HOCHSCHULE HEILBRONN

TECHNIK WIRTSCHAFT INFORMATIK

Geschäftsprozessautomatisierung der Sterilgutaufbereitung mit der BPMN 2.0 am Beispiel der Open Source Camunda BPM-Plattform

Master Thesis im Studiengang Medizinische Informatik

Hochschule Heilbronn und Universität Heidelberg

Referent: Prof. Dr. Christian Fegeler

Korreferent: Prof. Dr. Martin Haag

Dennis Lotz

Heilbronn, Februar 2015

Dennis Lotz
Nordstraße 8
74076 Heilbronn

Danksagung

Ich möchte mich bei allen bedanken, die mich während meines Studiums und beim Anfertigen dieser Arbeit unterstützt haben. Mein besonderer Dank gilt dabei meiner Freundin Agnieszka Krasnodębska, die immer an mich geglaubt und mich während meines Studiums unterstützt hat. Meinen Professoren Herrn Fegeler, möchte ich für die wissenschaftliche Unterstützung und die konstruktiven Vorschläge danken. Bei Herrn Peter und Frau Laun möchte ich mich besonders für die Unterstützung in moralischer und organisatorischer Hinsicht bedanken.

Der Thomas Gessmann-Stiftung gilt mein Dank für die finanzielle Unterstützung während der Arbeiten an meiner Masterthesis. Ohne das Stipendium wäre es nicht möglich gewesen diese Arbeit zu schreiben.

Herzlichsten Dank euch allen!

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig und nur unter Zuhilfenahme der ausgewiesenen Hilfsmittel angefertigt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach anderen gedruckten oder im Internet verfügbaren Werken entnommen sind, habe ich durch genaue Quellenangaben kenntlich gemacht.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und ist auch noch nicht veröffentlicht.

Ort, Datum Unterschrift

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

Tabellenverzeichnis

Abkürzungen

ACM	Adaptive Case Management
API	Application Programming Interface
BPEL	WS-Business Process Execution Language
BPMI	Business Process Management Initiative
BPMN	Business Process Model and Notation
CMMN	Case Management Model and Notation
EPK	Ereignisgesteuerte Prozesskette
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
JDBC	Java Database Connectivity
JUEL	Java Unified Expression Language
OMG	Object Management Group
RFID	Radio-frequency identification
UML	Unified Modeling Language
WAR	Web Archive
XML	Extensible Markup Language

Zusammenfassung

Die BPMN 2.0 bietet die Möglichkeit ausführbare Prozesse zu modellieren. Die so modellierten Prozesse können zur Automatisierung genutzt werden. Dies kann die Transparenz erhöhen und die Fehleranfälligkeit reduzieren. Im Klinischen Umfeld wird die Möglichkeit der Prozessautomatisierung mit Hilfe der BPMN 2.0 bisher nicht genutzt.

Im Rahmen dieser Arbeit wird am Beispiel eines Sterilgutaufbereiters evaluiert ob sich die BPMN 2.0 als Modellierungssprache, Signavio als Modellierungswerkzeug und die Camunda BPM als Business-Process-Engine eignen, Prozesse im klinischen Umfeld abzubilden. Der Schwerpunkt dieser Arbeit liegt auf der Umsetzung der modellierten Prozesse mit Hilfe der Camunda BPM und der anschließenden Evaluation.

Im Anschluss an die Analyse des Beispielszenarios werden mit Hilfe von Signavio BPMN 2.0 konforme Modelle erstellt. Anschließend werden die typischen Elemente der einzelnen Prozesse identifiziert. Die Modelle werden daraufhin mit der Camunda BPM zu automatisierbaren Prozessen umgesetzt.

Das Beispielszenario konnte in Teilprozesse zerlegt und anschließend automatisiert werden. Die dabei aufgetretenen Probleme wurden analysiert und die notwendigen Anpassungen der Prozesse wurden vorgenommen.

Die Evaluation zeigt, dass die Kombination aus BPMN 2.0, Signavio und der Camunda BPM geeignet ist standardisierte Prozesse aus dem klinischen Umfeld in automatisierte Prozesse umzusetzen.

1 Einleitung

Das erste Kapitel dieser Arbeit ist die Einleitung. Zuerst findet eine Einführung in die Thematik statt. Dabei wird auf die Problematik der Prozessautomatisierung im klinischen Umfeld eingegangen. Im zweiten Abschnitt wird die Zielsetzung der Arbeit dargestellt. Der Überblick über den Aufbau der Arbeit bildet den Abschluss dieses Kapitels.

1.1 Einführung und Problemstellung

Im klinischen Umfeld ist die Qualität der Prozesse besonders wichtig, da sie einen direkten Einfluss auf die Behandlung von Patienten haben (vgl. [1]).

Doch was sind Prozesse? Prozesse besitzen einen definierten Ablauf. Das bedeutet im Einzelnen, dass es neben einem definierten Anfang und einem definierten Ende auch einen zeitlichen und logischen Ablauf gibt. Es gibt eine oder mehrere Ein- und Ausgaben. Die Form der Ein- und Ausgaben ist dabei von der Art des Prozesses und der Umgebung, in der er stattfindet, abhängig (vgl. [2], [3])

„Geschäftsprozesse sind die zur Erstellung von Produkten und Leistungen erforderlichen betrieblichen Abläufe“ [4].

Nimmt man diese Definition von Thomas Allweyer als Grundlage so findet man Geschäftsprozesse in Unternehmen aller Größen. Sie gehören damit zum Kerngeschäft jedes Unternehmens und nehmen somit einen besonders hohen Stellenwert ein. Die Prozesse werden dabei meistens besonders in ein Unternehmen eingebunden und bei der strategischen Ausrichtung berücksichtigt (vgl. [5]).

Um die Qualität von Geschäftsprozessen zu bewerten kann Prozessmanagement eingesetzt werden. Es ist ein systematischer Ansatz um Geschäftsprozesse zu erfassen, gestalten, auszuführen, dokumentieren, messen, überwachen und zu steuern (vgl. [6]). Prozessmanagement kann auch zur Verbesserung der Qualität von Geschäftsprozessen beitragen. Durch die Erhebung und Darstellung der Prozesse im Gesamten werden sich alle Beteiligten über den genauen Ablauf bewusst und können diese besser bewerten. Es hat sich gezeigt, dass für den Erfolg von Geschäftsprozessmanagement vor allem ein strukturiertes Vorgehen und geeignete Werkzeuge eine große Rolle spielen (vgl. [7]).

Es gibt eine ganze Reihe an Werkzeugen für Geschäftsprozessmanagement. Meistens werden dafür Ereignisbasierte Prozessketten (EPKs), Petri Netze, UML Aktivitätsdiagrammen oder die BPMN eingesetzt. Es hat sich gezeigt, dass die BPMN eine Daseinsberechtigung als Notation besitzt (vgl. [8], [9], [10], [11]). Mit Hilfe der BPMN 2.0 ist es möglich Prozesse zu automatisieren (vgl. [12]). Dies ist ein enormer Vorteil gegenüber anderer Notationen, welche eine rein fachliche Darstellung der Prozesse ermöglichen.

Es gibt inzwischen eine Vielzahl an Software-Lösungen, die eine Ausführung von BPMN 2.0 Prozessen ermöglichen. Dazu zählen Produkte von Firmen wie SAP, Adobe, Bosch SI und Camunda. Die Qualität dieser Software-Lösungen unterscheidet sich jedoch stark und der Funktionsumfang muss immer in Bezug auf das Einsatzgebiet bewertet werden (vgl. [13]).

Auch im Gesundheitswesen nimmt Prozessmanagement einen immer größer werdenden Stellenwert ein. Viele Organisationen haben das Potential bereits erkannt und versuchen fortlaufend ihre Prozesse zu verbessern (vgl. [14]). Im klinischen Umfeld wird die BPMN 2.0 bisher überwiegend zur Prozesserfassung, -dokumentation und -optimierung eingesetzt. Die Automatisierung von klinischen Unterstützungsprozessen steht gerade erst am Anfang.

1.2 Zielsetzung

Im Rahmen dieser Arbeit soll die Möglichkeit zur Prozessautomatisierung im klinischen Umfeld, am Beispiel eines Dienstleisters für Sterilisationsgut, mit Hilfe der Camunda BPM-Plattform evaluiert werden.

Dabei gilt es zunächst zu klären, ob die BPMN 2.0 geeignet ist um Prozesse aus dem klinischen Umfeld abzubilden. Danach soll erörtert werden ob die Camunda BPM den Anforderungen gerecht werden kann.

Dazu soll im ersten Schritt ein Beispielszenario, mit Hilfe der BPMN 2.0 Notation, in Geschäftsprozesse überführt werden. Im Anschluss sollen diese Geschäftsprozesse auf der Camunda BPM-Plattform ausgeführt werden.

