JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

SOWI

Sozial- und Wirtschaftswissenschafliche
Fakultät

Prototypische Umsetzung der Automatisierung eines Prozesses für Nach- und Aufrüstung von Komponenten einer Formgebungsmaschine

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science (MSc)

im Masterstudium

Wirtschaftsinformatik

Eingereicht von:
Karl-Heinz Stöckler

Angefertigt am:
Institut für Wirtschaftsinformatik - Data & Knowledge Engineering

Beurteilung:
o.Univ.-Prof. Dr. Michael Schrefl

Mitwirkung:
Dr. Christoph Schütz

Linz, August 2015

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, Juli 2015 *Karl-Heinz Stöckler*

Kurzfassung Auf Formgebungsmaschinen können Komponenten nach- und aufgerüstet werden. Dies wird jedoch nur in Ausnahmefällen von Kunden in Anspruch genommen, da die Kunden diese Möglichkeit nicht kennen oder nicht wissen, dass es nützliche Zusatzkomponenten für ihre Maschinen gibt. Bei den Zusatzkomponenten bzw. Zusatzoptionen unterscheidet man zwischen Software- und Hardwarekomponenten. Mit dieser Arbeit wird ein Prototyp einer Plattform geschaffen, welche Benutzern die Möglichkeit bietet, Zusatzoptionen aufzulisten und zu bestellen.

In einem ersten Schritt wird der dafür notwendige Prozess, angefangen von der Auflistung der Komponenten bis hin zu deren Auslieferung, erhoben und modelliert. Es wird eine Werkzeugauswahl durchgeführt, um ein geeignetes Prozessautomatisierungs-Werkzeug für die Implementierung des Prototyps auszuwählen. Mit diesem Werkzeug wird der Teil des Prozesses, der sich mit internen Benutzern und mit den Softwarekomponenten befasst, umgesetzt.

Somit wird mit dieser Arbeit gezeigt, dass sich das Patent mit der Patentnummer AT 514527 A1 2015-01-15 umsetzen lässt. Des Weiteren werden Erklärungen und Vorschläge für den Ausbau, die Erweiterung und Verbesserung des erstellten Prototyps gegeben.

Abstract There is the possibility to retrofit and upgrade additional components on injection molding machines. Customers are not using this opportunity very often because they do not know about this possibility or they do not know which possible components for their machines are available. This additional components can be differentiated into software and hardware components. With this thesis a prototype for a platform which offers the opportunity for users to view and order possible components for their machine, has been created.

As a first step the process starting with creating and viewing a list of possible options and ending with the delivery, has been assimilated and modelled. A tool selection has been created to choose an appropriate tool for the implementation of the prototype. With this tool the part of the process which addresses the usage of internal users and the order of software components, has been developed.

Therefore this thesis shows that the patent with the patent number AT 514527 A1 2015-01-15 can be implemented. Furthermore it describes and offers suggestions how this prototype can be extended, upgraded and improved.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1

Einleitung

Formgebungsmaschinen, in Folge auch Spritzgussmaschinen genannt, sind mit einer Steuerung ausgerüstet, diese Steuerungskomponente ist die Schnittstelle zwischen Mensch und Maschine. Mit dieser Komponente kann die Maschine gesteuert und überwacht werden. Für diese Steuerungen bzw. Maschinen gibt es auch Zusatzoptionen, welche die Kunden bei der Bestellung der Maschine auswählen und kaufen können. Diese Zusatzoptionen können reine Software, reine Hardware oder Hardware mit benötigter Software sein. Diese Maschinen werden dann mit den gewählten Optionen angefertigt, die benötigten Softwareoptionen werden zusammengefügt und in der Steuerung der Maschine installiert. Danach wird die Maschine zu den Kunden geliefert und dort in Betrieb genommen.

Derzeit bleibt die Maschine dann meist in diesem Software- und Hardware-Zustand. Änderungen gibt es derzeit in den folgenden Fällen: Benötigen die Kunden Ersatzteile, so können sie eine Anfrage über die Hotline, eine Niederlassung oder die Kundenplattform econnect stellen. Die zweite Möglichkeit, die zu einer Änderung führen kann, funktioniert über eine Hotline-Anfrage. Hiervon gibt es auch wieder verschiedene Ausprägungen. Wie schon erwähnt, können die Kunden Ersatzteile bestellen. Des Weiteren können sie Problemfälle und Störungen melden, was dazu führen kann, dass neue Hardware und/oder Software benötigt wird. Die dritte Variante ist, dass die Kunden eine neue Funktion bzw. Option wünschen und bei der Hotline bezüglich dieser nachfragen.

Es gibt aktuell jedoch keine Möglichkeit, außer der aktiven Nachfrage der Kunden bei der Hotline, dass die Kunden über Zusatzfunktionen und Zusatzoptionen für deren Maschinen informiert werden. Mit diesem Projekt soll den Kunden eine Plattform angeboten werden, welche für deren Maschinen, die möglichen Zusatzoptionen auflistet. In weiterer Folge soll diese Plattform den Kunden auch die Möglichkeit bieten, diese Optionen, ähnlich wie in einem App-Store, zu bestellen. Der Ablauf der Bestellung bis hin zur Auslieferung soll soweit als möglich automatisiert werden.

Die Arbeit ist wie folgt aufgebaut: Im Kapitel 1.1 wird auf die Aufgabenstellung eingegangen und diese beschrieben. Nachdem im Unterkapitel 1.2 die Anforderungen, die für diese Arbeit relevant sind, definiert werden, wird im Un-

terkapitel 1.3 auf die Zielsetzung eingegangen und das Ziel der Arbeit gesetzt. Der Prozess, der erhoben und anhand eines BPMN Prozesses modelliert wurde, wird in Kapitel 2 abgebildet und erklärt. Im Kapitel 4 wird auf die Automatisierung des Prozesses eingegangen. Dazu gehören die im Unterkapitel 3 beschriebene Werkzeugauswahl, die Beschreibung der Prozessautomatisierung im Unterkapitel 4.1 sowie die im Unterkapitel 4.2 enthaltene Servicebeschreibung. Die Werkzeugauswahl ist wiederum unterteilt in eine Werkzeugübersicht mit Vorauswahl, eine Bewertung der Vorauswahl, Nutzwertanalyse und Entscheidung über ein Werkzeug, wohingegen die Prozessautomatisierung die Implementierungsdetails und die Systemarchitektur enthält. In den Implementierungsdetails werden zuerst die Prozessvariablen aufgelistet und erklärt, die bei der Umsetzung verwendet worden sind, danach werden die umgesetzten und noch nicht umgesetzten Prozessschritte erklärt und es wird auf die Anpassungen und Konfigurationen eingegangen die getätigt worden sind. Es folgt das abschließende Kapitel 5 mit Fazit, Schlussfolgerung und Ausblick.

1.1 Aufgabenstellung

Diese Arbeit basiert auf dem Patent mit der Patentnummer AT 514527 A1 2015-01-15 [Fritz et al., 2013]. Es soll die teilweise Implementierung des im Patent beschriebenen Verfahrens zur Nachrüstung von Komponenten für eine Formgebungsmaschine durchgeführt werden.

Mit einer App-Store-ähnlichen Plattform soll den Kunden eine Möglichkeit geboten werden, sich über Zusatzoptionen für deren Spritzgussmaschinen zu informieren, ohne dafür den Kundenservice kontaktieren zu müssen. Dieser App-Store soll den Kunden alle möglichen Zusatzoptionen für deren Maschinen auflisten und ihnen in Folge auch die Möglichkeit bieten, die Bestellung einer Option auszulösen.

Mit Zusatzoptionen sind in diesem Fall Software- und Hardwarekomponenten gemeint. Die Auslieferung von Hardwarekomponenten ist ohnehin nur physisch möglich. Die Auslieferung von reinen Softwareoptionen könnte jedoch statt mit einem Speichermedium auch über das Internet erfolgen. Softwareoptionen können zum Beispiel Überwachungsprogramme oder Erfassungsprogramme für den Energieverbrauch sein.

Der große Unterschied zu anderen App-Stores ist in diesem Fall, dass die Apps bzw. Optionen, die den Kunden angeboten werden, nicht modular sind und damit nicht einfach zu der bestehenden Software hinzu installiert werden können. Nur der Source Code der Optionen ist modular, jedoch nicht die fertigen Softwarepakete. Eine weitere Schwierigkeit ist, dass es in der Software und zwischen den Optionen sehr viele Abhängigkeiten und Konfigurationen gibt, welche erst beim Kompilieren und Bauen der Software aufgelöst werden können. Daher ist es nicht möglich, einfach nur die neue Option auszuliefern, sondern es muss zu den bestehenden Software-Sourcen der Source der neuen Option hinzugenommen werden und das ganze Paket neu kompiliert und gebaut werden. Es wird dann ein Update-Paket

erstellt, welches alle notwendigen Änderungen beinhaltet, und dieses wird dann ausgeliefert.

Im Idealfall soll dieser App-Store direkt auf der Maschinensteuerung laufen. Damit dies machbar ist, muss die Maschine jedoch mit dem Internet verbunden sein, denn die Erstellung der Optionsliste ist sehr komplex und benötigt viele Informationen, die nicht direkt auf der Maschine gespeichert sind. Einige der Maschinen bzw. Steuerungen sind mit dem Internet oder mit einem Netzwerk verbunden. Die Anzahl der vernetzten Maschinen ist derzeit eher gering, jedoch wird sie in den nächsten Jahren stetig steigen, da die Vernetzung in der heutigen Zeit immer wichtiger wird. Um Kunden bzw. deren Maschinen nicht auszuschließen, wenn diese nicht mit dem Internet verbunden sind, soll der App-Store auch auf einem PC lauffähig sein. In Abbildung 1.1 wird veranschaulicht, wie dies aussehen soll. In der Abbildung links unten ist ersichtlich, dass Benutzer über einen PC, der mit dem Internet verbunden ist, auf die Applikation zugreifen und sich Optionen für eine Maschine auflisten lassen können. Links oben ist ersichtlich, dass es Maschinen gibt, die direkt mit dem Internet verbunden sind und somit die Möglichkeit bieten, direkt von der Maschine aus die Applikation aufzurufen. In der Mitte der Abbildung ist ein Server dargestellt, auf welchem die Applikation läuft. Dieser Server bzw. die Applikation übernimmt die Prozesssteuerung und führt Aufgaben aus wie z. B. mögliche Optionen auflisten, Angebotserstellung, Softwaregenerierung. Dieser Server kommuniziert auch mit anderen Servern und Datenbanken, um benötigte Daten zu bekommen oder um diverse Aufgaben, wie z. B. das Abspeichern der Softwaresicherung, ausführen zu lassen.

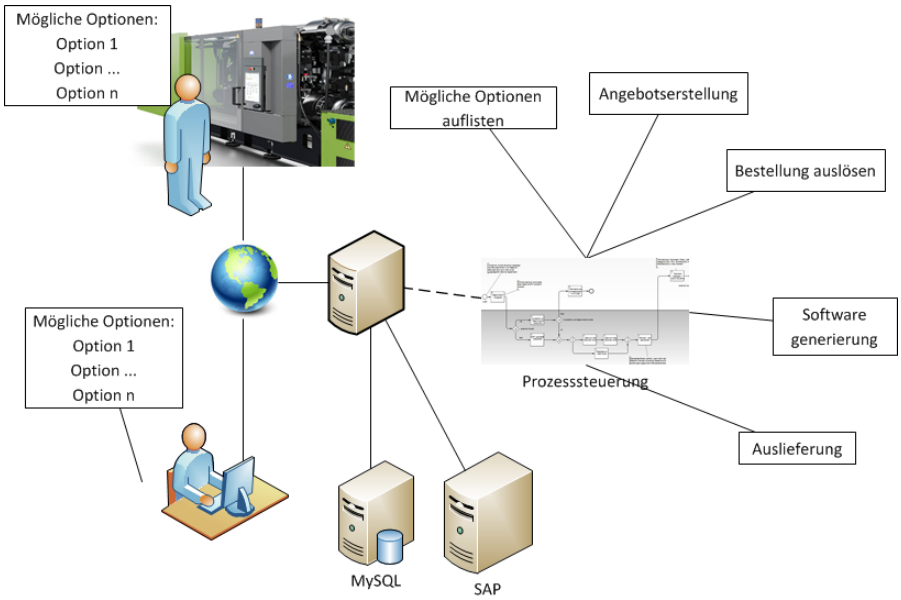

Abbildung 1.1. Skizze Aufgabenstellung

Wie im Kapitel 1 schon erwähnt, gibt es das econnect Kundenportal. Auf den ersten Blick sieht es so aus, als würde es sich sehr gut dafür eignen, um die Funktionalität des App-Stores einfach in dieses Portal zu integrieren. Das Problem hierbei ist jedoch, dass dieses Portal einen Fat-Client verwendet, welcher vorher installiert werden muss und nicht so einfach auf einer Spritzgussmaschine installiert werden kann. Daher ist die direkte Integration in das econnect Kundenportal keine Lösung, es soll jedoch möglich sein, mittels eines Links vom econnect Kundenportal auf das neue System zu gelangen.

Der Prozess von der Abfrage der Zusatzoptionsliste bis hin zur Auslieferung soll soweit als möglich automatisiert werden. Bei reinen Standardsoftwareoptionen sollte es möglich sein, obwohl sich im Prozess viele komplexe Schritte befinden, den Prozess fast zur Gänze automatisiert durchzuführen. Die Generierung der Software oder alleine schon die Erstellung der Optionsliste ist ein sehr komplexer Schritt. Handelt es sich um eine Sonderoption, so werden wahrscheinlich mehrere manuelle Schritte stattfinden müssen, da hierbei neue Software geschrieben oder bestehende Software angepasst werden muss. Im Prozess soll auch eine Unterscheidung zwischen internen und externen Benutzern gemacht werden. Bei internen Kunden, bei welchen es sich um Mitarbeiter der Firma Engel handelt, fällt der Part mit Angebot, Bestellung und Bezahlung weg.

Für die Umsetzung ist es auch notwendig, verschiedene Business Prozess Management Systeme (BPMS) zu evaluieren und zu analysieren, um ein geeignetes Werkzeug für die Prozessablaufsteuerung auswählen zu können. Dieses Werkzeug wird dann für die Implementierung bzw. Automatisierung des Prozesses benötigt und verwendet.

1.2 Anforderungen

Die wichtigste Anforderung ist, dass das System keine kostenpflichtigen Werkzeuge oder Frameworks verwendet. Für die Entwicklung, den ersten internen Probebetrieb bzw. das Pilotprojekt soll das System ohne anfallende Kosten betrieben werden können. Sollte zu einem späteren Zeitpunkt das System auch für Kunden extern zur Verfügung stehen, dann wäre es zwar nach wie vor von Vorteil, wenn das System kostenlos betrieben werden kann, jedoch ist es keine verpflichtende Anforderung mehr.

Das System soll, aus Sicherheits- und Datenschutzgründen, nicht Cloudbasiert, sondern lokal lauffähig sein. Für die Entwicklung soll das System lokal auf einem Rechner aufgesetzt werden können und in späterer Folge soll es dann auf einem unternehmensinternen Server laufen.

Wichtig ist auch, dass sich die Benutzeroberfläche anpassen lässt, damit das Design der Corporate Identity des Unternehmens entsprechend angepasst werden kann.

Die Modellierung des Prozesses soll mittels Business Process Modeling Notation (BPMN) durchgeführt werden. Eine weitere Anforderung ist, dass externe Services von dem Prozess aufgerufen werden können.

Wie schon im Kapitel 1.1 beschrieben, soll die Applikation sowohl von einer Spritzgussmaschine als auch von einem PC zugänglich sein. Die Applikation soll ohne jegliche Installation eines Clients oder Ähnlichem aufgerufen und verwendet werden können. Daher soll die Applikation eine Webplattform sein, welche mit einem Standardbrowser geöffnet werden kann.

Für die Evaluierung der Werkzeuge und der Auswahl eines geeigneten Werkzeugs für die Umsetzung ist es sinnvoll, noch weitere Anforderungen bzw. Kriterien zu berücksichtigen. Diese werden dann im Kapitel 3 definiert und für die Auswahl verwendet.

Um das System benutzen zu können, soll eine einmalige Registrierung der Benutzer erfolgen. Bei der Registrierung soll eine Sicherheitsüberprüfung eingebaut werden, damit nur interne Benutzer oder Unternehmenskunden sich registrieren können.

1.3 Zielsetzung

Das Ziel für diese Arbeit ist es, einen ersten lauffähigen Prototyp für die Ablaufsteuerung des Prozesses für interne Benutzer zu erstellen. Der Prozess soll für die internen Benutzer komplett erhoben, abgebildet und umgesetzt werden. Für externe Benutzer soll der Prozess überblicksmäßig erhoben und in BPMN abgebildet werden. Das heißt, der Prozess soll für diesen ersten Prototypen soweit umgesetzt werden, dass interne Benutzer das System verwenden können. Der restliche Prozess für die externen Benutzer und für Hardware-relevante Komponenten wird für den ersten Prototypen nicht umgesetzt. Für diesen Prototypen werden auch nur kostenfreie Optionen herangezogen. Es werden keine Optionen berücksichtigt, welche dokumentationsrelevant sind.

Die wichtigsten Elemente, die zeigen sollen, dass das im Kapitel 1.1 genannte Patent umsetzbar ist, sind im Prozess für die internen Benutzer enthalten. Zu diesen Elementen zählt das Zur-Verfügung-Stellen der Optionsliste sowie das Generieren der Software. Hinter diesen beiden Elementen stecken jeweils mehrere komplexe Schritte, die durchgeführt werden müssen. Für den ersten Prototypen werden noch keine Optionen in der Optionsliste vorhanden sein, welche Hardware beinhalten, das heißt, am Anfang konzentriert sich die Arbeit auf die reinen Softwarekomponenten.

Für die externen Benutzer müsste der Prozess, wie im Kapitel 2 ersichtlich, zum Beispiel um die Schritte für die Angebotsgenerierung und Bestellung erweitert werden. Diese Abläufe sind im Unternehmen jedoch standardisiert und müssten in den Prozess entsprechend integriert werden. Da diese Abläufe aber lediglich den bürokratischen Overhead darstellen und mit der Grundfunktionalität des Patentes nicht viel zu tun haben, werden sie für den Prototypen außen vor gelassen und sind für diese Arbeit nur insofern relevant, dass der Prozess überblicksmäßig erhoben wird.

Bevor der Prozess umgesetzt und lauffähig gemacht werden kann, ist eine Werkzeugauswahl durchzuführen, welche verschiedenen Werkzeuge berücksichtigt

und ein geeignetes für die Umsetzung auswählt. Ein weiteres Ziel ist es, einen Prototypen für die Registrierung von internen Benutzern zur Verfügung zu stellen, damit diese sich registrieren und das System verwenden können.

1.4 State-of-the-Art

Geschäftsprozessautomatisierung wird von [Gartner, 2015] als die Automatisierung von komplexen Prozessen und Funktionen, die über die konventionelle Datenmanipulation und Datenverarbeitung hinausgehen, definiert. Für diese Automatisierung werden komplexe und fortgeschrittene Werkzeuge und Technologien verwendet.

Laut [Ter Hofstede et al., 2009] wird Geschäftsprozessmanagement und somit auch die Prozessautomatisierung durch die steigende Produktivität und durch die Anforderung der Kostenersparnis immer wichtiger werden. Die Prozessautomatisierung wird oft auch als Workflowmanagement beschrieben und ist dafür zuständig, Prozesse soweit als möglich zu automatisieren. Mit Werkzeugen können Systeme entworfen werden, welche die Aufgaben übernehmen, um die anfallenden Arbeitsschritte zum richtigen Zeitpunkt an die richtige Ressource weiterzugeben, wobei eine Ressource entweder ein Mensch oder ein anderes System sein kann. Der Prozessfortschritt kann überwacht werden und somit entsteht die Möglichkeit, zu erkennen, wenn in einem Prozess etwas schiefläuft. Im Hintergrund können verschiedenste Ereignisse, wie zum Beispiel der Abschluss einer Aufgabe, geloggt werden. Durch die Prozessautomatisierung können Kosten und Zeit eingespart werden. Dafür ist es notwendig, den Workflow aufzunehmen und alle Aspekte zu betrachten. Die individuellen Aktivitäten und Aufgaben müssen erfasst werden, die Abarbeitungsreihenfolge muss definiert werden und die benötigten Ressourcen müssen angegeben werden. Es müssen auch jegliche Daten und Informationen, welche benötigt und weitergegeben werden, erfasst werden.

[Dumas et al., 2013] beschreibt Prozessautomatisierung so, dass im breiten Sinn alle denkbaren Teile eines Ablaufs in einem Prozess automatisiert werden, angefangen von einfachen Tätigkeiten, innerhalb einer einfachen Prozess-Aktivität, bis hin zur koordinierten Automatisierung von gesamten, komplexen Prozessen.

Mittels der Prozessmodellierungssprache Business Process Modeling Notation (BPMN) können Prozesse abgebildet und modelliert werden. Viele Prozessautomatisierungs-Werkzeuge unterstützen diese Sprache, welche von der Object Management Group (OMG) gepflegt wird. BPMN ist eine grafische Notation, welche sehr verbreitet ist um Prozesse darzustellen [Dijkman et al., 2011]. Voraussetzung, um diese Arbeit lesen zu können, ist, Wissen über BPMN 2.0 zu besitzen. Zur Aneignung dieses Wissens wird auf [Allweyer, 2010] verwiesen. Vor allem ist es wichtig, zu wissen, wie die einzelnen Elemente aussehen und wie diese bezeichnet werden. Das Wissen sollte so weit fortgeschritten sein, dass Prozessdiagramme gelesen und verstanden werden können. Es sollten auch die Begriffe, welche BPMN für die einzelnen Elemente definiert hat, verstanden werden.

Prozessautomatisierungs-Werkzeuge sind Software-Werkzeuge, in welche die oben beschriebenen Informationen eingegeben und abgebildet werden können. Das Werkzeug übernimmt dann die Aufgabe, dass die Arbeitsschritte den richtigen Ressourcen zur richtigen Zeit und in der richtigen Reihenfolge zugewiesen werden [Ter Hofstede et al., 2009]. [Dumas et al., 2013] fokussieren sich auf spezielle Prozessautomatisierungs-Werkzeuge, welche sich auf das Wissen über Prozess-Aktivitäten und wie diese voneinander abhängen spezialisieren. Systeme die mit solchen Werkzeugen erstellt werden, werden auch als „process-aware informations systems“ bezeichnet, was so viel heißt wie prozessbewusste oder prozessgesteuerte Informationssysteme. Diese Systeme werden auch als Business Prozess Management Systeme (BPMS) bezeichnet. Es gibt auch andere prozessgesteuerte Informationssysteme wie Customer Relationship Management (CRM) Systeme und Enterprise Ressource Planning (ERP) Systeme. Spezifisch bei einem BPMS ist, dass es die explizite Beschreibung eines Geschäftsprozesses, in der Form eines Prozessmodells, nutzt, um die Aktivitäten im Prozess zu koordinieren und zu steuern. Die Aufgabe eines solchen Werkzeugs ist es, automatisierte Geschäftsprozesse so zu koordinieren, dass alle Aufgaben zur richtigen Zeit von der richtigen Ressource ausgeführt werden.

Der Leistungsumfang solcher Prozessautomatisieruns-Werkzeuge ist unterschiedlich. Manche Werkzeuge bieten die komplette Auswahl des gesamten Prozess Lebenszyklus an. Einfachere Werkzeuge bieten nur das Modellieren und Automatisieren des Prozesses an, wohingegen komplexere System auch das Simulieren, Überwachen und das Integrieren von anderen Applikationen anbieten. Jedoch die Hauptfunktionalität eines solchen Werkzeuge bleibt nach wie vor die Automatisierung eines Geschäftsprozesses [Dumas et al., 2013].

In Abbildung 1.2 ist die Architektur eines typischen BPMS zu sehen. Die Hauptkomponenten solch eines Systems sind die Execution Engine, die Task-liste, das Prozessmodellierungswerkzeug und das Administrations- und Überwachungswerkzeug. Das zentrale Element in einem BPMS ist die Execution Engine, welche mit externen Services kommunizieren kann und verschiedene Funktionalitäten liefert. Diese Funktionalitäten sind ausführbare Instanzen eines Prozesses zu erstellen, Aufgaben an die verschiedenen Prozessbeteiligten zuzuweisen, um den Prozess vom Anfang bis zum Ende auszuführen, benötigte Prozssdaten automatisch abzurufen und zu speichern und automatische Tasks anderen Softwaresystemen zuzuweisen. Mit dem Prozessmodellierungswerkzeug können Prozessmodelle erstellt und bearbeitet werden. Des Weiteren können zusätzliche Informationen eingegeben werden wie z.B. Ein- und Ausgabedaten, Prozessbeteiligte und Geschäftsregeln, welche einem Prozess oder einer Aktivität zugeordnet werden können. Ein Modellierungswerkzeug kann auch Prozessmodelle in einem Repository ablegen und von dort wieder laden. Die Taskliste ist die Benutzeroberfläche, welche den Prozessbeteiligten zur Verfügung steht. Dort werden ihnen ihre zugeordneten offenen Aufgaben angezeigt und können dort auch abgearbeitet werden. Die externen Services repräsentieren Funktionen, die von anderen Systemen zur Verfügung stehen, d.h. die Execution Engine kann die Funktionen dieser Services verwenden und aufrufen. Somit kann die Execution Engine Aufgaben von anderen Systemen

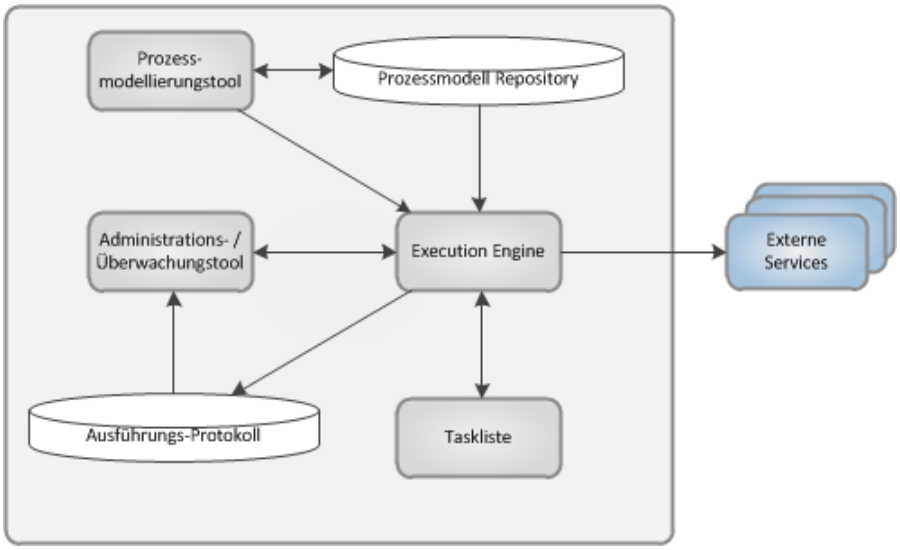

Abbildung 1.2. BPMS-Architektur [Dumas et al., 2013]

abarbeiten lassen. Mit dem Administrations- und Überwachungswerkzeug werden die administrativen Tätigkeiten erledigt sowie die Prozesse überwacht. Somit kann die Benutzerverwaltung durchgeführt und auch der Prozessfortschritt eingesehen werden. Es besteht die Möglichkeit, in die Prozesse einzugreifen und Aufgaben von einem Benutzer zu einem anderen zu verschieben [Dumas et al., 2013].

Die Windows Communication Foundation (WCF) ist laut [Selly et al., 2006] eine Technologie von Microsoft für verteilte Systeme. Es ist ein einheitliches Programmiermodell, welches die Erstellung von dienstorientierten Anwendungen ermöglicht. WCF ermöglicht es, Lösungen zu erstellen, welche zuverlässig, sicher und transaktiv sind. Diese dienstorientierten Lösungen können auch über verschiedene Plattformen und in bestehende Lösungen integriert werden [Microsoft, 2015]. Die Dienste und Services, welche von Applikationen wie z. B. einer Prozessautomatisierung verwendet werden, können mittels WCF erstellt und zur Verfügung gestellt werden. WCF ist auch für das Erstellen von Service Oriented Architectures (SOA) geeignet.

