen von Prozessmodellen

Zur Ausführung des fertigen Prozessmodells wird die vorhandene Komponente des Prototypen aus [MGKR15] verwendet. In der Client-Anwendung wird hierzu über die REST-Schnittstelle (*launchScenario(prmID)*) eine Prozessinstanz angelegt, die über die Prozessmodell-ID (prmID) aus der Datenbank bezogen wird. Die Ausführung der Prozessinstanz kann der Anwender in der Ausführungsüberwachung nachverfolgen, wie in Abbildung 5.8 dargestellt ist. Hierbei ist zu beachten, dass sich die Aktivitäten der Prozessinstanz von denen im Prozessmodell unterscheiden können. Dieser Umstand, der vor allem die maschinennahen Subprozesse betrifft, wird in Kapitel 5.4.3 näher erläutert.

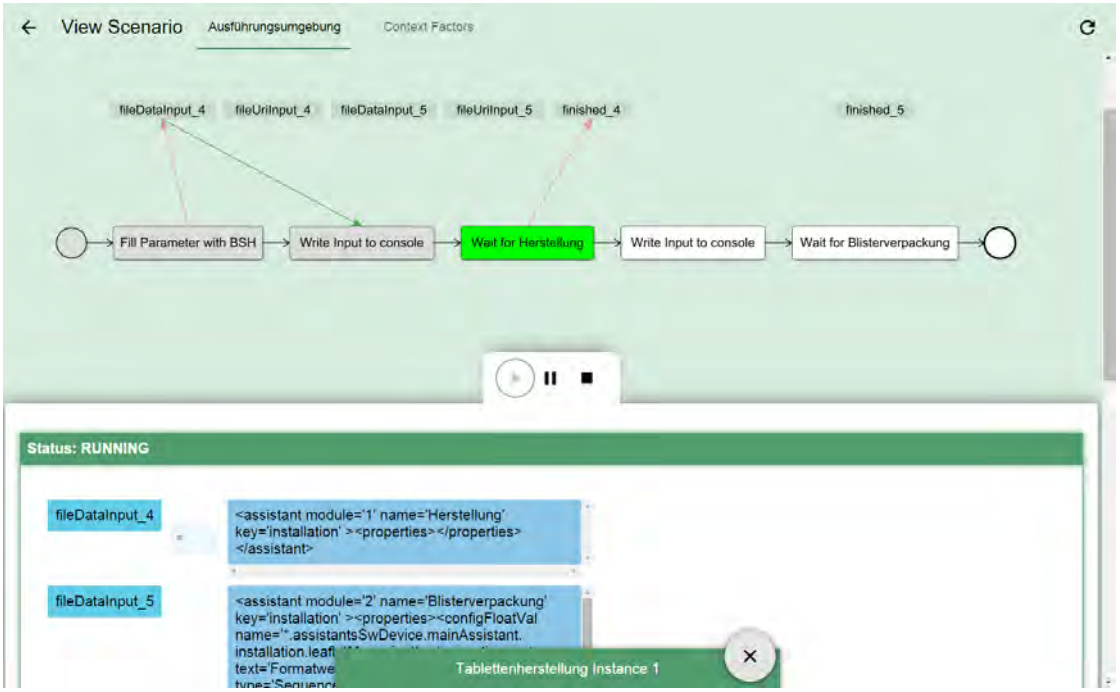


Abbildung 5.8.: Implementierung - Ausführung einer Prozessinstanz

Die Aktivitäten der Prozessinstanz werden vom PrMS ausgeführt und in der Ausführungsüberwachung visualisiert. Die Aktivität, die derzeitig bearbeitet wird, ist hierbei grün markiert, während die bereits abgeschlossenen Aktivitäten grau hinterlegt sind. Im unteren Bereich der Ausführungsumgebung werden die Datenobjekte, die in der Prozessinstanz verwendet werden, angezeigt. Beispielsweise die Assistenten, die durch eine SmartControl-Maschine ausgeführt werden sollen. Die Prozessinstanz in Abbildung 5.8 wartet auf den Abschluss der Aktivität „Herstellung“.

## 5.4. Server-Anwendung Prototyp

Zur Speicherung von Prozessmodellen und Ausführung von Prozessinstanzen benötigt die Client-Anwendung ein entsprechendes Loggen auf der Server-Seite. Um das in Kapitel 4.3 entworfene Konzept als Proof-of-Concept Prototypen zu implementieren, wird die Server-Anwendung der Ausgangsbasis (vgl. Kapitel 5.1) verwendet und ergänzt.

### 5.4.1. REST-Schnittstelle

Die Server-Anwendung bietet eine REST-Schnittstelle an, um Clients die Kommunikation mit ihm via HTTP zu ermöglichen. Diese Schnittstelle ermöglicht damit Methoden zur Interaktion mit Prozessmodellen. Im Folgenden sind diese Methoden aufgelistet, dabei wird zunächst die URI der Ressource angegeben und darunter die verfügbaren HTTP-Methoden mit den Typen, die sie als Ein- bzw. Ausgabe akzeptieren.

- **rest/PrM**
  - *GET (Ausgabe: application/json)*
    Diese Methode ruft alle Prozessmodelle des Servers ab und gibt sie als Liste zurück.
  - *POST (Eingabe: text/plain)*
    Diese Methode nimmt den Namen des Prozessmodells entgegen und legt für diesen Namen ein Prozessmodell in der Datenbank an.
- **rest/PrM/{processmodel-id}**

  - *GET (Ausgabe: application/xml)*
    Diese Methode gibt das Prozessmodell, passend zum Pfad-Parameter, als XML zurück.

  - *GET (Ausgabe: application/json)*
    Diese Methode gibt das Prozessmodell, passend zum Pfad-Parameter, im JSON-Format zurück. Dies erleichtert die Bearbeitung in der Client-Anwendung. Dazu muss der Accept-Header auf „application/json“ gesetzt sein.

  - *PUT (Eingabe: application/json)*
    Diese Methode sucht über den Pfad-Parameter das passende Prozessmodell und verändert es dem übergebenem JSON-Objekt entsprechend. Die Struktur des JSON-Objektes ist am Ende von Kapitel 5.3 definiert und besteht aus Name, SVG-Element, Objektmodell des Graphen, Subprozess-Array und Parameter-Array. Diese Objekte können jeweils dem Konstruktor des entsprechenden Java-Objektes übergeben werden, das in Kapitel 5.2 erläutert werden.

  - *DELETE*
    Diese Methode löscht das im Pfad-Parameter angegebene Prozessmodell aus der Datenbank.

- **rest/PrM/{processmodel-id}/preview**

  - *GET (Ausgabe: text/plain)*
    Diese Methode gibt das SVG-Element des im Pfad-Paramters angegebenen Prozessmodells als Text aus. Dies kommt bei der Vorschau des Prozessmodells zum Einsatz.

Darüber hinaus wurde die Rest-Schnittstelle zur Instanziierung von Prozessmodellen im Vergleich zum Prototypen aus [MGKR15] so angepasst, dass die Prozessmodelle aus dieser Arbeit verwendet werden können. Die Umsetzung ist in Kapitel 5.4.3 geschildert.

### 5.4.2. Datenhaltung

Die Server-Anwendung nutzt für die Datenhaltung eine H2-Datenbank [H215]. Um die Daten auf der Datenbank zu bearbeiten, wird die Klasse „PrMStore“ verwendet, die im Session-Servlet per Injektion eingebunden wird. Der *PrMStore* sorgt für das Persistieren der Prozessmodell-Objekte in der Datenbank, wobei sie Elemente mit primitiven Datentypen (z.B. String oder Integer) direkt speichert und komplexe Datentypen (beispielsweise ein Set von Graphen-Elementen) vorher serialisiert werden müssen. Dies bedeutet, die Datentypen werden in ein Format umgewandelt (hier XML), das in der Datenbank gespeichert werden kann. Die Prozessmodell-Objekte werden über die Rest-Schnittstelle angelegt und dort über die Methode *session.addPrM(prm, true)* in der Datenbank gespeichert.

Für die Serialisierung müssen Objekte eine XML-konforme Struktur einhalten, die direkt in den Klassen definiert ist. Listing 5.3 zeigt beispielhaft drei Klassenattribute der Klasse PrM. Dabei wird der Name der Klasse bei der XML-Darstellung als XML-Attribut und die Beschreibung und die Graphen-Elemente als XML-Elemente gespeichert. Für die Graphen-Elemente wird zusätzlich ein *XMLElementWrapper* benötigt, der diese als Unterelemente des Objekts „Graph“ kennzeichnet.

Listing 5.3: Definition von Klassenattributen

```
1  ...
2  @XmlAttribute(required = true)
3  private String name;
4  @XmlElement(name = "Description",required = false)
5  private String description;
6  ...
7  @XmlElementWrapper(name = "Graph")
8  @XmlElement(name = "GraphElement",required = false)
9  private Set<GraphField> graph = new LinkedHashSet<>();
10 ...
```

### 5.4.3. Prozessausführung

Wurde ein Prozessmodell in der Datenbank gespeichert, kann eine Prozessinstanz erstellt werden. Dies geschieht über die REST-Schnittstelle *ScenarioListEndpoint*, die im Prototypen von [MGKR15] vorhanden ist und abgeändert wurde. Die *POST*-Methode dieser Schnittstelle nimmt nun eine Prozessmodell-ID entgegen und lädt zunächst das passende Prozessmodell aus der Datenbank. Außerdem wird eine Liste von Key-Value-Paaren übergeben, die mit den Auftragsparametern des Prozessmodells verknüpft werden müssen.