Die Evaluation der Camunda BPM soll auf den Erkenntnissen, die während der Umsetzung gewonnen werden, erfolgen.

1.3 Aufbau der Arbeit

Diese Arbeit ist in die folgenden Kapitel eingeteilt.

Kapitel 2 – Grundlagen
Hier werden die Grundlagen, die zum Verständnis der Arbeit beitragen, vermittelt. Dazu gehören neben der BPMN 2.0 der Signavio Editor und die Camunda BPM. Die Werkzeuge, welche zur Camunda BPM gehören, werden einzeln vorgestellt.

Kapitel 3 – Beispielszenario
In diesem Kapitel wird das verwendete Beispielszenario kurz präsentiert.

Kapitel 4 – Methodik und Ergebnisse
Im vierten Kapitel werden die Methodik und die daraus resultierenden Ergebnisse dargestellt. Die Umsetzung des Beispielszenarios in einen ausführbaren Prozess besitzt dabei den größten Anteil.

Kapitel 5 – Diskussion
Hier werden die Ergebnisse diskutiert. Es wird eine Bewertung der Arbeit vorgenommen und diskutiert ob die Arbeit der Zielsetzung gerecht wird. Des Weiteren werden alternative Modellierungen diskutiert. Zum Schluss wird ein Ausblick auf Fragestellungen für weitere Forschungsprojekte gegeben.

2 Grundlagen

In diesem Kapitel werden die, zum Verständnis der Arbeit notwendigen, Grundlagen vermittelt.

Es wird dabei die BPMN 2.0 als Standard zur Prozessmodellierung und die Prozessmodellierung mit Hilfe der BPMN 2.0 vorgestellt. Der Signavio Editor und die Camunda BPM werden präsentiert. Es wird auf die einzelnen Werkzeuge, aus denen die Camunda BPM-Plattform besteht, eingegangen.

2.1 BPMN

Die BPMN, welche zu Beginn noch als Abkürzung für Business Process Modeling Notation stand, wurde ursprünglich als grafische Repräsentation der Business Process Modeling Language (BPML) konzipiert. Die BPML diente der Beschreibung von Prozessen, konnte sich aber nicht gegen BPEL[1] durchsetzen und wurde fortan nicht mehr weiterentwickelt. Der erste Entwurf wurde 2004 von Stephen A. White, der zur dieser Zeit bei IBM arbeitete, vorgestellt. Die Entwicklung wurde von der Business Process Management Initiative (BPMI) vorangetrieben. Die Entwicklung der BPMN und die BPMI sind im Laufe der Zeit von der Object Management Group (OMG) übernommen worden und 2006 wurde die erste offizielle Version der BPMN 1.0 veröffentlicht. Mit Version 2.0, welche im März 2011 von der OMG verabschiedet und veröffentlicht wurde kamen einige Neuerungen dazu. Neben neuen Elementen und Diagrammtypen wurde ein Metamodell hinzugefügt und der Name zu Business Process Model and Notation geändert. Das Metamodell beschreibt die verschiedenen BPMN Konstrukte und deren Beziehungen untereinander mit UML Diagrammen. Mit der Einführung des Metamodells wurde ein einheitliches Austauschformat für BPMN Diagramme definiert. Als wichtigste Neuerung der BPMN 2.0 ist allerdings zu nennen, dass mit ihr eine Ausführungssemantik beschrieben wurde. Diese Ausführungssemantik ermöglicht neben der fachlichen Modellierung von Prozessen auch eine technische Umsetzung. Dadurch können BPMN 2.0 Diagramme mit Hilfe von Business Process Engines direkt ausgeführt werden (vgl. [15], [12]).

2.1.1 Grafische Notation

Die BPMN 2.0 spezifiziert verschiedene Modellierungselemente. Mit Hilfe dieser Elemente lassen sich Prozesse exakt beschreiben. Die Elemente unterteilen sich in verschiedene Kategorien; Pools bzw. Lanes, Aktivitäten, Flows, Gateways, Ereignisse und Datenobjekte. Die Elemente der einzelnen Kategorien werden noch weiter unterschieden. Eine ausführliche Beschreibung der einzelnen Elemente ist im Rahmen dieser Arbeit nicht möglich. Im Folgenden werden kurz die einzelnen Kategorien umrissen. Um Missverständnisse bei den Diagrammen zu vermeiden werden die Elemente und Kategorien, welche für das Verständnis notwendig sind genauer erklärt.

[1] BPEL ist eine auf XML-basierende Sprache, die zur Orchestrierung von Webservices verwendet wird. BPEL wurde bereits 2002 von IBM, BEA System und Microsoft eingeführt (vgl. [25])

Pools und Lanes

Pools und Lanes beschreiben Zuständigkeiten. Ein Pool kann in mehrere Lanes unterteilt werden. Jede Lane kann wiederum durch weitere Lanes unterteilt werden. Pools werden oftmals für die Darstellung von Unternehmen oder Abteilungen verwendet. Im Standard ist die genaue Verwendung allerdings offen gelassen. So kann jeder selbst entscheiden, was ein Pool bzw. eine Lane repräsentieren soll. In Abbildung 1 ist ein Pool mit zwei Lanes dargestellt.

Abbildung 1: BPMN 2.0 Element Pool und Lanes
Quelle Eigene Darstellung

Aktivitäten

Aktivitäten stellen einzelnen Aufgaben im Prozess dar. Sie werden innerhalb von Pools platziert und so diesem zugeordnet. Die Darstellung von Aktivtäten ist ein Rechteck mit abgerundeten Ecken. Call Activities sind global definierte Teilprozesse oder Aufgaben. Durch sie lässt sich die Komplexität in Diagrammen verringern, da wiederkehrende Teilprozesse ausgelagert werden können. Im Gegensatz zu Aufgaben besitzen Call Activities eine dickere Umrandung. In Abbildung 2 sind verschiedene Aufgaben und Call Activities dargestellt.

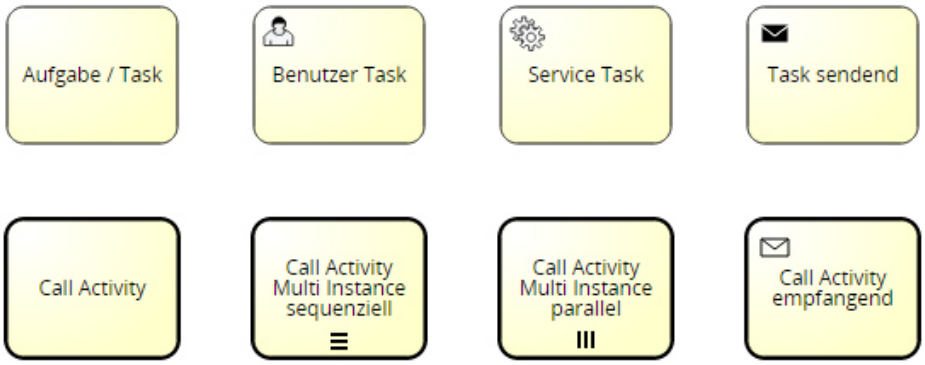


Abbildung 2: BPMN 2.0 Element Aktivitäten
Quelle: Eigene Darstellung

Aktivitäten können mit Hilfe von Typen und Markierungen genauer definiert werden. Typen werden im oberen linken Bereich einer Aktivität angezeigt. Sie definieren die Art der Aktivität.

Benutzer Tasks werden üblicherweise von einem Menschen in Zusammenarbeit mit einer Workflow Engine ausgeführt. Service Tasks symbolisieren Aufrufe von externen Applikationen oder Maschinen. Mit Hilfe der Typen lässt sich auch definieren, ob eine Aktivität Nachrichten versendet oder empfängt. Markierungen werden im unteren Bereich von Aktivitäten eingefügt. Aktivitäten, welche mehrmals ausgeführt werden sollen, werden als Multi Instance bezeichnet und können können entweder parallel oder sequentiell aufgerufen werden.

Flows

Flows werden in der BPMN 2.0 durch Pfeile symbolisiert. Es gibt zwei verschiedene Arten von Flows. Der Sequence Flow (Sequenzfluss) und der Message Flow (Nachrichtenfluss). Sequenzflüsse verbinden Aktivitäten und definieren so den Ablauf des Prozesses. Sequenzflüsse dürfen die Kanten von Pools nicht überschreiten. Nachrichtenflüsse definieren den Informationsaustausch durch Nachrichten auch über Poolgrenzen hinweg.