Die CC300 Steuerung ist die neue Steuerung der Engel Formgebungsmaschinen. [Engel, 2015a] gibt an, dass sich mit dieser neuen Steuerung die komplexen und hocheffizienten Maschinen durch eine klare Menüführung, intuitive Bedienung und eine smarte Steuerung ähnlich leicht wie ein Smartphone bedienen lassen. Das zentrale Bedienelement ist das so genannte e-move, mit diesem lassen sich geschwindigkeitssensible und millimetergenaue Steuerungen von Bewegungen vornehmen [Engel, 2015b]. Die Steuerung ist mit einem 21“ Full-HD Touch-Display ausgestattet, welches mit einem Sicherheitsglas eine schmutzunempfindliche und robuste Oberfläche bietet.

AngularJS ist ein JavaScript-Framework und es handelt sich dabei um eine Bibliothek, die in JavaScript geschrieben ist. Es kann mit dem Skript Tag <script> in HTML-Seiten direkt eingefügt oder als JavaScript-Datei eingebunden werden [w3schools, 2015]. Laut [Behring and Poddebniak, 2015] soll AngularJS die Entwicklung von Web-Frontends erleichtern und das serverseitige Zusammensetzen von HTML-Seiten wird nicht benötigt. Es erlaubt auch die Entkopplung von Back- und Frontend. [Lavin, 2014] erwähnt, dass es sich bei AngularJS um ein populäres JavaScript-Framework handelt, welches seit ein paar Jahren von der Open Source Community viel verwendet und gut angenommen wird. Daher hat es in den letzten Jahren auch viele Adaptierungen und Erweiterungen gegeben. Zum Beispiel wurde die Bibliothek Bootstrap in AngularJS integriert, um die Frontend-Entwicklung zu vereinfachen. Mittels Datenbindungen, Validierungen und sogenannten „Response Handlern“, welche auf Benutzer-Aktionen reagieren, können Benutzer-Interaktionen behandelt werden.

Seit einigen Jahren befindet sich die Industrie in einem großen Wandel welcher als Industrie 4.0 bezeichnet wird [Sendler, 2013]. Bei diesem Wandel geht es darum, dass die Industrie, deren Produkte und Dienstleistungen mit Software durchdrungen werden und gleichzeitig Dienste und Produkte über das Internet und andere Netze verbunden werden. Dadurch entstehen auch neue Produkte und Dienste, welche die Arbeit und das Leben der Menschen, vor allem den Umgang mit Technik, Technologien und Produkten, verändern. Für Industrie 4.0 ist aber auch eine grundlegende Anpassung und Veränderung der Produktion und Produktentwicklung notwendig, um zu gewährleisten, dass diese neuen Technologien wirtschaftlich nutzbringend und qualitativ hochwertig umgesetzt werden. Wichtig hierfür ist, dass die Industriebetriebe die Details von Industrie 4.0 miteinander vereinbaren. Dies bedeutet, dass ein einheitliches Verständnis über Begriffe wie „Cyber-Physical Systems (CPS)“, „vierte industrielle Revolution“, Internet der Dinge“ und viele mehr, geschaffen werden muss. Dabei sollte auch darauf geachtet werden, dass diese Begriffe keine leeren Schlagwörter werden.

[Dorst and BITKOM, 2012] beschreibt den Begriff Industrie 4.0 als Ansatz, um die Produktion und deren Umfeld durch Informations- und Kommunikationstechnologie (ITK) zu vernetzen. Der Fokus liegt dabei auf der globalen Fertigung und der zugehörigen Verteilung und Logistik, mit dem Ziel der Flexibilisierung und Automatisierung. Laut [Dorst and BITKOM, 2012] wird auch von einer 4. Industriellen Revolution gesprochen, da diese Entwicklung weitreichende Auswirkungen auf Produktivität, Technologie, Arbeisorganisation und Wissenschaft hat. Die 4. Industrielle Revolution hat auch Auswirkungen auf die Anforderungen an den Menschen, da sich die Arbeitswelt und -kultur damit rasch verändern. Das heißt, darauf muss sich die Industrie bei der Ausbildung und Rekrutierung von Fachkräften einstellen. Um die Terminologien abzustimmen und systemübergreifende Lösungen, die auch Zusammenarbeit erlauben, zu entwickeln, ist es notwendig, eine Verbindung zwischen den Softwarearchitekten, Wirtschaftsinformatikern und Industrieinformatikern herzustellen.

2

Modellierung des Prozesses

In diesem Kapitel wird nun der erhobene Prozess anhand von Abbildungen und zugehörigen Beschreibungen dargestellt. Zuerst wird der abstrakte Prozess dargestellt und beschrieben und danach wird auf die einzelnen Teilprozesse noch genauer eingegangen. Dieser abstrakte Hauptprozess wird als „Webstore“ bezeichnet. Wie schon in 1.3 definiert wird der Prozess für interne und externe Benutzer erhoben und dargestellt. Die Prozessabbildungen beinhalten graue Aufgaben, in Folge auch Task genannt, dies bedeutet, dass diese Aufgaben nur für Hardware-relevante Optionen oder nur für externe Benutzer benötigt werden. Das heißt, diese Tasks sind mit dem Prototypen nicht umgesetzt worden.

2.1 Hauptprozess: Webstore

Der Hauptprozess ist in den Abbildungen 2.1, 2.2, 2.3 und 2.4 zu sehen. Es gibt hier die drei Lanes Benutzer, System, CSD-VET. Benutzer können sowohl interne als auch externe Benutzer sein, bei internen Benutzern handelt es sich um Engel-Mitarbeiter. Das System übernimmt die Service-Tasks und die Automatisierung. Der CSD-VET ist die Vertriebsabteilung von Engel. Der Hauptprozess Webstore beginnt in Abbildung 2.1, sobald sich die Benutzer in das System eingeloggt und den Prozess mittels der Eingabe einer Fabrikationsnummer gestartet haben. Eine Fabrikationsnummer oder auch Maschinennummer genannt ist die Seriennummer einer Formgebungsmaschine. Nachdem die Benutzer die Fabrikationsnummer eingegeben haben, wird eine Sicherheitsüberprüfung durchgeführt, diese ist im Kapitel 2.2 genauer beschrieben. Der nächste Schritt ist das Generieren der Optionsliste, dazu mehr im Kapitel 2.3.

Im Task „Optionsliste anzeigen/Optionen auswählen“ bekommen die Benutzer eine Übersicht über die bereits installierten und über die möglichen Optionen. Es werden den Benutzern auch Detailinformationen wie z. B. der Name der Option und die Version angezeigt. Die Benutzer haben die Möglichkeit, aus den möglichen Optionen eine oder mehrere auszuwählen. In dem Service-Task „Optionen evaluieren“ werden nun die ausgewählten Optionen evaluiert. Diese Evaluierung

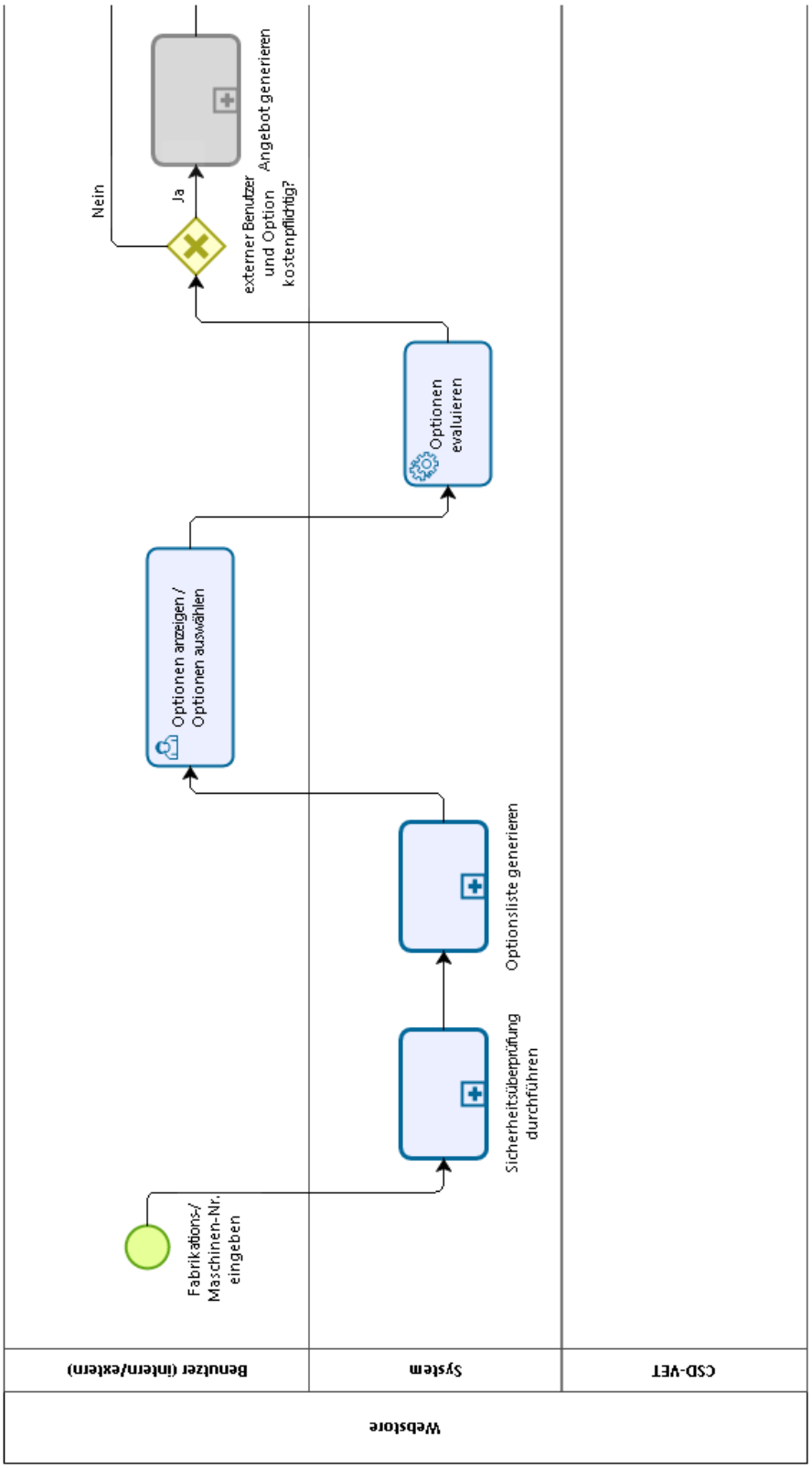

Abbildung 2.1. Hauptprozess Webstore Prozessteil 1/4

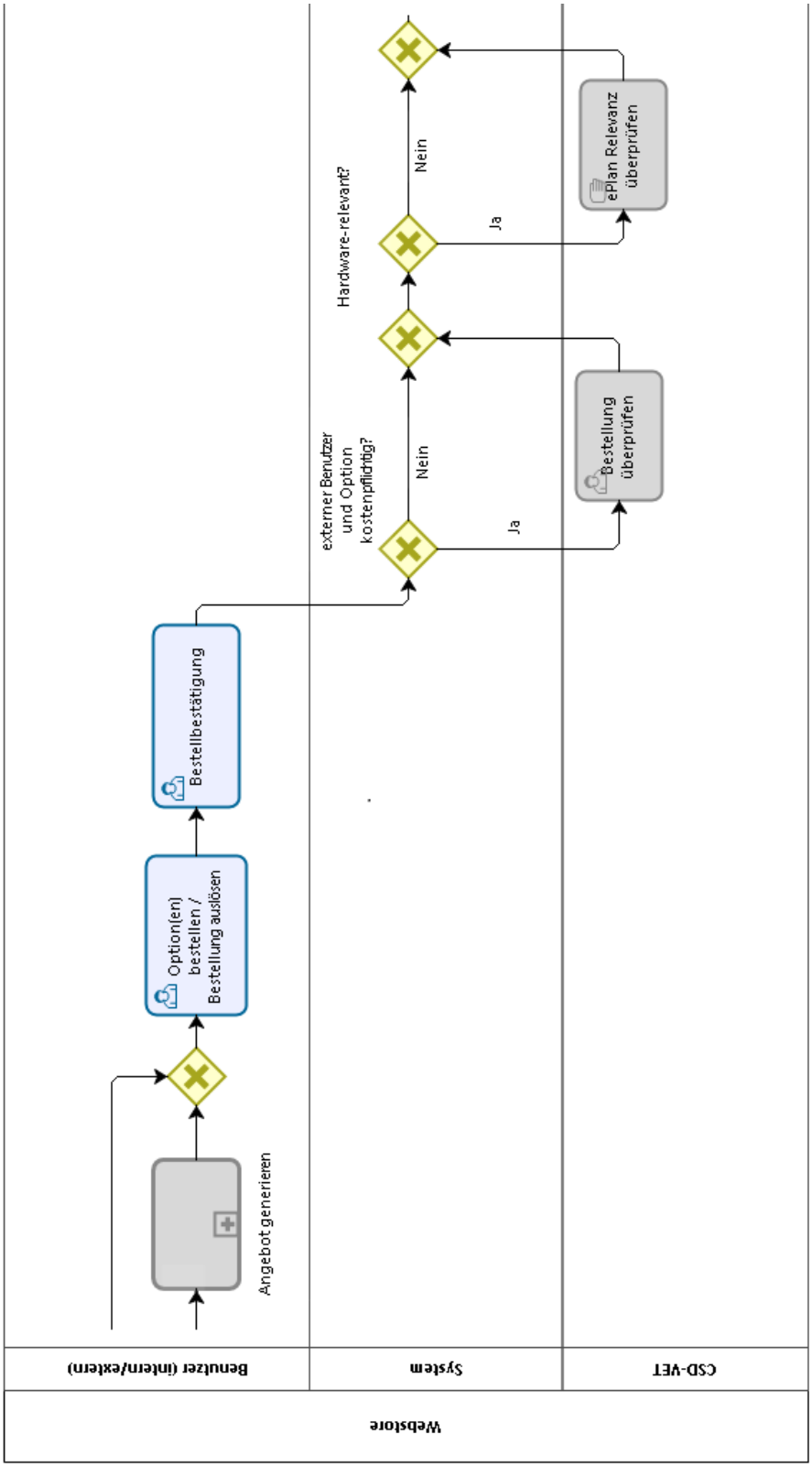

Abbildung 2.2. Hauptprozess Webstore Prozessteil 2/4

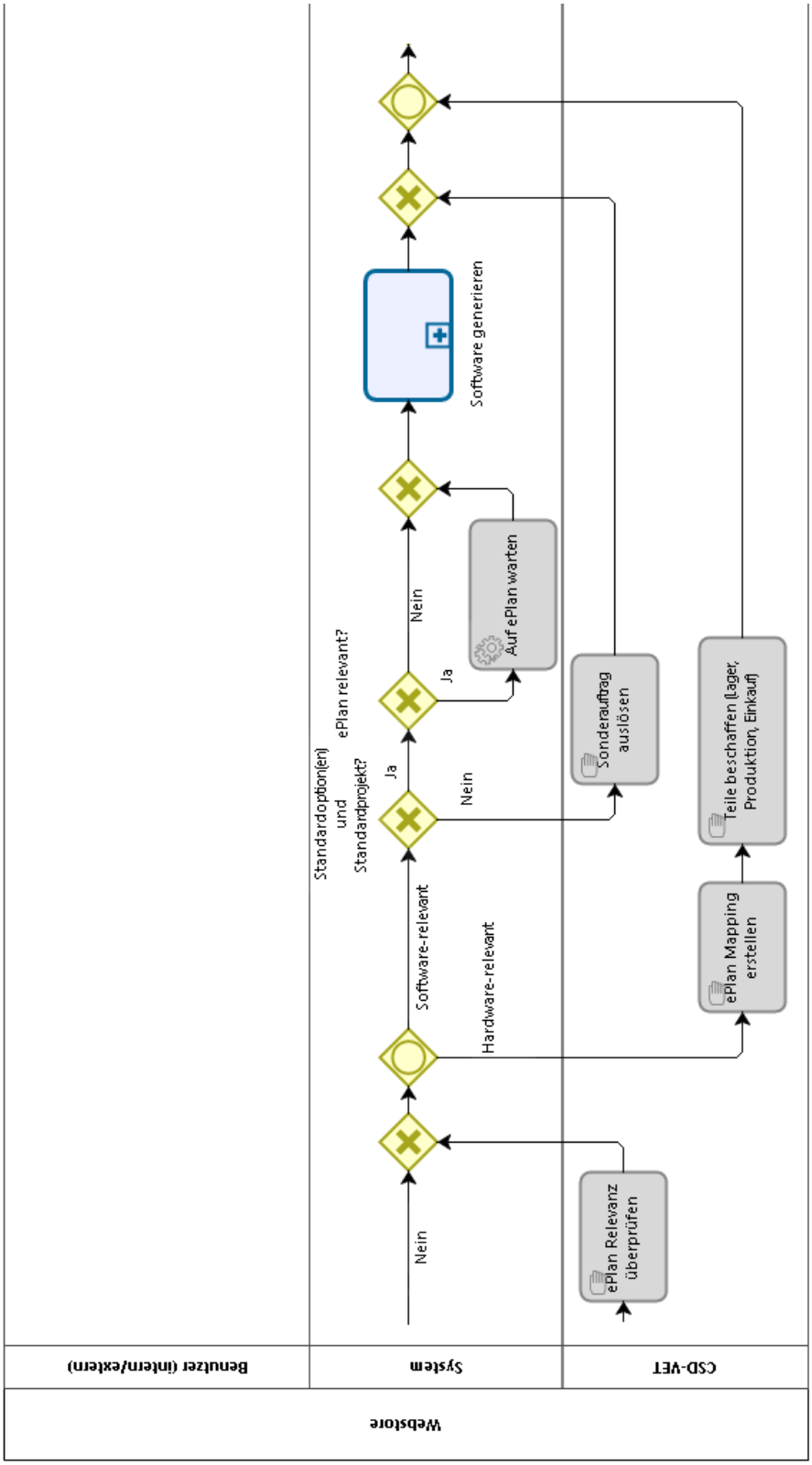

Abbildung 2.3. Hauptprozess Webstore Prozessteil 3/4

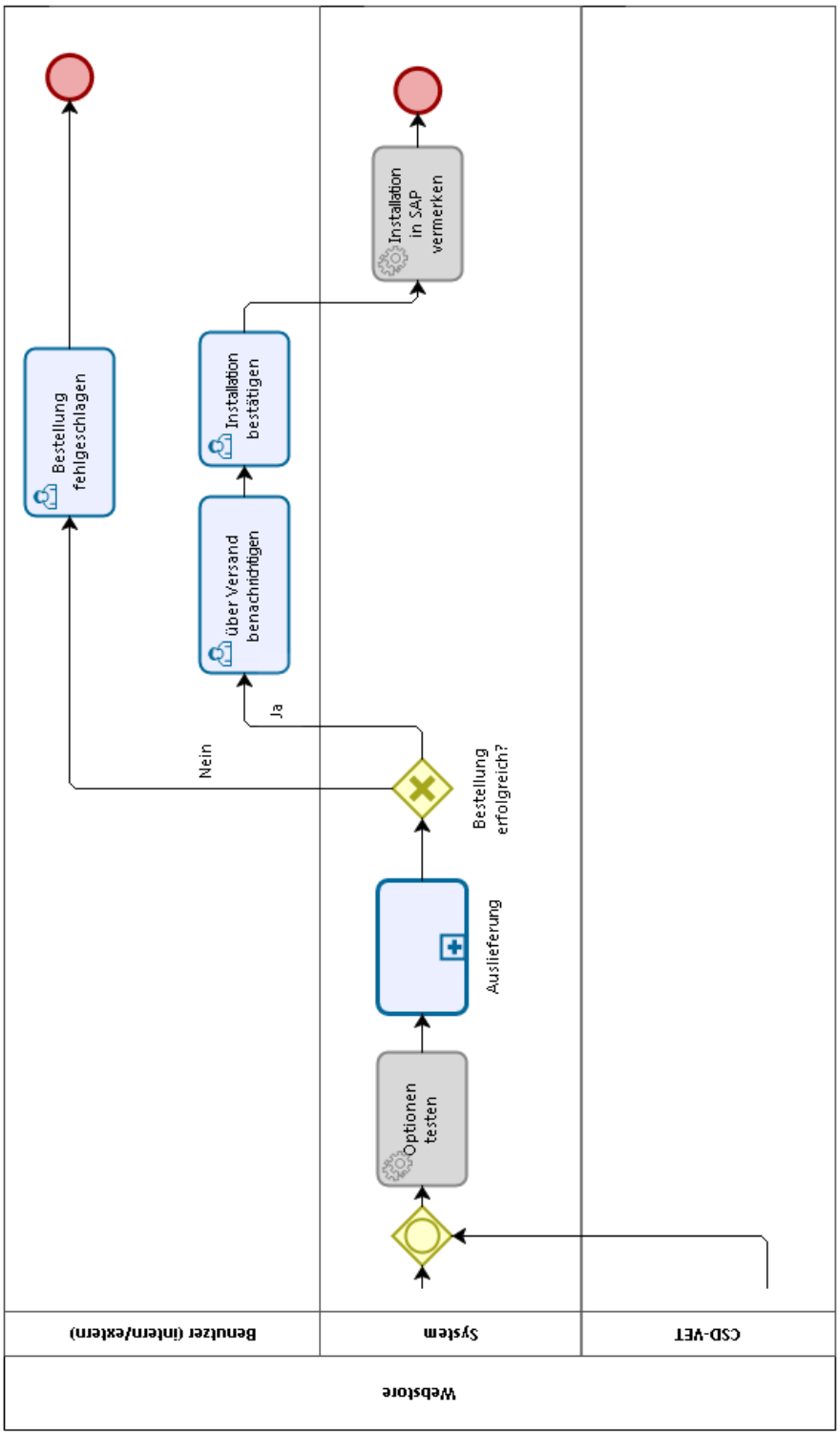

Abbildung 2.4. Hauptprozess Webstore Prozessteil 4/4

dient dazu, diverse Informationen aus den Optionen herauszufiltern und z. B. herauszufinden, ob die gesamte Auswahl der Optionen kostenpflichtig ist oder nicht.

Mittels dem Exklusiven Gateway wird nun entschieden, ob es sich bei den Benutzern um externe Kunden handelt und ob die Option kostenpflichtig ist. Handelt es sich um eine kostenfreie Option oder um einen internen Benutzer, dann ist der Teilprozes der „Angebotsgenerierung“ nicht notwendig. Es wird nur in dem Fall, dass es sich um externe Kunden und um eine kostenpflichtige Option handelt, ein Angebot generiert. Die Angebotsgenerierung ist im Kapitel 2.4 dargestellt.

In Abbildung 2.2 wird der Prozess fortgesetzt. Nach der Angebotsgenerierung bekommen die Benutzer nun den Task „Option(en) bestellen/Bestellung auslösen“ zu sehen. Über diese Aufgabe bekommen sie eine Übersicht über ihre Bestellung bzw. wenn ein Angebot generiert worden ist, wird dieses Angebot herangezogen, um den Benutzern diese Übersicht anzuzeigen. Mit diesem Task lösen die Benutzer dann eine Bestellung aus. Im nächsten Schritt „Bestellbestätigung“ erhalten die Benutzer eine Bestätigung ihrer Bestellung und Informationen über die weitere Vorgehensweise. Es erfolgt dann eine Überprüfung, ob es sich um externe Benutzer und um eine oder mehrere kostenpflichtige Optionen handelt. Ist dies der Fall, bekommt der CSD-VET, hierbei handelt es sich um die Vertriebsabteilung von Engel, die Aufgabe zugewiesen, die Bestellung zu überprüfen. Im Fall von internen Benutzern oder kostenfreien Optionen ist diese Überprüfung nicht notwendig. Es erfolgt dann eine Überprüfung, ob Hardware-relevante Optionen in der Bestellung inkludiert sind. Wenn ja, dann ist eine Überprüfung notwendig, ob der ePlan geändert werden muss. Beim ePlan handelt es sich um eine Konfigurationen, in welcher festgelegt wird, welcher Hardware Ein-/Ausgang zu welcher Software-Variable zugeordnet wird. Handelt es sich um reine Softwareoptionen, ist keine Änderung im ePlan notwendig.

Mit dem Inklusiven Gateway wird, je nachdem, ob es sich um reine Software-, reine Hardware-Optionen oder um eine Mischung beider handelt, nur einer der beiden Pfade oder beide Pfade ausgeführt. Bei Hardware-relevanten Optionen muss ein ePlan Mapping erstellt und auch die Teilebeschaffung durchgeführt werden. Die Teilebeschaffung erfolgt entweder direkt aus dem Lager, die Teile werden eingekauft oder selbst produziert. Im Falle von Software-Optionen ist es notwendig, zu überprüfen, ob es sich um Standardoptionen bzw. um ein Standardprojekt handelt oder nicht. Mit Standardoption und Standardprojekt ist gemeint, dass die Software bzw. die Spritzgussmaschine, auf welcher die neuen Optionen installiert werden sollen, keine kundenspezifischen Anpassungen hat. Wenn es sich nicht um den Standard handelt, muss ein Sonderauftrag ausgelöst und bearbeitet werden. Bei Standardprojekten muss im Falle, dass das Projekt ePlan-relevant ist, gewartet werden bis der ePlan fertig erstellt worden ist, bevor mit dem Software-Generieren begonnen werden kann. Bei nicht ePlan-relevanten Projekten kann gleich mit dem Software-Generieren, siehe Kapitel 2.5, begonnen werden. Nach der Generierung der Software wird der Prozess wieder mit dem Hardware-relevanten Pfad synchronisiert. Wurde ein Sonderauftrag ausgelöst, wird dieser anstelle des Software-Generierens durchgeführt und das Ergebnis dieses Tasks ist ebenfalls ei-

ne generierte Software. Jedoch handelt es sich hierbei um eine kundenspezifische Spezialsoftware.

Der letzte Teil des Hauptprozesses wird in Abbildung 2.4 dargestellt. Nachdem die Software erstellt und die Hardware produziert, eingekauft oder aus dem Lager besorgt worden ist, soll ein automatischer Test auf einem Simulator durchgeführt werden. Dieser Test soll dazu dienen, zu gewährleisten, dass die neuen Optionen auch lauffähig sind. Nach den Tests erfolgt dann die Auslieferung der Optionen an den Benutzer. Details dazu sind im Kapitel 2.6 erklärt. Nach der Auslieferung wird überprüft, ob die Bestellung komplett erfolgreich abgearbeitet und versandt worden ist. Hat es irgendwo einen Fehler oder ein Problem gegeben, so werden die Benutzer darüber informiert, dass ihre Bestellung fehlgeschlagen ist. Bei einer erfolgreichen Bestellung werden die Benutzer benachrichtigt, dass ihre Bestellung versandt worden ist. Der letzte Schritt, den die Benutzer nach dem Erhalten der Bestellung durchführen müssen, ist zu bestätigen, dass die Optionen auf der Maschine installiert worden sind. Dies ist notwendig, damit Engel dann die Installation in SAP vermerken kann, um einen aktuellen Stand über die Maschine zu haben. Es ist wichtig, immer einen aktuellen Stand zu haben, denn ansonsten ist nicht klar welches Backup für die Maschine aktuell gültig ist.

2.2 Teilprozess: Sicherheitsüberprüfung durchführen

Im Teilprozess der Sicherheitsüberprüfung, welcher in Abbildung 2.5 ersichtlich ist, wird sichergestellt, dass nur berechtigte Personen Zugriff auf den Prozess haben. Nachdem die Benutzer eine Fabrikationsnummer eingegeben haben, wird dieser Teilprozess ausgelöst. Es wird zwischen externen und internen Benutzern unterschieden. Im Falle von externen Benutzern erfolgt eine Überprüfung, ob die eingegebene Fabrikationsnummer der Kundennummer des Benutzers zugeordnet ist. Somit kann überprüft werden, ob diese Spritzgussmaschine diesem Benutzer gehört. Handelt es sich um interne Benutzer, also Engel-Mitarbeiter, so wird überprüft, ob die SAP-Logindaten noch im Active Directory existieren. Diese Überprüfung ist notwendig, damit sichergestellt werden kann, dass Benutzer das System nicht mehr verwenden können, nachdem sie die Firma Engel verlassen haben. Wenn die Überprüfung fehlgeschlagen ist, werden die Benutzer darüber informiert und der Prozess endet an dieser Stelle. Bei einer erfolgreichen Überprüfung wird der Hauptprozess fortgesetzt.