Abbildung 5.9 verdeutlicht anhand einer Grafik das Prinzip nach dem in der Server-Anwendung Prozessinstanzen ausgeführt werden und stellt die Erweiterung vor, die in dieser Arbeit entwickelt wurde um Prozessmodelle (vgl. Kapitel 5.2) zu instanziieren. Der Server enthält Prozess-Templates, CPFs und Szenarios. Die Prozess-Templates wurden bisher in AristaFlow modelliert und in den Server importiert. Prozess-Templates stellen die Basis-Prozessmodelle der CPFs dar. Zur Ausführung der Prozessinstanz auf dem Prozessmanagementsystem AristaFlow wird ein Szenario, ausgehend von einem CPF-Objekt erstellt.

Dieses Prinzip kann einfach erweitert werden, indem aus dem Prozessmodell, das in Kapitel 5.2 erläutert wurde, ein neues Prozess-Template generiert und anschließend ein CPF-Objekt erstellt wird. Dieses CPF-Objekt verwendet das generierte Prozess-Template als Basis-Prozessmodell und beinhaltet keine Kontextinformationen. Ein Szenario, das auf dieses Objekt aufbaut, führt das in der Client-Anwendung modellierte Prozessmodell aus.

Dieses Vorgehen wurde in der REST-Schnittstelle implementiert. Dabei wird das Prozess-Template bei jeder Instanziierung des Prozessmodells mit der Methode *updateBaseProcess(prm, parameterMapping)* am PrMS aktualisiert, da das Prozessmodell möglicherweise bearbeitet wurde. Weder die CPF- noch die Szenario-Struktur werden durch diese Erweiterung verändert. Dadurch bleibt die Flexibilität, die CPFs durch Erweiterungsgebiete, Kontext-Situationen, Prozess-Fragmente und Injektions-Spezifikationen bereitstellen, erhalten und kann auch mit den hier entwickelten Prozessmodellen verwendet werden.

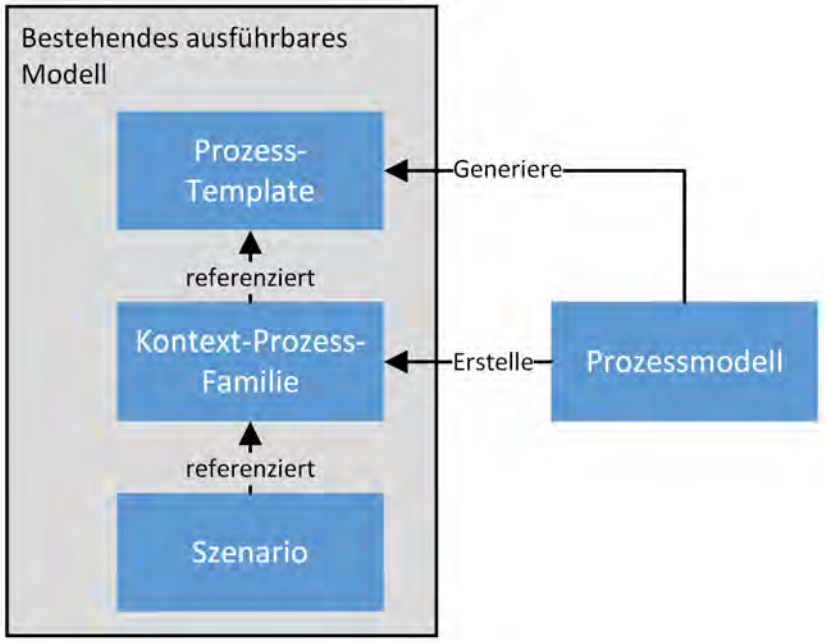


Abbildung 5.9.: Erweiterung des Server Ausführungsmodells

Eine der Herausforderung zur Ausführung von Instanzen von Verfahrensprozessen, ist also das Erstellen der Prozess-Templates aus den Prozessmodellen. AristaFlow benötigt dazu ein XML-Dokument, das aus den Elementen *Knoten* (nodes), *Datenelemente* (dataElements), *Kanten* (edges), *Datenkanten* (dataedges) und *strukturierte Daten* (structuralData) besteht. Der Aufbau der Elemente ist in Listing 5.4 zu sehen. Im Bereich *nodes* befinden sich alle Knoten-Elemente, wie beispielsweise Ereignisse und Aktivitäten und im Bereich *dataElements* befinden sich alle Daten-Elemente. In den Bereichen *edges* und *dataEdges* wird definiert, wie die Knoten- bzw. die Daten-Elemente untereinander verknüpft sind. Zuletzt ist im Bereich *structuralData* die Struktur der Knoten-Elemente definiert. Das Prozess-Template wird durch ein XML-Schema validiert und bei einer Abweichung vom vorgegebenen Schema kann das Prozessmodell nicht ausgeführt werden. Zu diesem Zweck implementiert der Proof-of-Concept Prototyp die neue Klasse *TemplateFactory*. Diese nimmt ein Prozessmodell-Objekt entgegen und erstellt daraus ein passendes Prozess-Template als XML-Dokument. Da es sich hierbei

um eine sehr komplexe Aufgabe handelt, fokussiert sich der Prototyp zunächst auf die Subprozesse des Prozessmodells.

Listing 5.4: Aufbau eines Prozess-Templates

```
<template ...>
  <nodes>
    ...
  </nodes>
  <dataElements>
    ...
  </dataElements>
  <edges>
    ...
  </edges>
  <dataEdges>
    ...
  </dataEdges>
  <startNode>n0</startNode>
  <endNode>n1</endNode>
  <structuralData>
    ...
  </structuralData>
</template>
```

Bei der Generierung des Prozessmodells muss weiterhin berücksichtigt werden, dass einige Anpassungen notwendig sind, um das Modell ausführbar zu machen. Wie in Abbildung 5.10 zu sehen ist, wird zu Beginn des Prozess-Templates eine zusätzliche Aktivität benötigt, die die Werte der Auftragsparameter in AristaFlow-Datenobjekte schreibt. Diese können ohne die Interaktion mit einem Anwender als Eingabe für andere Aktivitäten verwendet werden. Außerdem werden Subprozesse in zwei Aktivitäten aufgeteilt. Die erste Aktivität überträgt den Inhalt des Subprozesses (z.B. den Assistenten „Blisterverpackung“ aus Kapitel 3.1) an die Kommunikations-Anwendung, während die zweite Aktivität

auf eine Antwort der Kommunikations-Anwendung wartet, um die Prozessinstanz fortzuführen. Diese Antwort soll von der Kommunikations-Anwendung gesendet werden, sobald der Subprozess abgearbeitet wurde (z.B. sobald SmartControl meldet, dass der Assistent abgeschlossen wurde). Da der Prototyp die Kommunikations-Anwendung nicht vollständig implementiert, werden diese Aktivitäten allerdings dadurch substituiert, dass der Assistent in der Ausführungsumgebung angezeigt wird und dort durch den Anwender manuell bestätigt wird.

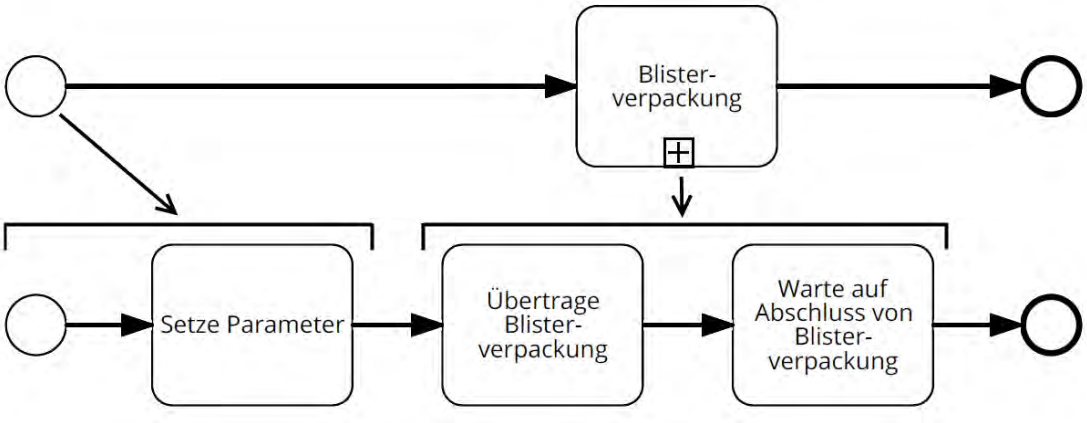


Abbildung 5.10.: Umwandeln eines Prozessmodells in ein Prozess-Template für AristaFlow

## 5.5. Kommunikations-Anwendung

Im entwickelten Konzept ist die Kommunikations-Anwendung als Komponente zwischen der Ausführungssteuerung und den am Verfahrensprozess beteiligten Maschinen beschrieben (vgl. Kapitel 4.4). Sie besteht aus einer Sammlung von Wrappern, die die maschinennahen Subprozesse bei der Ausführung an die Maschinen übertragen und auf deren Abarbeitung warten. Im Rahmen dieser Arbeit wurde die Schnittstelle zwischen dem Maschinenbediensystem SmartControl und dessen Wrapper definiert und ein Ansatz vorgestellt, der die Korrektheit der Schnittstelle aufzeigt.