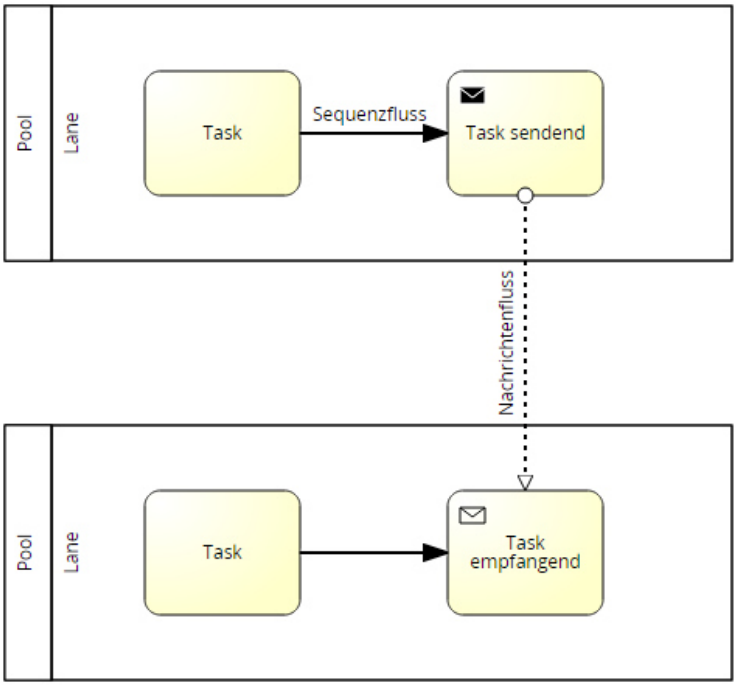


Abbildung 3: BPMN 2.0 Element Flows
Quelle: Eigene Darstellung

Abbildung 3 zeigt wie die Reihenfolge von Tasks mit Hilfe von Sequenzflüssen festgelegt und der Nachrichtenaustausch zwischen zwei Tasks mit Nachrichtenflüssen realisiert wird.

Gateways

Gateways ermöglichen es den Sequenzfluss zu verzweigen. Dadurch lässt sich der Ablauf von Prozessen steuern. Gateways werden grafisch durch eine Raute dargestellt. Sie besitzen oftmals mehrere eingehende - und ausgehende Sequenzflüsse. Die Entscheidung, auf welchen Sequenzfluss das Token[2] geleitet wird basiert üblicherweise auf Prozessvariablen.

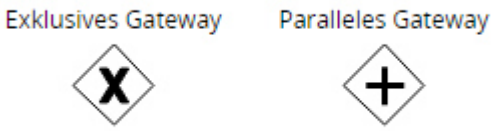


Abbildung 4: BPMN 2.0 Element Gateways
Quelle: Eigene Darstellung

Die mit Abstand häufigsten Gateways sind das Exklusive – und das Parallele Gateway, welche in Abbildung 4 gezeigt werden. Bei Exklusiven Gateways wird das Token auf einen der ausgehenden Sequenzflüsse weitergeleitet. Bei Parallelen Gateways hingegen wird das Token aufgeteilt und jede ausgehende Kante wird von einem Teiltoken beschritten.

Ereignisse

Es werden drei Arten von Ereignissen unterschieden. Start-, Zwischen und Endereignisse. Jeder Prozess besitzt mindestens ein Start- und ein Endereignis. Sie werden durch Kreise symbolisiert und mit Sequenzflüssen in den Prozessablauf eingebunden. Wie in Abbildung 5 zu sehen werden Startereignisse durch eine einfache Linie, Zwischenereignisse durch eine doppelte Linie und Endereignisse durch eine dicke Linie dargestellt.

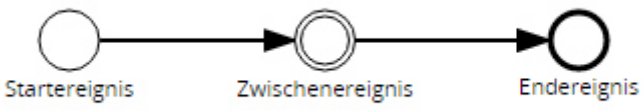


Abbildung 5: BPMN 2.0 Element Ereignisse Arten
Quelle: Eigene Darstellung

[2] Das Token Konzept wird in Kapitel 2.1.2 erklärt.

Es gibt verschiedene Typen von Ereignissen. In Abbildung 6 werden die, im Rahmen dieser Arbeit verwendeten, Typen dargestellt. Das Timer Ereignis, wird zu einem bestimmten Zeitpunkt, das Fehler Ereignis bei einem Fehler und das Nachrichten Ereignis beim Eintreffen einer Nachricht ausgeführt.

Abbildung 6: BPMN 2.0 Element Ereignisse Typen
Quelle: Eigene Darstellung

Datenobjekte

Datenobjekte werden verwendet um Informationen grafisch darzustellen. Sie besitzen keine Ausführungssemantik. Wie in Abbildung 7 zu sehen ist, werden Datenobjekte durch ein Rechteck mit eingeknickter Ecke symbolisiert. Datenspeicher werden durch einen Zylinder mit Ringen im oberen Bereich dargestellt. Da sie keine Bedeutung bei der Ausführung besitzen wurde zugunsten der Lesbarkeit auf die Verwendung von Datenobjekten verzichtet.

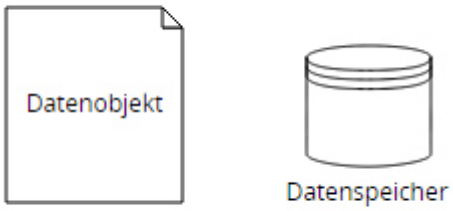


Abbildung 7: BPMN 2.0 Element Datenobjekte
Quelle: Eigene Darstellung

2.1.2 Token Konzept

Das Token Konzept ist ein theoretisches Konstrukt, welches verwendet wird um die Prozessstruktur zu verstehen. Mit Hilfe des Tokens kann man verstehen welche Pfade im Process beschritten werden und warum. Stellt man sich das Prozessdiagramm als Landkarte mit Straßen vor, so entspräche das Token einem Fahrzeug, welches die verschiedenen Straßen langfährt. Es kann an Kreuzungen abbiegen und fährt vom Start bis Ziel. Das Token repräsentiert an welcher Stelle sich der Prozess gerade befindet. Beim Start einer Prozessinstanz wird immer genau ein Token erzeugt.

Im Verlauf des Prozesses gibt es allerdings auch Stellen (Parallele Gateways) an denen das Token geklont werden muss und danach zwei Tokens existieren. Danach spricht man von Teiltokens. Um eine Prozessinstanz zu beenden, müssen alle Token bzw. Teiltoken ein Endereignis erreichen. Einem Token können Variablen, sogenannte Prozessvariablen, mitgegeben werden. Eine Prozessvariable ist ein Wert, welcher im Verlauf des Prozesses von Aktivitäten ausgewertet werden kann. Sie sind notwendig, um an Exklusiven Gateways eine Entscheidung treffen zu können (vgl. [16]).

2.2 Signavio Process Editor

Die Modellierung der BPMN 2.0 Diagramme soll mit dem Signavio Process Editor erfolgen. Der webbasierte Editor wird seit 2009 von der Firma Signavio GmbH entwickelt. Inzwischen liegt er in der Version 8.40 vor. Der Editor unterstützt neben der Modellierung von BPMN 2.0 Diagrammen unter anderem auch Ereignisgesteuerte Prozessketten und Petri Netze (vgl. [17]).

Die Oberfläche des Editors ist sehr intuitiv gehalten. Nach dem Login befindet man sich im Workspace. Die Oberfläche des Workspace ist in Abbildung 8 dargestellt. Auf der linken Seite befindet sich der Workspace des Benutzers, welcher in einer Ordnerstruktur angelegt ist. Rechts daneben befindet sich eine Übersicht aller Diagramme des aktuellen Ordners. Direkt darüber ist eine Werkzeugleiste. Über diese Werkzeugleiste lässt sich eine Vielzahl an Aufgaben erledigen. Es können neue Diagramme erstellt, bestehende Diagramme bearbeitet oder exportiert werden. Es können auch Diagramme in den Workspace importiert werden. Es befinden sich auch Werkzeuge zur automatisierten Berichtserstellung und Freigabe von Diagrammen darunter. Im unteren Bereich wird eine Vorschau des ausgewählten Prozesses präsentiert. Mit einem Doppelklick auf ein Diagramm wechselt die Ansicht in den Bearbeitungsmodus.