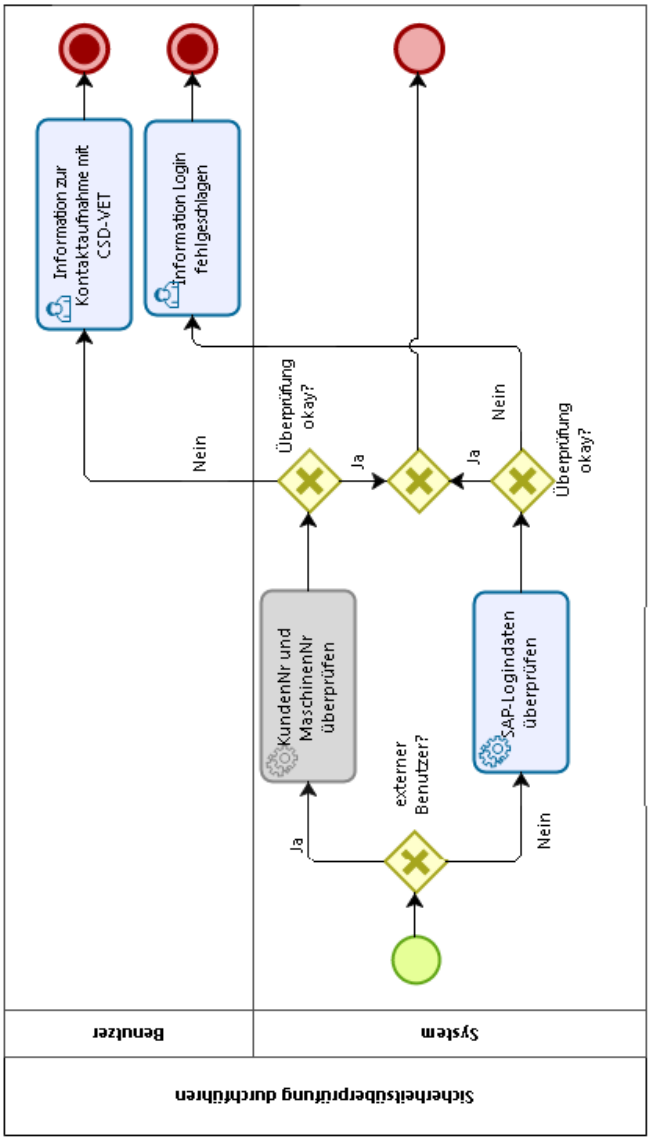

Abbildung 2.5. Teilprozess Sicherheitsüberprüfung

2.3 Teilprozess: Optionsliste generieren

Bei dem Teilprozess „Optionsliste generieren“, siehe Abbildung 2.6, handelt es sich um Arbeitsschritte, die vom System im Hintergrund ausgeführt werden. Es gibt zwei parallele Pfade, die gleichzeitig ausgeführt werden können. In einem Pfad werden zwei Webservices aufgerufen. Der eine Webservice dient dazu, die bereits installierten Optionen auszulesen und zurückzuliefern. Der andere liefert eine Liste von möglichen Zusatzoptionen, d.h. eine Liste von Optionen, die auf dieser Maschine hinzuinstalliert werden können. Im zweiten Pfad wird eine Preisliste von der Vertriebsabteilung eingelesen. Im Task „Optionsliste generieren“ werden alle diese Informationen und Listen dann verknüpft und zusammengefügt.

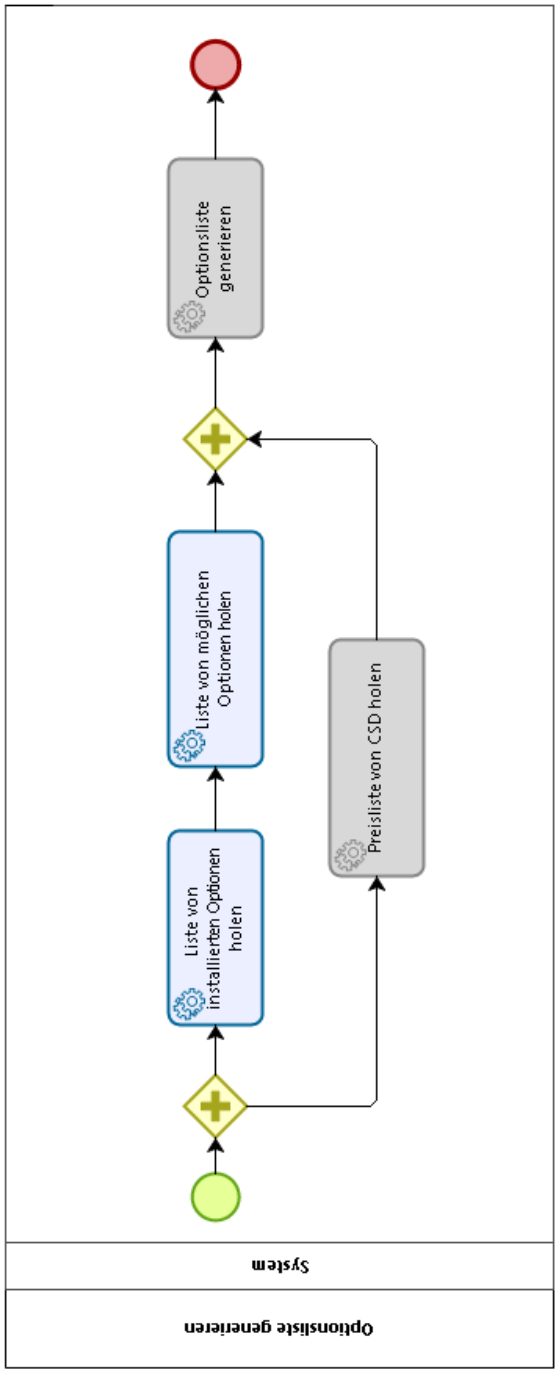

Abbildung 2.6. Teilprozess Optionsliste generieren

2.4 Teilprozess: Angebot generieren

Mit dem in Abbildung 2.7 ersichtlichen Teilprozess wird die Erstellung eines Angebots abgebildet. Wie schon im Hauptprozess, siehe Kapitel 2.1, erwähnt, wird ein Angebot nur dann generiert, wenn es sich um kostenpflichtige Optionen handelt, die von externen Benutzern ausgewählt worden sind. Mit dem Exklusivem Gateway wird überprüft, ob zusätzliche Schritte notwendig sind oder ob gleich eine Angebotsanfrage gestellt werden kann. Im Fall, dass der Preis und/oder die Lieferzeit zu einer Option nicht vorhanden ist, müssen die Kunden eine Preisanfrage stellen. Diese Preisanfrage wird dann von der Vertriebsabteilung bearbeitet. Wenn es sich um keine Standardoptionen oder um einen Sonderwunsch handelt, so müssen die Kunden diesbezüglich eine Anfrage stellen. Diese Sonderwunschanfrage wird dann auch von der Vertriebsabteilung bearbeitet. Alle drei Pfade werden dann wieder zusammengeführt und es sollten nun alle benötigten Informationen vorhanden sein, damit die Kunden eine Angebotsanfrage erstellen und absenden können. Diese wiederum wird wieder von der Vertriebsabteilung bearbeitet und es wird ein Angebot angelegt. Dieses Angebot wird dann im Hauptprozess für die Bestellübersicht verwendet.

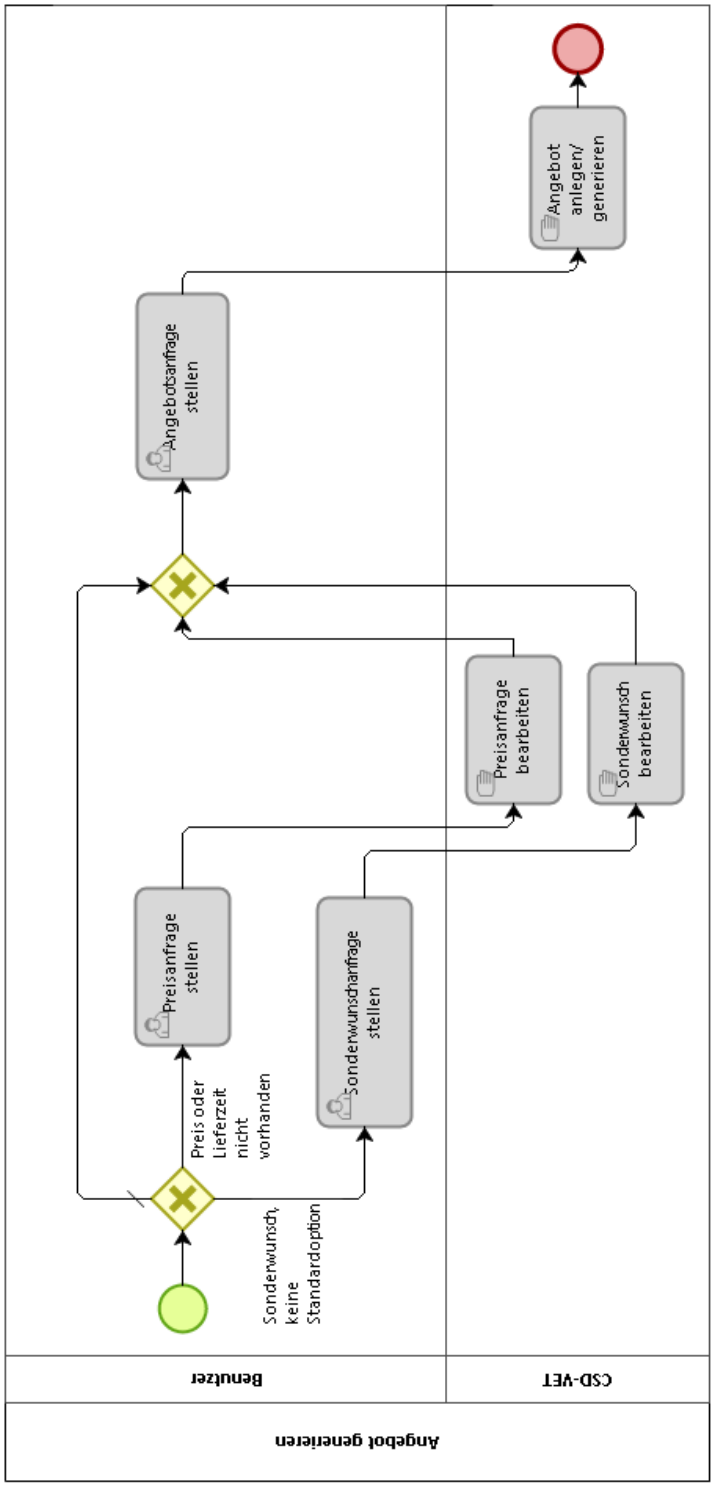

Abbildung 2.7. Teilprozess Angebot generieren

2.5 Teilprozess: Software generieren

Der Teilprozess „Software generieren“ wird in Abbildung 2.8 und 2.9 dargestellt. Nachdem im Hauptprozess die Bestellbestätigung akzeptiert worden ist und im System bzw. auch in der Vertriebsabteilung diverse Überprüfungen und Schritte, wie die Überprüfung der Bestellung, stattgefunden haben, wird dieser Teilprozess aufgerufen.

Hier gibt es zu dem Hauptpfad, der mit dem Task „Finale Sicherung aus SAP laden“ in Abbildung 2.8 startet, einen optionalen Pfad, welcher nur ausgeführt wird, wenn mindestens eine der ausgewählten Optionen dokumentations-relevant ist. Dies bedeutet, dass die Dokumentation für diese Spritzgussmaschine geändert und angepasst werden muss. In dem Pfad der Dokumentations-Generierung wird zuerst der Auftrag für die Dokumentation freigegeben. Es folgt dann eine Prüfung, ob es sich um sicherheitsrelevante Optionen handelt, das heißt, ob durch Hinzufügen dieser Optionen irgendwelche besonderen Sicherheitsrisiken entstehen, welche in die Dokumentation aufgenommen werden müssen. Ist dies der Fall so ist es notwendig, dass die Dokumentation auch in Papierform erstellt werden muss. Die Dokumentation wird zusätzlich immer als elektronisches eHelp erzeugt.

Der Hauptpfad wird immer durchlaufen und startet damit, dass die finale Sicherung aus SAP geladen wird. Dieser Schritt sowie alle folgenden Schritte im Hauptpfad werden durch Webservices abgearbeitet. Im zweiten Service-Task wird die Projektumgebung vorbereitet, welche für die Software-Generierung benötigt wird. Als nächstes ist es notwendig, die Optionsliste aus der finalen Sicherung zu laden. Diese Optionsliste enthält die Informationen aller Optionen, die in der vorherigen Installation auf der Maschine installiert worden sind. Es wird nun diese Optionsliste um die neuen Optionen ergänzt. Mit dieser Optionsliste werden dann die Source Dateien, welche für die einzelnen Optionen benötigt werden, aus dem SAP geladen.

In Abbildung 2.9 geht es mit dem Schritt „(IMM) Projekt erstellen, kompilieren“ weiter. Es wird damit, wie der Name des Task schon sagt, das Projekt erstellt und dann kompiliert, dies wird auch durch einen Webservice erledigt. Wurde der Nebenpfad mit der Erstellung der Dokumentation durchgeführt, wird dieser hier wieder mit dem Hauptpfad synchronisiert. Im nächsten Schritt „Target generieren“ wird ein Update-Paket erstellt, dieses Paket enthält nun auch die neuen Optionen. Die letzten drei Schritte des Software-Generierens können dann parallel ausgeführt werden. Es werden die finale und die Source Sicherung abgespeichert, welche dann als Backup dienen, und es wird das Update-Paket auf einen FTP Server hochgeladen.

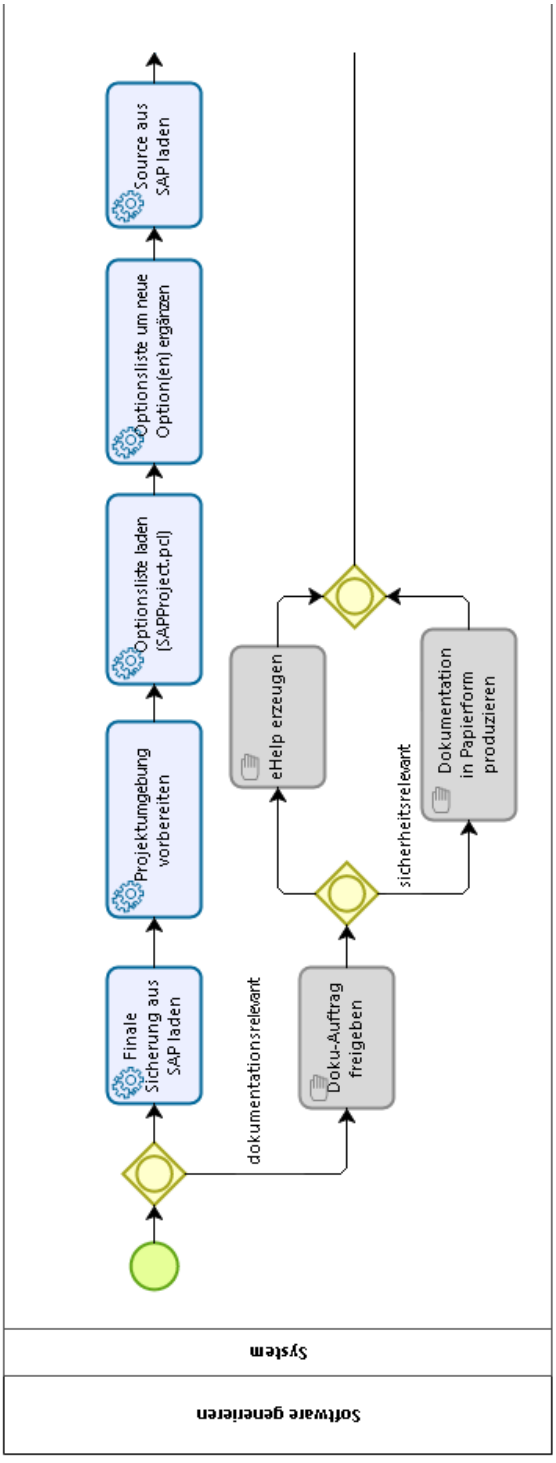

Abbildung 2.8. Teilprozess Software generieren Prozessteil 1/2

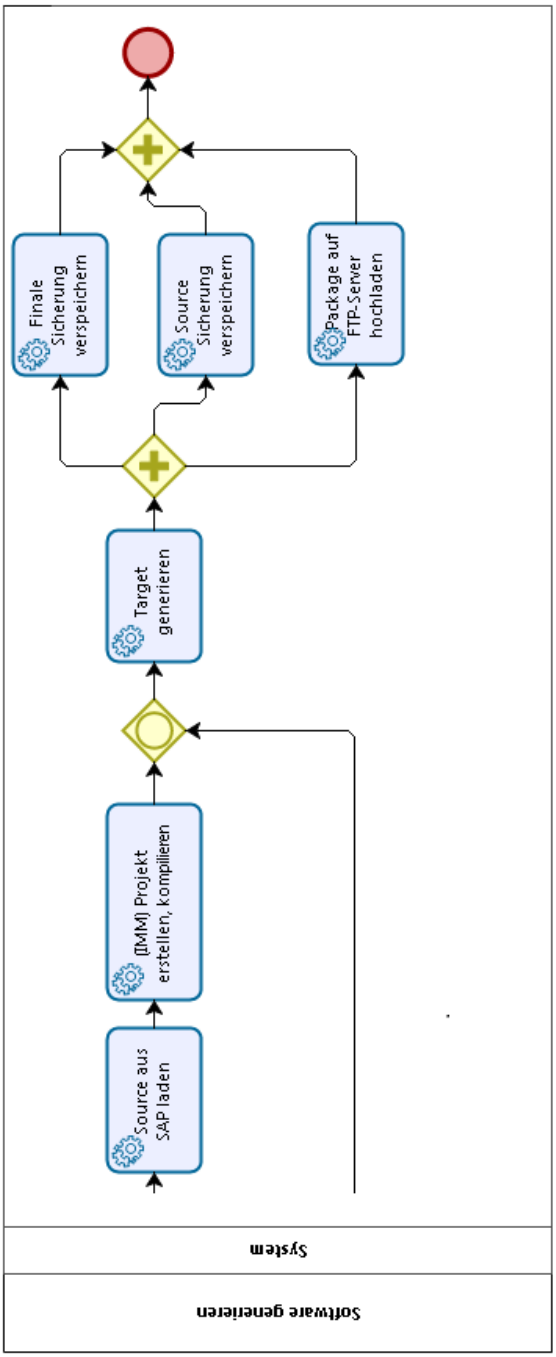

Abbildung 2.9. Teilprozess Software generieren Prozessteil 2/2

2.6 Teilprozess: Auslieferung

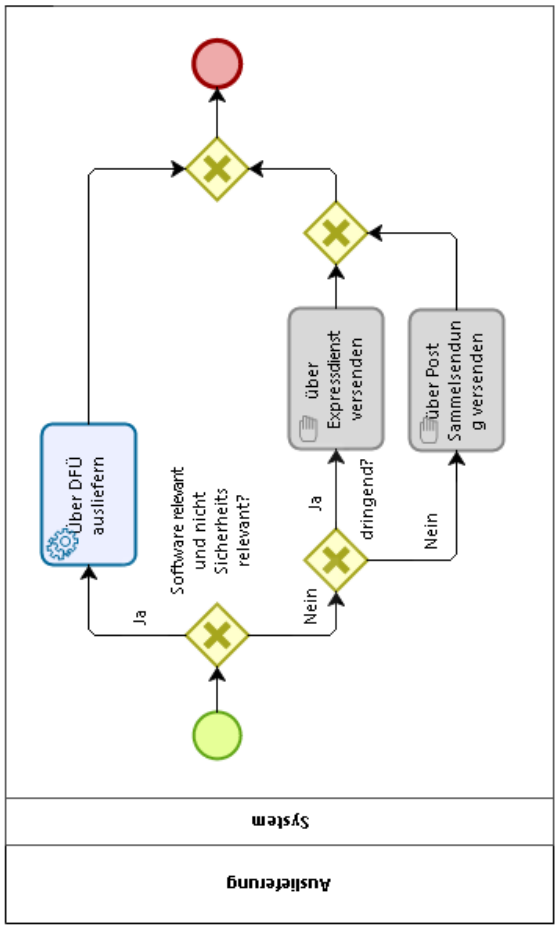

Abbildung 2.10. Teilprozess Auslieferung

Nachdem die Optionen im Hauptprozess durch den Service-Task „Optionen testen“ erfolgreich getestet wurden, startet der Teilprozess „Auslieferung“, dieser ist in Abbildung 2.10 zu sehen. Es wird hier an erster Stelle überprüft, ob es sich um Software-relevante und nicht sicherheitsrelevante Optionen handelt. Wenn dies der Fall ist, dann wird die Software über Datenfernübertragung (DFÜ) ausgeliefert. Dies bedeutet, dass die Benutzer einen Link erhalten, über welchen sie sich das Update-Paket herunterladen können. Im anderen Fall, also wenn es sich um Hardware-Optionen und/oder sicherheitsrelevante Optionen handelt, dann wird

eine physische Auslieferung durchgeführt. Bei einer physischen Auslieferung wird unterschieden, ob es ein dringender oder ein nicht dringender Auftrag ist. Je nachdem erfolgt die Versendung über den Expressdienst oder über eine Post-Sammelsendung.

3

Werkzeugauswahl

Für die Umsetzung bzw. die Automatisierung des Prozesses wird ein Business Prozess Management (BPM) Werkzeug verwendet. In diesem Kapitel werden WErkzeuge aus den Bereichen Business Prozess Management Systeme (BPMS) und Business Prozess Automation (BPA) betrachtet und anhand von Kriterien evaluiert. Die Evaluierung setzt sich aus drei Elementen und zwei Schritten zusammen, da es sehr viele verschiedene Anbieter von BPM Werkzeugen gibt.

Um eine Liste von BPM Werkzeugen zu bekommen, wurden Werkzeuge ausgewählt, die in [Dumas et al., 2013, Kapitel 9] angeführt sind und als geeignet erschienen. Es wurden dann noch einige Werkzeuge durch eine Internetrecherche ergänzt. In weiterer Folge wurden Werkzeuge aufgenommen, welche von diversen Kontaktpersonen, die im BPM-Umfeld tätig sind, empfohlen wurden.

Wie schon erwähnt, findet die Evaluierung in zwei Schritten statt. Im ersten Schritt wird eine Übersicht über verschiedene Werkzeuge erstellt. Es werden Kriterien definiert und deren Ausprägungen für die einzelnen Werkzeuge erhoben. Diese Kriterien sollen dazu dienen, um eine erste grobe Auswahl durchführen zu können. Im zweiten Schritt sollen die ausgewählten Werkzeuge detaillierter analysiert, in der Praxis getestet und anhand von zusätzlichen Kriterien evaluiert werden. Um eine endgültige Entscheidung bezüglich eines Werkzeugs zu treffen, wird dann eine Nutzwertanalyse herangezogen.

3.1 Werkzeugübersicht und Vorauswahl

Die Liste von Werkzeugen ist folgendermaßen entstanden: Aus [Dumas et al., 2013] wurden die Werkzeuge Bonitasoft, Bizagi BPM Suite, Perception Software BPMOne, YAWL, Oracle SOA Suite und der IBM Business Process Manager aufgenommen. Weitere Werkzeuge wurden nach Emmpfehlungen von Kntaktpersonen aufgenommen. Hierzu wurden Personen von der Johannes Kepler Universität befragt, welche mit BPM zu tun haben. Weiters wurden Personen auf der Plattform XING kontaktiert, welche im BPM-Umfeld tätig sind. Durch ein Gespräch mit einem Universitätsmitarbeiter der Johannes Kepler Universität wurde Webratio und

Signavio zu der Liste hinzugefügt. [BPMasters, 2015] hat das Werkzeug Camunda empfohlen. Von [Open-Text, 2015] wurde Activiti und von [MDBA, 2015] wurde GBTec BIC Platform empfohlen. Durch eine Internetrecherche konnten noch Werkzeuge wie SemTalk, IBM Blueworks live, Oracle BPM Suite, Pega und K2 BPM Software aufgenommen werden.

3.1.1 Kriterien

Es werden nun die Kriterien für den ersten Schritt der Vorauswahl aufgelistet und kurz erklärt:

- **BPMN Support:** Mit diesem Kriterium wird angegeben, ob das Werkzeug die Business Process Modeling Notation (BPMN) unterstützt.
- **Simulation:** Kann mit diesem Werkzeug ein Prozess simuliert werden?
- **Ausführen (Execution):** Bietet das Werkzeug die Möglichkeit, den Prozess lauffähig und ausführbar zu machen?
- **Monitoring:** Ermöglicht das Werkzeug die Beobachtung und Überwachung der laufenden Prozesse?
- **Verwendung von externen Services:** Erlaubt das Werkzeug den Aufruf von externen Services?
- **Import/Export:** Erlaubt das Werkzeug den Import und Export des Prozesses, zum Beispiel im Format XPDL? Dieses Kriterium wird eher benachteiligt behandelt, da sich herausgestellt hat, dass zwar viele Werkzeuge diverse Import- und Export-Funktionalitäten bieten, es jedoch in den meisten Fällen zu Problemen führt, wenn man den Prozess von einem Werkzeug in ein anderes importieren will, da die Hersteller unterschiedliche Formate verwenden. Der Vollständigkeit halber werden die Formate jedoch auch aufgelistet. Hinzuzufügen ist, dass hier Formate wie *.pdf oder *.jpg nicht betrachtet werden, sondern nur Formate, welche theoretisch zum Austausch von Prozessmodellen verwendet werden könnten.
- **Kosten:** Mit diesem Kriterium soll ermittelt werden, ob das Werkzeug etwas kostet oder nicht. Für diese Pilotstudie lautet die Anforderung, dass das Werkzeug kostenlos (Freeware) sein soll, zumindest soll es möglich sein, einen lauffähigen Prototypen zu erstellen, ohne etwas dafür bezahlen zu müssen. Wenn in späterer Folge dann das Pilotprojekt erweitert werden soll, dann ist es kein Problem, wenn Kosten anfallen.

Die folgenden Kriterien sind zwingend zu erfüllen: BPMN Support, Ausführung (Execution) und externe Services. Wenn bei einem Werkzeug entdeckt wird, dass eines dieser Kriterien nicht erfüllt werden kann, dann scheidet dieses Werkzeug automatisch aus und die anderen Kriterien sind nicht mehr unbedingt erhoben worden. Die Erhebung dieser Kriterien erfolgt anhand einer Literaturrecherche bzw. wird anhand der Hersteller-Webseiten ermittelt. Nachfolgend werden nun einige verschiedene Werkzeuge anhand der obigen Kriterien analysiert. Im Unterkapitel „Vorauswahl“ wird das Ergebnis der Analyse in einer Tabelle dargestellt.

3.1.2 Werkzeuge

Bonitasoft

Laut [Bonitasoft, 2015] wird in der Modellierungssprache BPMN 2.0 gearbeitet. Auch alle anderen Kriterien Simulation, Ausführung, Monitoring und externe Services werden erfüllt. In der kostenlosen Community Edition werden die Grundfunktionalitäten für die erwähnten Kriterien angeboten. Es gibt auch noch bessere Editionen, welche Zusatzfunktionalitäten wie z. B. User Profil Management und Anpassung des Designs erlauben. Die anderen Editionen sind nicht kostenlos, bieten jedoch etwas mehr Funktionalität. Der Preis für diese ist jedoch leider nicht ersichtlich. Diesbezüglich musste eine Anfrage an Bonitasoft gestellt werden. Die Anfrage wurde nach ein paar Tagen beantwortet.