### 5.5.1. SmartControl Schnittstelle

Um im Allgemeinen maschinennahe Subprozesse in SmartControl als Assistent ausführen zu können, muss zunächst ein passendes Format festgelegt werden, in dem diese übertragen werden. SmartControl verwendet zum Einlesen seiner Assistenten ein XML-Format, das beim Erstellen des SmartControl-Plugins berücksichtigt wurde. Um das benötigte Format zu verdeutlichen, zeigt Listing 5.5 den korrespondierenden Assistenten zu dem Szenario aus Kapitel 3.1 bzw. der Abbildung 5.6. Der Wurzelknoten des XML-Dokuments ist durch das XML-Tag „<assistant>“ gekennzeichnet und erfordert die drei Attribute *module*, *name* und *key*. Hierbei ist zu beachten, dass für den Proof-of-Concept-Ansatz das Attribut „key“ den Text „installation“ als Inhalt benötigt, da die derzeitige Schnittstelle von SmartControl dies erwartet. Die Schritte, wie beispielsweise *Formatwechsel*, werden im Assistenten mit dem XML-Tag *„step“* und die Prozedur-Schritte als Widgets vom Typ *SequenceStepContainer* dargestellt. Die Prozedur-Schritte enthalten *Label*-Widgets und *Treenode*-Widgets. Die Erstellung des XML-Assistenten erfolgt auf Seiten des Clients. Sie ist in das SmartControl-Plugin integriert und wird bei der Übertragung des Prozessmodells an den Server mitgesendet. Derzeit gibt es noch kein XML-Schema von Uhlmann, um das Format des Assistenten zu validieren.

Um ein neues Maschinenfeld als Steuerelement in den Assistenten einzufügen, wie es in Kapitel 4.5 beschrieben ist, wird eine Ergänzung des XML-Formats benötigt. Jedes neue Maschinenfeld erhält einen Namen und muss zu Beginn in dem Element *properties* mit entsprechendem Typ definiert sein. Im Beispiel von Listing 5.5 ist dies *leafletMagazine* (Zeile 7). Hierbei wird die Pfadangabe *„*.assistantsSwDevice.mainAssistant.installation“* bei der Initialisierung des Subprozesses an den Namen angefügt (Zeile 16), da SmartControl diesen Pfad benötigt, um das neue Feld in den Maschinenbaum einzufügen. Bei dem neuen Maschinenfeld handelt es sich um ein Textfeld, weshalb es unter *properties* als Element vom Typ *configStringVal* angegeben ist. Die Typen, die der Proof-of-Concept Prototyp dabei unterstützt, sind: *configStringVal*, zur Darstellung von Texten, *configFloatVal*, zur Darstellung von Zahlen, und*configListVal*, zur Darstellung von Auflistungen.

Listing 5.5: XML Schnittstelle zu SmartControl

```
<?xml version="1.0" encoding="UTF-8"?>
<assistant module='1'
    name='Blisterverpackung' key='installation' >
  <properties>
    <configStringVal name='leafletMagazine'/>
  </properties>
  <step text='Formatwechsel' key='Formatwechsel'>
  ...
  <widget type='SequenceStepContainer'>
    <widget type='label'>
       <text key=''>Prospekte bereitstellen</text>
    </widget>
    <widget type='treenode' label='true' >
      <url>*.assistantsSwDevice.mainAssistant.
        installation.leafletMagazine</url>
    </widget>
  </widget>
  ...
  </step>
  <step text="Produktion" key="Produktion">
    ...
  </step>
  ...
</assistant>
```

### 5.5.2. Proof-of-Concept-Ansatz

Das Konzept aus Kapitel 4.4 sieht an dieser Stelle vor, den Inhalt des XML-Dokument aus Listing 5.5 über eine REST-Schnittstelle an die Kommunikationskomponente zu senden, damit der Assistent dort über einen Wrapper in SmartControl eingelesen und

bearbeitet werden kann. Für den Proof-of-Concept-Ansatz wird der Assistent jedoch lediglich in der Ausführungsumgebung angezeigt und muss manuell in SmartControl eingefügt werden.

# 6

# Diskussion

Dieses Kapitel diskutiert die Ergebnisse der Masterarbeit und eruiert zunächst zu welchem Grad die Anforderungen aus Kapitel 3.3 realisiert und welche Konzepte aus Kapitel 4 im Prototypen implementiert wurden. Weiterhin werden verwandte Arbeiten diskutiert und ihr Zusammenhang mit dieser Arbeit aufgearbeitet.

## 6.1. Proof-of-Concept

Nachdem der Proof-of-Concept Prototyp entwickelt wurde gilt festzustellen, inwiefern er den in Kapitel 3 beschriebenen Anforderungen und dem in Kapitel 4 aufgestellten Konzept entspricht. Dabei werden zunächst die Anforderungen verglichen und diskutiert ob diese zur Genüge erfüllt wurden. Anschließend wird anhand eines Beispiels beschrieben, wie die Schnittstelle zu SmartControl funktioniert.

### 6.1.1. Anforderungsvergleich

Nachfolgend werden die in Kapitel 3.3 tabellarisch aufgelisteten funktionalen und nichtfunktionalen Anforderungen mit dem entwickelten Konzept und implementierten Prototypen verglichen. Sofern eine Anforderung nicht bzw. teilweise umgesetzt ist, wird zudem erläutert, weshalb dies der Fall ist.

#### Funktionale Anforderungen

Die funktionalen Anforderungen aus Tabelle 3.8 nennen zunächst das Erstellen und Speichern von Prozessmodellen, Subprozessen und Aktivitäten. Das Erstellen dieser

drei Elemente ist durch den Modellierungseditor sowohl im Konzept (vgl. Kapitel 4.2), als auch im Prototypen (vgl. Kapitel 5.3.2) gewährleistet. Der Editoranwender legt im Modellierungseditor zunächst, wie gefordert, den Kontrollfluss des Prozessmodells an und modelliert anschließend die Subprozesse im SmartControl-Plugin detailliert. Die Speicherfunktion persistiert das Prozessmodell einschließlich der modellierten Subprozessen und Aktivitäten in der Datenbank, wie es in Kapitel 5.4 beschrieben ist. Hierbei wird das Prozessmodell durch den Einsatz der relationalen H2-Datenbank versioniert. Die Anforderung „Prozessmodell freigeben“ hingegen wurde zwar im Konzept bei der Darstellung von Prozessmodellen berücksichtigt, jedoch im Proof-of-Concept Prototyp nicht umgesetzt. Hierzu ist die Integration der Benutzerverwaltung von SmartControl notwendig, um die Berechtigungen eines Anwenders zu überprüfen. Dies stand nicht im Hauptfokus der Masterarbeit und wurde deshalb bei der Entwicklung des Prototypen zurückgestellt.

Weitere funktionale Anforderungen an die Modellierungsumgebung stellen das Referenzieren und Einfügen von Maschinenparametern, Auftragsparametern und Steuerelementen dar. Die Verwaltung von Auftragsparametern wurde konzipiert, umgesetzt und in Kapitel 5.3.2 beschrieben. Das Anlegen und Verwalten der Auftragsparameter geschieht dabei durch eine Tabelle, in der die Auftragsparameter mit einem Typen der SmartControl Typ-Bibliothek angelegt werden. Im Prozessmodell können Auftragsparameter über einen textbasierten Ansatz eingefügt werden. Das Einfügen von Maschinenparameter und anderen Steuerelemente wird durch ein Drag&Drop-Verfahren umgesetzt. Maschinenparameter können innerhalb des SmartControl-Plugins als existierende und neue Maschinenfelder referenziert werden, wie es in Kapitel 4.5 beschrieben ist. Durch die graphische Modellierung von Prozessmodellen und die Orientierung am SmartControl-Assistenten, bei der Modellierung von Subprozessen, wird dem Editoranwender bereits eine Visualisierung des Prozessmodells gegeben. Dies erfüllt daher die Anforderung „Vorschau zu Prozessmodell anzeigen“ (vgl. Kapitel 3.3).