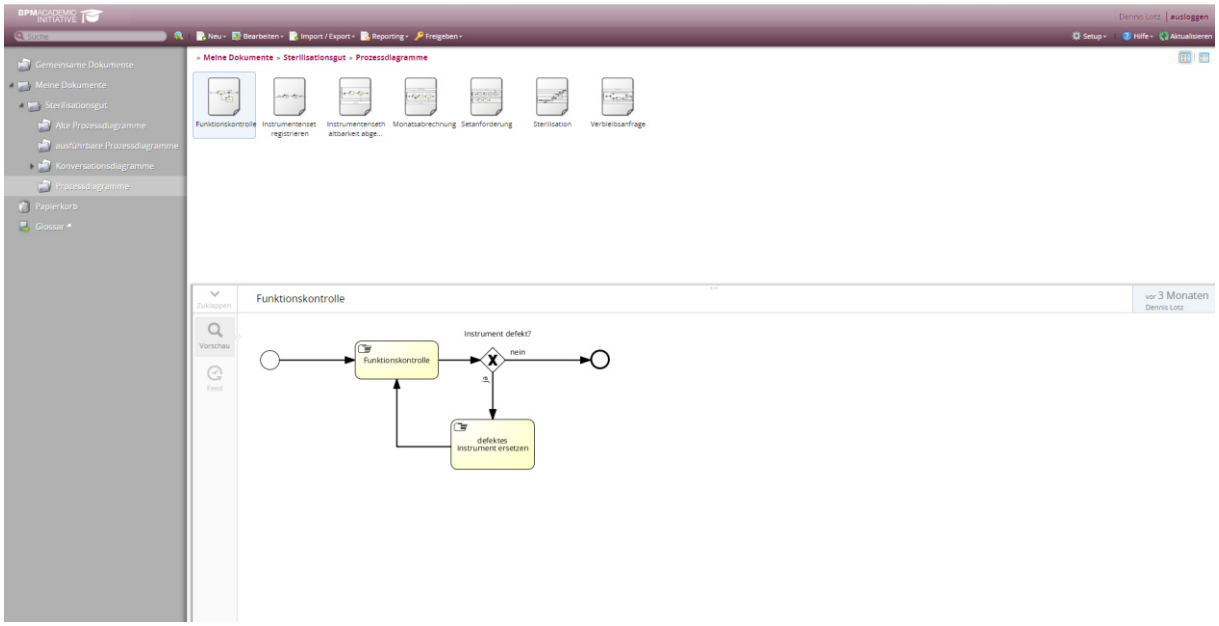


Abbildung 8: Benutzeroberfläche Signavio Process Editor – Workspace
Quelle: Screenshot Signavio Process Editor Version 8.40

Im Bearbeitungsmodus können Diagramme modelliert werden. Abbildung 9 zeigt die entsprechende Oberfläche bei der Bearbeitung eines BPMN 2.0 Diagramms. Auf der linken Seite befindet sich eine Liste mit allen vorhandenen Elementen der BPMN 2.0 Notation. Im Bereich in der Mitte kann das Diagramm erstellt werden. Die Elemente lassen sich per Drag and Drop dort platzieren. Auf der rechten Seite können die Attribute des aktuell ausgewählten Elements eingesehen und verändert werden. Dort lässt sich z.B. der Typ eines Tasks bestimmen.

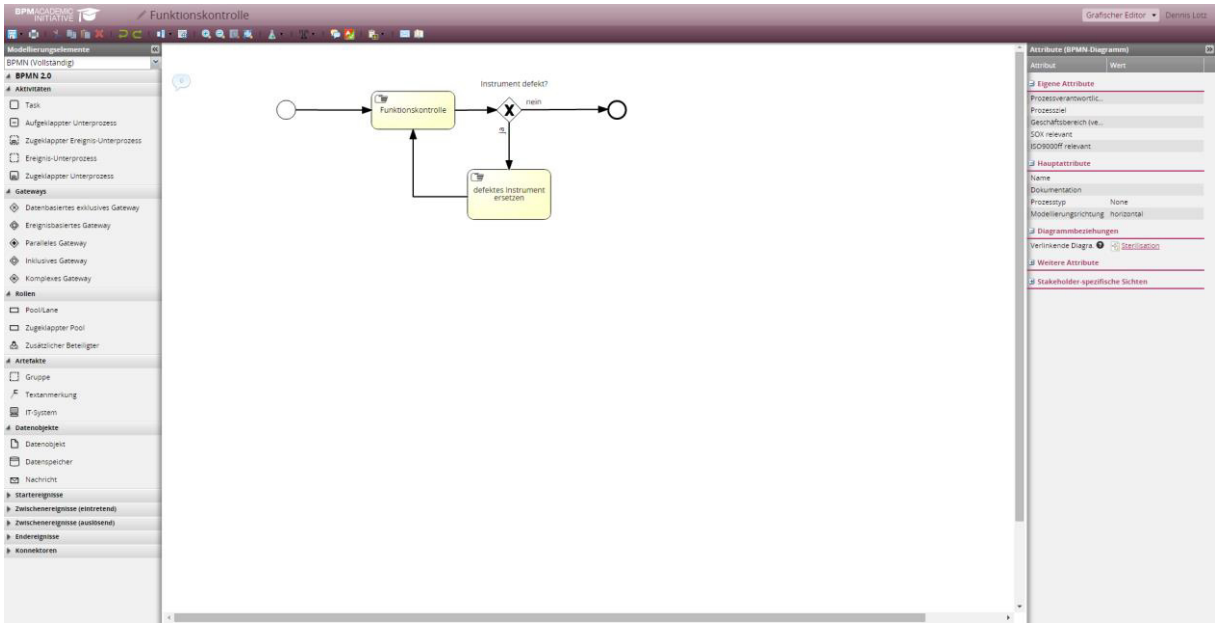


Abbildung 9: Benutzeroberfläche Signavio Process Editor – Bearbeitungsmodus
Quelle: Screenshot Signavio Process Editor Version 8.40

Der Signavio Process Editor besitzt auch eine eingebaute Syntaxprüfung. Diese wird ausgeführt, wenn ein Diagramm gespeichert werden soll. Weiterhin verfügt der Editor über eine Versionierung der Diagramme. Beim Speichern kann ein Änderungskommentar eingegeben werden. Der Speicherdialog ist in Abbildung 10 zu sehen.

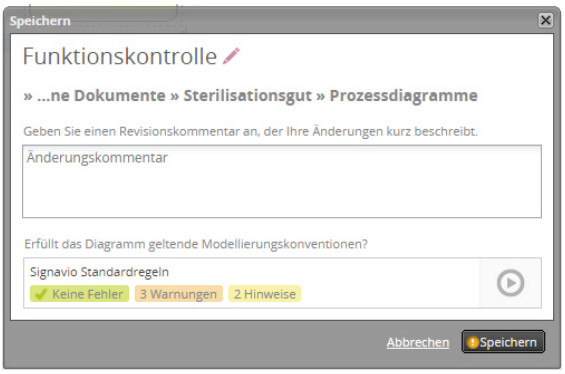


Abbildung 10: Benutzeroberfläche Signavio Process Editor Speicherdialog
Quelle: Screenshot Signavio Process Editor Version 8.40

Die Ergebnisse der Prüfung auf Modellierungskonventionen, darunter fällt auch die Syntaxprüfung, sind im unteren Teil zu sehen. Mit einem Klick auf den Button rechts der Ergebnisse werden die Empfehlungen zur Modellierung, wie in Abbildung 11 zu sehen, im Diagramm angezeigt.

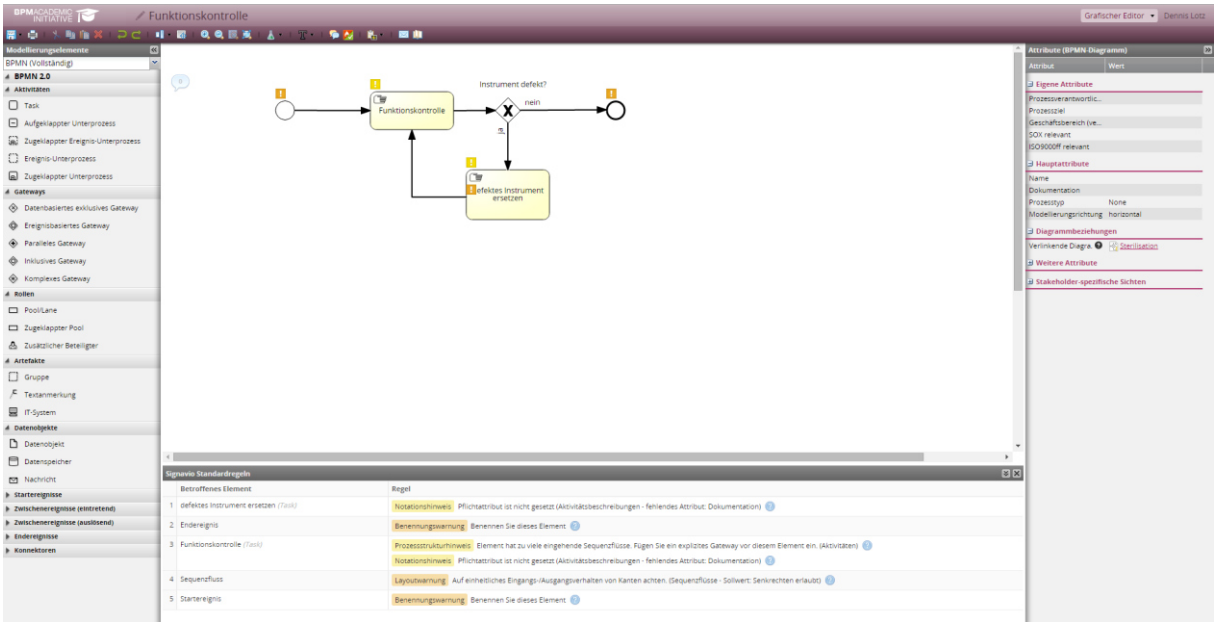


Abbildung 11: Benutzeroberfläche Signavio Process Editor – Modellierungskonventionen
Quelle: Screenshot Signavio Process Editor Version 8.40

Der Signavio Process Editor unterstützt auch die Simulation von BPMN 2.0 Diagrammen. Mit Hilfe der Simulation kann überprüft werden, ob es Deadlocks im Diagramm gibt. Auch Sequenzflüsse, welche nie beschritten werden lassen sich so sehr schnell identifizieren.