Der Preis wird grundsätzlich in zwei Kategorien eingeteilt. Die erste Kategorie beinhaltet den Gold Support und die zweite Kategorie den Platinum Support. Der Unterschied liegt z. B. darin, dass der Platinum Support 24/7 verfügbar ist und der Gold Support nur werktags von 09:00 Uhr bis 17:00 Uhr oder dass die Reaktionszeit kürzer ist. Eine weitere Untergliederung des Preises erfolgt dann bezüglich der Subscription Packs, welche den oben beschriebenen Editionen entsprechen. Des Weiteren steigt der Preis mit der Anzahl von Fällen, die abgearbeitet werden müssen, angefangen von 10.000 Fällen/Jahr bis hin zur unlimitierten Anzahl von Fällen. Die Preisspanne reicht von € 15.000 für den Gold Support, der niedrigsten (Teamwork) Edition für 10.000 Fällen/Jahr, bis hin zu € 135.000 für den Platinum Support, der höchsten (Performance) Edition für eine unlimitierte Anzahl von Fällen.

Bizagi

[Bizagi, 2015] verwendet auch BPMN als Modellierungssprache. Es unterstützt auch die Kriterien Simulation, Ausführung, Monitoring und das Aufrufen von externen Services. Die Bizagi Suite setzt sich aus drei Teilen zusammen: der Modeler, das Studio und die Engine. Die ersten beiden sind kostenlos in der Community Edition inkludiert. Der Modeler wird, wie der Name schon vermuten lässt, dafür verwendet, den Prozess zu modellieren. Mit dem Studio wird das Prozessmodell in eine lauffähige Applikation verwandelt. Die Engine wird dann dazu verwendet, den lauffähigen Prozess produktiv im Unternehmen einzusetzen. Die Engine kostet pro Arbeitsplatz € 615. Es gibt auch die Möglichkeit, eine größere Menge von Benutzerlizenzen zu erwerben. Die Lizenz kann auch um € 239 auf ein Jahr begrenzt gekauft werden.

SemTalk

Ein auf Visio aufbauendes Modellierungswerkzeug ist [Semtation, 2015], welches BPMN als Modellierungssprache anbietet. Es ermöglicht auch die Simulation eines Prozesses. Die Möglichkeit, den Prozess auszuführen oder zu überwachen, gibt es leider nicht. Jedoch kann eine Business Process Execution Language (BPEL)

Datei generiert werden, welche zum Beispiel auf einem BizTalk Server ausgeführt werden kann. Auf der Homepage kann eine 30 Tage Testversion heruntergeladen werden, jedoch findet man keine Informationen darüber, was nach diesen 30 Tagen passiert. Der Hersteller gibt an, dass verschiedene Formate, die auf XML basieren, importiert und exportiert werden können.

Signavio

Auch [Signavio, 2015] ist ein reines Modellierungswerkzeug. Für die Modellierung wird BPMN verwendet und es gibt auch eine Komponente, welche die Simulation des Prozesses ermöglicht. Wie bei SemTalk gibt es hier auch eine 30 Tage Testversion, danach ist das Werkzeug kostenpflichtig. Es gibt drei Editionen, für die jeweils die Preise pro User und Monat angegeben sind. Die Preise reichen von rund € 82 für die niedrigste Edition bis hin zu €200 für die höchste Edition. Signavio bietet Mashup- und REST-APIs, um den Prozesseditor in BPMS Werkzeuge wie Camunda einzubinden.

Webratio

Das Werkzeug [Webratio, 2015] bietet zwei Funktionalitäten, welche miteinander Verknüpft werden können. Zum einen gibt es die Funktionalität, einen Prozess in BPMN zu modellieren und lauffähig zu machen, und zum anderen eine Möglichkeit, mit der standardisierten Modellierungssprache Interaction Flow Modeling Language (IFML) das Userinterface zu erstellen. Diese beiden Funktionen können in der Webratio Web Platform Edition miteinander verbunden werden.

Dieses Werkzeug ist Eclipse-basiert und ermöglicht die Erstellung des Prozesses mittels BPMN sowie die Ausführung, das Monitoring und das Einbinden von externen Services. Ob Webratio die Simulation des Prozesses zulässt, konnte nicht herausgefunden werden. Es gibt die Web Platform Edition, die Community und die Professional Edition. Die Kosten belaufen sich auf circa € 200 im Monat und pro Person. Um den Preis für die Enterprise Edition zu eruieren, wurde eine Anfrage an den Kundenservice von Webratio gestellt. Die Enterprise Edition kostet € 9.100 pro Jahr und Person. Diese Edition inkludiert einen Support und Upgrades.

BIC Platform GBTec

Alle festgelegten Kriterien des ersten Schrittes werden von [GBTec, 2015] erfüllt. Dieses Werkzeug ist eine Plattform, welche aus mehreren Werkzeugen besteht, wie z. B. ein Modellierungserkzeug und ein Prozessportal. Das Modellierungswerkzeug ist Browser-, aber nicht Cloud-basiert und die Daten werden auf dem lokalen Rechner abgelegt. Es wird auch nicht zwingend eine Internetverbindung benötigt, sondern es kann auch Offline gearbeitet werden. Es gibt eine kostenlose Variante des Modellierungswerkzeugs, das in der Plattform vorhanden ist, jedoch können alle anderen Werkzeuge, aus dem die Plattform besteht, nicht kostenlos erworben werden. Bezüglich der Kosten müsste eine Anfrage gestellt werden. Da

die Anforderung jedoch ist, dass der Prozess als Prototyp mit einem kostenlosen Werkzeug erstellt werden soll, wird hier keine Anfrage bezüglich des Preises gestellt. Laut [MDBA, 2015] ist es auch sehr aufwendig und es benötigt einiges an Zeit, das Werkzeug zu installieren und zu konfigurieren.

Camunda

Der Support von BPMN ist von [Camunda, 2015] gewährleistet. Dieses Werkzeug ist Eclipse-basiert und es bietet ein sehr flexibles Framework an. Die Benutzeroberfläche kann z. B. mit JSF oder jedem anderen beliebigen Java GUI-Framework implementiert werden. Es gibt auch die Möglichkeit den Prozess mittels REST anzusprechen und eine ganz andere Client Technologie zu verwenden. Es gibt eine kostenlose Version von Camunda und eine Enterprise Version. Grundsätzlich beinhaltet die kostenlose Version schon die wichtigsten Funktionalitäten und ist zudem Open Source.

Die Enterprise Version liefert zusätzliche Optionen wie z.B. 24/7 Support, Wartung durch Patches, das Laufen des Prozesses auf einem IBM Websphere Applikation Server oder Oracle WebLogic Application Server. Der Preis der Enterprise Version muss leider auch angefragt werden. Es wurde diesbezüglich eine Anfrage geschickt. Die Kosten werden für die Enterprise Version auf einer Payper-Use Basis berechnet. Dies wird anhand der tatsächlich durchlaufenden Prozessmodelle gemessen. Für die Berechnung werden die aktivierten BPMN Flusselemente herangezogen und es wird berücksichtigt, wie komplex die Prozesse sind. Camunda bietet keinen Formular Designer, jedoch können Formulare automatisch generiert werden, indem man für Tasks Formular Felder definiert. Anhand dieser Felder wird dann automatisch eine HTML Seite generiert. Es können aber auch manuell HTML-Seiten erstellt werden, um die Formulare individuell zu gestalten.

Activiti

Wie Camunda ist auch [Activiti, 2015] Java-basiert, es erfüllt die Anforderungen BPMN Modellierung, Ausführen und Monitoring des Prozesses sowie das Einbinden von externen Services. Des Weiteren ist das Werkzeug kostenlos und auch Open Source. Ein Nachteil dieses Werkzeugs ist, dass es keinen grafischen Editor gibt, um Formulare bzw. Screens für den User zu erstellen. Es gibt zwar von [Alfesco-Activiti, 2015] eine Plattform, welche das unterstützt, diese steht jedoch kostenlos nur in einer 30 Tage Testversion zur Verfügung. Bezüglich des Preises müsste wiederum eine Anfrage gestellt werden.

Sonstige Werkzeuge

Bei einigen der Werkzeugen wurde festgestellt, dass diese die Modellierungssprache BPMN nicht unterstützen. Diese Werkzeuge wurden dann bezüglich der anderen Kriterien nicht mehr genauer betrachtet, da sie ohnehin nicht zur Umsetzung in Frage gekommen wären. In dieser Kategorie gibt es folgende Werkzeuge: Pega von Pegasystems [Pegasystems, 2015], BPMOne von Perceptive

Software [Perceptive-Software, 2015], YAWL [Pegasystems, 2015] und K2 BPM [K2, 2015].

Andere Werkzeuge wiederum sind zu schwergewichtig, um sie in die Werkzeugauswahl aufzunehmen. Mit zu schwergewichtig ist hier gemeint, dass das Werkzeug nicht in Frage kommt, da es für den zweiten Schritt der Evaluierung zu aufwendig wäre, das Werkzeug zu installieren und zu konfigurieren. In diese Kategorie fallen die beiden Oracle Werkzeuge SOA Suite [Oracle, 2015b] und Business Process Management Suite [Oracle, 2015a] sowie die beiden IBM Werkzeuge Business Process Manager [IBM, 2015b] und Blueworks [IBM, 2015a]. Allein schon bei der Erhebung der Kriterien für diese Werkzeuge kann man feststellen, wie schwergewichtig diese Werkzeuge sind, da es sehr schwierig ist, Informationen über die Kriterien zu finden. Eine Anfrage an IBM, um Informationen zu bekommen, ist leider ohne jegliche Rückmeldung geblieben.

3.1.3 Vorauswahl

Tabelle 3.1 zeigt eine Übersicht der oben beschriebenen Werkzeuge. Die Werkzeuge sind grob nach ihrer Relevanz bzw. nach der empfundenen Tauglichkeit für die Umsetzung gereiht, wobei hinzuzufügen ist, dass es nicht bedeutet, dass das Werkzeug an erster Stelle besser geeignet ist als das an zweiter Stelle. Es wurde jedoch festgestellt, dass die zuerst aufgelisteten Werkzeuge sich besser eignen als die am Ende der Liste. Ein „+“ kennzeichnet, dass ein Werkzeug ein Kriterium erfüllt, wohingegen ein „-“ angibt, dass es das Kriterium nicht erfüllt. Die leeren Zellen konnten entweder nicht erhoben werden oder wurden nicht erhoben, da sich schon im Vorhinein herausstellte, dass dieses Werkzeug nicht geeignet ist.

Die an den ersten vier Positionen gereihten Werkzeuge in der Tabelle 3.1 eignen sich am besten für die Umsetzung. Die vier Werkzeuge Bonitasoft, Bizagi, Webratio und Camunda werden im Kapitel 3.2 im zweiten Evaluierungsschritt noch näher betrachtet. Das Werkzeug Activiti und die BIC Platform würden sich wahrscheinlich auch für die Umsetzung eignen. Diese beiden Werkzeuge werden jedoch nicht betrachtet. Der Grund dafür ist, dass bei Activiti kein Formular Editor in der kostenlosen Version enthalten ist. Bei der BIC Platform ist nur der Designer, das Werkzeug, um den Prozess abzubilden, kostenlos vorhanden. Soll der Prozess ausführbar gemacht werden, muss das ganze Werkzeug gekauft werden. Die restlichen Werkzeuge wie z. B. Signavio, Oracle BPM Suite oder K2 BPM Software eignen sich aus einem der folgenden Gründe nicht:

- Das Werkzeug ist zu schwergewichtig und es würde alleine schon zu viel Aufwand bedeuten, das Werkzeug zu installieren und zu konfigurieren.
- Das Werkzeug ist nur ein Modellierungswerkzeug, aber bietet keine Unterstützung, um den Prozess auch ausführbar zu machen.
- Das Werkzeug unterstützt die BPMN Modellierungssprache nicht.

Werkzeug	BPMN Support	Simulation	Ausführung	Monitoring	ext. Services	Kosten
Bonitasoft	+	+	+	+	+	kostenlos, bessere Editionen kosten von € 15.000 - € 135.000
Bizagi	+	+	+	+	+	Modeler & Studio sind kostenlos, die Execution Engine kostet € 615 pro Benutzerlizenz oder jährlich € 239
Webratio	+	-	+	+	+	Community Edition ist kostenlos, die Professional Edition kostet ab € 200 pro Monat und Person, die Enterprise Edition kostet € 9.100 im Jahr pro Person
Camunda	+	-	+	+	+	kostenlos, Enterprise Version kostenpflichtig (pay-per-use)
GBTec BIC	+	+	+	+	+	nur Designer ist kostenlos, Plattform Edition kostenpflichtig
Activiti	+	+	+	+	+	kostenlos (kein Formulareditor), Enterprise kostenpflichtig
Signavio	+	+	-	-	-	30 Tage Testversion
SemTalk	+	+	-	-	-	30 Tage Testversion
BPMOne	-					
YAWL	-					
Oracle SOA Suite	-					
IBM BPM	+					
IBM Blueworks	+					
Oracle BPM Suite	-					
Pega	-					
K2 BPM Software	-					

Tabelle 3.1. Taxonomie

3.2 Bewertung der Vorauswahl

In diesem Kapitel werden nun die Werkzeuge, welche durch die Vorauswahl ausgewählt wurden, bewertet. Nachfolgend werden die Kriterien für diese Bewertung bzw. für den zweiten Schritt der Werkzeugauswahl aufgelistet und kurz erklärt:

- **Datenbank:** Um den Prozessstatus und Ablauf speichern zu können, wird eine Datenbank benötigt. Auch Datenobjekte, welche im Prozess benötigt werden, müssen irgendwo abgespeichert werden. Mit diesem Kriterium wird angegeben, welche Datenbanken für das Werkzeug verwendet werden können.

- **Anpassung/Erstellung von Formularelementen:** Für die Erstellung der Benutzerinterfaces werden so genannte Formulare verwendet. Mit diesem Kriterium soll geprüft werden, ob es möglich ist, eigene Formularelemente zu erstellen oder bestehende Elemente anzupassen.
- **Anpassung des Designs:** Die Benutzerinterfaces des ausführbaren Prozesses sollen angepasst werden können. Zum Beispiel lassen sich mittels CSS die Farben, Logos, usw. an einen Firmenstandard anpassen.
- **Usability:** Dieses Kriterium gibt an, wie benutzerfreundlich das Werkzeug zu bedienen ist. Die Auswertung dieses Kriteriums kann nicht hundertprozentig objektiv erfolgen, da die Usability-Faktoren auch von vielen persönlichen Eigenschaften abhängen. Hier ist die Benutzerfreundlichkeit des Werkzeugs aus Sicht der Entwickler gemeint.
- **Erweiterbarkeit/Mächtigkeit:** Mit diesem Kriterium wird angegeben, wie mächtig ein Werkzeug ist, also wie offen das Werkzeug ist, um andere Technologien und andere Frameworks einzusetzen und einzubinden, bzw.wie starr das Werkzeug an sich ist und wie viel Spielraum es den Entwicklern gibt, um etwas umzusetzen. Als Beispiel kann hier genannt werden, ob es die Möglichkeit gibt, eine eigene Taskliste zu erstellen.

Die vier Werkzeuge aus dem Ergebnis der Vorauswahl bzw. des ersten Evaluierungsschrittes werden hier nun anhand dieser Kriterien evaluiert. Die Werkzeuge wurden installiert und es wurde je ein kleiner Beispielprozess umgesetzt.

3.2.1 Bizagi

Bizagi bietet die Möglichkeit, eine Oracle oder Microsoft SQL Datenbank zu verwenden. Das Werkzeug ist nicht Cloud-basiert. Es gibt auch die Möglichkeit, eigene Formularelemente zu erstellen. Hierfür müssen jedoch Widgets erstellt werden, welche dann eingebunden werden können. Es gibt auch schon vorgefertigte Widgets, welche verwendet werden können, wie z. B. Google Maps. Will man das Bizagi Logo durch sein eigenes Firmenlogo ersetzen, muss man dafür etwas bezahlen. Im Bereich Usability ist Bizagi sehr gut, denn man wird von der Modellierung des Prozesses bis hin zum laufenden Prozess Schritt für Schritt geführt. Um den Prozess jedoch produktiv einsetzen zu können, muss die Execution Engine gekauft werden. Mit dem gratis Studio kann der Prozess zwar lokal auf einem Rechner ausgeführt werden, jedoch kann der Prozess nicht auf einen Server gelegt werden.

3.2.2 Bonitasoft

In Bonitasoft können sehr viele verschiedene Datenbanken verwendet werden, unter anderem MS SQL, Oracle, H2, IBM DB2 und noch einige mehr. Ähnlich wie bei Bizagi können Widgets erstellt und eingebunden werden, um Elemente, die von dem Werkzeug nicht schon zur Verfügung gestellt werden, hinzuzufügen. Bonitasoft ist auch sehr benutzerfreundlich gestaltet und es ist einfach zu bedienen. Die Usability ist intuitiv und Bonitasoft lässt den Benutzern nicht viel Spielraum,

um Fehler zu machen. Ein Nachteil dieses Werkzeugs ist jedoch, dass in der kostenlosen Version nicht die Möglichkeit besteht, das Design der Userinterfaces anzupassen.

3.2.3 Webratio

Die Community Edition von Webratio kann von der Homepage gratis heruntergeladen und installiert werden. Der einzige Nachteil der freien Community Edition ist, dass es Cloud-basiert ist, somit kann das Deployment nur in der Cloud durchgeführt werden. Die Enterprise Version läuft auch in der Cloud, bietet aber im Gegensatz zu den anderen Editionen die Möglichkeit, die Applikation lokal in der eigenen Umgebung laufen zu lassen. Jedoch ist diese nicht kostenlos. Webratio ist sehr offen und mächtig, denn es können Datenbanken wie Oracle, MS SQL, IBM DB2, MySQL, Derby und noch einige mehr verwendet werden. Es gibt auch eine ODBC Schnittstelle, mittels dieser kann nahezu jede beliebige Datenbank verwendet werden.

Für die Erstellung der Formularseiten wird in Webratio die Sprache „The Interaction Flow Modeling Language“ (IFML) verwendet. Mit dieser standardisierten Sprache werden Websites erstellt. Um die Seiten individuell zu gestalten und an Unternehmensfarben anzupassen, gibt es die Möglichkeit, ein so genanntes Style Project zu erstellen. Dieses Project ermöglicht es dann, mittels HTML und CSS das Design individuell zu gestalten.

Webratio ist ein mächtiges Werkzeug, da es die oben genannten Möglichkeiten bietet. Bezüglich Usability kann gesagt werden, dass es, nachdem man sich etwas mit dem Werkzeug beschäftigt hat, einfach zu verwenden und zu bedienen ist.

3.2.4 Camunda

Um Camunda verwenden zu können, muss die Camunda BPM Platform heruntergeladen und installiert werden. Zusätzlich muss noch eine Eclipse Version von der Camunda Homepage heruntergeladen werden, welche den Camunda BPMN 2.0 Modeler enthält. Camunda ist nicht Cloud-basiert, es verwendet jedoch Maven, um Projekt-Abhängigkeiten zu definieren. Standardmäßig verwendet Camunda eine H2 Datenbank. Durch das Ändern der Verbindungsdaten in der Konfigurationsdatei des Servers können aber auch andere Datenbanken verwendet werden. Hierfür gibt es vorgefertigte SQL Skripts, die zum Erstellen der benötigten Tabellen und Einträge in der Datenbank sorgen. Folgende Datenbanken werden unterstützt:

- IBM DB2
- Microsoft SQL
- MySQL
- Oracle
- PostgreSQL

Zum Punkt Usability ist zu sagen, dass es etwas umständlich ist, den Prozess auszutesten. Es wird der Prozess in Eclipse modelliert und alle benötigten Einstellungen werden vorgenommen. Danach muss die Applikation mit Maven gebaut und auf den Server (z. B. JBoss oder Tomcat) kopiert werden. Es gibt jedoch auch Konfigurationsmöglichkeiten, damit Maven das erstellte File gleich direkt am Server ablegt. Ansonsten ist die Usability sehr gut, nach einem kurzen Tutorial Video kennt man die wichtigsten Funktionen und wie man diese verwendet. Es gibt auf Google Groups auch eine Seite, auf welcher Fragen gestellt werden können, und man erhält relativ rasch eine Antwort, meist innerhalb eines Werktags. Formularelemente können individuell gestaltet werden, es gibt die Möglichkeit, HTML-Seiten zu erstellen, und somit können dann beliebige Formularelemente angelegt werden.

Für das Benutzerinterface können auch eigene Frameworks und Technologien wie z. B. JSF verwendet werden. Zur Anpassung des Designs können, wie gerade mit den Formularelementen erwähnt, komplett unabhängige Formulare erstellt und designed werden. Von Camunda gibt es auch schon eine vorgefertigte Applikation, um die Tasks für die Benutzer anzuzeigen, sowie auch ein Cockpit, wo die Prozesse überwacht werden können. Diese Taskliste kann mittels CSS angepasst werden. Camunda ist sehr mächtig, da viele Elemente der Architektur ausgetauscht werden können, wie man an der Abbildung 3.1 sehr gut erkennen kann. Alle strichlierten Elemente welche in der Grafik ersichtlich sind, können ausgetauscht werden. „Custom Application“ heißt, dass man hier selbst eine Applikation erstellen kann, welche die Schnittstelle zu den Benutzern darstellt, und diese spricht dann Camunda mittels REST oder einer Java API an. Der Cycle ermöglicht es, diverse Werkzeuge zur Modellierung von BPMN Modellen einzubinden. Solche Werkzeuge fallen dann in „Business Modeler“. Signavio ist ein Werkzeug, welches mittels des Cycles als Modellierungswerkzeug eingebunden werden kann. Somit wird auch eine Trennung zwischen dem fachlichen Modellieren und dem technischem Prozess selbst ermöglicht, wobei der Cycle dafür sorgt, dass diese beiden Teile synchronisiert werden. Es kann dann auch noch ein beliebiges „File Repository“ für die Versionskontrolle von Daten verwendet werden, wie z. B. SVN oder GIT.

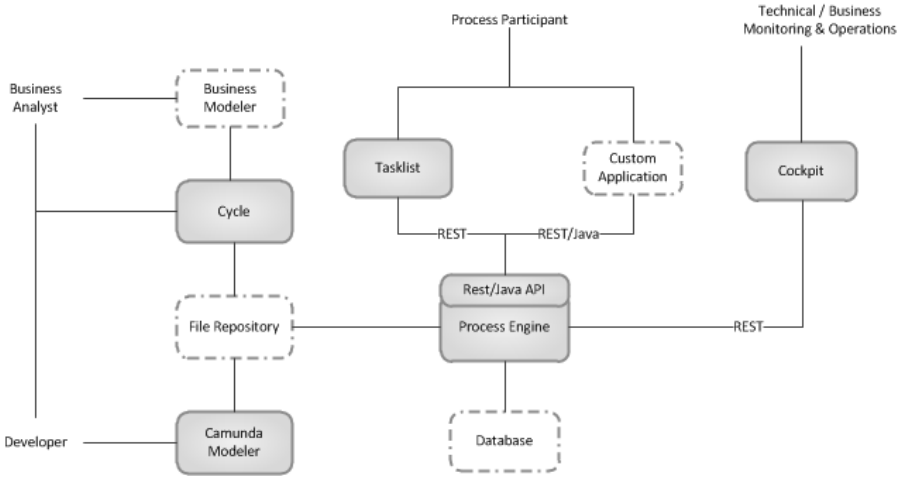

Abbildung 3.1. Camunda Architektur Übersicht [Camunda, 2015]

3.3 Nutzwertanalyse

In diesem Kapitel wird nun eine Nutzwertanalyse über die vier Werkzeuge, welche im zweiten Schritt der Evaluierung genauer betrachtet worden sind, durchgeführt. Laut [Heinrich and Stelzer, 2009] ist die Nutzwertanalyse ein systematischer Planungsansatz, welcher eine Unterstützung bietet, aus einer Menge von Alternativen die optimale Variante unter Berücksichtigung mehrerer Kriterien auszuwählen.

[Wieczorrek and Mertens, 2011] beschreibt den Verfahrensablauf für die Nutzwertanalyse in fünf aufeinanderfolgenden Schritten. In den folgenden Unterkapiteln wird jeder dieser Schritte kurz erklärt und für die Bewertung der Werkzeuge angewendet.

3.3.1 Schritt 1: Bestimmung der Zielkriterien (Zielkriterienbestimmung)

Im ersten Arbeitsschritt werden die quantitativen bzw. qualitativen Beurteilungskriterien festgelegt, welche zur Beurteilung der Alternativen wichtig sind [Wieczorrek and Mertens, 2011]. Im Kapitel 1.2 sind schon einige Anforderungen, die das fertige System erfüllen soll, definiert worden. Des Weiteren wurden auch in den Kapiteln 3.1 und 3.2 Kriterien definiert, welche relevant für Werkzeuge sind. Aus diesen drei Kapiteln wurden folgende Kriterien ausgewählt:

- Datenbanken
- Anpassung von Formularelementen
- Anpassung des Designs
- Usability
- Erweiterbarkeit/Mächtigkeit

Die Kosten der einzelnen Werkzeuge im Detail können nicht sinnvoll in die Nutzwertanalyse aufgenommen werden, da die Lizenzmodelle der einzelnen Werkzeughersteller sehr unterschiedlich sind. Bei Bonitasoft gibt es beispielsweise keine Einschränkung der Anzahl der User, wohingegen bei Bizagi pro User eine Lizenz gekauft werden muss. Camunda wiederum hat ein Pay-per-Use Modell. Da für die Entwicklung des Prototypen ein kostenloses Werkzeug verwendet werden soll, werden die Kosten gar nicht in der Nutzwertanalyse berücksichtigt.

3.3.2 Schritt 2: Bestimmung der Ausprägungen der Beurteilungskriterien

Dieser Schritt dient dazu, diesen Kriterien für die verschiedenen Alternativen nun Ausprägungen zuzuweisen. Diese Ausprägungen können bei manchen Kriterien nur durch Schätzverfahren ermittelt werden [Wieczorrek and Mertens, 2011]. In der Tabelle 3.2 sind nun die Ausprägungen der Zielkriterien für die einzelnen Werkzeuge ersichtlich.

Kriterien \ Alternativen	Bizagi	Bonitasoft	Webratio	Camunda
Datenbanken	Oracle, MS SQL	Access, MS SQL, Oracle, HSQL, H2, PostgreSQL, IBM DB2, Sybase, Informix, AS400, Ingres, Teradata	Apache Derby, PostgreSQL, IBM DB2, MS SQL, Oracle, Informix, Sybase Adaptive Server, Teradata, ODBC, c-treeACE SQLx	H2, IBM DB2, MS SQL, MySQL, Oracle, PostgreSQL
Anpassung von Formularelementen	Ja	Ja	Ja	Ja
Anpassung des Designs	Nein	Ja	Ja	Ja
Usability	Sehr gut	Sehr gut	Gut	Gut
Erweiterbarkeit/ Mächtigkeit	Mittelmäßig	Mittelmäßig	Mächtig	Sehr mächtig

Tabelle 3.2. Ausprägungen der Kriterien

Die verschiedenen Ausprägungen wurden durch das Experimentieren mit den einzelnen Werkzeuge erhoben, Details können im Kapitel 3.2 nachgelesen werden.

3.3.3 Schritt 3: Bestimmung der Teilnutzwerte (Teilnutzenbestimmung)

Im dritten Schritt werden diese Werte nun in einer einheitlichen Bewertungsskala ausgedrückt. Dies kann zum Beispiel durch das Benotungssystem oder ein

Punktesystem erfolgen. Diese Werte werden als Teilnutzen oder Scores bezeichnet [Wieczorrek and Mertens, 2011]. Es wird hier als Bewertungsskala das Benotungssystem herangezogen, also eine ordinale Skala. Es gibt dabei Ausprägungen von 1 bis 5, wobei 1 der beste und 5 der schlechteste Wert ist, den ein Werkzeug für ein Kriterium bekommen kann. Bei den Kriterien, die nur ein Ja oder Nein als Ausprägung haben können, wird dafür 1 und 5 herangezogen. In der Tabelle 3.3 sind nun die Ausprägungen der Werkzeuge für die einzelnen Kriterien ersichtlich.