Tabelle 3.9 definiert die funktionalen Anforderungen an die Ausführungsumgebung. Das Konzept beschreibt in Kapitel 4.3 das Instanziieren von Prozessmodellen und die korrekte Abarbeitung von Prozessinstanzen, wobei die Auftragsparameter, wie gefordert, beim Instanziieren mit einbezogen werden. Im Konzept wird die Ausführung einer Prozess-

instanz durch den Einsatz eines PrMS gelöst. In der Implementierung handelt es sich bei diesem PrMS um AristaFlow (vgl. Kapitel 5.4.3), das die vollständige und angemessene Ausführung der Prozessinstanzen gewährleistet. Das Bearbeiten von maschinennahen Aktivitäten des Prozessmodells, d.h. das Senden dieser Aktivitäten an die richtige Maschine und die dortige Ausführung, wird durch die Kommunikations-Anwendung gelöst (siehe Kapitel 4.4). Da der Prototyp hier allerdings nur ein Proof-of-Concept-Ansatz beinhaltet, wie in Kapitel 5.5 erläutert ist, wurde diese Anforderung damit teilweise umgesetzt. Um jedoch zu zeigen, dass die Modellierung der SmartControl-Assistenten korrekt umgesetzt wurde, ist in Kapitel 6.1.2 die Darstellung des Beispiel-Assistenten aus Kapitel 5.5 beschrieben. Die nächste Anforderung - das Signieren einer Aktivität - ist indirekt durch die Benutzerverwaltung von SmartControl erfüllt, da diese den Operator eines Assistenten identifiziert. Die Benutzerkennung wird durch die Kommunikations-Anwendung an das Prozessmanagementsystem zurückgegeben, wenn ein Signatur-Steuerelement im SmartControl-Plugin eingefügt wurde. Das PrMS speichert alle Daten einer Aktivität und somit auch die Identifikation des aktuellen Operators, der diese ausgeführt hat. Zuletzt bietet die in Kapitel 5.1 beschriebene Ausgangsbasis der Arbeit in der Ausführungsumgebung ein Verfahren, um Prozessinstanzen jederzeit zu pausieren und zu einem beliebigen Zeitpunkt wieder aufzunehmen. Dies erfüllt die Letzte der funktionalen Anforderungen (*Prozessinstanz unterbrechen*).

**Nicht-funktionale Anforderungen**

Tabelle 3.10 enthält die nicht-funktionalen Anforderungen an das System zur Modellierung und Ausführung von Verfahrensprozessen. Die Anforderungen sind dabei in die Bereiche Benutzbarkeit, Entwicklungsspezifisch und Sicherheit gegliedert.

Um die *Bedienbarkeit* und *Erlernbarkeit* des entwickelten Systems für den Anwender zu optimieren, wurden die Möglichkeiten, die dieses ihm bietet, zunächst gering gehalten und beziehen sich auf die nötigsten Funktionen. Außerdem wird ihm die Bedienbarkeit durch eine einheitliche Navigation und den strukturierten Aufbau der Modellierung, die das Modellieren des Gesamtprozessmodells und das der Subprozesse aufteilt, erleichtert. Damit der Editoranwender den Bezug zu SmartControl nicht verliert, wurden

im Konzept Richtlinien zur Usability (vgl. Kapitel 4.6) des Prototypen festgelegt, die sich dabei an SmartControl orientieren. Beispielsweise wird für die Gestaltung die SmartControl Farbtablette verwendet. Die Anforderung der Internationalisierung ist durch den Einsatz einer deutschen und einer englischen Sprach-Datei gelöst. Diese enthalten jeweils alle verwendeten Texte des Systems, damit diese je nach Sprachwahl angezeigt werden können. Das entwickelte System kann dadurch jederzeit um weitere Sprachen erweitert werden.

Die entwicklungsspezifischen Anforderungen sollen eine optimale Integration des Modellierungseditors in SmartControl sicherstellen. Obwohl die Client-Anwendung nicht, wie SmartControl, in Python programmiert ist und auf einem anderen Applikationsserver ausgeführt wird, können die beiden Web-Anwendungen durch die Verwendung von Hyperlinks in einem Webbrowser integriert werden. Sollen in zukünftigen Projekten Komponenten aus SmartControl, wie beispielsweise der Maschinenbaum, im Modellierungseditor verwendet werden, kann dies über Web Services realisiert werden. Eine weitere Anforderung der Integration verlangt, dass die Server-Anwendung während der Ausführung einer Prozessinstanz die Kommunikation mit SmartControl ermöglichen muss. Dies wurde bei der Konzeption der Kommunikations-Anwendung aus Kapitel 4.4 berücksichtigt, zudem ist eine passende XML-Struktur der Schnittstelle in Kapitel 5.5 definiert. Zur Integration der Maschinen in den Modellierungseditor werden Plugins für jedes Maschinenbediensystem verwendet. Die Beschaffenheit dieser Plugins wurde im Konzept (vgl. Kapitel 4.5) erläutert und im Prototypen beispielhaft für SmartControl (vgl. Kapitel 5.3.2) umgesetzt.

In Hinsicht auf den Punkt Sicherheit, stellt die Server-Anwendung eine sichere Datenhaltung bereit. Prozessmodelle können ausschließlich durch die implementierten Funktionen der Modellierungsumgebung verändert werden. Die Vertraulichkeit der Daten ist aber nur dann gewährleistet, solange der Server, auf dem das entwickelte System ausgeführt wird, gesichert ist und nur berechtigte Anwender zulässt. Was die Korrektheit der Prozessinstanzen betrifft, so lässt das PrMS AristaFlow die Instanziierung von Prozessmodellen nur dann zu, wenn diese keine Fehler beinhalten. Die inhaltliche Richtigkeit eines Prozessmodells vor der Instanziierung muss allerdings von den Pharmazeuten überprüft und bestätigt werden.

### 6.1.2. Schnittstelle zu Smart Control

Um aufzuzeigen, dass das modellierte Prozessmodell des Prototypen auf SmartControl ausführbar ist, hat Uhlmann im ersten Schritt des dynamischen Ladens von Assistenten eine vereinfachte Schnittstelle in SmartControl geschaffen. Diese Schnittstelle ermöglicht es, das XML-Dokument eines Assistenten zur Laufzeit manuell einzufügen und den Ablauf dieses Assistenten zu simulieren.

Abbildung 6.1 zeigt in den markierten Bereichen die Darstellung des XML-Dokuments aus Listing 5.5 in SmartControl. Der Prototyp wird auf keiner realen, sondern auf einer virtuellen Maschine ausgeführt, weshalb im oberen Bereich von SmartControl keine Maschine anzeigt wird.