Da diese Funktion im Rahmen dieser Arbeit keine weitere Relevanz besitzt wird nicht weiter darauf eingegangen.

Im Rahmen der Signavio BPM Academic Initiative können Studenten, Professoren und Forscher den Signavio Process Editor kostenlos benutzen.

2.3 Camunda BPM

Camunda bietet mit der Camunda BPM eine Plattform zur Ausführung von Arbeitsabläufen und Geschäftsprozessen. Camunda BPM ist eine leichtgewichtige und sehr flexible Open Source Plattform. Sie bietet eine Java API und lässt sich sehr gut in Spring und Java EE Projekte integrieren. Durch die Java API ist eine Interaktion zwischen eigenen Java Applikationen und der Camunda BPM möglich. Dadurch erreicht die Camunda BPM eine sehr große Flexibilität und Erweiterbarkeit (vgl. [18], [13], [19]).

Die Plattform besteht aus verschiedenen Komponenten, die benötigt werden um aus BPMN 2.0 Modellen einen ausführbaren Prozess zu erzeugen. Die verschiedenen Komponenten werden im Folgenden genauer vorgestellt.

2.3.1 Process Engine

Die Process Engine bildet den zentralen Bestanteil der Camunda BPM. Mit Hilfe der Process Engine ist es möglich BPMN 2.0 Diagramme auszuführen. Sie interpretiert die Prozesse und führt diese aus. Dabei wertet sie den Kontrollfluss von Prozessen aus und entscheidet welcher Task als nächstes auszuführen ist. Sie bietet über definierte Schnittstellen die Möglichkeit um mit den Prozessen zu interagieren. Diese Schnittstellen werden z.B. von der Tasklist und dem Cockpit verwendet um eine grafische Administration der Process Engine zu ermöglichen. So können z.B. neue Instanzen eines Prozesses erzeugt oder laufende Instanzen dargestellt werden.

2.3.2 Modeler

Der Modeler spielt eine zentrale Rolle für die Überführung von BPMN 2.0 Diagrammen in ausführbare Prozesse. Er wird als Plugin für die Eclipse IDE angeboten. Mit Hilfe des Modelers können die von der Process Engine benötigten Informationen in BPMN 2.0 Diagramme eingebracht werden.

In Abbildung 12 ist ein Screenshot der Oberfläche des Camunda Modeler Plugins für Eclipse abgebildet. Auf der linken Seite befindet sich, wie gewohnt bei Eclipse, die Verzeichnisstruktur. Auf der rechten Seite befindet sich eine Werkzeugpalette, die einem die in BPMN 2.0 spezifizierten Komponenten bereitstellt. Im unteren Bereich ist das Properties Panel zu sehen. Dieses Panel ist die Grundlage für die Aufbereitung von bereits modellierten BPMN 2.0 für die Ausführung.

Grundsätzlich ist es möglich mit dem Modeler zu modellieren. Die Usability lässt allerdings noch schwer zu wünschen übrig. Es gibt keine Möglichkeiten das Diagramm zu vergrößern oder zu verkleinern. Bei kleinen Diagrammen stellt dies kein Problem dar. Bei größeren Diagrammen büßt man im Camunda Modeler doch sehr an Übersichtlichkeit ein. Die Sequenzflüsse von und zu den Tasks lassen sich nicht sauber anordnen, so dass nicht gerade Sequenzflüsse entstehen können.

Für die Aufbereitung der Diagramme ist der Camunda Modeler unumgänglich. Das Property Panel bietet zu dem jeweils ausgewählten Element die Möglichkeit die für die Ausführung notwendigen Informationen einzutragen. Bei Gateways werden andere Eingabefelder dargestellt als bei Aktivitäten, da bei der Ausführung unterschiedliche Informationen benötigt werden. So werden einem zu jedem Zeitpunkt die relevanten Möglichkeiten der Aufbereitung angezeigt. Es ist zwar möglich die notwendigen Informationen auch direkt in die XML Datei einzutragen. Dieses Vorgehen erweist sich allerdings als nicht praktikabel, da es unübersichtlich, umständlich und fehleranfällig ist.

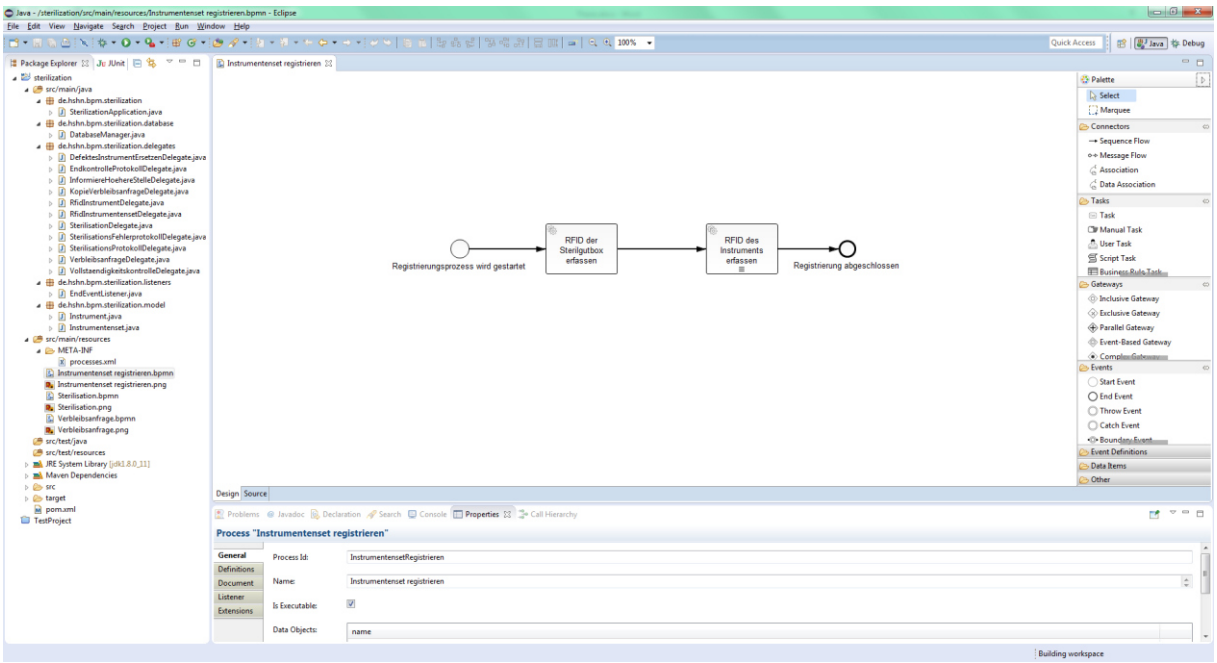


Abbildung 12: Benutzeroberfläche Camunda Modeler
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

2.3.3 Tasklist

Die Camunda Tasklist bildet die Schnittstelle zwischen dem Benutzer und den ihm zugeordneten Tasks. Sie bietet eine Übersicht über alle anstehenden Aufgaben und unterstützt so den Benutzer bei der Abarbeitung dieser.

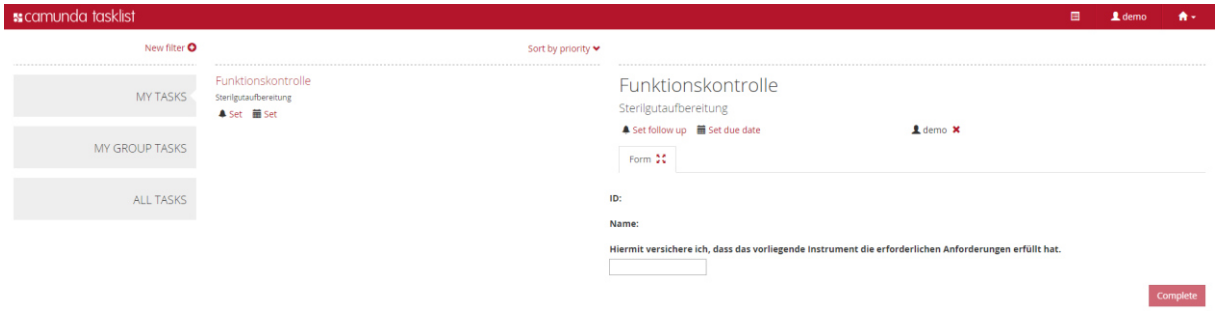


Abbildung 13: Benutzeroberfläche Camunda Tasklist
Quelle: Screenshot Camunda Tasklist Version 7.2-alpha5

In Abbildung 13 ist die Oberfläche der Tasklist zu sehen. Auf der linken Seite kann der Benutzer auswählen welche Tasks angezeigt werden sollen. Er hat dabei die Auswahl zwischen den eigenen, den Tasks seiner Benutzergruppe und allen Tasks. Es können auch weitere Filter erstellt werden. Diese werden dann ebenfalls auf der linken Seite angezeigt.