Kriterien \ Alternativen	Bizagi	Bonitasoft	Webratio	Camunda
Datenbanken	4	2	1	3
Anpassung von Formularelementen	1	1	1	1
Anpassung des Designs	5	1	1	1
Usability	1	1	2	2
Erweiterbarkeit/Mächtigkeit	3	3	2	1

Tabelle 3.3. Teilnutzwerte

3.3.4 Schritt 4: Bestimmung der Nutzwerte (Nutzwertermittlung)

Für die Zusammenführung der Teilnutzwerte bzw. Einzelnoten zu einem Gesamtnutzwert bzw. zu einer Gesamtnote gibt es verschiedene Möglichkeiten [Wieczorrek and Mertens, 2011]. Hier wird folgende Möglichkeit verwendet um die Noten zu gewichten: Zuerst wird für jedes Kriterium erhoben, wie wichtig es im Vergleich zu jedem anderen ist. Daraus ergibt sich ein Faktor und dieser wird dann dazu verwendet, die Noten zu gewichten.

Schritt 4a: Bestimmen der Kriteriengewichte

Die Wichtigkeit der Kriterien wird ermittelt, indem man jedes mit jedem gegenüberstellt. Das geschieht folgendermaßen:

Ist das Kriterium 1 (z. B. Datenbanken) wichtiger als Kriterium 2 (z.B. Usability)?

Wenn „Ja“ dann schreibt man in das entsprechende Feld eine „2“,
Wenn „Nein“ dann eine „0“,
Wenn „Gleich“ dann eine „1“.

Tabelle 3.4 zeigt nun die ausgefüllten Felder. Zum Beispiel ist das Kriterium Datenbanken weniger wichtig als Usability, daher steht in diesem Feld eine „0“. Umgekehrt wiederum ist das Kriterium der Usability wichtiger als Datenbanken, daher steht in diesem Kästchen eine „2“.

Hat man jedes Kriterium mit jedem anderen verglichen, dann berechnet man das Gewicht für jedes Kriterium, indem man alle einzelnen Ausprägungen aufsummiert. Für Datenbanken wäre das dann 2+2 = 4. Dieses Gewicht dividiert man

	Datenbanken	Anpassung von Formularelementen	Anpassung des Designs	Usability	Erweiterbarkeit / Mächtigkeit	Gewicht	Gewichtungsfaktor
Datenbanken	x	2	2	0	0	4	4/20 = 0,20
Anpassung von Formularelementen	0	x	1	1	1	3	3/20 = 0,15
Anpassung des Designs	0	1	x	1	1	3	3/20 = 0,15
Usability	2	1	1	x	0	4	4/20 = 0,20
Erweiterbarkeit/Mächtigkeit	2	1	1	2	x	6	6/20 = 0,30
Summe						20	

Tabelle 3.4. Kriterien-Gegenüberstellung

nun durch das Gesamtgewicht (20) und somit erhält man einen Faktor für das Kriterium. Dieser Faktor wird im nächsten Schritt mit der Ausprägung multipliziert.

Schritt 4b: Nutzwertberechnung

In diesem Schritt wird nun der Nutzwert berechnet. Hierzu werden die Teilnutzwerte aus Tabelle 3.3 mit dem berechneten Gewichtungsfaktor aus Tabelle 3.4 multipliziert. In Tabelle 3.5 ist dies ersichtlich.

Kriterien \ Alternativen	Bizagi	Bonitasoft	Webratio	Camunda	Gewichtungsfaktor
Datenbanken	4 * 0,20 = 0,80	2 * 0,20 = 0,40	1 * 0,20 = 0,20	3 * 0,20 = 0,60	0,20
Anpassung von Formularelementen	1 * 0,15 = 0,15	1 * 0,15 = 0,15	1 * 0,15 = 0,15	1 * 0,15 = 0,15	0,15
Anpassung des Designs	5 * 0,15 = 0,75	1 * 0,15 = 0,15	1 * 0,15 = 0,15	1 * 0,15 = 0,15	0,15
Usability	1 * 0,20 = 0,20	1* 0,20 = 0,20	2 * 0,20 = 0,40	2 * 0,20 = 0,40	0,20
Erweiterbarkeit/ Mächtigkeit	3 * 0,30 = 0,90	3* 0,30 = 0,90	2 * 0,30 = 0,60	1 * 0,30 = 0,30	0,30
Nutzwert (Summe)	**2,8**	**1,8**	**1,5**	**1,6**	

Tabelle 3.5. Nutzwertberechnung

Es wird dann die Summe über alle Kriterien für ein Werkzeug gebildet und somit erhält man den Nutzwert für das Werkzeug.

3.3.5 Beurteilung der Vorteilhaftigkeit

Im letzten Schritt der Nutzwertanalyse werden nun die Nutzwerte miteinander verglichen und gegenübergestellt. Da eine ordinale Skala verwendet wurde, ist der niedrigste Wert der beste. Das Werkzeug Webratio hat den niedrigsten Wert von 1,5 und ist somit das am besten geeignete Werkzeug für die Umsetzung.

3.4 Werkzeugentscheidung

Laut dem Ergebnis der Nutzwertanalyse wäre das Werkzeug Webratio für die Umsetzung heranzuziehen und somit aus wissenschaftlicher und methodischer Sicht das optimale Werkzeug für die Umsetzung dieser Arbeit. Zusätzlich bietet Webratio das standardisierte und methodische Vorgehen IFML, um die Gestaltung und Navigation des Benutzerinterfaces zu definieren. Es existiert jedoch auch eine Anforderung von Engel, die festlegt, dass das Werkzeug nicht Cloud-basiert sein

darf. Webratio ist in der kostenlosen Community Edition Cloud-basiert. Es gibt von Webratio die Enterprise Edition, welche nicht Cloud-basiert, jedoch kostenpflichtig ist. Daher wird das Werkzeug Webratio nicht für die Umsetzung herangezogen, sondern Camunda, welches mit dem Nutzwert von 1,6 an zweiter Stelle liegt.

4

Automatisierung des Prozesses

Im Kapitel 3 wurde die Auswahl eines geeigneten Werkzeugs erläutert. In diesem Kapitel wird auf die Prozessautomatisierung eingegangen und anschließend die Durchführung der Automatisierung sowie die erstellten und verwendeten Services erklärt.

4.1 Prozessautomatisierung

Wie in Unterkapitel 3.4 festgehalten, wurde Camunda als Werkzeug für die Umsetzung und die Automatisierung des Prozesses gewählt. In den folgenden Unterkapiteln wird zuerst auf die Implementierungsdetails eingegangen. Danach wird die Systemarchitektur anhand einer Abbildung dargestellt und beschrieben.

4.1.1 Implementierungsdetails

Im Kapitel 2 ist der Prozess abgebildet und man kann anhand der grauen Elemente erkennen, welche Aufgaben für den Prototyp umgesetzt worden sind und welche nicht. Die grauen Elemente wurden noch nicht umgesetzt. Der Prozess ist in BPMN abgeildet. Hinter den einzelnen Benutzer-Tasks stecken jeweils HTML-Formulare, mit welchen die Benutzer-Formulare erstellt werden. In manchen dieser HTML-Formulare werden JavaScript und AngularJSCode Fragmente benötigt. Die Service-Tasks verwenden Java-Klassen, welche Funktionen liefern die vom Prozess automatisch abgearbeitet werden.

Es wird in diesem Kapitel auf die umgesetzten Tasks eingegangen und es werden einige Besonderheiten und Details dazu erklärt. Im Weiteren wird auf die noch nicht umgesetzten Tasks eingegangen. Zuerst werden jedoch die im Prozess benötigten Variablen beschrieben.

Prozessvariablen

Variable	Datentyp	Beschreibung
initiator	String	Name des Benutzers, der den Prozess gestartet hat. Wird für die Zuweisung der Tasks benötigt.
customerNo	String	Kundennummer eines externen Kunden. Wird für Sicherheitsüberprüfung benötigt.
fabNr	String	Fabrikations- bzw. Maschinennummer. Wird für viele Service-Tasks als Übergabeparameter benötigt.
language	String	Browsersprache. Wird für das Steuern der Sprache von Beschriftungen benötigt.
eMail	String	Email-Adresse des Benutzers. Wird für Auslieferung des Update-Pakets benötigt.
extKunde	Boolean	Externer oder interner Benutzer. Wird benötigt, um im Prozess zwischen ext./int. Benutzern zu unterscheiden.
loginCorrect	Boolean	Sicherheitsüberprüfung erfolgreich/fehlgeschlagen.
installed-OptionsList	Object	Bereits installierte Optionen. Wird für die Anzeige der inst. Optionen benötigt.
possible-OptionsList	Object	Mögliche Optionen. Wird für die Anzeige und Auswahl von Optionen benötigt.
priceList-OptionsList	Object	Preisliste der Optionen. Wird für die Erweiterung der Liste von möglichen Optionen benötigt.
optionsList	Object	Optionsliste, Kombination von Listen installierte Optionen, mögliche Optionen und Preisliste.
optionSelected	Double	Mindestens eine mögliche Option ausgewählt. Wird für Fortführen des Prozesses benötigt.
selectedPossible-OptionsList	Object	Selektierte mögliche Optionen. Wird z. B. in Anzeige für Bestellübersicht und zum Software-Generieren benötigt.
priceAvailable	Boolean	Ist ein Preis für alle selektierte mögliche Optionen vorhanden? Wird für Preisanfrage benötigt.
free	Boolean	Sind alle selektierte mögliche Optionen kostenlos? Wird für Entscheidung, ob Angebot generiert werden muss, benötigt.
shippingPeriod-Available	Boolean	Lieferzeit für alle selektierte mögliche Optionen vorhanden? Wird für Preisanfrage benötigt.
type	String	Software, Hardware oder beides. Wird für Entscheidungen benötigt, ob Software generiert und/oder Hardware besorgt werden muss.

Tabelle 4.1: Prozessvariablen

standardOption	Boolean	Sind alle selektierte mögliche Optionen Standardoptionen? Wird für Entscheidung über Sonderauftrag benötigt.
standardProject	Boolean	Ist die Maschine eine Standardmaschine? Wird für Entscheidung über Sonderauftrag benötigt.
dokuRelevant	Boolean	Sind die selektierte mögliche Optionen dokumentationsrelevant? Wird für Entscheidung über Dokumentation erstellen benötigt.
securityRelevant	Boolean	Sind die selektierte mögliche Optionen sicherheitsrelevant? Wird für Entscheidung über Dokumentation in Papierform erstellen benötigt.
urgent	Boolean	Dringende Auslieferung erforderlich? Wird für Entscheidung bei der Auslieferung benötigt.
ePlan	Boolean	Sind die selektierte mögliche Optionen ePlan relevant? Wird für Entscheidung, ob auf ePlan gewartet werden muss, benötigt.
orderSucc	Boolean	Gesamte Bestellung erfolgreich durchgeführt. Wird für Benachrichtigung der Benutzer benötigt.
agb	Boolean	AGBs müssen akzeptiert werden. Wird für Fortführen des Prozesses benötigt.

Tabelle 4.1: Prozessvariablen (Fortsetzung)

In Tabelle 4.1 werden alle Prozessvariablen aufgelistet und es wird kurz erklärt, wofür diese Variablen benötigt bzw. verwendet werden. Die erste Spalte beinhaltet den Variablennamen, wie er im Prozess verwendet wird. In der zweiten Spalte wird der Datentyp der Variable eingetragen und in der letzten Spalte eine kurze Beschreibung zu der Variable.

Einige der Variablen, wie z. B. „installedOptionsList“, „possibleOptionsList“ usw., haben als Datentyp „Object“ eingetragen. Hier handelt es sich um komplexe Datentypen. Für die „installedOptionsList“ ist dies der komplexe Datentyp „InstalledOption“, welcher eine installierte Option repräsentiert. Eine installierte Option enthält Informationen wie z. B. einen deutschen und englischen Optionsnamen, die Optionsnummer und die Version. Bei der „possibleOptionsList“ handelt es sich um den Datentyp „PossibleOption“, dieser repräsentiert eine mögliche Option. Die mögliche Option enthält die gleichen Informationen wie die installierte Option, jedoch um ein paar mehr. Zum Beispiel wird zusätzlich die Lieferzeit, der Preis, Standardoption (Ja/Nein) und der Typ (Hardware/Software) gespeichert. Die „selectedPossibleOptionsList“ besteht aus einer Liste von Objekten vom Typ „PossibleOption“.

Die beiden Variablen „priceListOptionsList“ und „optionsList“ werden derzeit im Prozess nicht verwendet, da sie für die prototypische Umsetzung nicht benötigt werden. Allerdings sind sie im Prozess schon berücksichtigt, damit der Prozess leicht erweitert werden kann, und auch die Datentypen „PriceListOption“ und „Option“ für diese beiden Listen wurden schon angelegt. Erster beinhaltet In-

formationen wie die Lieferzeit und den Preis einer Option. Der zweite Datentyp beinhaltet alle Informationen aus den drei Datentypen „InstalledOption“, „PossibleOption“ und „PriceListOption“.

Die Verwaltung der Prozessvariablen erfolgt durch Camunda. Für den Sicherheitsaspekt wird beachtet, dass sicherheitskritische Variablen nicht in den HTML-Formularen aufscheinen. Diese dürfen auch nicht als versteckte Felder in den HTML Formularen verwendet werden, denn sie könnten dann von den Benutzern verändert werden und dadurch Risiken hervorrufen. In dieser Arbeit wurden zwei versteckte Felder, die als Hilfsvariablen in den HTML-Formularen dienen, verwendet. Zu diesen sicherheitskritischen Variablen zählen einige Variablen, die den Prozessfluss steuern, wie „loginCorrect“, „free“ oder „extKunde“. Diese Variablen steuern den Prozess und sollen auf keinen Fall von den Benutzern selbst verändert werden können.

Umgesetzte Tasks

Hier wird nun auf die bereits umgesetzten Tasks eingegangen, d.h. auf die Elemente des gesamten Prozesses, welche im Kapitel 2 nicht grau markiert sind. Durch die Absatzüberschriften wird festgelegt, auf welchen Teil der Modellierung des Prozesses sich die Beschreibung der Elemente bezieht. Hauptsächlich wird auf die Elemente eingegangen, bei denen irgendwelche Besonderheiten implementiert worden sind. Es wurden schon alle Gateways und zugehörigen Variablen erstellt und implementiert, sodass der Fluss bei den Verzweigungen abhängig von den Variablen gesteuert wird. Manche Variablen werden jedoch einstweilen fest einprogrammiert, da dieser Prototyp z. B. keine dokumentationsrelevanten Optionen unterstützt. Des Weiteren stecken hinter allen Service-Tasks schon Webservice-Aufrufe auch wenn diese als nicht umgesetzt (grau) markiert sind. Am Webservice befinden sich auch schon Schnittstellen, welche diese Aufrufe entgegennehmen. Das heißt, es sind schon einige Vorbereitungen getroffen, um den Prototypen zu erweitern.

Hauptprozess: Webstore

Der Prozess wird mit dem Start Event „Fabrikations-/Maschinen-Nr. eingeben“ gestartet und es wird auch die Variable „initiator“ gesetzt. In diesem Schritt wird auch mittels einem JavaScript, welches in Listing 4.1 ersichtlich ist, die Variable „language“ gesetzt. Diese Variable befindet sich in einem versteckten Feld, siehe Listing 4.2.

Camunda hat die Eigenschaft, dass es Benutzerformulare, nach dem Abschließen, so lange offen lässt, bis die nachfolgenden Service-Tasks im Hintergrund abgearbeitet sind. Daher wurde in dieses Formular ein Lade-Symbol eingebaut, welches eingeblendet wird, sobald der Benutzer den Task abschließt. Das Listing 4.3 zeigt den Code, welcher dieses Symbol ein- und ausblendet.

Da das Generieren der Listen lange dauert, könnte das Generieren zu einem späteren Zeitpunkt optimiert werden. Zum Beispiel könnte man in der Datenbank

```
<script language="javascript">
  var browserLang = navigator.language + " " +
    navigator.userLanguage + " " +
    navigator.browserLanguage + " " +
    navigator.systemLanguage;
  browserLang = browserLang.toUpperCase()
  document.getElementById("language").value = browserLang;
</script>
```

Listing 4.1. JavaScript für das Setzen der Variable „language“

```
<input type="hidden" id="language" class="form-control"
    cam-variable-type="String" cam-variable-name="language"
    name="language" value="EN" />
```

Listing 4.2. HTML für die versteckte Variable „language“

```
<script cam-script type="text/form-script">
  camForm.on('submit', function() {
    document.getElementById("loading")
      .style.display = "block";
  });
  camForm.on('submit-success', function() {
    document.getElementById("loading")
      .style.display = "block";
  });
  camForm.on('submit-error', function() {
    document.getElementById("loading")
      .style.display = "none";
  });
</script>
<div id="loading" class="col-md-12 text-center"
 style="display:none">
   <span class="glyphicon glyphicon-refresh
    glyphicon-refresh-animate"></span>
</div>
```

Listing 4.3. HTML und AngularJS Code für das Lade-Symbol

alle Fabrikationsnummern, die irgendwann schon einmal abgefragt worden sind, speichern. Dann könnte man automatisch periodisch, zum Beispiel einmal am Tag oder einmal pro Woche und am besten irgendwann in der Nacht, die Generierung der Optionsliste laufen lassen und dann die Optionsliste zu den Fabrikationsnummern speichern. Wird nun für eine Fabrikationsnummer, die schon einmal abgefragt worden ist, eine Abfrage durchgeführt, muss man sich die Optionsliste nicht generieren lassen, sondern kann die gespeicherte Liste laden. Eine weitere Möglichkeit wäre, schon vorab für alle Maschinen die Optionslisten zu generieren und zu speichern. Interessant wird das vor allem, wenn das System produktiv den Kunden zur Verfügung gestellt wird und viele Kunden dieses System verwenden.

Teilprozess: Sicherheitsüberprüfung durchführen

Das nächste Element im Prozess ist der Exklusive Gateway, welcher zwischen externen und internen Benutzern unterscheidet. Auf dieses Element wird, wie in Abbildung 4.1 ersichtlich, ein Listener gehängt. Beim Start des Gateways, d.h. bevor die Überprüfung im Gateway durchgeführt wird, wird dieser Listener ausgeführt. Die Expression ruft die in der Klasse „userBean“ enthaltene Methode „loadUserDataFromDB“ auf und übergibt dieser die „execution“. Wie in Listing 4.4 ersichtlich, werden mittels einer Abfrage auf die Datenbank die Benutzerdaten des eingeloggten Benutzers geladen und in einem „User“ Objekt gespeichert. Es werden die Email-Adresse, die Kundennummer und die Information über internen oder externen Benutzer aus dem Benutzerobjekt in die Prozessvariable gespeichert.

In diesem Listing ist eine SQL Abfrage ersichtlich, welche einfach durch Verknüpfung von Texten zusammengebaut wird. Wird eine SQL Abfrage so erstellt, kann es zu SQL Injections kommen. Bei SQL Injections handelt es sich um eine Technik die bei Attacken auf Webseiten sehr häufig vorkommt. Dabei wird durch uneingeschränkte Benutzereingaben die Query Logik verändert. Diese Sicherheitslücke wird von [Shar and Tan, 2013] im Detail beschrieben und sie geben auch Vorschläge, wie diese Angriffe verhindert werden können. In dem Fall, der in Listing 4.4 ersichtlich ist, dürfte es jedoch keine Probleme geben, da die Variablen, die hier in die SQL Abfrage mit hinein gekommen sind, nicht von einem Benutzer verändert werden können. Bei dem „User.class.getName()“ hat der Benutzer keinen Einfluss, da hier der Klassenname einer Java-Klasse verwendet wird. Das „delegateExecution.getVariable(“initiator“)“ liest eine Prozessvariable aus, welche vom Benutzer nie gelesen oder verändert werden kann, daher sollte hier kein Problem bezüglich SQL Injection auftreten.

Als nächstes wird nun der Service-Task „SAP Logindaten überprüfen“ ausgeführt. In diesem Task wird die Funktion „checkSAPLoginCredentials“ des Webservice „OptionStore“, mittels des Codes, der in Listing 4.5 ersichtlich ist, aufgerufen. Zuerst wird ein Service Locator angelegt und gleich initialisiert. Dann wird ein „Stub“ bzw. ein Client erstellt und über den Service Locator wird dieser Client dann geliefert. Mittels des Clients kann dann die gewünschte Funktion des Webservices aufgerufen werden. Der Aufruf liefert ein Ergebnis zurück, welches

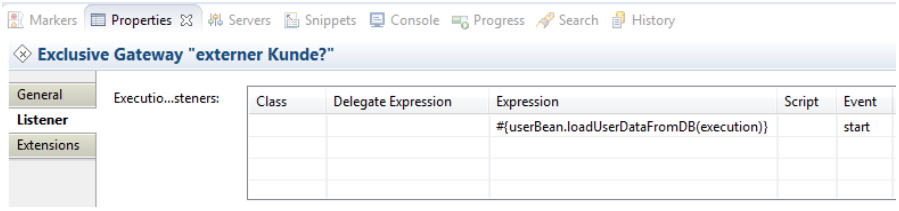

Abbildung 4.1. Start Listener in Gateway für eMail, extKunde und customerNo

```
public boolean loadUserDataFromDB(DelegateExecution
                    delegateExecution) {
  String q = "SELECT p from " + User.class.getName()
      + " p where id_ = '" + delegateExecution
      .getVariable("initiator") + "'";
  Query query = entityManager.createQuery(q);
  User user = (User) query.getSingleResult();

  delegateExecution.setVariable("eMail",
    user.getEmail());
  delegateExecution.setVariable("customerNo",
    user.getCustomerNo());

  if (user.getExtKunde() == 1) {
    delegateExecution.setVariable("extKunde", true);
  } else {
    delegateExecution.setVariable("extKunde", false);
  }
  return true;
}
```

Listing 4.4. Java Code für das Speichern von Benutzerdaten aus der Datenbank in die Prozessvariablen

dann in die Variable „loginCorrect“ gespeichert wird. Nähere Informationen zu diesem Service sind im Kapitel 4.2.1 zu finden.

Die beiden Benutzer-Tasks „Information zur Kontaktaufnahme mit CSD-VET“ und „Information Login fehlgeschlagen“ werden im Fall, dass die Sicherheitsüberprüfung fehlgeschlagen ist, angezeigt. Je nachdem, ob es sich um interne oder externe Benutzer handelt, wird der eine oder andere Task angezeigt. Hier bekommen die Benutzer lediglich eine Information, dass die Sicherheitsüberprüfung fehlgeschlagen ist. Ist die Sicherheitsüberprüfung erfolgreich gewesen, wird die Optionsliste generiert und der Prozess fortgeführt.

```
OptionStoreServiceLocator optionStoreServiceLocator =
    new OptionStoreServiceLocator();
IOptionStore optionStoreStub;
Boolean result;
try {
  optionStoreStub = optionStoreServiceLocator
      .getBasicHttpBinding_IOptionStore();
  result = optionStoreStub.checkSAPLoginCredentials(
      delegateExecution
        .getVariable("initiator").toString());
  delegateExecution.setVariable("loginCorrect", result);
} catch (Exception e) {
  e.printStackTrace();
}
```

Listing 4.5. Java Code für Webservice Aufruf „checkSAPLoginCredentials“

Teilprozess: Optionsliste generieren

Hinter dem Service-Task „Liste von inst. Optionen holen“ steckt wiederum ein Webservice Aufruf. Die Aufrufe der Webservices sind immer gleich, es ändern sich lediglich die Funktion, die aufgerufen wird, die Übergabeparameter, der Rückgabewert und die Prozessvariable, in welcher der Rückgabewert gespeichert wird, siehe Listing 4.6. Dieses Listing zeigt den Aufruf für den Service „Liste von inst. Optionen holen“, welcher in Kapitel 4.2.3 genauer beschrieben ist. Eine Besonderheit bei diesem Code-Ausschnitt ist auch, dass hier eine Liste in Form eines Arrays zurückgegeben wird. Damit Camunda solch ein Array in Benutzer-Formularen anzeigen kann, wird es als JSON serialisiert, bevor es in die Prozessvariable „installedOptionsList“ gespeichert wird.

Auch hinter dem Service-Task „Liste von möglichen Optionen holen“ steckt wieder ein Webservice Aufruf. In Kapitel 4.2.4 wird dieser Service näher beschrieben. Das Ergebnis dieses Aufrufs wird in der Prozessvariable „possibleOptionsList“ gespeichert.

Hauptprozess: Webstore

Nachdem die Service-Tasks der Optionsgenerierung abgeschlossen sind, geht es im Hauptprozess weiter. Das nächste Element im Prozess ist der Benutzer-Task „Optionen anzeigen/Optionen auswählen“. Hinter diesem Benutzer-Task stecken einige Besonderheiten. Da dem Benutzer eine Liste von installierten und von möglichen Optionen angezeigt werden soll, müssen diese in eine geeignete AngularJS Variable gespeichert werden. Die Prozessvariablen sind wie vorher erwähnt in JSON serialisiert und werden wie in Listing 4.7 ersichtlich in eine Variable geladen. In diesem Code-Ausschnitt wird der Code für die möglichen Optionen herangezogen, um zu zeigen, wie dies funktioniert.

```
OptionStoreServiceLocator optionStoreServiceLocator
    = new OptionStoreServiceLocator();
IOptionStore optionStoreStub;
InstalledOption[] installedOptionsArray;
try {
  optionStoreStub = optionStoreServiceLocator
    .getBasicHttpBinding_IOptionStore();
  installedOptionsArray = optionStoreStub
    .getInstalledOptions(delegateExecution
    .getVariable("fabNr").toString());
  ObjectValue typedInstalledOptionsList =
    Variables.objectValue(installedOptionsArray)
      .serializationDataFormat(
        SerializationDataFormats.JSON).create();
  delegateExecution.setVariable(
    "installedOptionsList", typedInstalledOptionsList);
} catch (Exception e) {
  e.printStackTrace();
}
```

Listing 4.6. Java Code für Webservice Aufruf „getInstalledOptions“

```
var variableManager = camForm.variableManager;

camForm.on('form-loaded', function() {
  variableManager.fetchVariable('possibleOptionsList');
});

camForm.on('variables-fetched', function() {
  $scope.possibleOptionsList = variableManager
    .variable('possibleOptionsList').value;
});

camForm.on('submit', function() {
  angular.forEach($scope.possibleOptionsList,
    function(possibleOption) {
      delete possibleOption.$$hashKey;
    });
});
```

Listing 4.7. AngularJS Code um Variablen zu laden

Sobald das Formular geladen wird, wird die Variable „possibleOptionsList“ abgerufen. Wenn die Variablen abgerufen werden, dann wird der Wert der Variable in eine Angular Scope Variable gespeichert, damit das Formular damit arbeiten kann. Der letzte Schritt, der im Listing 4.7 zu sehen ist, wird ausgeführt, sobald das Formular abgeschlossen wird. In diesem Schritt wird der so genannte „hashKey“ entfernt, der von Angular hinzugefügt wird, um Änderungen an den Daten nachzuverfolgen. In Listing 4.8 ist zu sehen, wie diese Variable dann in einer Tabelle verwendet wird, um die Liste von möglichen Optionen anzuzeigen. Dies passiert mit dem Code „<tr ng-repeat=“possibleOption in possibleOptionsList“>“.

Eine weitere Besonderheit in dem Benutzer-Task ist die Lösung, um festzustellen, ob mindestens eine mögliche Option selektiert wurde, da der Prozess nur fortgesetzt werden kann, wenn mindestens eine Option selektiert worden ist. Hierfür wird die versteckte Variable „optionSelected“ verwendet, welche in Listing 4.9 angelegt wird. Es handelt sich hier um eine erforderliche („required“) Variable, d.h. wenn diese nicht gesetzt ist, dann kann das Formular nicht abgeschlossen und der Prozess nicht fortgefahren werden. Jedes Mal, wenn sich an der Selektion etwas ändert (siehe Code „ng-change=“changeSelection()““ in Listing 4.8), wird die Funktion „changeSelection“ ausgeführt. Somit wird das Listing 4.10 jedes Mal ausgeführt, sobald sich an der Selektion der möglichen Optionen etwas ändert.

```
<div id="possibleOptions" style="display:block">
  <table id="possibleOptions">
    <thead>
      <tr>
        <th translate="Sel" title="Selektion">Sel.</th>
        <th translate="OptNo">Option</th>
        <th translate="Name" ng-show="language">Name DE</th>
        <th translate="Name" ng-show="!language">Name EN</th>
      </tr>
    </thead>
    <tr ng-repeat=
      "possibleOption in possibleOptionsList">
      <td><input type="checkbox"
        ng-model="possibleOption.selected"
        ng-change="changeSelection()"/></td>
      <td>{{possibleOption.optionNr}} </td>
      <td ng-show="language">{{possibleOption
        .label_de}}</td>
      <td ng-show="!language">{{possibleOption
        .label_en}}</td>
    </tr>
  </table>
</div>
```

Listing 4.8. HTML Code für die Liste von möglichen Optionen

In diesem Listing werden alle möglichen Optionen durchiteriert, um zu kontrollieren, ob mindestens eine davon selektiert wurde, und abhängig davon die Variable „optionSelected“ gesetzt.