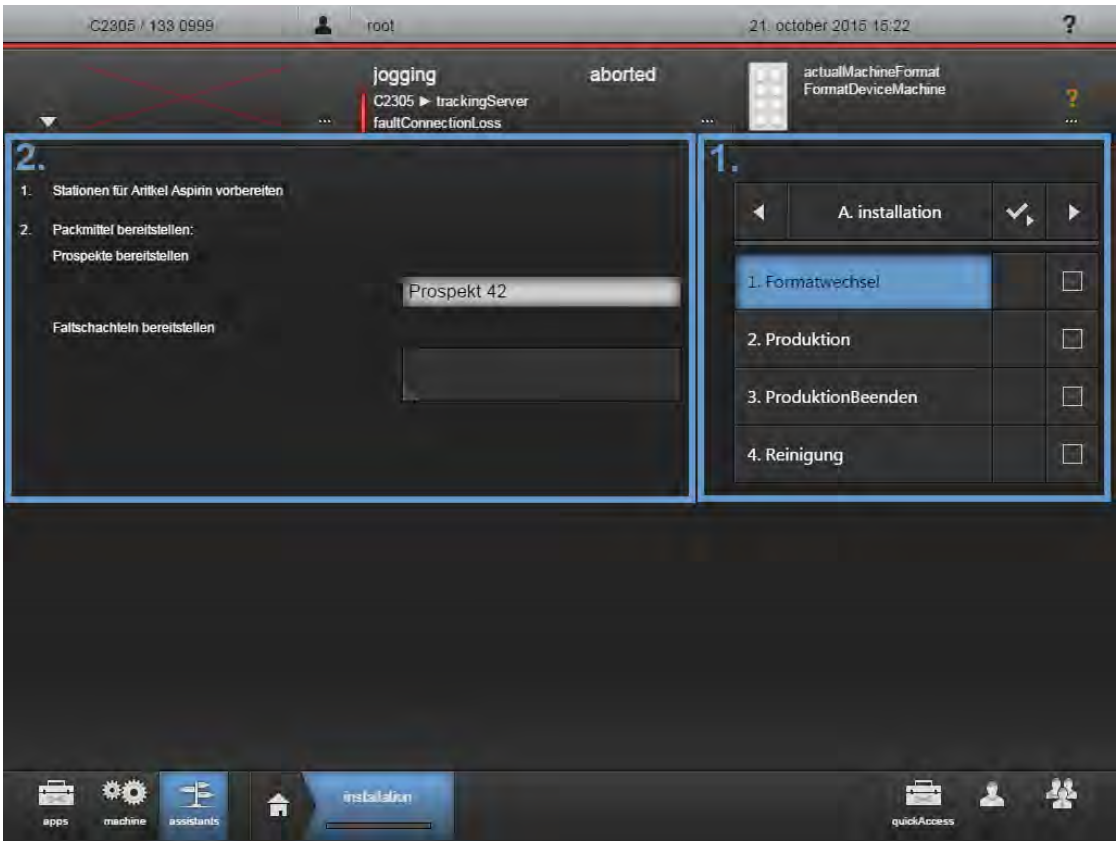


Abbildung 6.1.: SmartControl - Visualisierung der XML-Schnittstelle

Um den Assistenten in SmartControl einzubauen, muss, wie in der Schnittstellenbeschreibung (Vgl. Kapitel 5.5) erklärt, das Attribut „key“ den Text „Installation“ enthalten. Dieser ist über der Schrittanzeige (1) und unten in der Navigationsleiste zu sehen. Die im Prozessmodell definierten Schritte finden sich auf der rechten Seite (1) und können dort nacheinander abgearbeitet werden. In Bereich (2) von Abbildung 6.1 befinden sich die Arbeitsanweisungen, die abhängig von ihrem Widget-Typ dargestellt werden. Widgets vom Typ *SequenceStepContainer* werden mit einer Nummerierung aufgelistet und enthalten eingerückt weitere Widgets. Widgets vom Typ *Label* zeigen ihren Inhalt als Text an. Die Anzeige eines *Treenode*-Widgets hängt vom Typ des Maschinenfeldes ab. Bei einem neuen Maschinenfeld wird der Datentyp (Text, Zahl oder Liste) im *properties*-Element angegeben und es wird eine entsprechende Eingabemaske angezeigt. Die Anzeige eines existierenden Maschinenfelds ist durch den Maschinenbaum der jeweiligen Maschine definiert. In Abbildung 6.1 wird das Maschinenfeld *leafletMagazine* aus Listing 5.5 durch ein Textfeld dargestellt, das den Typ der Faltschachteln (in diesem Beispiel *„Prospekt 42“*) als Eingabe erwartet. Das andere Maschinenfeld verweist hingegen mit seinem Pfad auf einen Schalter der Blistermaschine, der die Faltschachteln einzieht.

Abbildung 6.2 zeigt die Schrittanzeige, nachdem der erste Schritt abgearbeitet wurde und der zweite („Produktion beginnen“) aktiv ist. Der Abschluss eines Schrittes wird durch einen Hacken und eine Fortschrittsleiste dargestellt.

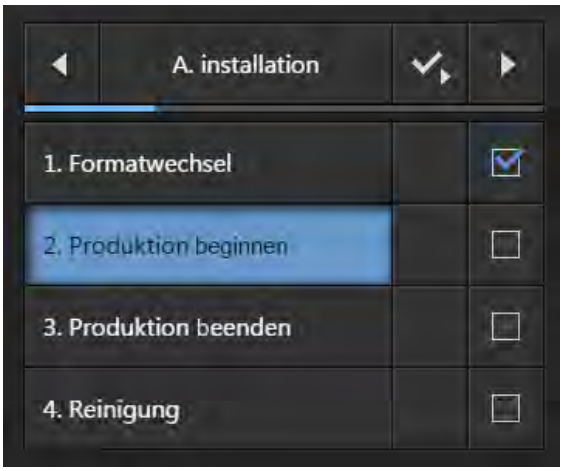


Abbildung 6.2.: SmartControl - Visualisierung der XML-Schnittstelle

## 6.2. Related Work

Um die Ergebnisse dieser Arbeit zu diskutieren werden im folgenden wissenschaftliche Arbeiten betrachtet, die sich mit Themen befassen, die für diese Arbeit relevant sind. Zunächst wird das Themengebiet MES untersucht, da dieses bereits in Kapitel 2.2 genannt wurde und Teil der Grundlagen dieser Arbeit ist. Daraufhin werden die Prozessmodellierung und Prozesskonfiguration betrachtet, da diese, im Zusammenhang mit dem Modellierungseditor, von Bedeutung sind und Erweiterungspotential für den Modellierungseditor beinhalten.

### 6.2.1. Manufacturing Execution Systems

In [BDC08] wird ein MES als Schnittstelle zwischen Geschäftslogik und Produktion definiert, zwischen den physikalischen Subsystemen (Maschinen) und den hochrangigen Entscheidungssystemen (z.B. ERP). Das MES kontrolliert den Produktionsprozess, indem es direkt die Steuerung der physikalischen Subsysteme übernimmt. Deshalb sollte die Entwicklung bzw. Parametrisierung des MES individuell auf diese angepasst sein. Zudem stellt [BDC08] ein MES vor, das eine flexible Reaktionsfähigkeit und Wirtschaftlichkeit erzielen soll, indem es die Konzepte von *Holonic Manufacturing Systems* (HMS) nutzt. Darunter sind Produktionssysteme zu verstehen, die unter Verwendung intelligenter Systeme (z.B. Multi-Agenten Systeme) das Verhalten des Produktionsprozesses vorhersagen können, um beispielsweise rechtzeitig auf mögliche Fehler zu reagieren [BG08]. Um dieses Ziel zu erreichen werden Flexibilität, Rekonfiguration und Evolution des Produktionsprozesses als wichtige Charakteristiken eines MES genannt.

[SSM06] führt bei der Definition eines MES an, dass es dessen Aufgabe ist, Managementsysteme und Systeme auf Maschinenebene zu integrieren, um das Konzept „intelligenter Produkte“ (engl: smart-products) umzusetzen. Unter intelligenten Produkten ist ein Objekt der Produktion zu verstehen, das seinen Zustand und seinen Zweck kennt. MES nehmen daher eine zentrale Rolle im Thema Industrie 4.0 ein. Es ist ihre Aufgabe eine *Smart Factory* durch die Integration aller Management- und Maschinensysteme in einem komplexen Produktionsprozess zu ermöglichen.

Um einen technischen Vergleich des entwickelten Prototypen zu einem MES anzustellen, wurde [MFT09] herangezogen, das die typische Architektur eines MES beschreibt. Hierzu existiert ein Server-basiertes System mit einer relationalen Datenbank und einer webbasierten Benutzerschnittstelle, die auf einem Applikationsserver ablaufen. Vergleicht man diese Architektur mit dem Architekturkonzept aus Kapitel 4.1, lassen sich entsprechend Gemeinsamkeiten feststellen. Zum Beispiel die Aufteilung der Server-Anwendung in die verschiedene Komponenten (Prozessmanagementsystem, Kommunikations-Anwendung) und eine webbasierten Client-Anwendung mit der über eine serviceorientierte Schnittstelle kommuniziert wird. [MFT09] beschreibt dabei unter anderem detailliert das Datenmanagement innerhalb eines MES, die Schnittstellen zu anderen Systemen, wie beispielsweise ERP, und den Aufbau eines User Interfaces.

### 6.2.2. Prozessmodellierung

Ein Hautaugenmerk dieser Arbeit ist es, Anwender ohne Kenntnisse von technischen Prozessmodellen einen solchen intuitiv modellieren zu lassen. Es ist daher wichtig, im Modellierungseditor eine fachliche Prozesssicht bereitzustellen, die alle erforderlichen Modellierungsoptionen liefert und trotzdem einfach zu verstehen und bedienen ist. Diese Herausforderung wurde im Konzept dadurch gelöst, dass der Editoranwender bei der Modellierung des Gesamtprozesses diesen zunächst in maschinennahe Subprozesse aufteilt. Vorgesehen ist dabei die Aufteilung anhand der jeweils eingesetzten Maschinen vorzunehmen. Dem Editoranwender ist es aber auch gestattet, große Aufgaben an einer Maschine in Form von mehreren Subprozessen darzustellen.

Sobald der Editoranwender den Gesamtprozess erfolgreich in sinnvolle Subprozesse aufgeteilt hat, stellt sich der Anspruch, ihm die Modellierung der Aktivitäten zu ermöglichen. Das dazu gewählte Konzept lehnt sich an die Assistentenumgebung aus SmartControl an. In der Literatur lassen sich ähnliche Ansätze finden. In [Kol15] beschreibt der Autor einen formular-basierten Ansatz, der ein Prozessmodell aus fachlicher Sicht kapselt. Dabei werden Aktivitäten als Rechtecke mit einem zentrierten Titel dargestellt. Die Rechtecke sind vertikal angeordnet, was die Reihenfolge der Abarbeitung der Aktivitäten definiert. Außerdem benötigt diese Darstellung kein Start- und Ender-

eignis. Es beginnt mit der obersten Aktivität und endet mit der untersten. Während die entwickelte Modellierungsumgebung lediglich diese Art der Abarbeitung verwendet (Vgl. Kapitel 5.3.2), führt [Kol15] den Ansatz weiter und ermöglicht die Verwendung von UND-Verzweigungen, ODER-Verzweigungen, Schleifen und Datenobjekte. Eine UND-Verzweigung wird durch die horizontale Anordnung zweier Rechtecke dargestellt. Dabei haben alle Rechtecke, die parallel ablaufen, zusammen dieselbe Breite wie das Rechteck oberhalb der UND-Verzweigung. Eine ODER-Verzweigung besteht ebenfalls aus nebeneinander angeordneten Rechtecken, allerdings sind diese in der Mitte durch eine Entscheidungsabfrage (z.B. Artikelnummer = "123456"?) verbunden und bieten je eine Antwortmöglichkeit an. Eine Schleife ist durch einen Pfeil gekennzeichnet, der die Aktivitäten, die wiederholt werden sollen, umschließt. Datenobjekte werden in eine Aktivität als Eingabe- und Ausgabeparameter geschrieben und können in der Ansicht eingeklappt werden. Zur Modellierung eines formular-basierten Prozessmodells beschreibt [Kol15] eine Drag&Drop Methodik, die der des Prototypen dieser Arbeit ähnelt, um Steuerelemente in das Formular des SmartControl Plugins einzufügen. Es wird darauf hingewiesen, dass der Ansatz der formular-basierten Modellierung einfach zu verstehen ist und Anwendern besonders die Änderung von Prozessmodellen erleichtert [Kol15].