Nach der Auswahl eines Filters wird rechts davon eine Liste mit den Tasks, die den Kriterien des Filters entsprechen, angezeigt. Diese Liste kann nach verschiedenen Merkmalen sortiert werden.

Wird ein Task ausgewählt erscheint auf der rechten Seite eine Übersicht der Taskeigenschaften. Dazu gehören neben dem Namen des Tasks und dem Namen des Prozesses auch die Task Form. Es können der zugewiesene User, die Frist für die Bearbeitung und der Zeitpunkt für eine Wiedervorlage definiert werden. Die Camunda Tasklist unterstützt standardmäßig vier verschiedene Arten von Task Forms; Embedded Task Forms, Generated Task Forms, External Task Forms und Generic Task Forms. Task Forms dienen der Darstellung und Eingabe von Informationen.

Embedded Task Forms werden mit HTML und JavaScript modelliert und mit der eigenen Process Application ausgeliefert. Generated Task Forms basieren auf XML Metadaten in den BPMN Diagrammen. External Task Forms werden nicht mit der Process Application ausgeliefert, sondern werden von einer anderen Applikation bereitgestellt. Beim Aufruf wird der Benutzer dorthin weiter geleitet. Generic Task Forms werden automatisch erzeugt, wenn keine andere Task Form vorhanden ist. Dem Benutzer werden alle vorhandenen Prozessvariablen angezeigt (vgl. [18]).

Im Rahmen dieser Arbeit werden ausschließlich Embedded Task Forms verwendet.

2.3.4 Cockpit

Das Camunda Cockpit ermöglicht es vorhandene Prozessdefinitionen zu verwalten. Nach dem Start des Camunda Cockpits wird eine Übersicht aller Prozessdefinitionen dargestellt. Diese Übersicht ist in Abbildung 14 abgebildet.

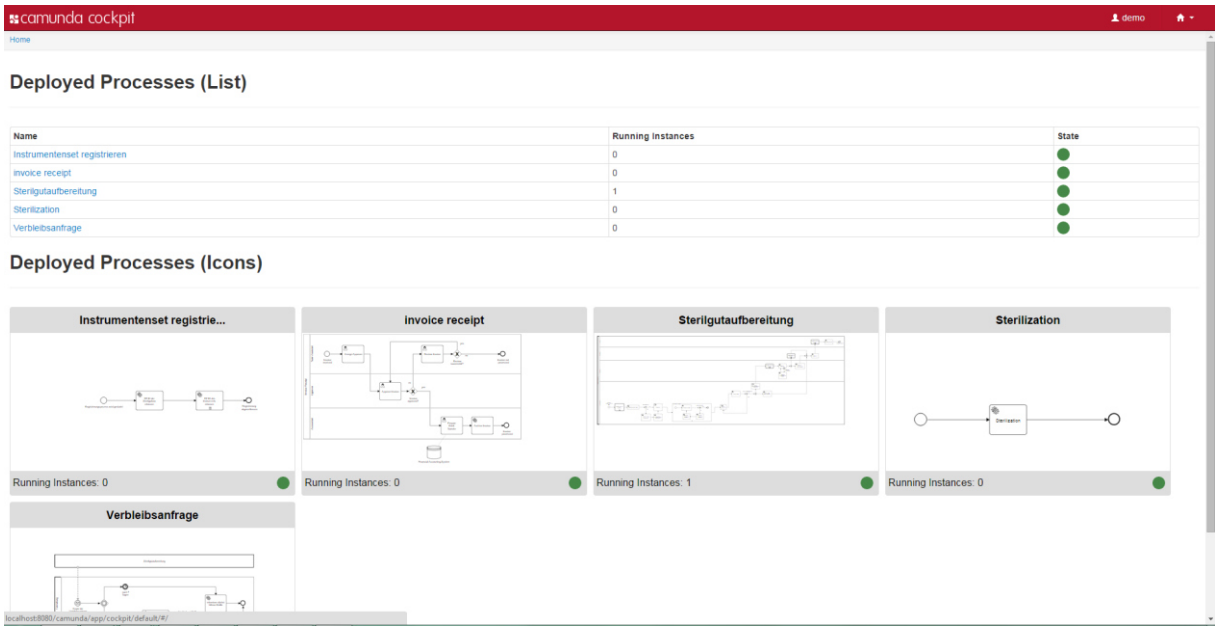


Abbildung 14: Benutzeroberfläche Camunda Cockpit Übersicht Prozessdefinitionen
Quelle: Screenshot Camunda Cockpit Version 7.2-alpha5

Klickt der Benutzer auf eine Prozessdefinition wird diese geöffnet und ihm eine Vorschau des Prozessdiagramms angezeigt. Im Diagramm sind die Stellen, an denen die Tokens der einzelnen Instanzen gerade stehen, markiert. Wie in Abbildung 15 zu sehen ist befindet sich unterhalb des Diagramms eine Liste mit allen aktiven Instanzen.

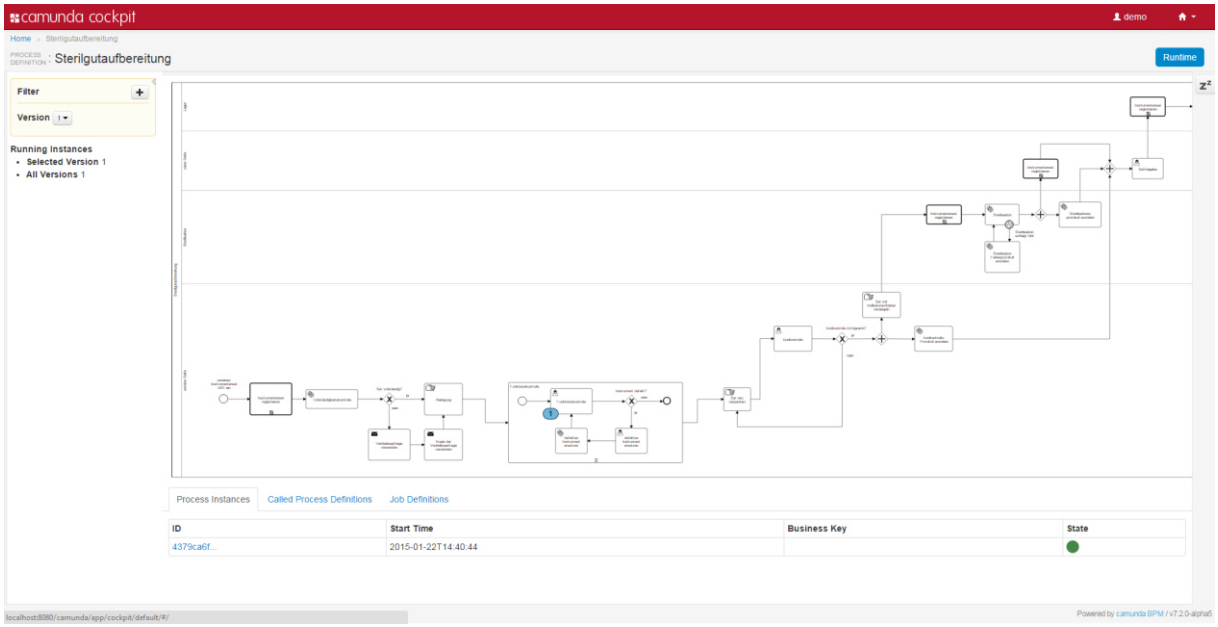


Abbildung 15: Benutzeroberfläche Camunda Cockpit Übersicht Prozessinstanzen
Quelle: Screenshot Camunda Cockpit Version 7.2-alpha5

Wählt man eine der Instanzen aus der Liste aus wird eine Übersicht der Prozessinstanz geöffnet. In Abbildung 16 ist diese Übersicht dargestellt. Neben der Vorschau des Prozessdiagramms wird eine Liste mit allen vorhandenen Prozessvariablen präsentiert. Der Inhalt der Prozessvariablen lässt sich sowohl auslesen als auch verändern. Es können auch neue Prozessvariablen hinzugefügt werden. Dadurch kann der Benutzer steuernd auf den Prozess einwirken.