Eine weitere Besonderheit ist, dass es einen englischen und deutschen Beschreibungs-Text für die Optionen gibt. Grundsätzlich sind in der Tabelle, welche in Listing 4.8 zu sehen ist, beide Texte enthalten, jedoch wird mit dem „ng-show=“language““ abhängig von der Variable „language“ gesteuert, welcher der beiden Texte angezeigt werden soll. Mit dem Code von Listing 4.11 ist zu sehen, wie diese Variable auf „true“ oder „false“ gesetzt wird.

Die letzte Besonderheit, die der Task „Optionen anzeigen/Optionen auswählen“ enthält, ist, die Liste von installierten und möglichen Optionen auf- und zuklappen zu können. Anhand des Beispiels der möglichen Optionen wird erklärt, wie dies gelöst wird. In Listing 4.12 sind zwei Links zu sehen, welche beim Klicken auf eine JavaScript-Funktion zugreifen. Diese JavaScript-Funktion ist in Listing 4.13 zu sehen. In der Funktion wird mittels einer „switch-case“-Verzweigung erkannt, von welchem Element die Funktion aufgerufen wird. Dadurch wird die Variable für das Zeichen, ob der Inhalt auf- oder zugeklappt ist, gesetzt und auch die Variable für den Block, der aus- oder eingeblendet werden soll. Mit der „if“-Abfrage wird erkannt, ob auf- oder zugeklappt ist, und dann das Gegenteil ausgeführt, d.h. wenn aufgeklappt ist, wird zugeklappt und umgekehrt.

```
<input id="optionSelected" cam-variable-type="Double"
  cam-variable-name="optionSelected"
  name="optionSelected" required type="hidden"/>
```

Listing 4.9. HTML für die versteckte Hilfsvariable „optionSelected“

```
$scope.changeSelection = function() {
  $scope.optionSelected = "";
  angular.forEach($scope.possibleOptionsList,
    function(possibleOption) {
    if (possibleOption.selected) {
      $scope.optionSelected += "1";
    }
  });
};
```

Listing 4.10. AngularJS Code für die versteckte Hilfsvariable „optionSelected“

```
var browserLang = navigator.language + " " +
  navigator.userLanguage + " " +
  navigator.browserLanguage + " " +
  navigator.systemLanguage;
browserLang = browserLang.toUpperCase()
if (browserLang.indexOf("DE") != -1){
  $scope.language = true;
} else {
  $scope.language = false;
}
```

Listing 4.11. AngularJS Code zum Setzen der Variable „language“

Beim nächsten Schritt, der im Hauptprozess ausgeführt wird, handelt es sich um den Service-Task „Optionen evaluieren“. In diesem Schritt werden alle ausgewählten Optionen für die nachfolgenden Prozessschritte evaluiert. Damit ist

```
<a id="toggleSignPoss"
  href='javascript:toggle("possibleOptions");'> v </a>
<a id="displayPossibleOptions"
  href='javascript:toggle("possibleOptions");'
  translate="possibleOptionen">Mögliche Optionen</a>
```

Listing 4.12. HTML Code für das Auf- und Zuklappen der möglichen Optionen

```
function toggle(element) {
  var ele;
  var text;
  switch (element) {
  case "possibleOptions":
    ele = document.getElementById("possibleOptions");
    text = document.getElementById("toggleSignPoss");
    break;
  }
  if (ele.style.display == "block") {
    ele.style.display = "none";
    text.innerHTML = " > ";
  } else {
    ele.style.display = "block";
    text.innerHTML = " v ";
  }
}
```

Listing 4.13. JavaScript Code für das Auf- und Zuklappen der möglichen Optionen

gemeint, dass alle Variablen, die in den nächsten Schritten benötigt werden, gesetzt werden. Welche Variablen das sind und wie sie gesetzt werden ist in Listing 4.14 zu sehen.

Die Variablen „dokuRelevant“, „securityRelevant“, „urgent“, „ePlan“ und „orderSucc“ werden hier initialisiert. Bis auf die Variable „orderSucc“ behalten diese Variablen, für den Prototypen, diesen initialen Wert. Der Wert der Variable „standardProject“ wird über die Webservice-Funktion „isStandardProject“ gesetzt. Um die Werte für die restlichen Variablen zu bekommen, ist es notwendig, alle selektierten möglichen Optionen durchzuiterieren. Mit dem ersten „if-else-if-else“-Block wird ermittelt, ob für alle Optionen ein Preis vorhanden ist oder nicht. Des Weiteren wird auch ermittelt, ob alle Optionen kostenlos sind oder nicht. Wenn der Preis kleiner als 0 ist, dann bedeutet das, dass kein Preis angegeben ist. Wenn der Preis gleich 0 ist, dann heißt das, dass der Preis vorhanden und die Option kostenlos ist. Ist der Preis größer als 0, so handelt es sich um eine Option, die kostenpflichtig ist. Somit wird die Variable „priceAvailable“ gesetzt und enthält nur dann ein „true“, wenn für alle Variablen ein Preis vorhanden ist. Genauso wird die Variable „free“ gesetzt und bekommt am Ende aller Schleifendurchläufe nur dann ein „true“, wenn für alle Optionen ein Preis vorhanden und dieser 0 ist.

In dem zweiten Block mit den verschachtelten Verzweigungen wird die Variable „type“ gesetzt, welche den Typ der Optionen angibt. Diese Variable enthält, nachdem die Schleife vollkommen durchgelaufen ist, den Wert „SW“, wenn es sich ausschließlich um Software-Optionen gehandelt hat. Sie enthält den Wert „HW“, wenn es ausschließlich Hardware-Optionen waren und den Wert „SWHW“, wenn es sich um eine Mischung aus beiden handelt. Des Weiteren wird auch die Variable „standardOption“ gesetzt, welche nur dann den Wert „true“ enthält, wenn alle Optionen Standard-Optionen sind.

Der nächste Schritt ist ein Exklusives Gateway, welches überprüft, ob es sich um einen externen Kunden und um eine kostenpflichtige Option handelt. Derzeit wird hier immer die Verzweigung gewählt, welche die Angebotsgenerierung überspringt. Die Benutzer-Tasks „Option(en) bestellen/Bestellung auslösen“ und „Bestellbestätigung“ beinhalten keine speziellen Implementierungen.

Wie schon erwähnt gibt es das Problem, dass Camunda einen Benutzer-Task so lange anzeigt, bis alle folgenden Service-Tasks abgeschlossen sind. Da das Software-Generieren, je nach Serverauslastung, mehrere Minuten und auch Stunden dauern kann, würde der Benutzer dann so lange die „Bestellbestätigung“ sehen, bis die Generierung abgeschlossen ist. Um dies zu verhindern, wurde gleich nach der „Bestellbestätigung“ ein Timer Ereignis eingefügt, welches 1 Sekunde wartet. Durch dieses Ereignis wird der Benutzer-Task sofort nach dem Abschließen geschlossen. Die nachfolgenden Exklusive Gateways laufen, in der aktuellen Implementierung, die Verzweigungen direkt bis zum Service-Task „Software generieren“ durch. Es geht nun mit „Software generieren“ und der „Auslieferung“ weiter.

```
boolean dokuRelevant             = false;
boolean securityRelevant         = false;
boolean urgent                   = false;
boolean ePlan                    = false;
boolean orderSucc                = false;
boolean standardProject          = true;
boolean free                     = true;
boolean priceAvailable           = true;
boolean shippingPeriodAvailable = true;
String type                      = null;
boolean standardOption           = true;

standardProject = optionStoreStub
        .isStandardProject(delegateExecution
        .getVariable("fabNr").toString());

for (PossibleOption selectedPossibleOption :
          selectedPossibleOptionsArray) {
  if (selectedPossibleOption.getPrice() < 0) {
    priceAvailable &= false;
    free &= false;
  } else if (selectedPossibleOption.getPrice() == 0) {
    priceAvailable &= true;
    free &= true;
  } else {
    priceAvailable &= true;
    free &= false;
  }
  if (selectedPossibleOption
      .getDeliveryPeriodeDays() < 0){
    shippingPeriodAvailable &= false;
  } else {
    shippingPeriodAvailable &= true;
  }

  if (type == null){
    type = selectedPossibleOption.getType();
  } else {
    if (selectedPossibleOption
        .getType().equals("SWHW")){
      type = selectedPossibleOption.getType();
    }
    if (type.equals("SWHW") ||
    type.equals(selectedPossibleOption.getType())){
    } else {
      type = "SWHW";
    }
  }
  standardOption &= selectedPossibleOption
      .getStandardOption();
}
```

Listing 4.14. Java Code für das Evaluieren der möglichen Optionen

Teilprozess: Software generieren und Teilprozess: Auslieferung

Aus Performance-Gründen wurden die Schritte, die für die Generierung der Software und für die Auslieferung notwendig sind, in eine Webservice-Funktion zusammengepackt. Es gibt zwar am Webservice für jeden Schritt eine einzelne Funktion, die aufgerufen werden könnte, jedoch benötigt jede der Webservice-Funktionen einige Parameter, die erst aus diversen Files ausgelesen werden müssen. Daher wurde eine Funktion („upgradeProject“) erstellt, die aufgerufen wird und dann nacheinander die Funktionen, aufruft die für das „Software generieren“ und die „Auslieferung“ benötigt werden. Das Ergebnis, welches vom Webservice-Aufruf zurückgeliefert wird, wird in die Variable „orderSucc“ gespeichert und legt fest, ob die Funktion erfolgreich war oder nicht. Nähere Informationen zu den einzelnen Funktionen sind im Unterkapitel 4.2.7 zu finden.

Es geht dann im Hauptprozess weiter mit dem Exklusiven Gateway, welches mittels der Variable „orderSucc“ überprüft, ob die Bestellung erfolgreich war. Falls die Bestellung nicht erfolgreich war, werden die Benutzer darüber informiert. Wenn sie erfolgreich war, werden die Benutzer mit dem Benutzer-Task „über Versand benachrichtigen“ über die Versendung informiert. Im letzten Schritt bestätigen die Benutzer, dass die Optionen an der Maschine installiert worden sind. Hinzuzufügen ist, dass alle Gateways im Prozess schon funktionsfähig sind. Sie verwenden auch schon die richtigen Variablen, welche angeben, welche Verzweigung genommen werden soll. Diese Variablen sind einstweilen so gesetzt, dass der richtige Pfad für diesen Prototypen durchlaufen wird.

Noch nicht umgesetzte Tasks

In diesem Kapitel wird nun auf alle Teilprozesse und Tasks eingegangen, die noch nicht umgesetzt worden sind. Es werden Umsetzungsvorschläge dargestellt und Erklärungen dazu gegeben. Des Weiteren wird auch auf schon umgesetzte Tasks eingegangen und erklärt, wie diese erweitert werden können, damit der Prozess z. B. auch Hardware-relevante Optionen unterstützt. Wie schon erwähnt, sind im gesamten Prozess, einschließlich den Teilprozessen, alle Gateways umgesetzt worden. Es sind auch schon alle Tasks im Prozess modelliert, jedoch teilweise ohne jegliche Funktion. Im Prozess wurden auch einige Manuelle-Tasks abgebildet. Bei diesen Tasks ist noch eine genauere Erhebung notwendig, was hier genau passieren soll. Grundsätzlich ist zu sagen, dass dies für alle Manuellen-Tasks im Prozess zutrifft, d.h. für diese Tasks ist es notwendig, noch genauer zu erheben, wie dies umgesetzt werden soll bzw. kann. Der Grund dafür ist, dass es sich hier meist um Aufgaben handelt, die schon in irgendeiner Form im Unternehmen durchgeführt werden. Teile dieser Aufgaben wurden schon erhoben, jedoch nicht in den Prozess und somit auch nicht in diese Arbeit mit aufgenommen, da im Detail geklärt werden muss, wie diese Aufgaben in den Prozess integriert werden können.

Hauptprozess: Webstore

Hier wird auf die noch nicht umgesetzten Tasks des Hauptprozesses, welcher in Unterkapitel 2.1 ersichtlich ist, eingegangen. Sobald die beiden noch nicht umge-

setzten Service-Tasks des Teilprozesses „Optionsliste generieren“ umgesetzt worden sind, ist es auch notwendig, den Benutzer-Task „Optionen anzeigen/Optionen auswählen“ anzupassen. Hierfür gibt es hier mehrere Lösungsmöglichkeiten. Die erste Umsetzungsidee ist, dass mit dem Service-Task „Optionsliste generieren“ aus den drei Listen mögliche, installierte und Preislisten-Optionen eine gesamte Liste generiert und diese dann angezeigt wird. In diesem Fall ist es notwendig, den Benutzer-Task anzupassen, sodass dieser dann nicht wie derzeit die beiden Listen installierte und mögliche Optionen getrennt anzeigt, sondern eine Liste angezeigt wird, die alle Informationen enthält. Eine weitere Lösungsmöglichkeit wäre, dass beim „Optionsliste generieren“ nur die Liste der möglichen Optionen um die Informationen der Preisliste ergänzt wird. In diesem Fall könnte der Benutzer-Task unverändert bleiben, jedoch müsste die Schnittstelle des Webservices angepasst werden, damit er die ergänzte Liste von möglichen Optionen zurückgibt.

Der Benutzer-Task „Optionen anzeigen/Optionen auswählen“ muss auch erweitert, werden sobald der Teilprozess „Angebot generieren“ umgesetzt wird. In den Benutzer-Task „Optionen anzeigen/Optionen auswählen“ muss dann etwas eingebaut werden, um den Benutzern die Möglichkeit zu bieten, einen Sonderwunsch einzugeben. Hiermit ist gemeint, dass die Kunden, wenn sie sich etwas Bestimmtes für ihre Maschine wünschen, es aber nicht unter den möglichen Optionen finden, eine Anfrage stellen können, um herauszufinden, ob sie dies bekommen können. Eine mögliche Umsetzungsvariante wäre, einfach zu den möglichen Optionen eine Sonderwunsch-Option hinzuzufügen, welche dann im Teilprozess „Angebot generieren“ berücksichtigt wird.

In dem gesamten Prozess ist die Bezahlung der Optionen noch nicht berücksichtigt, da nicht klar war, wie die Bezahlung erfolgen soll. Es gibt alleine schon mehrere Möglichkeiten für den Zeitpunkt, wann die Bezahlung erfolgen soll. Zum Beispiel könnte man es so umsetzen, dass zuerst die Bezahlung eingegangen sein muss, bevor der Prozess weiterläuft, im Gegensatz dazu, dass die Zahlung parallel zum Prozess passieren kann und vor der Auslieferung auf den Zahlungseingang gewartet wird. Auch für die Umsetzung selbst stehen mehrere Möglichkeiten zur Auswahl. Soll für die Abwicklung der Bezahlung ein Drittanbieter herangezogen werden oder soll dies selbst umgesetzt werden? Da all diese Fragen noch nicht geklärt sind, wurde die Bezahlung noch nicht im Prozess berücksichtigt.

Nachdem der Teilprozess „Angebot generieren“ umgesetzt worden ist, muss auch der Benutzer-Task „Option(en) bestellen/Bestellung auslösen“ angepasst bzw. erweitert werden. Der Task muss so erweitert werden, dass er die Informationen aus dem Angebot auch berücksichtigen kann. Für die Umsetzung ist es wahrscheinlich am besten, wenn die Angebotsgenerierung so gestaltet wird, dass die Informationen aus dem Angebot direkt in die Liste der möglichen Optionen aufgenommen werden. Falls dies möglich ist, dann ist es nicht notwendig, den Benutzer-Task „Option(en) bestellen/Bestellung auslösen“ anzupassen, da dieser ohnehin die möglichen Optionen auflistet.

Der Benutzer-Task „Bestellung überprüfen“ ist noch nicht umgesetzt, als Umsetzung könnte hier ein Benutzerformular mit der Bestellübersicht erstellt und den Vertriebsmitarbeitern angezeigt werden. Grundsätzlich ist aber auch zu überlegen,

ob dieser Schritt überhaupt notwendig ist, da es alleine darum geht, dass die Bestellung, bevor sie durchgeführt wird, nochmals kontrolliert wird. Wahrscheinlich ist es sinnvoll, diesen Schritt in einer ersten Testphase zu verwenden, aber dann nach einiger Zeit auszubauen, um den Prozess nicht unnötig aufzuhalten.

Wenn es sich um Hardware-relevante Optionen handelt, ist es notwendig zu überprüfen, ob sich am ePlan etwas ändert, dies geschieht mit dem Task „ePlan Relevanz überprüfen“. Hier ist wiederum ein manueller Task verwendet worden, wobei dieser Task höchstwahrscheinlich ein einfacher Benutzer-Task sein wird, in dem jemand einträgt, ob die Optionen ePlan-relevant sind oder nicht. Der Schritt „ePlan Mapping erstellen“ wird dann durchgeführt, wenn sich herausgestellt hat, dass die Optionen ePlan-relevant sind. Das ePlan Mapping ist die Zuordnung von Hardware-Ein- und Ausgängen zu Software-Variablen. Wie dieses Mapping genau erstellt wird, muss noch erhoben werden. Für die Umsetzung ist es wahrscheinlich ausreichend, die Erstellung des ePlan Mappings irgendwo anzutriggern, da es diese ePlan Mapping Erstellung bereits gibt. Der Service-Task „auf ePlan warten“ wird dafür benötigt, dass das Software-Gnerieren erst dann ausgeführt wird, wenn der ePlan erstellt worden ist, denn dieser wird in die Software integriert. Dieser Service-Task, um auf den ePlan zu warten, ist derzeit noch nicht umgesetzt und es gibt dazu auch keinen Webservice. Der Grund dafür ist, dass es noch unklar ist, wie dies am besten gehandhabt wird. Es gäbe z. B. die Möglichkeit, dass der Benutzer nach dem Erstellen des ePlan Mappings (Task „ePlan Mapping erstellen“) in einem Benutzerformular einträgt, dass er dies nun erledigt hat. Dann könnte dieser Eintrag von dem Service, der auf den ePlan wartet, abgefragt werden bzw. der Prozess könnte ummodelliert werden und das Warten könnte mit einem Signal gelöst werden. Eine andere Möglichkeit wäre, auf den Service-Task einen Webservice zu legen, der sich um das Warten kümmert, nachsieht ob das Mapping erstellt wurde, und den Prozess dann weiterlaufen lässt.

Für die beiden Tasks „Sonderauftrag auslösen“ und „Teile beschaffen (Lager, Produktion, Einkauf)“ gibt es derzeit schon Abläufe und Prozesse, welche genau diese Aufgabe übernehmen. Jedoch sind diese derzeit nur in SAP abgebildet. Es ist unklar, ob und wie diese Abläufe in den Webstore integriert werden können oder ob diese neu abgebildet und nachgebaut werden müssen. Hierzu sind noch Evaluierungen und Erhebungen durchzuführen, bevor diese Abläufe in den Prozess eingebaut werden können. Zur Umsetzung können deshalb derzeit keine Vorschläge gemacht werden.

Die beiden Service-Tasks „Optionen testen“ und „Installation in SAP vermerken“ sind noch nicht umgesetzt worden, in den Unterkapiteln 4.2.17 und 4.2.19 sind diese Services näher erklärt. Hinter beiden Tasks stecken wieder Webservices, welche die Aufgaben übernehmen sollen, und deren Aufrufe wurden schon implementiert. Das heißt, es müssen die Funktionen dieser beiden Webservices implementiert werden, wobei erstere schwierig umzusetzen sein wird, da es derzeit noch keine Möglichkeit im Unternehmen gibt, solche Tests voll automatisch ausführen zu lassen, d.h. dies müsste auch zuerst umgesetzt werden. Dieser Service-Task könnte auch durch einen Benutzer-Task ausgetauscht und das Testen manuell durchgeführt werden. Dies würde jedoch bedeuten, dass hier wieder einiges an

Zeitverzögerung hinzukommt und auch Ressourcen, in Form von Testern, benötigt werden. Für die Umsetzung des anderen Service-Tasks „Installation in SAP vermerken“ ist es nur ein Datenbankfeld in SAP erforderlich mit welchem man kennzeichnen kann, dass die Installation durchgeführt worden ist. Da es solch ein Feld noch nicht gibt, müsste dieses hinzugefügt und eine Schnittstelle geschaffen werden, um dieses Feld setzen zu können. Der Webservice kann dann auf diese Schnittstelle zugreifen und das Feld setzen.

Teilprozess: Sicherheitsüberprüfung durchführen

Im Teilprozess der Sicherheitsüberprüfung, welcher in Kapitel 2.2 ersichtlich ist, muss noch der Service-Task „KundenNr und MaschinenNr überprüfen“ umgesetzt werden. Dazu gibt es, wie in Unterkapitel 4.2.2 beschrieben, schon eine Webservice-Schnittstelle, diese ist jedoch noch ohne Funktion. Dieser Service-Task dient dazu, bei externen Benutzern überprüfen zu können, ob dieser Benutzer berechtigt ist, für die eingegebene Spritzgussmaschine den Webstore zu benutzen. Die Überprüfung soll mittels Kontrolle, ob die Maschinennummer der Kundennummer zugeordnet ist, erfolgen. Um diesen Task umzusetzen, reicht es, im Webservice die Funktion auszuprogrammieren.

Teilprozess: Optionsliste generieren

Dieser Teilprozess, siehe auch 2.3, funktioniert grundsätzlich schon, jedoch gibt es noch zwei Tasks in diesem Teilprozess, welche noch nicht umgesetzt worden sind. Es handelt sich hierbei um die beiden Service-Tasks „Preisliste von CSD holen“ und „Optionsliste generieren“, welche vom Webservice zur Verfügung gestellt werden. In den Unterkapiteln 4.2.5 und 4.2.6 sind diese Services beschrieben. Die Services werden schon richtig aufgerufen, jedoch ist deren Funktion noch nicht implementiert. Das heißt wiederum, es reicht, wenn deren Funktion im Webservice erstellt wird.

Der Task, um die Preisliste zu holen, dient dazu, aktuelle und kundenspezifische Preise und Lieferzeiten für die Optionen zu erhalten. Es soll dann die Liste der möglichen Optionen mit den Informationen aus der Preisliste angereichert werden. Dies geschieht mit dem Task „Optionsliste generieren“, bei dem die Preislisten-Informationen mit den möglichen und installierten Optionen zusammengeführt werden sollen.

Damit auch Hardware-relevante Optionen angezeigt und unterstützt werden, müssen die beiden Webservices, die in den Unterkapiteln 4.2.3 und 4.2.4 zu finden sind, erweitert werden. Für die installierten Optionen muss die Funktion im Webservice überarbeitet werden. Dabei ist jedoch zu beachten, dass derzeit hier eine Verbindung zwischen den Daten fehlt, damit eindeutig festgestellt werden kann, welche Software-Option zu welcher Hardware-Option gehört. Nähere Informationen dazu sind im Unterkapitel 4.2.3 zu finden. Um die möglichen Optionen auf Hardware-relevante Optionen zu erweitern, ist es ausreichend, wenn die Hardware-Option in der XML-Datei eingetragen wird, in der sich die möglichen Optionen befinden, da es sich dabei derzeit um eine statische Liste handelt.

Teilprozess: Angebot generieren

Der gesamte Teilprozess der Angebotsgenerierung, siehe Kapitel 2.4, wurde noch nicht umgesetzt. Beim Starten dieses Teilprozss wird überprüft, ob ein Preis oder eine Lieferzeit nicht vorhanden ist oder ob es sich um einen Sonderwunsch oder eine Nicht-Standardoption handelt.

Falls der Preis und/oder die Lieferzeit nicht vorhanden ist, soll ein Benutzerformular angezeigt werden, welches den Benutzern erlaubt, eine Preisanfrage zu stellen. Diese Preisanfrage gelangt dann in die Vertriebsabteilung und wird dort mit dem manuellen Task „Preisanfrage bearbeiten“ bearbeitet. Hier wurde einstweilen ein manueller Task verwendet, da noch genauere Erhebungen gemacht werden müssen, wie diese Aufgabe umgesetzt werden kann bzw. soll. Das Ergebnis dieses Tasks soll sein, dass für alle Optionen nun die Preise und Lieferzeiten vorhanden sind. Als Umsetzungsvorschlag könnte hier ein Benutzerformular dienen, welches die gewählten Optionen des Kunden auflistet, und die Vertriebsmitarbeiter geben die fehlenden Preise und Lieferzeiten ein.

In dem Fall, dass beim Gateway ein Sonderwunsch oder eine Spezialoption bzw. ein Nicht-Standardprojekt einlangt, sollen die Kunden ebenfalls ein Benutzerformular angezeigt bekommen, mit welchem sie dann eine Anfrage stellen können. Die Vertriebsmitarbeiter sollen dann diese Anfrage bearbeiten und dafür eine Option mit Preis und Lieferzeit anlegen. Dafür würde sich auch wieder ein Benutzerformular eignen, in welches die Options-Informationen eingegeben werden können.

Auch der Benutzer-Task „Angebotsanfrage stellen“ wurde noch nicht umgesetzt. Um diesen Task umzusetzen, reicht ein einfaches Benutzerformular, das eine Übersicht über die ausgewählten Optionen gibt und dem Kunden mitteilt, dass er mit diesem Formular ein Angebot einholen kann. Danach folgt der manuelle Task „Angebot anlegen/generieren“, wie bei allen manuellen Tasks ist hier eine genauere Erhebung notwendig. Wichtig bei diesem Schritt ist, dass das Angebot auch in SAP abgelegt werden muss. Für die Umsetzung könnte ein Benutzerformuar verwendet werden, in welches die Daten für das Angebot eingegeben werden. Aus diesen Daten wird dann ein Angebot generiert und das Angebot wird mittels einem Webservice in SAP abgespeichert. Da die Angebotsgenerierung nichts Neues ist, muss evaluiert werden, ob es eventuell möglich ist, hier etwas Bestehendes einzubauen.

Teilprozess: Software generieren

Im Unterkapitel 2.5 sind die Aufgaben ersichtlich, welche für den Teilprozess des Software-Generierens benötigt werden. Zu der Software-Generierung gehört auch die Erstellung der Dokumentation. Es gibt hierzu auch bereits einen bestehenden Prozess, dieser wurde grob durch die drei manuellen Schritte „Doku-Auftrag freigeben“, „eHelp erzeugen“ und „Doku in Papier produzieren“ nachgebildet. Jedoch sind hier auch noch einige Erhebungen und Evaluierungen notwendig, um herauszufinden, wie die Erzeugung der Dokumentation am besten in den Prozess integriert werden kann. Der bestehende Prozess ist derzeit in SAP abgebildet und

es muss evaluiert werden, ob und wie dieser Ablauf in den Webstore integriert werden kann.

Teilprozess: Auslieferung

Der Teilprozess der Auslieferung ist im Unterkapitel 2.6 zu sehen. Darin wurden die beiden Tasks „über Expressdienst versenden“ und „über Post-Sammelsendung versenden“ noch nicht umgesetzt. Es wird keine Möglichkeit geben, diese beiden Schritte zu automatisieren, und es werden manuelle Aufgaben bleiben. Am sinnvollsten erscheint es, hierfür Benutzerformulare zu erstellen, welche anzeigen, was wohin versendet werden soll. Nachdem der Benutzer die Sendung erstellt und versendet hat, bestätigt er das Formular und der Prozess wird fortgesetzt.

Anpassung und Konfiguration

In diesem Unterkapitel werden nun einige Details für diverse Anpassungen und Konfigurationen erklärt.