Die Verwendung von fachlichen Prozesssichten, um komplexe technische Prozesse ungeschulten Personen darzustellen, findet sich in der automatischen Generierung der technischen Prozess-Templates (Vgl. Kapitel 5.4.3) wieder. Während der Editoranwender den fachlichen Subprozess „Blisterverpackung“ angibt, besteht dieser in der Prozessinstanz aus „Übertrage Blisterverpackung“ und „Warte auf Abschluss von Blisterverpackung“. Außerdem beinhaltet die Prozessinstanz eine Aktivität zum Setzen der Parameter, die dem Editoranwender nicht angezeigt wird. [KR13] stellt einen Ansatz vor, um große, unübersichtliche Prozessmodelle durch das Erstellen von Prozesssichten zu vereinfachen. Dies wird dadurch erreicht, dass für den Anwender irrelevante Teile des Prozessmodells in der Prozesssicht verborgen werden und mehrere detaillierte Aktivitäten in einer abstrakten Aktivität aggregiert werden. Die Prozesssichten sind dabei personalisiert, d.h. die Entscheidung welche Teile verborgen, bzw. aggregiert werden, hängt davon ab, welche Rolle dem Benutzer zugewiesen ist.

### 6.2.3. Prozesskonfiguration

Die Wiederverwendung von Prozessmodellen für unterschiedliche Aufträge kommt in der Praxis häufig vor, da sich z.B. der Produktionsprozess zweier verschiedener Medikamentensorten in dem Auslassen bestimmter Produktionsschritte unterscheidet. Um die zeitaufwändige separate Modellierung ähnlicher Prozesse zu vermeiden, spielt die Prozesskonfiguration an dieser Stelle eine wichtige Rolle. Damit ist gemeint, dass aufgrund eines gemeinsamen Basis Prozessmodells mehrere Varianten existieren können [RW12].

[HBR10] stellt anhand des Provop Frameworks zur Modellierung und Verwaltung großer Geschäftsprozesse einen Ansatz vor, bei dem zuerst ein Basis Prozessmodell ausgewählt wird, aus dem durch die Anwendung der Operationen Einfügen (INSERT), Löschen (DELETE) und Bewegen (MOVE) verschiedene Prozessvarianten erstellt werden können. Bei der Auswahl des Basis Prozessmodells können dabei fünf verschiedene Methoden angewandt werden. So wird, beispielsweise, die am häufigsten verwendete Prozessvariante zum Basis Prozessmodell erklärt oder dieses wird so erstellt, dass die Anzahl der Änderungen aller Varianten minimal wird. Um Prozess-Fragmente durch die drei genannten Operationen zu bearbeiten, werden sogenannte „adjustment points“ in das Basis Prozessmodell eingefügt, die Bereiche kennzeichnen, in denen Änderungen vorgenommen werden können.

Die Prozesskonfiguration wurde in dieser Arbeit so umgesetzt, dass ein Prozessmodell (z.B. für die Tablettenproduktion) erstellt wird und die Konfiguration bei der Instanziierung durch Auftragsparameter erfolgt. Der Editoranwender kann allerdings von den Konzepten aus [HBR10] profitieren, wenn diese so genutzt werden, dass er zu Beginn der Prozessmodellierung als Ausgangspunkt ein Basis Prozessmodell erhält, das er lediglich an den konkreten Fall anpasst. Außerdem kann dieser Ansatz von Unternehmen zur Qualitätssicherung verwendet werden, indem dem Editoranwender ein Basis Prozessmodell vorgegeben wird, das Aktivitäten beinhaltet, die nicht entfernt werden können und zwischen denen der Editoranwender spezifischen Aktivitäten einfügt.

# 7

# Zusammenfassung und Ausblick

In diesem Kapitel werden das entworfene Konzept und der implementierte Prototyp zusammengefasst. Zudem wird ein detaillierter Ausblick darüber gegeben, welche Prototyp-Erweiterungen notwendig sind, um das Konzept vollständig zu implementieren und welche Erweiterungspotentiale das entwickelte Konzept bietet.

## 7.1. Zusammenfassung

Das Ziel dieser Arbeit war es, ein System zur Modellierung und Ausführung von Verfahrensprozessen zu konzipieren und einen entsprechenden Proof-of-Concept Prototypen zu entwickeln. Die Modellierung der Verfahrensprozesse soll dabei die Produktion über die gesamte Linienebene beinhalten und Produktionsmaschinen integrieren können. Hierbei wurden zunächst *Smart Factories* und *Manufacturing Execution Systeme* betrachtet, da diese im Angesicht von *Industrie 4.0* eine bedeutsame Rolle innehaben und Ansätze bieten wie Verfahrensprozesse aussehen sollen. Für die Ausführung von Verfahrensprozessen ist ein Prozessmanagementsystem notwendig, weshalb das Thema des Prozessmanagements für diese Arbeit wichtig war und in in Kapitel 2.1 behandelt wurde. Außerdem wurde das Maschinenbediensystem SmartControl betrachtet, um einen Einblick in die Praxis zu erhalten und die Anforderungen, die eine Produktionsmaschine an die Modellierung und Ausführung von Verfahrensprozessen stellt, zu erfassen (vgl. Kapitel 2.3).

Vor der Konzeption des Systems zur Modellierung und Ausführung von Verfahrensprozessen wurden Anforderungen erhoben. Hierbei wurden zunächst ein Szenario in Absprache mit und in Anlehnung an die Prozessabläufe der Firma Uhlmann aufgestellt (vgl. Kapitel

3.1), um die realitätsnahen Anforderungen der Praxis erheben zu können. Aufgrund des Szenarios wurden User Stories angelegt, die die Anforderungen aus Benutzersicht darstellen (vgl. Kapitel 3.2). Auf die Benutzersicht folgte die technische Betrachtung des zu entwickelnden Systems, in Form von funktionalen und nicht-funktionalen Anforderungen (vgl. Kapitel 3.3). Diese Aufgaben erforderten viele Besprechungen mit Fachpersonal der Firma Uhlmann und einen großen Arbeitsaufwand, da die Konzeption und Entwicklung eines Systems zur Modellierung und Ausführung von Verfahrensprozessen viele Möglichkeiten bietet und abgegrenzt werden musste, was im Rahmen dieser Arbeit berücksichtigt werden sollte.

Die Entwicklung des Konzeptes begann mit der Ermittlung aller notwendigen Komponenten: Einem Prozessmanagementsystem, einer Client- und Server-Anwendung und einer Kommunikations-Anwendung, die über Web Services mit dem PrMS und über Wrapper mit den Maschinen bzw. SmartControl kommuniziert (vgl. Kapitel 4.1). Das Konzept befasst sich intensiv mit den Bestandteilen der Client-Anwendung (vgl. Kapitel 4.2), da diese zur Modellierung von Prozessmodellen und den Arbeitsabläufen der Maschinen im Vordergrund der Arbeit steht. Durch Mockups werden die Funktionalitäten des Modellierungseditors genau festgelegt. Ein wichtiger Aspekt des Konzeptes ist außerdem die Anbindung von SmartControl und damit im Allgemeinen von Maschinen an das System. Hierzu wurde die Art der Anbindung durch die generische Kommunikations-Anwendung und eine Schnittstellenbeschreibung definiert (vgl. Kapitel 4.4 und 4.5).

Die Systemimplementierung wurde anhand eines Proof-of-Concept Prototypen vorgenommen, der auf einem bestehenden Projekt der Universität Ulm beruht und aus einer Client- und einer Server-Anwendung besteht (vgl. Kapitel 5.1). Der Client wurde als Web-Anwendung in den Technologien HTML, CSS und CoffeeScript entwickelt und implementiert ein Menü zur Auswahl, Anlage und Instanziierung von Prozessmodellen, einem Modellierungseditor zur Bearbeitung von Prozessmodellen inklusive der Arbeitsabläufe der Maschinen und einer Ausführungsüberwachung, die aus dem bestehenden Projekt übernommen wurde (vgl. Kapitel 5.3). Die Server-Anwendung dient zur Persisitierung und Ausführung der Prozesse (vgl. Kapitel 5.4). Zur Kommunikation mit der Client-Anwendung stellt der Server eine REST-Schnittstelle zur Verfügung, über die Prozessmodelle gespeichert, abgerufen und instanziiert werden und die Ausführung

von Prozessinstanzen gesteuert werden kann. Für die Speicherung der Prozessmodelle integriert die Server-Anwendung eine Datenbank und ein geeignetes Datenmodell und für die Ausführung das Prozessmanagementsystem AristaFlow. Der Prototyp definiert außerdem das Schnittstelle-Format zu SmartControl (vgl. Kapitel 5.5).