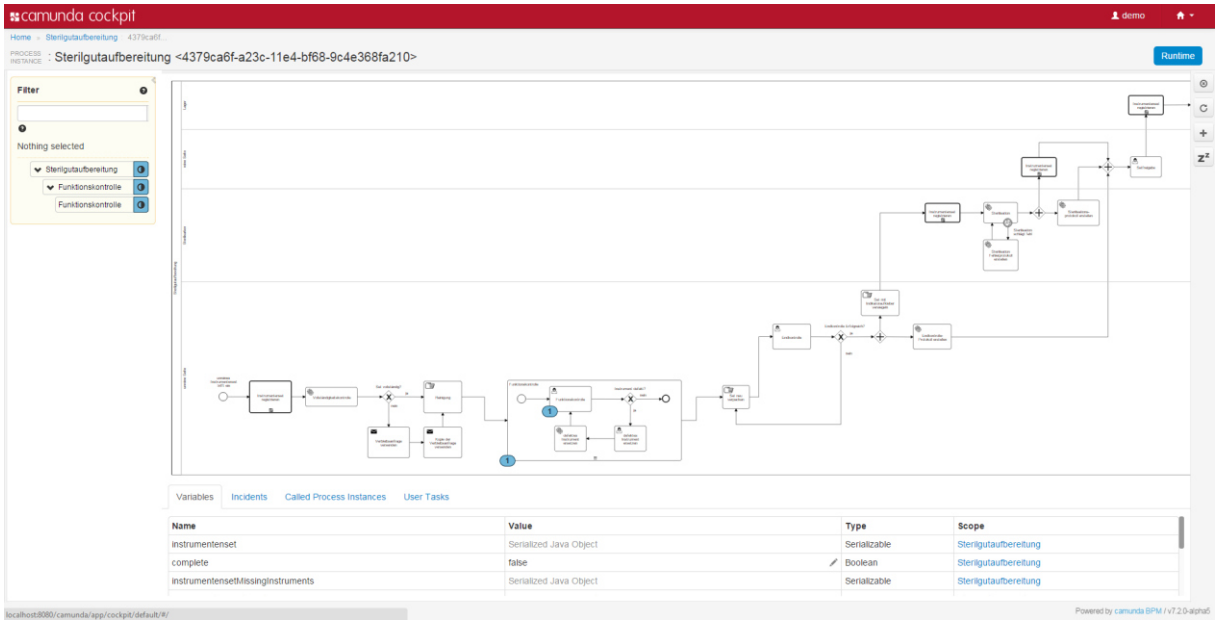


Abbildung 16: Benutzeroberfläche Camunda Cockpit Übersicht Prozessvariablen
Quelle: Screenshot Camunda Cockpit Version 7.2-alpha5

2.3.5 Cycle

Camunda bietet mit Cycle eine Möglichkeit BPMN 2.0 Diagramme konsistent zu halten. Die Modellierung und die Umsetzung werden oft von verschiedenen Mitarbeitern einer Firma durchgeführt. Damit alle Beteiligten dieselbe Prozessdefinition besitzen muss diese konsistent gehalten werden.

Cycle erlaubt es einen Roundtrip zwischen einem BPMN 2.0 Editor und dem Camunda Modeler herzustellen. Beim Roundtrip werden die Diagramme aus dem Editor in den Modeler exportiert. Dort werden sie mit den für die Ausführung notwendigen Informationen angereichert und anschließend wieder zurück in den Editor übertragen. Dazu müssen in Cycle sogenannte Connectoren definiert werden. Ein Connector ermöglicht den Zugriff auf einen Speicherort mit Prozessdiagrammen. Für einen kompletten Roundtrip sind zwei Connectoren notwendig.

3 Beispielszenario

Im Rahmen dieser Arbeit wird ein, an die Realität angelehntes, aber vereinfachtes Beispielszenario verwendet. Das Szenario wurde von Prof. Dr. Christian Fegeler als Übungsaufgabe für die Vorlesung Geschäftsprozessmodellierung konzipiert. Im Anhang als A.1 Beispielszenario ist die vollständige Aufgabenbeschreibung abgedruckt, da die Quelle ansonsten für den Leser nicht frei zugänglich ist.

Das Szenario beschreibt einen Sterilgutdienstleister, welcher als Servicegesellschaft eines Klinikverbundes Kunden mit Sterilgut versorgt. Es werden sowohl Kunden im Verbund, als auch außerhalb davon beliefert. Es wird der Prozess der Aufbereitung von Sterilgut beschrieben. Die Aufbereitung der Instrumentensets ist ein komplizierter Prozess mit vielen Beteiligten. Neben einer Vollständigkeits- und Funktionskontrolle wird auch die Reinigung und Zusammenstellung der neuen Sets beschrieben. Daneben gibt es auch organisatorische Prozesse, die nicht direkt mit der Aufbereitung zusammenhängen. Dazu gehören die monatliche Abrechnung, die Setanforderung und die Fristenkontrolle von Verbleibsanfragen.

Für das Verständnis ist es zuträglich die Aufgabenstellung im Gesamten durchzulesen. Im späteren Verlauf der Arbeit werden nur die jeweils relevanten Auszüge der Beispielaufgabe aufgeführt.

4 Methodik und Ergebnisse

In diesem Kapitel werden die Methodik und die erarbeiteten Ergebnisse der einzelnen Arbeitsabschnitte vorgestellt. Das Kapitel ist in drei Abschnitte eingeteilt, welche den Arbeitsverlauf wiederspiegeln.

Zuerst wird im Einzelnen auf die Modellierung eingegangen. Im zweiten Teil geht es um die Umsetzung der zuvor modellierten Diagramme mit Hilfe der Camunda BPM. Bei der Umsetzung werden das Datenbankkonzept, die Process Application, die Prozessvariablen und die ausführbaren Diagramme im Einzelnen dargestellt. Bei den Diagrammen werden die jeweiligen, notwendigen Änderungen anhand der einzelnen Diagramme dargestellt und erläutert. Im dritten Abschnitt des Kapitels befindet sich die Evaluation. Zur Evaluation gehört neben dem Evaluationskonzept die Präsentation der Ergebnisse.

4.1 Modellierung

Die Modellierung des Beispielszenarios erfolgt mit Hilfe des Signavio Editors. Das Beispielszenario wird gedanklich in kleinere Teilprozesse zerlegt, welche einzeln modelliert werden. Die einzelnen Teilprozesse werden über BPMN eigene Mittel, wie Nachrichtenaustausch und Call Activities, miteinander verknüpft. Die Syntaxüberprüfung des Signavio Editor wird verwendet um eine syntaktisch korrekte Modellierung sicherzustellen.

Nachfolgend werden die einzelnen modellierten Diagramme dargestellt und erläutert. Dabei werden die jeweils relevanten Textpassagen der Aufgabenstellung zitiert[3].

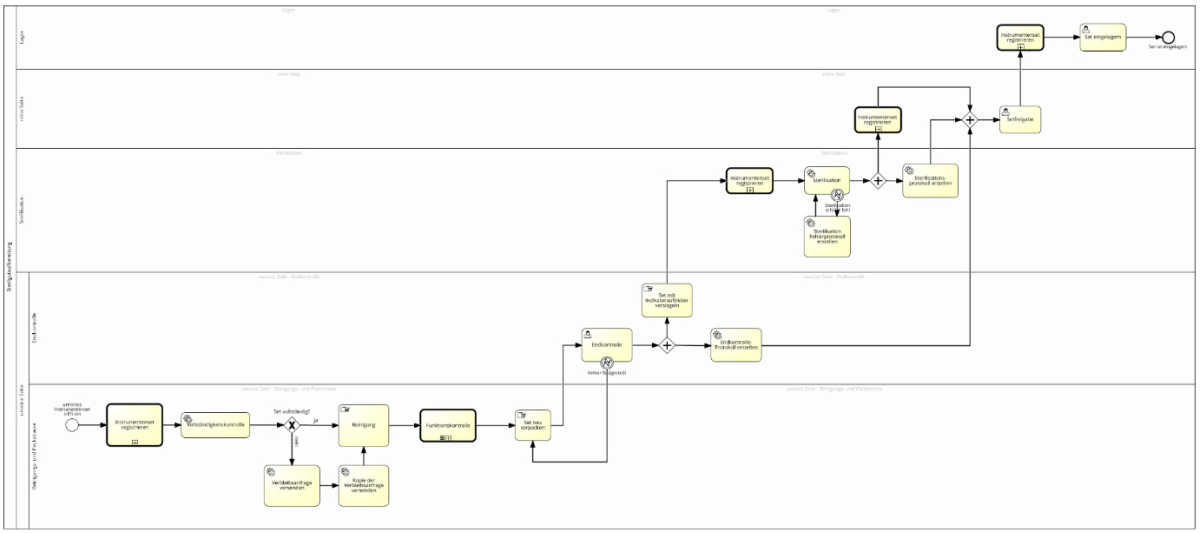


Abbildung 17: Prozessdiagramm Sterilisation
Quelle: Eigene Darstellung siehe Anhang A.2

In Abbildung 17 wird der Prozess der Sterilgutaufbereitung dargestellt[4]. Die Sterilgutaufbereitung ist in vier Lanes aufgeteilt, die die verschiedenen Arbeitsbereiche wiederspiegeln. Die Lane *unreine Seite* ist in zwei weitere Lanes unterteilt. Diese Unterteilung ist notwendig um die Auflagen der Endkontrolle zu erfüllen. Der Prozess wird gestartet wenn ein unreines Instrumentenset bei der Sterilgutaufbereitung eintrifft. Danach wird das Instrumentenset in der Lane *unreine Seite* innerhalb der Call Activity *Instrumentenset registrieren* registriert. Daraufhin folgt der Servicetask *Vollständigkeitskontrolle*. In diesem Task wird das vorliegende Instrumentenset auf Vollständigkeit überprüft. Dazu wird eine Datenbank, in welcher die Instrumentensets hinterlegt sind, abgefragt und mit dem vorliegenden Instrumentenset verglichen.