Für diesen Prototypen wurden die von Camunda bereitgestellten Komponenten Tasklist, Cockpit und Admin verwendet und es wurden hier keine eigenen Oberflächen entwickelt. Um das Design der Taskliste individuell anzupassen und die Farben sowie das Logo mittels CCS zu ändern, ist folgendes Vorgehen notwendig: Am JBoss Server läuft, wie in Abbildung 4.4 ersichtlich, die Camunda webapp, diese enthält die Tasklist. Da Camunda Open Source ist, ist der Source Code natürlich frei zugänglich. Im Source der Camunda webapp gibt es im Ordner „WebContent/app“ einen Ordner „tasklist“, dieser beinhaltet den Source Code für die Taskliste. Darin gibt es dann den Ordner „styles“ welcher die Dateien „styles“ und „user-styles.css“ enthält. Die erste Datei enthält das aktuelle Design der Taskliste und zweitere ist dafür gedacht, mittels CSS das Design zu überschreiben und individuell zu gestalten. Der Source Code kann entweder direkt heruntergeladen, die Änderungen durchgeführt, eine war-Datei erstellt und auf den Server kopiert werden oder es kann die bestehende war-Datei entpackt, verändert und wieder eine war-Datei erstellt werden.

In Camunda gibt es die Möglichkeit, die Sprache der Taskliste abhängig von der Sprache des Browser zu definieren. Um diese Möglichkeit auszunutzen, wird wie folgt vorgegangen. Zu beachten ist, dass es hier nur um die Taskliste geht und nicht um den Inhalt, der in der Taskliste angezeigt wird, d.h. die Sprache der eingebetteten Formulare wird nicht durch diese Möglichkeit angepasst. Wie schon beim Anpassen des Designs benötigt man den Source der Taskliste, welcher in der Camunda webapp enthalten ist. Es gibt im Ordner „tasklist“ einen Ordner „locales“, in diesen wird einfach die gewünschte Sprachdatei (de.json, en.json) hineinkopiert. Für einige Sprachen gibt es schon Übersetzungen die man sich herunterladen kann. Nachdem die gewünschte Sprache in den Ordner kopiert worden ist, wird der Code in der „config.js“ Datei, welche im Ordner „scripts“ zu finden ist, noch angepasst und die Sprache ergänzt. In Listing 4.15 ist der Code ersichtlich, welcher notwendig ist, um die unterstützten Sprachen anzuführen.

```
"locales": {
    "availableLocales": ["en", "de"],
    "fallbackLocale": "en"
}
```

Listing 4.15. Taskliste Sprachanpassung

Hier ist es genau wie bei der Anpassung des Designs, es muss wieder eine war-Datei erstellt und auf den Server kopiert werden. Es können eine Reihe von Sprachen sowie eine Fallback-Sprache definiert werden. Für jede der angegebenen Sprachen muss es natürlich im Ordner „locales“ eine Sprachdatei geben.

Um auch die Sprache der Formularinhalte anzupassen, können die Sprachdateien (de.json, en.json), welche für die Sprachanpassung der Taskliste erwähnt wurden, einfach um die zusätzlich benötigten Übersetzungen erweitert werden. Hierzu müssen die Key-Value-Pairs ergänzt werden, ein Beispiel ist in Listing 4.16 zu sehen.

Hier ist zu sehen, dass für den Key „FabNr“ die deutsche Übersetzung „Maschinennr./Fabrikationsnr.“ und die englische Übersetzung „Maschine no./Serial no.“ definiert wird. Danach können in den Formularen diese definierten Labels verwendet werden. In Listing 4.17 ist zu sehen, wie der Key „FabNr“ nun in dem Translate Attribut verwendet wird, um die Sprache der Formularinhalte anzupassen.

Genau wie bei den anderen Anpassungen muss hier auch wieder eine war-Datei erstellt und auf den Server kopiert werden. Im Formular wird dann abhängig von der im Browser eingestellten Sprache der deutsche oder englische Text für das Label angezeigt. Um auch die Sprache der Tasknamen abhängig von der Browsersprache steuern zu lassen, wurde ein Workaround verwendet. In Camunda wurde eine Klasse „TaskNameService“ erstellt, welche die Methode „notify“ enthält, diese ist in Listing 4.18 zu sehen.

```
Datei de.json:  "FabNr": "Maschinennr./Fabrikationsnr."
Datei en.json:  "FabNr": "Maschine no./Serial no."
```

Listing 4.16. Formularinhalte Sprachanpassung

```
<label for="fabNr" translate="FabNr">Maschinen Nummer</label>
```

Listing 4.17. HTML Code für Formularinhalte Sprachanpassung

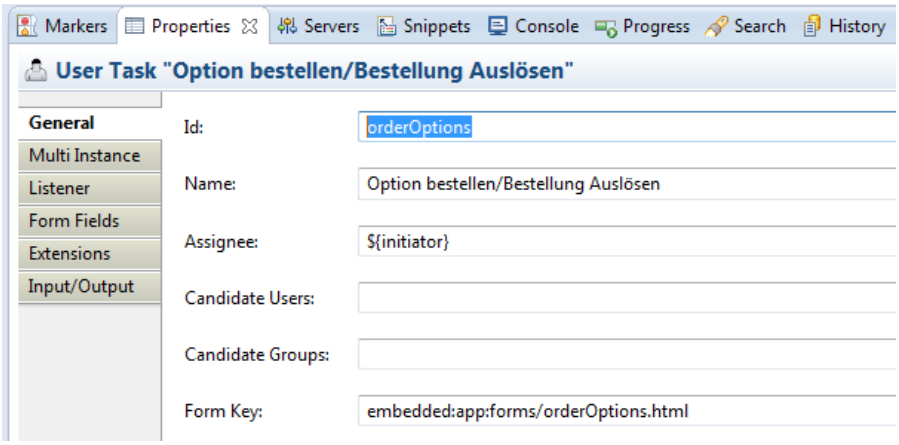

Abbildung 4.2. Taskname übersetzen: Task Id

Zuerst wird aus den Prozessvariablen die Browsersprache ausgelesen. Dann wird die Task Id, welche in Abbildung 4.2 zu sehen ist, in die Variable „taskDefinitionKey“ gespeichert. Es wird abhängig von der Browsersprache der Webservice entweder mit dem Parameter für Deutsch oder Englisch aufgerufen. Genauere Informationen zu dem Webservice sind im Unterkapitel 4.2.20 zu finden. Der Webservice gibt dann die Übersetzung für diesen Task zurück und dann wird der

```
public void notify(DelegateTask delegateTask) {
    String browserlanguage =  delegateExecution
      .getVariable("language").toString();
    String taskDefinitionKey = delegateTask
      .getTaskDefinitionKey();
    String taskName;

    optionStoreStub = optionStoreServiceLocator
      .getBasicHttpBinding_IOptionStore();
    if (browserlanguage.contains("DE")){
      taskName = optionStoreStub
        .getTranslation(taskDefinitionKey, "DE");
    } else {
      taskName = optionStoreStub
        .getTranslation(taskDefinitionKey, "EN");
    }
    delegateTask.setName(taskName);
}
```

Listing 4.18. Java Code für Übersetzung von Tasknamen

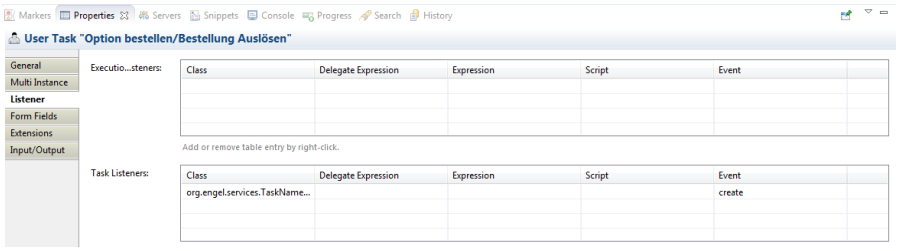

Abbildung 4.3. Taskname übersetzen: Tasklistener hinzufügen

Name des Tasks mit dieser Übersetzung überschrieben. Somit können auch die Tasknamen sprachabhängig definiert werden.

Damit die Klasse und somit auch die Methode zum Übersetzen aufgerufen wird, muss bei jedem Benutzer-Task, der übersetzt werden soll, diese Klasse als Tasklistener hinzugefügt werden, siehe Abbildung 4.3.

Da es einige Zeit dauern kann, bis die Webservices, speziell der Service „Software generieren“, eine Antwort zurückliefern, ist es notwendig, einige Konfigurationen durchzuführen. Für den Webservice-Aufruf selbst muss in der Klasse, indem das Binding für den Webservice definiert ist, das Timeout für den Aufruf hinaufgesetzt werden. Beim Webservice OptionStore muss in der Klasse „BasicHttpBinding_IOptionStoreStub“ für den gewünschten Aufruf das Timeout erhöht werden. In Listing 4.19 ist hierzu ein Beispiel zu sehen.

In diesem Listing geht es konkret um den Service-Aufruf „Upgrade Projekt“. Hier wird der Aufruf erstellt und es werden einige Eigenschaften konfiguriert. Die Zeile „_call.setTimeout(36000000);“ wurde hinzugefügt, um das Timeout für diesen Aufruf auf 10 Stunden einzustellen.

Auch der JBoss Server selbst hat ein Timeout für solche Operationen. Weiters versucht der JBoss Server nach einer definierten Zeit, den Aufruf erneut zu senden, falls er innerhalb einer bestimmten Periode keine Antwort bekommen hat. Das heißt, auch hier sind Konfigurationen für die Webservice-Aufrufe notwendig. Hierzu werden die nachfolgenden Einstellungen in der „standalone.xml“ Konfigurationsdatei getätigt.

```
public java.lang.Boolean upgradeProject(...)
  throws java.rmi.RemoteException {
    org.apache.axis.client.Call _call = createCall();
    _call.setOperation(_operations[9]);
    _call.setUseSOAPAction(true);
    ...
    _call.setTimeout(36000000);
```

Listing 4.19. Java Code für Timeout von Webservice Binding

```
<subsystem xmlns="urn:jboss:domain:transactions:1.3">
  <core-environment>
    <process-id>
      <uuid/>
    </process-id>
  </core-environment>
  <recovery-environment
     socket-binding="txn-recovery-environment"
     status-socket-binding="txn-status-manager"/>
  <coordinator-environment default-timeout="36000"/>
</subsystem
```

Listing 4.20. Konfiguration JBoss Transaction Timeout

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.4">
  ...
  <pools>
    <bean-instance-pools>
      <strict-max-pool name="slsb-strict-max-pool"
        max-pool-size="20"
        instance-acquisition-timeout="600"
        instance-acquisition-timeout-unit="MINUTES"/>
    </bean-instance-pools>
  </pools>
  ...
</subsystem>
```

Listing 4.21. Konfiguration JBoss EJB Timeout

In Listing 4.20 ist zu sehen, wie das Timeout für die Transaktionen gesetzt wird. Der Wert wird in Sekunden angegeben und das Timeout wird auf 36000 Sekunden und somit auf 10 Stunden gesetzt.

Zusätzlich wurden die Timeout-Einstellungen für EJB erhöht, wie in Listing 4.21 zu sehen ist. Das Timeout wurde hier ebenfalls auf 10 Stunden eingestellt.

Am JBoss Server gibt es auch für Camunda eine Einstellung, die getätigt werden muss, damit diese Einstellung auch an die längere Dauer der Webservice-Aufrufe angepasst ist. In Listing 4.22 wird gezeigt, wie die „lockTime“ auf 10 Stunden gesetzt wird. Diese Einstellung bezieht sich auf den „job-executor“, welcher für das Ausführen des Aufruf des Webservices zuständig ist. Diese „lockTime“ entspricht der Wartezeit, bis der „job-executor“ den Aufruf erneut ausführt, falls er in dieser Wartezeit noch keine Antwort zurückbekommen hat. Das heißt, ist diese Wartezeit niedriger als die Zeit die der Webservice für eine Antwort benötigt, so wird ein erneuter Aufruf gemacht. Dies führt jedoch zu einem Problem, wenn die Antworten vom Webservice zurückkommen, denn dann gibt es zu einer Prozessinstanz und speziell zu dem Task, der den Aufruf durchgeführt hat, mehrere

```
<subsystem xmlns="urn:org.camunda.bpm.jboss:1.1">
  ...
  <job-executor>
  ...
    <property name="lockTimeInMillis">
      36000000
    </property>
  ...
  </job-executor>
  ...
</subsystem>
```

Listing 4.22. Konfiguration JBoss Camunda „job-executor“

Antworten und der Prozess weiß dann nicht, welche Antwort die richtige ist. Daher ist es wichtig, dass diese Zeit auch auf den selben Wert wie die Timeouts gesetzt werden, denn wenn ein Timeout entsteht, wird der aktuelle Aufruf abgebrochen und ein neuer Versuch gestartet.

In diesem Kapitel wurden nun die Implementierungsdetails mit den Prozessvariablen, den schon umgesetzten und noch nicht umgesetzten Tasks sowie die Anpassungen und Konfigurationen erklärt. Im nächsten Kapitel wird die Systemarchitektur dargestellt.

4.1.2 Systemarchitektur

In der Abbildung 4.4 ist die Architektur des erstellten Systems zu sehen. Das System ist auf dem EMSSTARTUP Server installiert. Dieser Server ist ein Microsoft Windows Server.

Auf diesem Server laufen die drei erstellten WCF Webservices OptionStore, IMMSGServiceWeb und SAPServiceProvider. Diese Dienste kommunizieren miteinander mittels Simple Object Access Protocol (SOAP). Mit dem SAPServiceProvider wird ein Hilfsdienst zur Verfügung gestellt, welcher mittels SOAP- und einer .net-Schnittstelle auf das Engel SAP-System zugreift. Dieser Service stellt Funktionalitäten zur Verfügung, welche im Webstore benötigt werden, um diverse Schreib- und Leseoperationen im SAP durchzuführen. Der IMMSGServiceWeb Webservice wird hauptsächlich für die Software-Generierung benötigt, und dieser selbst wiederum benötigt den SAPServiceProvider, um auf SAP zuzugreifen. Der OptionStore Webservice dient als Koordinator und als Schnittstelle zwischen IMMSGServiceWeb, SAPServiceProvider und der UserRegistration und dem Webstore. Somit stellt der OptionStore die für den Webstore und die Benutzerregistrierung benötigten Services zur Verfügung und ruft wiederum Funktionalitäten des IMMSGServiceWeb und des SAPServiceProvider auf.

Auf dem EMSSTARTUP Server ist ein JBoss Server installiert, auf welchem die Camunda webapp, der Webstore und die UserRegistration laufen. Die UserRegistration ist eine Webapplikation, welche für die Benutzerregistrierung zuständig

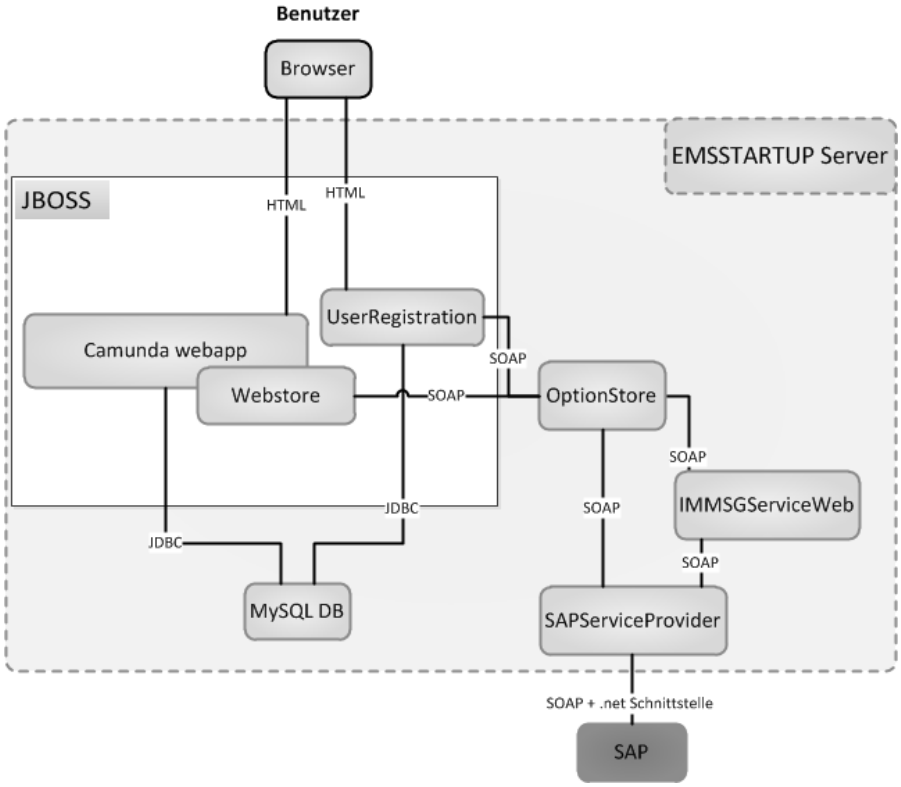

Abbildung 4.4. Systemarchitektur

ist. Diese Applikation verwendet den OptionStore mittels SOAP, um zu überprüfen, ob der Benutzer berechtigt ist, sich im System zu registrieren. Nähere Informationen dazu sind im Kapitel 4.2 zu finden. Weiters greift diese Applikation mittels eines JDBC Drivers auf eine MySQL Datenbank zu, welche ebenfalls auf dem EMSSTARTUP Server installiert ist. In diese Datenbank werden die Benutzerdaten geschrieben und später von der Camunda webapp wiederum mittels JDBC gelesen und verwendet. Die Benutzer können mittels eines Browsers die Applikation zur Benutzerregistrierung verwenden.

Wie schon erwähnt, läuft auf dem JBoss Server die Camunda webapp und der Webstore. Die Camunda webapp stellt eine Taskliste zum Starten von Prozessen und Abarbeiten von Prozessschritten, ein Cockpit zum Überwachen von Prozessen und eine Administrationsoberfläche für die Verwaltung von Benutzern und deren Rechten zur Verfügung. Der Benutzer kann diese Oberflächen mittels eines Browsers aufrufen und verwenden. Mit dem Camunda Modellierungswerkzeug wurde ein Prozess erstellt und automatisiert. Dieser automatisierte Prozess wird als Webstore am JBoss Server installiert und zur Camunda webapp hinzugehängt.

```
bool CheckSAPLoginCredentials(string username);
```

Listing 4.23. Schnittstelle für die Funktion: SAP-Logindaten überprüfen

```
bool CheckCustomerNoMaschineNo(string customerNo,
                    string maschineNo);
```

Listing 4.24. Schnittstelle für die Funktion: KundenNr und MaschinenNr überprüfen

Somit können Benutzer über die Taskliste diesen Prozess starten und den Prozess wie in Kapitel 2 ersichtlich durchführen.

4.2 Services

In diesem Kapitel werden nun die Services bzw. Funktionen beschrieben, welche von den Webservices zur Verfügung gestellt und für den Prozess benötigt werden. Es werden auch Funktionen kurz erklärt, für die bereits die Schnittstellen generiert worden sind, um den Prozess später schnell und einfach erweitern zu können.

4.2.1 SAP Logindaten überprüfen

Mit diesem Service wird überprüft, ob der User berechtigt ist, den Prozess auszuführen. In Listing 4.23 ist die Schnittstelle zu dieser Funktion ersichtlich.

Der Benutzername wird als Übergabeparameter benötigt und der Service verwendet diesen, um im Active Directory nachzusehen, ob dieser Benutzer existiert. Anhand dieser Kontrolle liefert der Service dann eine Boolsche Variable zurück, welche angibt, ob der Benutzer existiert oder nicht.

4.2.2 KundenNr und MaschinenNr überprüfen

Um zu überprüfen, ob eine Formgebungsmaschine einem Kunden gehört, wird dieser Service verwendet. Diese Funktion kann mittels der in Listing 4.24 ersichtlichen Schnittstelle aufgerufen werden.

Die Funktion bekommt die Maschinennummer und die Kundennummer als Übergabeparameter und gibt dann zurück, ob diese Maschine dem Kunden zugeordnet ist oder nicht. Derzeit ist diese Funktion noch nicht umgesetzt, da diese nur für externe Benutzer benötigt wird. Um diesen Service anbieten zu können, müsste ein Zugriff auf SAP erfolgen, welcher die Zuordnung zwischen Kundennummer und Maschinennummer überprüft.

```
List<InstalledOption> GetInstalledOptions(
                    string maschineNo);
```

Listing 4.25. Schnittstelle für die Funktion: Liste von installierten Optionen holen

4.2.3 Liste von installierten Optionen holen

Die Liste von installierten Optionen wird durch diesen Service zurückgeliefert. Mit der in Listing 4.25 ersichtlichen Schnittstelle wird dieser Service verwendet.

Als Übergabeparameter ist hier nur die Maschinennummer notwendig. Grundsätzlich wurde die Liste von installierten Optionen folgendermaßen erhoben: Es ist notwendig, drei XML-Dateien, die teilweise 70 Megabyte groß sind, zu analysieren. In der ersten XML-Datei sind zu jeder Maschinennummer alle Optionsnummern aufgelistet, welche auf dieser Maschine installiert sind. Somit entsteht eine Liste von Optionsnummern. Die zweite XML-Datei enthält Detailinformationen zu jeder Optionsnummer, die existiert, d.h. aus dieser XML-Datei werden dann die Detailinformationen herausgeholt. Um nun auch die Detailinformationen der zugehörigen Software zu bekommen, wird die dritte Liste benötigt. Findet man in dieser Liste einen passenden Eintrag, so wird eine installierte Option erzeugt und in eine Liste eingetragen, welche dann als Rückgabewert zurückgegeben wird.

Es wurde jedoch festgestellt, dass die Zuordnung der letzten beiden XML-Dateien derzeit nicht eindeutig erfolgen kann, da nicht ersichtlich ist, welche Software zu welcher Option gehört. Es handelt sich hier um eine 1:N Beziehung, somit kann eine Software einer von mehreren Optionen zugeordnet sein. Bei der initialen Generierung und Installation ist es klar, welche Software zu welcher Option gehört, jedoch ist diese Information später nicht reproduzierbar. Da für diesen Prototyp die Hardware-Optionen nicht relevant sind, wurde die Entscheidung getroffen, einstweilen einen anderen Ansatz zu wählen. Und zwar wird ab sofort, sobald für eine Maschine eine Software generiert wird, auch bei der initialen Generierung am Server hinterlegt, welche Software-Optionen auf dieser Maschine installiert sind. Da es derzeit ohnehin nicht möglich ist, nachträglich eine Software-Option eindeutig einer Option zuzuordnen und auch die Hardware-Optionen nicht berücksichtigt werden, wird einstweilen nicht mehr auf die XML-Dateien zugegriffen, sondern gleich direkt auf die Information, die ab sofort am Server hinterlegt wird.

4.2.4 Liste von möglichen Optionen holen

In Listing 4.26 ist die Schnittstelle zu der Funktion „Liste von möglichen Optionen holen“ ersichtlich. Mit diesem Service wird die Liste von Optionen, die auf der Maschine installiert werden können, evaluiert.

Hier wird wiederum nur die Maschinennummer als Übergabeparameter benötigt. In einer derzeit statischen XML-Datei wird eine Liste von Optionen angelegt, welche für die Evaluierung in Frage kommen, d.h. mittels dieser Liste

```
List<PossibleOption> GetPossibleOptions(
                    string maschineNo);
```

Listing 4.26. Schnittstelle für die Funktion: Liste von möglichen Optionen holen

```
List<PriceListOption> GetPriceListOptions();
```

Listing 4.27. Schnittstelle für die Funktion: Preisliste von CSD holen

wird grundsätzlich bestimmt, welche Optionen zur Verfügung stehen sollen. Der Grundgedanke dabei ist, dass diese Liste wachsen soll und laufend um Optionen ergänzt wird. Diese Liste enthält auch die Standard-Detailinformationen für die Optionen. In dem Service wird diese Liste als Ausgangsbasis verwendet und dann werden Überprüfungen durchgeführt, um festzustellen, ob diese Optionen auf der übergebenen Maschine möglich sind oder nicht. Zum Beispiel wird überprüft, ob die Version und der Typ der Maschine die Installation der Option zulässt. Die Optionen, welche nach der Überprüfung noch übrig sind, werden als Liste zurückgeliefert.

4.2.5 Preisliste von CSD holen

Mit dieser Funktion soll eine aktuelle Preisliste vom CSD bzw. der Vertriebsabteilung eingelesen werden. Diese Preisliste enthält kundenspezifische Preis- und Lieferzeitinformationen zu den einzelnen Optionen. Diese Preisliste ist derzeit noch in Arbeit und daher ist in diesem Service einstweilen nur die Schnittstelle vorhanden. In Listing 4.27 ist die Schnittstelle ersichtlich, um diese Funktion aufzurufen.

Derzeit enthält diese Schnittstelle keinen Übergabeparameter. Es ist auch derzeit noch unklar, ob überhaupt ein Übergabeparameter benötigt wird. Wenn welche benötigt werden, wird wahrscheinlich die Kundennummer sowie die Fabrikationsnummer oder eine Liste von Optionen als Parameter übergeben werden. Zurückgegeben wird dann eine Liste, welche die Preis- und Lieferzeitinformationen zu den Optionen enthält.

4.2.6 Optionsliste generieren

Dieser Service ist auch noch nicht umgesetzt worden, da es die im Unterkapitel 4.2.5 erwähnte Preisliste noch nicht gibt. Mit dieser Funktion sollen die drei Listen installierte Optionen, mögliche Optionen und Preis- und Lieferzeitinformationen miteinander verknüpft und ergänzt werden. Das heißt, zum Beispiel die Standardpreise der möglichen Optionen werden durch die Preise der Preisliste ersetzt. In Listing 4.28 ist die Schnittstelle für diesen Service ersichtlich.

Als Übergabeparameter werden die drei Listen installierte Optionen, mögliche Optionen und Preis- und Lieferzeitinformationen erwartet. Der Rückgabewert ist

```
List<option> GetOptionsList(
        List<InstalledOption> installedOption,
        List<PossibleOption> possibleOption,
        List<PriceListOption> priceListOption);
```

Listing 4.28. Schnittstelle für die Funktion: Optionsliste generieren

```
public bool LoadSourceFromSAP(string fabNo);
private bool LoadSourceFromSAP(Parameters par);
```

Listing 4.29. Beispiel für die Paramater anstelle der Fabrikationsnummer

eine neue Liste, welche die Verknüpfung der drei Listen darstellt. Dieser Service liefert jedoch noch keine Funktionalität.

4.2.7 Upgrade Projekt

Dieser Service führt die gesamte Generierung der Software sowie die Auslieferung durch. Um Ressourcen zu sparen und performanter zu sein, wurde dieser Service erstellt, welcher dann gleich nacheinander die Schritte des Software-Generieren und der Auslieferung durchführt. Es gibt auch die Möglichkeit, jeden Schritt einzeln aufzurufen, jedoch würde dies um einiges länger dauern und würde um einiges mehr an Ressourcen beanspruchen. Der Grund dafür ist, dass die einzelnen Services so gestaltet werden müssen, dass sie unabhängig voneinander funktionieren. Jeder dieser Services benötigt jedoch einige Parameter, welche in jedem Service zuerst gesetzt werden müssen. Daher wurde für jeden dieser Services eine nicht öffentliche Schnittstelle erstellt, welche anstelle der Fabrikationsnummern ein Objekt übergeben bekommt, das die Parameter enthält. Mit nicht öffentlicher Schnittstelle ist gemeint, dass diese nur innerhalb des Websevices aufgerufen werden kann, jedoch nicht von außerhalb. In Listing 4.29 wird ein Beispiel gezeigt, wie dies aussieht. Die erste Schnittstelle ist jene, die von außen zugänglich ist und als Übergabewert die Fabrikationsnummer erhält, und die zweite Schnittstelle, die nicht von Außen zugänglich ist,ist jene, welche die Parameter als Übergabewert verlangt. Erstere liest sich die Parameter mittels der Fabrikationsnummer aus und ruft dann die zweite, interne Schnittstelle auf. Somit wird im Prinzip in beiden Fällen der gleiche Code ausgeführt, jedoch müssen bei einem Fall vorher die Parameter eingelesen werden, was beim anderen erspart bleibt.

In Listing 4.30 ist die Schnittstelle zum Upgrade Projekt zu sehen.

Als Übergabeparameter werden die Maschinennummer, die selektierten möglichen Optionen, die Email-Adresse des Benutzers und die Sprache übergeben. Dieser Service liest zuerst die Parameter ein und ruft dann nach und nach die internen Schnittstellen der Services auf, die benötigt werden, um die Software zu generieren und auszuliefern. Zuerst wird der Service „Finale Sicherung aus SAP laden“ aufgerufen, um die finale Sicherung aus SAP zu laden. Als nächstes wird die

```
bool UpgradeProject(string maschineNo,
    List<PossibleOption> selectedPossibleOptions,
    string receiver, string language);
```

Listing 4.30. Schnittstelle für die Funktion: Upgrade Projekt

```
bool LoadFinalFromSAP(string fabNo);
```

Listing 4.31. Schnittstelle für die Funktion: Finale Sicherung aus SAP laden

Projektumgebung mit dem Service „Projektumgebung vorbereiten“ vorbereitet. Danach wird die Optionsliste mittels der Funktion „Optionsliste laden (SAPProject.pcl)“ geladen und mit der Funktion „Optionsliste um neue Option(en) ergänzen“ um die neuen Optionen ergänzt. Nachdem die Optionsliste ergänzt wurde, wird der Source für alle Optionen, inklusive der ergänzten Optionen, mit dem Service „Source aus SAP laden“ aus SAP geladen. Nun kann das Projekt durch den Service „(IMM) Projekt erstellen, kompilieren“ erstellt und kompiliert werden. Mit diesem Schritt wird auch gleich der Service „Target generieren“ durchgeführt. Danach wird das Paket durch den Service „Package auf FTP-Server hochladen“ auf den FTP Server hochgeladen und mittels „Über DFÜ ausliefern“ per Email versendet. Wurde die Email erfolgreich versandt, dann werden noch die Services ausgeführt, welche die Backup-Sicherung abspeichern. Um die finalen Dateien zu sichern, wird der Service „Finale Sicherung verspeichern“ verwendet und für die Source Dateien der „Source Sicherung verspeichern“-Service. Genauere Informationen zu all diesen Services sind in den folgenden Unterkapiteln zu finden.

4.2.8 Finale Sicherung aus SAP laden

Dieser Service lädt die finale Sicherung der Maschine aus dem SAP heraus. Diese Sicherung enthält alle bereits installierten Software-Optionen. In Listing 4.31 ist die Schnittstelle zu sehen, mit der diese Funktion aufgerufen wird.

Als Übergabeparameter reicht die Maschinennummer. Zuerst müssen die Parameter geladen werden. Danach wird die private Schnittstelle aufgerufen, welche als Übergabeparameter diese Parameter benötigt. Diese private Funktion übernimmt dann die Arbeit. Anhand der Parameter wird im SAP nach der aktuellsten finalen Backupsicherung gesucht und diese heruntergeladen. Diese Sicherung ist in einer Zip-Datei verpackt. Nach dem Herunterladen wird die Zip-Datei entpackt und somit kann auf die Daten der aktuellen Backupsicherung zugegriffen werden. Als Rückgabewert wird zurückgegeben, ob das Backup erfolgreich geladen und entpackt werden konnte.