Im Anschluss an die Implementierung des Prototypen, wurden das Konzept und der Prototyp mit den Anforderungen abgeglichen. Die Diskussion zeigt auf, dass alle Muss-Anforderungen berücksichtigt und bis auf die Kommunikationskomponente auch umgesetzt wurden (vgl. Kapitel 6.1). Da die Kommunikationskomponente nicht Teil der Implementierung war, wurde außerdem ein Proof-of-Concept-Ansatz vorgestellt, der veranschaulicht, dass die Anbindung von SmartControl über das erzeugte XML-Dokument funktioniert. Zusätzlich wurden verwandte Arbeiten vorgestellt und aufgezeigt, wie diese im Zusammenhang mit dieser Arbeit stehen, bzw. wie sie verwendet werden können, um diese Arbeit zu ergänzen (vgl. Kapitel 6.2).

Zusammenfassend legt diese Arbeit ein wichtiges Grundgerüst für ein System zur Modellierung und Ausführung von Verfahrensprozessen fest, das auf der Basis eines Maschinenbediensystems entwickelt wurde. Die geforderten Basisfunktionen wurden konzipiert, alle Muss-Anforderungen umgesetzt und dokumentiert. Der entwickelte Prototyp ermöglicht sowohl die Modellierung von Prozessmodellen als auch die Ausführung von Prozessinstanzen und durch die Konzeption einer XML-Schnittstelle kann SmartControl in diese integriert werden. Es sind viele Erweiterungen zum Ausbau der Funktionalität des Systems denkbar und eine Reihe von Erweiterungen wurde sowohl bei der Konzeption, als auch bei der Implementierung des Prototypen berücksichtigt.

## 7.2. Ausblick

Im Folgenden soll aufgezeigt werden, welche Erweiterung am Proof-of-Concept Prototypen aus Kapitel 5 vorgenommen werden müssen, um das entwickelte Konzept vollständig umzusetzen. Außerdem werden Erweiterungen vorgestellt, die das Konzept betreffen und in zukünftigen Arbeiten behandelt werden können. Dabei wird besonders auf den

Aspekt der Flexibilität eingegangen, da dieser im Prototypen von [MGKR15] bereits behandelt wurde.

### 7.2.1. Erweiterungen

#### Kommunikations-Anwendung

Um das Konzept vollständig umzusetzen, muss die Kommunikations-Anwendung implementiert werden (vgl. Kapitel 4.4). Dazu wird eine REST-Schnittstelle benötigt, die Aktivitäten von der Server-Anwendung erhält. Dies bedeutet, die Server-Anwendung muss so verändert werden, dass sie den Inhalt der Aktivitäten (z.B. SmartControl-Assistenten) nicht nur in der Ausführungsumgebung anzeigt, sondern an die REST-Schnittstelle sendet. Bei der Implementierung der Kommunikations-Anwendung muss neben der Architektur auf die Netzwerkkommunikation geachtet werden. Beispielsweise müssen Verbindungsfehlern so gehandhabt werden, dass sie die Ausführung der Prozessinstanz nicht einschränken. Außerdem wird ein SmartControl-Wrapper benötigt, der das Anlegen einer SmartControl-Aktivität erkennt und dazu dynamisch einen Assistenten in SmartControl anlegt. Die Entwicklung dieses Wrappers sollte systemnah mit SmartControl geschehen und über entsprechende Web-Technologien (z.B. Web Socket) eine Sitzung in SmartControl erhalten.

#### Einbinden einer Benutzerverwaltung

Um ein sinnvolles Freigabemodell von Prozessen zu ermöglichen, wird das Einbinden einer Benutzerverwaltung benötigt. Dies beinhaltet unter anderem das Anmelden an der Client-Anwendung und die Vergabe von Rollen, um Berechtigungen festzulegen. Die Struktur der Benutzerverwaltung sollte dabei nicht neu implementiert, sondern von SmartControl übernommen werden, damit beide Systeme auf eine gemeinsame Datenbasis zugreifen können und näher aneinander gekoppelt sind.

### Parameterkonfiguration

Für die Verwendung der Auftragsparameter innerhalb des Prozessmodells, ist im Prototypen eine textbasierte Referenzierung implementiert (vgl. Kapitel 5.3.2). Außerdem werden die Parameter im Prototypen im Subprozess-Plugin für SmartControl verwendet. Auf Grundlage dieser Auftragsparameterkonfiguration ist es denkbar, in einem zukünftigen Projekt ein benutzerfreundliches Verfahren zu entwickeln, um die Auftragsparameter in das Prozessmodell einzufügen. Zudem bietet die Art der Verwendung der Auftragsparameter Erweiterungspotential. Sie können auch für Entscheidungen im ODER Gatter verwendet werden, wie es im Konzept (vgl. Kapitel 4.2) beschrieben ist. Neben den Auftragsparametern kann auch das Einbinden der Maschinenparameter erweitert werden. Um diese in existierenden Maschinenfeldern im Prototypen zu verwenden, muss der genaue Pfad des Maschinenparameters im Maschinenbaum bekannt sein. Dies stellt für den Editoranwender ein Problem dar, da er eventuell nicht zu jedem benötigten Maschinenparameter den vollständigen Pfad kennt und es zu jeder Maschine eine Vielzahl von Parametern gibt. Dem Anwender sollte daher zur leichteren Bedienbarkeit ein geeignetes Auswahlmenü für Maschinenparameter angeboten werden. Dabei ist zu beachten, dass es sich bei den Maschinenparametern zum Teil um sensible Daten handelt und diese abhängig von der Rolle des Editoranwenders zugänglich sein sollen.

### Maschinen-Plugins

Ein Ziel dieser Arbeit war es, alle am Produktionsprozess beteiligten Maschinen direkt anzusprechen und die Arbeitsabläufe dieser Maschinen im Modellierungseditor zu modellieren. Sobald die Kommunikations-Anwendung implementiert und die Schnittstelle zu SmartControl vollständig umgesetzt wurde, kann damit begonnen werden, weitere Maschinenbediensysteme in dieses System zu integrieren und so diesem Ziel näher zu kommen. Der erste Schritt zur Anbindung eines neuen Maschinenbediensystems ist es, ein Schnittstellen-Format festzulegen, das im Modellierungseditor erstellt und über die Kommunikations-Anwendung übertragen werden kann. Dazu muss innerhalb der Modellierungsumgebung ein neues Plugin angelegt werden. Das Plugin soll sich dabei an dem Polymer-Element *editor-plugin-smartcontrol* orientieren. Die Struktur des

Plugins ist offen, muss allerdings auf einem JSON-Objekt beruhen, um es entsprechend speichern und laden zu können.

### 7.2.2. Flexibilität

Das vorgestellte Konzept aus Kapitel 4 fokussiert sich bei der Modellierung von Prozessmodellen auf einen statischen Ablauf. Bei der Ausführung einer Prozessinstanz ist fest vorgegeben, welche Aktivitäten ausgeführt werden und Unterschiede zweier Prozessinstanzen können nur durch die Verwendung anderer Auftragsparameter und ODER-Verzweigungen auftreten. Industrieunternehmen stehen jedoch häufig vor der Schwierigkeit, flexibel auf Kundenerwartungen oder Regularien zu reagieren, was durch einen dynamischeren Ablauf von Prozessinstanzen erleichtert werden kann [MGKR15]. Außerdem erfordert die hohe Nachfrage an kundenspezifischen Produkten eine Vielzahl an Prozessvarianten.

Ein Konzept, das sich mit der Flexibilität von Prozessinstanzen beschäftigt, ist Context-aware Process Injection (CaPI)[1] [MGKR15]. Diese kann aufgrund des aktuellen Kontexts der Prozessinstanz zur Laufzeit Prozess-Fragmente in das Basis-Prozessmodell einfügen. Die Ausgangsbasis des Prototypen aus Kapitel 5.1 implementiert dieses CaPI-Konzept. Sollte das in dieser Arbeit entwickelte Konzept im Rahmen einer weiteren Arbeit um Flexibilität erweitert werden, so kann die Implementierung des Prototypen aus [MGKR15] dazu verwendet werden, ohne dass die Modellierungsumgebung verändert werden muss.

[1] zu dt. Kontext-orientierte Prozess Injektion (vgl. Kapitel 5.1)

# A

# Quelltexte

Listing A.1: Datenmodell eines Graphen des Prozessmodell Editors

```
"Graph":
    [
        {
            "name": "start",
            "nodeType": "event"
        },
        {
            "name": "Droparea 1",
            "nodeType": "droparea",
            "dropid": 1,
            "dropmode": "head"
        },
        {
            "name": "Herstellung",
            "nodeType": "subprocess",
            "id": 1
        },
        {
            "name": "Droparea 2",
            "nodeType": "droparea",
            "dropid": 2,
            "dropmode": "normal"
```

```
23          },
24          {
25              "name": "Blisterverpackung",
26              "nodeType": "subprocess",
27              "id": 2
28          },
29          {
30              "name": "Droparea 3",
31              "nodeType": "droparea",
32              "dropid": 3,
33              "dropmode": "normal"
34          },
35          {
36              "name": "Gateway1",
37              "nodeType": "gateway",
38              "id": 1,
39              "gateType": "or",
40              "topCount": 1,
41              "botCount": 0,
42              "topContent":
43              [
44                  {
45                      "name": "Droparea 4",
46                      "nodeType": "droparea",
47                      "dropid": 4,
48                      "dropmode": "head"
49                  },
50                  {
51                      "name": "Endverpackung",
52                      "nodeType": "subprocess",
53                      "id": 3
```

```
                },
                {
                    "name": "Droparea 7",
                    "nodeType": "droparea",
                    "dropid": 7,
                    "dropmode": "normal"
                }
            ],
            "botContent":
            [
                {
                    "name": "Droparea 5",
                    "nodeType": "droparea",
                    "dropid": 5,
                    "dropmode": "head"
                }
            ],
            "topCondition": "packaging == true",
            "botCondition": "packaging == false"
        },
        {
            "name": "Droparea 6",
            "nodeType": "droparea",
            "dropid": 6,
            "dropmode": "normal"
        },
        {
            "name": "end",
            "nodeType": "event"
        }
    ]
```

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Literaturverzeichnis


[ACKM04] ALONSO, Gustavo ; CASATI, Fabio ; KUNO, Harumo ; MACHIRAJU, Vijay: *Web Services: Concepts, Architecture and Applications*. Berlin and Heidelberg [u.a.] : Springer, 2004

[BDC08] BLANC, Pascal ; DEMONGODIN, Isabel ; CASTAGNA, Pierre: A holonic approach for manufacturing execution system design: An industrial application. In: *Engineering Applications of Artificial Intelligence* 21 (2008), Nr. 3, S. 315–330

[BG08] BOTTI, Vicent ; GIRET, Adriana: *ANEMONA: A Multi-agent Methodology for Holonic Manufacturing Systems*. London : Springer-Verlag London, 2008

[Bit15] BITKOM: *Welches sind die wichtigsten IT-Trends des Jahres 2015?* http://de.statista.com/statistik/daten/studie/380843/umfrage/die-wichtigsten-trends-in-der-itk-branche/. Version: 2015. – Zuletzt besucht am 11.11.2015

[Bra05] BRANDL, Dennis: *The IT Implications of ISA 95 and ISA 99*. http://www.brlconsulting.com/Files/The%20IT%20Implications%20S95%20and%20S99.pdf. Version: 2005. – Zuletzt besucht am 14.05.2015

[Bun15] BUNDESMINISTERIUM FÜR WIRTSCHAFT UND ENERGIE: *Was ist Industrie 4.0?* http://www.plattform-i40.de/I40/Navigation/DE/Industrie40/WasIndustrie40/was-ist-industrie-40.html. Version: 2015. – Zuletzt besucht am 08.11.2015

[Cof15] COFFEESCRIPT: *Documentation*. http://coffeescript.org. Version: 2015. – Zuletzt besucht am 20.11.2015

[Col14] COLUMBUS, Louis: *Gartner's ERP Market Share Update Shows The Future Of Cloud ERP Is Now*. http://www.forbes.com/sites/louiscolumbus/2014/05/12/gartners-erp-market-

share-update-shows-the-future-of-cloud-erp-is-now/. Version: 2014. – Zuletzt besucht am 16.05.2015

[DDD15] DATA-DRIVEN-DOCUMENTS: *D3 Wiki*. https://github.com/mbostock/d3/wiki. Version: 2015. – Zuletzt besucht am 09.10.2015

[Dep97] DEPARTMENT OF HEALTH AND HUMAN SERVICES: CFR - Code of Federal Regulations Title 21: Part 11 / U.S. Food and Drug Administration. 1997. – Regierungsbericht

[DRRM+09] DADAM, Peter ; REICHERT, Manfred ; RINDERLE-MA, Stefanie ; GOESER, Kevin ; KREHER, Ulrich ; JURISCH, Martin: Von ADEPT zur AristaFlow BPM Suite - Eine Vision wird Realität: "Correctness by Constructionünd flexible, robuste Ausführung von Unternehmensprozessen. In: *EMISA Forum* 29 (2009), January, Nr. 1, S. 9–28

[FR10] FREUND, Jakob ; RÜCKER, Bernd: *Praxishandbuch BPMN 2.0*. 2., aktualisierte Aufl. München; Wien : Hanser, 2010

[H215] H2: *Database Engine*. http://www.h2database.com/html/main.html. Version: 2015. – Zuletzt besucht am 20.11.2015

[HBR10] HALLERBACH, Alena ; BAUER, Thomas ; REICHERT, Manfred: Capturing variability in business process models: the Provop approach. In: *Journal of Software Maintenance and Evolution: Research and Practice* 22 (2010), Nr. 6-7, S. 519–546

[Kol15] KOLB, Jens: *Abstraction, Visualization, and Evolution of Process Models*, Ulm University, PhD Thesis, 2015

[KR13] KOLB, Jens ; REICHERT, Manfred: A Flexible Approach for Abstracting and Personalizing Large Business Process Models. In: *Applied Computing Review* 13 (2013), Nr. 1, S. 6–17

[KWH13] KAGERMANN, Henning (Hrsg.) ; WAHLSTER, Wolfgang (Hrsg.) ; HELBIG, Johannes (Hrsg.): *Deutschlands Zukunft als Produktionsstandort sichern: Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0:*

*Abschlussbericht des Arbeitskreises Industrie 4.0*. Berlin : Forschungsunion im Stifterverband für die Deutsche Wirtschaft e.V, 2013

[MFT09] MEYER, Heiko ; FUCHS, Franz ; THIEL, Klaus: *Manufacturing Execution Systems: optimal design, planning, and deployment*. New York : McGraw-Hill, 2009

[MGKR15] MUNBROD, Nicolas ; GRAMBOW, Gregor ; KOLB, Jens ; REICHERT, Manfred: Context-Aware Process Injection: Enhancing Process Flexibility by Late Extension of Process Instances. In: *23rd International Conference on Cooperative Information Systems (CoopIS 2015)*, Springer, October 2015 (LNCS 9415), S. 127–145

[Nie06] NIELSEN, Jakob: *Prioritizing Web Usability*. Berkley, Calif. : New Riders, 2006

[Ora15] ORACLE: *JavaEE 7 Documentation*. `http://docs.oracle.com/javaee/7/index.html`. Version: 2015. – Zuletzt besucht am 20.11.2015

[Pol15] POLYMER: *API Developer Guide*. `https://www.polymer-project.org/0.5/docs/polymer/polymer.html`. Version: 2015. – Zuletzt besucht am 09.10.2015

[Pyt15] PYTHON: *Documentation 3.5.0*. `https://docs.python.org`. Version: 2015. – Zuletzt besucht am 13.11.2015

[Qoo15] QOOXDOO: *Documentation: Universal JavaScript Framework*. `http://qooxdoo.org/`. Version: 2015. – Zuletzt besucht am 13.11.2015

[RR07] RICHARDSON, Leonard ; RUBY, Sam: *RESTful Web Services*. Beijing; Köln [u.a.] : O'Reilly, 2007

[Rub12] RUBIN, Kenneth S.: *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Boston and New York [u.a.] : Addison-Wesley, 2012

[RW12] REICHERT, Manfred ; WEBER, Barbara: *Enabling Flexibility in Process-aware Information Systems*. Berlin and Heidelberg [u.a.] : Springer, 2012

[SSM06] SIMÃO, Jean M. ; STADZISZ, Paulo C. ; MOREL, Gérard: Manufacturing Execution Systems for customized production. In: *Journal of Materials Processing Technology* 179 (2006), Nr. 1–3, S. 268–275

[SW12] STRASSER, Artur ; WITTEK, Michael: IT-Compliance. In: *Informatik-Spektrum* 35 (2012), Nr. 1, S. 39–44

[Uhl14] UHLMANN PAC-SYSTEME GMBH & CO. KG: *Das Bediensystem SmartControl*. http://www.uhlmann.de/fileadmin/Redakteure_Website/02_Solutions/05_Automation_and_software/02_SmartControl/Datenblatt_SmartControl.pdf. Version: 2014. – Zuletzt besucht am 05.10.2015

[Vac11] VACZEK, David: Automating With MES. In: *Pharmaceutical & Medical Packaging News* 19 (2011), Nr. 8

[WAL08] WELGE, Martin K. ; AL-LAHAM, Andreas: *Strategisches Management: Grundlagen - Prozess - Implementierung*. Wiesbaden : Gabler, 2008

[Wes12] WESKE, Mathias: *Business Process Management: Concepts, Languages, Architectures*. Berlin; Heidelberg [u.a.] : Springer, 2012

Name: Manuel Fiedler Matrikelnummer: 844849

**Erklärung**

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Manuel Fiedler