```
bool PrepareProjectEnvironment(string fabNo);
```

Listing 4.32. Schnittstelle für die Funktion: Projektumgebung vorbereiten

```
bool ExtendOptionsListSAPProject(string fabNo,
    List<PossibleOption> selectedPossibleOptions);
```

Listing 4.33. Schnittstelle für die Funktion: Optionsliste um neue Option(en) ergänzen

```
bool LoadSourceFromSAP(string fabNo);
```

Listing 4.34. Schnittstelle für die Funktion: Source aus SAP laden

4.2.9 Projektumgebung vorbereiten

In Listing 4.32 ist die Schnittstelle zu sehen, um die Projektumgebung vorzubereiten.

Wie schon beim Laden der finalen Sicherung wird auch hier die Fabrikationsnummer übergeben. Zunächst werden wieder die Parameter geladen und dann wird die zugehörige private Funktion aufgerufen, welche diese Parameter als Übergabewert benötigt. Mit dieser Funktion wird die Projektstruktur vorbereitet, d.h. die Projekt-Metadateien werden vom Verzeichnis der finalen Sicherung in das Verzeichnis für das Sourceprojekt geladen.

4.2.10 Optionsliste um neue Option(en) ergänzen

Mit diesem Service wird die aktuelle Optionsliste um die neuen Optionen ergänzt. Listing 4.33 zeigt die Schnittstelle zu diesem Service.

Zusätzlich zu der Maschinennummer wird hier die Liste der selektierten möglichen Optionen mitübergeben. Wie gehabt werden dann wieder die Parameter geladen und die private Funktion aufgerufen. Die aktuelle Optionsliste befindet sich in der finalen Sicherung und enthält Informationen über die bereits installierten Optionen. Eine Erfolgsmeldung wird zurückgeliefert, sobald der Service die Liste erfolgreich ergänzt hat.

4.2.11 Source aus SAP laden

Dieser Service dient dazu, die Source-Dateien aus SAP zu laden, welche benötigt werden, um die Software erstellen und generieren zu können. Listing 4.34 zeigt die Schnittstelle zu dieser Funktion.

Als Übergabeparameter dient wieder die Maschinennummer und es werden auch wieder die Parameter geladen, bevor die private Funktion aufgerufen wird.

```
bool MakeProjectIMM(string fabNo);
```

Listing 4.35. Schnittstelle für die Funktion (IMM) Projekt erstellen

```
bool TriggerFinalBackup(string fabNo, string controlIndex);
```

Listing 4.36. Schnittstelle für die Funktion: Finale Sicherung verspeichern

Anhand der erweiterten bzw. ergänzten Optionsliste, welche sich auf dem Server befindet, werden nun alle Source-Dateien aus SAP geladen. Das heißt, in der Optionsliste sind alle Optionen aufgelistet, welche installiert werden sollen, und um diese zu installieren, wird der Source Code benötigt. Zurückgegeben wird, ob der Source erfolgreich aus SAP geladen werden konnte oder nicht.

4.2.12 (IMM) Projekt erstellen, kompilieren

Mit diesem Service wird das Projekt gebaut, der Build Prozess durchgeführt und alles kompiliert. In Listing 4.35 ist die Schnittstelle zu sehen, um diesen Service aufzurufen.

Hier reicht wiederum die Maschinennummer als Übergabeparameter. Zusätzlich zum Projekt bauen und kompilieren wird hier auch gleich das Target generieren durchgeführt, d.h. der Service „Target generieren“ ist in diesen Service integriert. Somit werden auch die Software-Pakete generiert und gebaut. Alle Einzelpakete werden zusammengefasst zu einem Update-Paket.

4.2.13 Target generieren

Dieser Service wurde nicht einzeln erstellt, sondern in den Service „(IMM) Projekt erstellen, kompilieren“ integriert. Der Grund für die Integration in den anderen Service ist, dass es keinen Sinn ergibt, das Generieren des Targets alleine auszuführen, beziehungsweise weil das Target-Generieren sowieso nur nach dem Projekt erstellen und kompilieren ausgeführt werden kann. Das heißt, dieser Service kann nicht ohne den anderen verwendet werden, und es ist auch nicht sinnvoll, den anderen Service alleine auszuführen, weil man damit nicht wirklich etwas anfangen kann.

4.2.14 Finale Sicherung verspeichern

Mit der in Listing 4.36 ersichtlichen Schnittstelle wird der Service zum Verspeichern der finalen Sicherung aufgerufen.

Übergeben wird die Fabrikationsnummer und es kann auch ein Kontrollindex übergeben werden. Dieser Kontrollindex ist derzeit optional und standardmäßig

```
bool TriggerSourceBackup(string fabNo);
```

Listing 4.37. Schnittstelle für die Funktion: Source Sicherung verspeichern

„1“. Da sich dies jedoch ändern kann, wurde die Möglichkeit geschaffen, den Kontrollindex zu übergeben.

Mit diesem Service wird die finale Sicherung in SAP verspeichert bzw. die Speicherung wird angetriggert. Es existiert ein Service, der dieses Speichern übernimmt, er benötigt sozusagen nur einen Auftrag, sodass er mit dem Verspeichern beginnt. Hierzu reicht es aus, dass in einem bestimmten Ordner eine Datei angelegt wird. Anhand des Dateinamens erkennt dieser bereits existierende Service, um welche Maschine es sich handelt, und somit weiß er auch genau, was er abspeichern und als Backup hinterlegen muss. Jede halbe Stunde läuft der Prozess durch und sieht nach, ob eine Datei angelegt wurde und ob er etwas abarbeiten muss. Nachdem das Backup erfolgreich verspeichert wurde, wird die Datei gelöscht. Für den Service „Finale Sicherung verspeichern“ heißt dies, es muss einfach nur die Datei angelegt werden und um den Rest kümmert sich der andere Service.

4.2.15 Source Sicherung verspeichern

Dieser Service funktioniert ähnlich wie der Service „Finale Sicherung verspeichern“. Die Schnittstelle, um den Service zu verwenden, ist in Listing 4.37 zu sehen.

Als Übergabeparameter reicht hier die Maschinennummer. Mit diesem Service werden die Sourcen, aus welchen die Software generiert worden ist, als Backup in SAP abgespeichert. Es wird eine Zip-Datei erstellt, welche die Sourcen enthält, die abgespeichert werden sollen. Handelt es sich bei der Software, die generiert worden ist, um reine Standardsoftware, dann ist es nicht notwendig, die gesamten Sourcen in die Zip-Datei zu packen. In diesem Fall wird eine Zip-Datei mit dem Inhalt „Standard“ erzeugt und diese abgespeichert. Standardsoftware ist unverändert und nicht maschinenabhängig, und daher muss diese nicht bei jeder Maschine abgespeichert werden, sondern es reicht hier ein Vermerk, dass es sich um reine Standardoptionen handelt.

4.2.16 Package auf FTP-Server hochladen

Damit die Software ausgeliefert werden kann, muss diese auf den FTP-Server hochgeladen werden. Mit der in Listing 4.38 ersichtlichen Schnittstelle kann diese Funktion aufgerufen werden.

Mit der Maschinennummer als Übergabeparameter weiß dieser Service, welche Datei er auf den FTP-Server hochladen soll. Nachdem die Datei erfolgreich hochgeladen worden ist, wird der Link zu der Datei am FTP-Server zurückgegeben.

```
string UploadUpdatePackage(string fabNo);
```

Listing 4.38. Schnittstelle für die Funktion: Package auf FTP-Server hochladen

```
bool TestOptions(string fabNo, string pathUpdatePackage);
```

Listing 4.39. Schnittstelle für die Funktion: Optionen testen

```
bool SendEmail(string fabNo, string message,
        string receiver, string subject);
```

Listing 4.40. Schnittstelle für die Funktion: Über DFÜ ausliefern

4.2.17 Optionen testen

Diese Funktion ist über die Schnittstelle, welche in Listing 4.39 ersichtlich ist, zugänglich.

An dieser Schnittstelle werden die Fabrikationsnummer und der Pfad zum Update-Paket angegeben. Dieser Service ist noch nicht umgesetzt. Der Service sollte das Update-Paket von dem Pfad abholen und auf einer virtuellen Maschine installieren. Diese virtuelle Maschine sollte vorher auf den Stand der Maschine mit der übergebenen Fabrikationsnummer gebracht werden. Die virtuelle Maschine repräsentiert dann die Maschine, für welches das Update bestimmt ist. Auf dieser Maschine sollten dann automatisierte Tests durchgeführt werden. Es sollte dann zurückgegeben werden, ob die Tests erfolgreich waren oder nicht.

4.2.18 Über DFÜ ausliefern

Mittels Datenfernübertragung, kurz DFÜ, liefert dieser Service das Update-Paket aus. Die Schnittstelle für den Aufruf ist in Listing 4.40 ersichtlich.

Als Übergabeparameter werden hier die Fabrikationsnummer, eine Nachricht, der Empfänger und der Betreff benötigt. Die Nachricht soll den Link zum Update-Paket am FTP-Server enthalten. Nachdem der Service „Package auf FTP-Server hochladen“ ausgeführt worden ist, wird der Link zum Server, auf de sich das Paket befindet zurückgegeben. Dieser Link soll dann in eine Nachricht verpackt werden, die an diese Auslieferungs-Funktion übergeben wird. Als Betreff wird ein Text übergeben, dieser kann z. B. folgendermaßen lauten: „Optionserweiterung ENGEL Spritzgießmaschine Fabrikationsnummer 123456“. Als Empfänger wird die Email-Adresse des Benutzers mitübergeben. Nachdem die Email erfolgreich versandt wurde, wird eine Erfolgsmeldung zurückgegeben.

```
bool UpdateInstalled(string fabNo);
```

Listing 4.41. Schnittstelle für die Funktion: Installation in SAP vermerken

```
string GetTranslation(string key, string language);
```

Listing 4.42. Schnittstelle für die Funktion: Tasknamen übersetzen

4.2.19 Installation in SAP vermerken

Dieser Service ist noch nicht umgesetzt. Jedoch wurde die Schnittstelle, welche in Listing 4.41 ersichtlich ist, schon vorbereitet.

Übergeben wird wiederum die Fabrikationsnummer. Dieser Service soll benutzt werden, um festzuhalten, dass die Option(en) auf der Maschine tatsächlich installiert worden ist/sind. Da es derzeit noch keine Möglichkeit gibt bzw. nirgends ein Datenbankfeld oder ähnliches vorhanden ist, um dies zu kennzeichnen, konnte dieser Service noch nicht umgesetzt werden.

4.2.20 Tasknamen übersetzen

Dieser Service wird benötigt, um für die Tasknamen eine geeignete Übersetzung zu bekommen. In Listing 4.42 ist die Schnittstelle zu diesem Service zu sehen.

Übergeben wird ein Key, welcher der Task Id entspricht, sowie das Kürzel der Sprache, in welche die Übersetzung erfolgen soll. Dieser Service liest dann aus dem richtigen Sprachfile die Übersetzung für die angeforderte Sprache anhand des übergebenen Keys aus und gibt die Übersetzung zurück.

5

Fazit, Schlussfolgerung und Ausblick

Mit dieser Arbeit wurde ein Prototyp entworfen, welcher den wichtigsten Teil, die automatisierte Generierung von Software, des Patents mit der Nummer AT 514527 A1 2015-01-15 ausführt. Auch die Schritte vor und nach der Generierung wurden umgesetzt, um internen Benutzern zu ermöglichen, Software für Formgebungsmaschinen generieren zu lassen. Somit können Installationspakete für die Installation von Software-Zusatzoptionen erstellt werden. Mit dieser Arbeit wurde gezeigt, dass das Patent von [Fritz et al., 2013] umgesetzt werden kann. Der erstellte Prototyp wurde als experimenteller Prototyp begonnen und kann nun als evolutionärer Prototyp weiter verwendet werden, um die Grundfunktionalitäten die schon gegeben sind, zu erweitern.

Der Prototyp kann vor allem so erweitert werden, dass auch externe Benutzer, also Unternehmenskunden, diesen verwenden können. Hierzu ist es notwendig, die Aufgaben, die im Prozess schon für die externen Benutzer modelliert wurden, mit Funktionalität zu füllen. Bei einigen Aufgaben wird es auch notwendig sein noch Analysen in Form von Erhebungen bei den betroffenen Abteilungen durchzuführen.

Damit Benutzer den Prototypen bzw. das System verwenden können, müssen sie registriert sein. Hierfür wurde eine Applikation entworfen, welche die Registrierung übernimmt. Derzeit können sich nur interne Benutzer damit registrieren, d.h. diese Applikation müsste erweitert werden, damit sich auch externe Benutzer registrieren können. Externe Kunden müssen bei der Registrierung eine Kundennummer angeben und diese muss in der Datenbank gespeichert werden. In der Datenbank wurde diesbezüglich schon ein Feld angelegt.

Eine weitere große Erweiterung, ist den Prozess auszubauen, um auch Hardwarerelevante Optionen über diese Plattform bestellen zu können. Hier wurden ebenfalls die Schritte im Prozess modelliert, welche notwendig sind um den Prozess für Hardware-Optionen zu erweitern. Auch hier gilt, dass diese mit Funktionalität gefüllt und eventuell Erhebungen in den betroffenen Abteilungen durchgeführt werden müssen.

Ausbaumöglichkeiten gibt es auch noch für die Generierung der Software-Dokumentation. Diese Dokumentationserstellung exisitiert bereits unternehmensintern, es müsste erhoben werden, ob und wie diese in den Prozess integriert

werden kann. Auch im Bereich des automatischen Testens des Update-Pakets steckt noch großes Potential, den Prozess auszubauen.

Im Unterkapitel „Noch nicht umgesetzte Tasks“ sind alle noch nicht umgesetzten Aufgaben beschrieben. In den Kapiteln „Services“ und „Umgesetzte Tasks“ werden die Services und die schon umgesetzten Aufgaben erklärt. Es werden in diesen drei Kapiteln Informationen und Vorschläge für die Umsetzung der noch offenen Schritte aufgelistet und erklärt. Auch so manche Verbesserungsvorschläge werden in diesen Kapiteln erwähnt.

Für den Prototypen gibt es nicht nur Erweiterungs-, sondern auch Verbesserungsmöglichkeiten. Der Schritt, in dem die Optionslisten generiert und zusammengestellt werden, ist komplex und beinhaltet aufwendige Aufgaben. Für die Optimierung dieser Generierung könnten zum Beispiel für alle Fabrikationsnummern diese Listen schon vorab generiert und gespeichert werden. Somit könnten die Listen direkt abgefragt werden ohne jeglichen Generierungs-Aufwand zum Zeitpunkt der Abfrage.

Auch der Prozess könnte anders aufgebaut werden, damit dieser übersichtlicher und leichter verständlich wird. Hierzu könnte der Prozess in mehrere zum Teil unabhängige Teilprozesse unterteilt werden. So könnte zum Beispiel die Generierung der Software in einen eigenen Prozess ausgelagert werden. Dann könnte der Hauptprozess den Prozess des Software-Generierens einfach verwenden. Ebenso könnten auch noch unabhängige andere Prozessteile, wie die Auslieferung, ausgelagert werden. Eine andere Variante wäre, den Prozess so aufzusplitten, dass die gesamte Bearbeitung, welche unternehmensintern abläuft, in einem eigenen Prozess modelliert wird. Um herauszufinden, wie die Aufteilung am besten ist oder ob es überhaupt sinnvoll ist, den Prozess aufzuteilen, müsste evaluiert und erhoben werden, welche Teile unabhängig vom gesamten Prozess existieren können. Weiters müsste auch darüber nachgedacht werden, ob diese Teile auch sinnvoll von etwaigen späteren anderen Prozessen, genutzt werden könnten.

Derzeit ist der Prozess so gestaltet, dass die Benutzer selbst den Prozess nicht mehr abbrechen können. Das heißt, derzeit bleibt eine laufende Prozessinstanz so lange bestehen, bis diese Instanz entweder abgeschlossen ist oder ein Administrator die Instanz entfernt hat. Der Prozess bzw. die Umsetzung könnte so erweitert werden, dass der Benutzer den Prozess abbrechen kann. Hierbei muss aber beachtet werden, dass ein Abbruch nur bis zur Bestellung möglich ist. Sobald die Bestellung abgeschickt und die Bestätigung angekommen ist, darf der Benutzer den Prozess nicht mehr abbrechen können, da eine Bestellung verbindlich ist.

Manche Funktionen der Webservices benötigen viel Zeit, da sie komplexe Aufgaben erledigen müssen. Dadurch sind an vielen Stellen im System, die mit den Webservices verbunden sind, Probleme mit Timeouts und ähnlichem entstanden. Wie im Kapitel 4.1.1 beschrieben wird, wurden diese Probleme durch Hinaufsetzen der Timeouts gelöst. Diese Lösung funktioniert zwar, ist aber nicht sehr elegant. Eine wesentlich bessere Variante wäre es mit den Webservices asynchron kommunizieren zu können. Es müsste hierzu evaluiert werden, ob und wie es möglich ist die WCF Webservices umzubauen, damit sie asynchrone Kommunikation zulassen. Des Weiteren müsste evaluiert werden, ob diese Webservice-Aufrufe in

den Webservices sowie die Service-Aufrufe in Camunda auch asynchron gestaltet werden können.

Eine weitere Verbesserungsmöglichkeit liegt im Bereich des Designs der Benutzeroberfläche der Camunda Taskliste sowie der Webapplikation, welche die Benutzerregistrierung übernimmt. Bei der Webapplikation für die Benutzerregistrierung wurde noch in keiner Weise auf das Aussehen der Benutzeroberfläche geachtet, da nur ein Prototyp für interne Benutzer entworfen worden ist. In dieser Webapplikation werden XHTML-Seiten für das Anzeigen des Seiteninhalts verwendet, d.h. mittels CSS könnte das Design verbessert werden.

Das Design in der Taskliste von Camunda ist mit dieser Arbeit schon etwas angepasst worden. Es wurde zum Beispiel das rote Camunda Design auf die Unternehmensfarben Grün und Schwarz angepasst. Auch das Camunda Logo wurde durch das Unternehmenslogo ersetzt. Dies sind jedoch rein experimentelle Umsetzungen, um zu zeigen, dass das Anpassen möglich ist und wie dies möglich ist, daher bedarf es noch der Verbesserung durch einen Web-Designer.

Bevor das System den Kunden zugänglich gemacht wird, sollten Sicherheitsaspekte und Risiken genauer betrachtet werden. Dafür sollte unter anderem analysiert werden, wann und wo Prozessvariablen gesetzt, angezeigt und verwendet werden. Dabei ist zu beachten, welche Variablen von den Benutzern gesehen und verändert werden dürfen. Ein weiterer Sicherheitsaspekt ist die Sicherheit des Systems selbst im Hinblick auf Passwörter, Protokolle und Firewall. Es gibt natürlich noch viele weitere Aspekte. Bevor diese Punkte im Detail betrachtet werden, ist eine Erhebung und Analyse sinnvoll, um alle für dieses System relevanten Aspekte zu ermitteln.

Sobald die Erweiterungen und Verbesserungen implementiert und getestet worden sind, eignet sich dieses System auch, um es den Kunden zugänglich zu machen.

Quellenverzeichnis

Activiti, 2015. Activiti (2015). Activiti Homepage. http://activiti.org/. Zugriff am 11.03.2015.

Alfesco-Activiti, 2015. Alfesco-Activiti (2015). Alfresco Activiti Homepage. http://www.alfresco.com/products/activiti. Zugriff am 11.03.2015.

Allweyer, 2010. Allweyer, T. (2010). *BPMN 2.0: introduction to the standard for business process modeling*. BoD–Books on Demand.

Behring and Poddebniak, 2015. Behring, R. and Poddebniak, D. (2015). Frontend Entwicklung mit AngularJS.

Bizagi, 2015. Bizagi (2015). Bizagi Homepage. http://www.bizagi.com/. Zugriff am 06.03.2015.

Bonitasoft, 2015. Bonitasoft (2015). Bonitasoft Homepage. http://www.bonitasoft.com/how-we-do-it/compare-editions. Zugriff am 06.03.2015.

BPMasters, 2015. BPMasters (2015). Kontaktperson von BPMasters. http://www.bpmasters.com/?lang=de. Kontakt via Xing im Februar 2015.

Camunda, 2015. Camunda (2015). Camunda Homepage. http://camunda.com/de/. Zugriff am 11.03.2015.

Dijkman et al., 2011. Dijkman, R., Hofstetter, J., and Koehler, J. (2011). Business Process Model and Notation–Third International Workshop, BPMN 2011. *Lecture Notes in Business Information Processing*, 95.

Dorst and BITKOM, 2012. Dorst, W. and BITKOM, E. (2012). Fabrik- und Produktionsprozesse der Industrie 4.0 im Jahr 2020. *IM Die Fachzeitschrift für Information, Management und Consulting, Nr*, (3):34–37.

Dumas et al., 2013. Dumas, M., La Rosa, M., Mendling, J., and Reijers, H. A. (2013). *Fundamentals of business process management*. Springer.

Engel, 2015a. Engel (2015a). Engel Austria GmbH CC300 Steuerung. http://www.engelglobal.com/de/at/cc-300.html. Zugriff am 10.06.2015.

Engel, 2015b. Engel (2015b). Engel Austria GmbH CC300 Steuerung Pressemitteilung. http://www.engelglobal.com/de/at/news-presse/news-pressemitteilungen/detail/news/detail/News/cc300dieneuesteuerungvonengel.html. Zugriff am 10.06.2015.

Fritz et al., 2013. Fritz, H., Lettner, G., and Kern, A. (2013). Verfahren zum Nachrüsten von Komponenten für eine Formgebungsmaschine. AT Patent Nr. AT 514527 A1 2015-01-15.

Gartner, 2015. Gartner (2015). Gartner IT Glossary Business Process Automation Homepage. http://www.gartner.com/it-glossary/bpa-business-process-automation. Zugriff am 08.06.2015.

GBTec, 2015. GBTec (2015). GBTec BIC Platform Homepage. http://www.gbtec.de/bpm-software/bic-platform.html. Zugriff am 11.03.2015.

Heinrich and Stelzer, 2009. Heinrich, L. J. and Stelzer, D. (2009). *Informationsmanagement. Grundlagen, Aufgaben, Methoden. 9. vollst. überarb. Aufl.* München: Oldenbourg.

IBM, 2015a. IBM (2015a). IBM Blueworks Homepage. https://www.blueworkslive.com/. Zugriff am 06.03.2015.

IBM, 2015b. IBM (2015b). IBM Business Process Manager Homepage. http://www-03.ibm.com/software/products/de/business-process-manager-family. Zugriff am 06.03.2015.

K2, 2015. K2 (2015). K2 BPM Homepage. http://www.k2.com/bpm-software. Zugriff am 06.03.2015.

Lavin, 2014. Lavin, J. (2014). *AngularJS Services*. Packt Publishing Ltd.

MDBA, 2015. MDBA (2015). Kontaktperson von MDBA Deutschland GmbH. http://www.mbda-careers.de/opencms/opencms/career/Unternehmen/MBDA-DE/. Kontakt via Xing im Februar 2015.

Microsoft, 2015. Microsoft (2015). Microsoft Windows Communication Foundation (WCF) Homepage. https://msdn.microsoft.com/de-de/library/dd456779%28v=vs.110%29.aspx. Zugriff am 08.06.2015.

Open-Text, 2015. Open-Text (2015). Kontaktperson von Open Text Austria GmbH. http://www.opentext.com/. Kontakt via Xing und telefonisches Gespräch Ende Februar/Anfang März 2015.

Oracle, 2015a. Oracle (2015a). Oracle Business Process Management Suite Homepage. http://www.oracle.com/us/technologies/bpm/suite/overview/index.html. Zugriff am 06.03.2015.

Oracle, 2015b. Oracle (2015b). Oracle SOA Suite Homepage. http://www.oracle.com/technetwork/middleware/soasuite/overview/index.html. Zugriff am 06.03.2015.

Pegasystems, 2015. Pegasystems (2015). Pegasystems Homepage. http://www.pega.com/. Zugriff am 06.03.2015.

Perceptive-Software, 2015. Perceptive-Software (2015). Perceptive Software BPMOne Homepage. http://www.perceptivesoftware.com/solutions/enterprise/business-process-management.html. Zugriff am 06.03.2015.

Selly et al., 2006. Selly, D., Troelsen, A., and Barnaby, T. (2006). Windows Communication Foundation. *Expert ASP. NET 2.0: Advanced Application Design*, pages 297–318.

Semtation, 2015. Semtation (2015). Semtation SemTalk Homepage. http://semtation.de/index.php/de/. Zugriff am 06.03.2015.

Sendler, 2013. Sendler, U. (2013). Industrie 4.0. *Berlin/Heidelberg*.

Shar and Tan, 2013. Shar, L. and Tan, H. B. K. (2013). Defeating SQL Injection. *Computer*, 46(3):69–77.

Signavio, 2015. Signavio (2015). Signavio Homepage. http://www.signavio.com/de/. Zugriff am 06.03.2015.

Ter Hofstede et al., 2009. Ter Hofstede, A. H., van der Aalst, W., Adams, M., and Russell, N. (2009). *Modern Business Process Automation: YAWL and its support environment*. Springer Science & Business Media.

w3schools, 2015. w3schools (2015). w2schools Homepage. http://www.w3schools.com/angular/angular_intro.asp. Zugriff am 11.03.2015.

Webratio, 2015. Webratio (2015). Webratio Homepage. http://www.webratio.com/site/content/en/home. Zugriff am 11.03.2015.

Wieczorrek and Mertens, 2011. Wieczorrek, H. W. and Mertens, P. (2011). *Management von IT-Projekten: von der Planung zur Realisierung*. Springer-Verlag.