[3] Bei der Beschreibung des Prozessdiagramms Sterilisation wird nicht aus der Aufgabenstellung zitiert. Aufgabenstellung siehe Anhang A.1
[4] Für eine größere Darstellung der Abbildung siehe Anhang A.2

Wenn die Vollständigkeitskontrolle negativ ausfällt wird eine Verbleibsanfrage an den Kunden versendet. In dieser Verbleibsanfrage werden die fehlenden Instrumente aufgeführt. Eine Kopie der Verbleibsanfrage wird an die Verwaltung verschickt, die sich um die Fristenkontrolle der Anfrage kümmert. Im Anschluss daran wird das Instrumentenset manuell gereinigt. Es folgt die *Funktionskontrolle*. Die *Funktionskontrolle* ist eine weitere Call Activity. Die Auslagerung in eine Call Activity ist an dieser Stelle nicht notwendig, bietet sich aber an, um die Komplexität des Prozessdiagramms zu reduzieren. Jetzt wird das Instrumentenset von einem Mitarbeiter der Sterilgutaufbereitung neu gepackt. Anschließend wechselt das Instrumentenset in die Lane *Endkontrolle*. Dieser Übergang soll sicherstellen, dass die *Endkontrolle* von einem anderen Mitarbeiter ausgeführt wird. Tritt bei der *Endkontrolle* ein Fehler auf, wird das unterbrechende angeheftete Fehlerereignis *Fehler festgestellt* ausgelöst und das Instrumentenset muss erneut gepackt werden. Ist die *Endkontrolle* erfolgreich, so wird jetzt der Sequenzfluss durch ein paralleles Gateway aufgeteilt. Am Gateway werden aus einem Token zwei neue Teiltoken erzeugt. Die Tasks *Set mit Indikatoraufkleber versiegeln* und *Endkontrolle-Protokoll erstellen* werden nun gleichzeitig angestoßen. Wenn das Set mit dem Indikatoraufkleber versiegelt wurde wechselt es in die Lane *Sterilisation*. Dort wird das Instrumentenset zuerst registriert und anschließend sterilisiert. Der Servicetask *Sterilisation* repräsentiert den Sterilisationsautomaten. Tritt während der Sterilisation ein Fehler auf, wird das angeheftete Fehlerereignis *Sterilisation schlägt fehl* aufgerufen, ein Fehlerprotokoll erstellt und die *Sterilisation* erneut ausgeführt. Nach erfolgreicher Sterilisation wird der Sequenzfluss durch ein weiteres Paralleles Gateway verzweigt. Das Sterilisationsprotokoll wird erstellt und das Instrumentenset wechselt in die Lane *reine Seite*. Dort wird es registriert. Am darauf folgenden parallelen, zusammenführenden Gateway werden die parallel laufenden Sequenzflüsse wieder vereint. Der Benutzertask *Setfreigabe* wird erst ausgeführt, wenn an allen eingehenden Kanten des parallelen Gateways ein Token eintrifft. Diese Modellierung stellt sicher, dass die Protokolle der Endkontrolle und Sterilisation vorliegen müssen bevor eine Freigabe des Instrumentensets erfolgen kann. Der Mitarbeiter prüft anhand der vorliegenden Protokolle ob das Set ordnungsgemäß gepackt und sterilisiert wurde. Nachdem er das Instrumentenset freigibt wechselt es die Lane und wird im *Lager* registriert. Im Lager wird das Instrumentenset eingelagert.

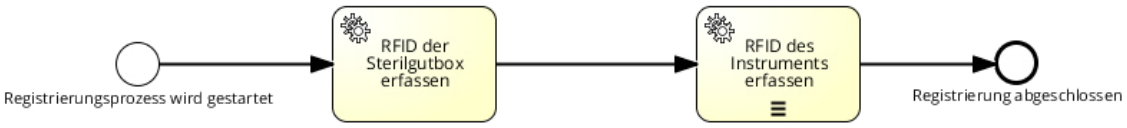


Abbildung 18: Prozessdiagramm Instrumentenset registrieren
Quelle: Eigene Darstellung

In Abbildung 18 wird der Vorgang der Registrierung eines Instrumentensets dargestellt. „Alle Instrumente, die zur Mehrfachnutzung aufbereitet werden können besitzen einen RFID über den auch die Registrierung abläuft. […] Gepackt wird ein Set in eine Sterilgutbox, auch diese hat einen RFID“ [20]. Die Registrierung ist als Call Activity modelliert. Call Activities sind eigenständige BPMN Prozesse, die von anderen Prozessen integriert werden können. Dies ermöglicht die Wiederverwendung an verschiedenen Stellen in anderen Prozessen und vermeidet dadurch redundante Modellierung. Die beiden Tasks *RFID der Sterilgutbox erfassen* und *RFID des Instruments erfassen* sind Service Tasks, da keine Interaktion mit Mitarbeitern des Sterilgutaufbereiters vorliegt. Der Task *RFID des Instruments erfassen* ist als sequentielle Mehrfachausführung gekennzeichnet. Dies bedeutet, dass jedes Instrument im Instrumentenset nacheinander mit Hilfe von RFID erfasst wird.

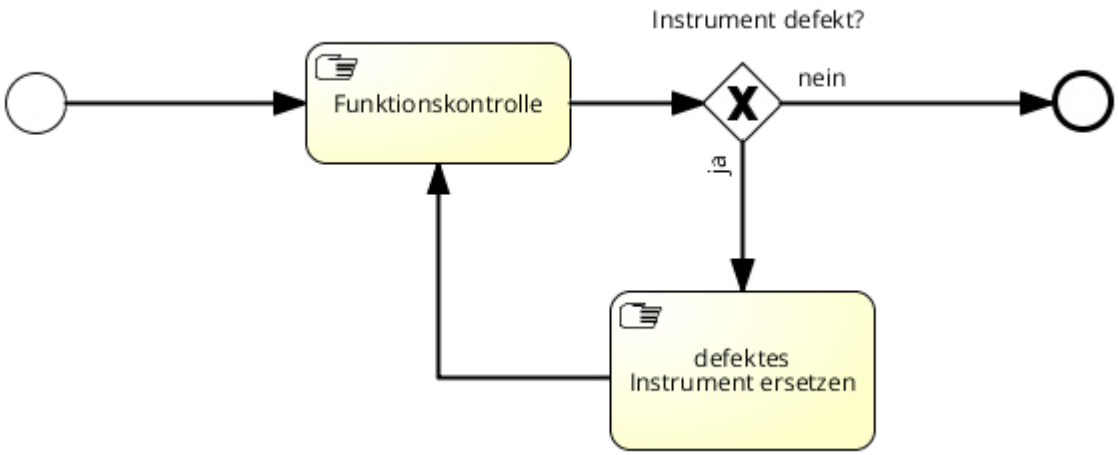


Abbildung 19: Prozessdiagramm Funktionskontrolle
Quelle: Eigene Darstellung

Abbildung 19 zeigt die Modellierung der Funktionskontrolle eines einzelnen Instruments. „Danach wird jedes Instrument des Sets einer Funktionskontrolle unterzogen, wenn ein Defekt festgestellt wird, wird das jeweilige Instrument ersetzt“ [20].

Die Tasks *Funktionskontrolle* und *defektes Instrument ersetzen* sind manuelle Tasks. Es findet keine Interaktion mit dem System statt. Wird ein defektes Instrument ersetzt so wird auch das das Ersatzinstrument einer Funktionskontrolle unterzogen um etwaige Fehlfunktionen auszuschließen. Dieses Vorgehen ist so nicht in der Beispielaufgabe beschrieben, erscheint aber logisch wenn man Fehlerquellen reduzieren möchte. Der Mehraufwand hält sich dabei in Grenzen und bringt einen Zugewinn an Qualität.

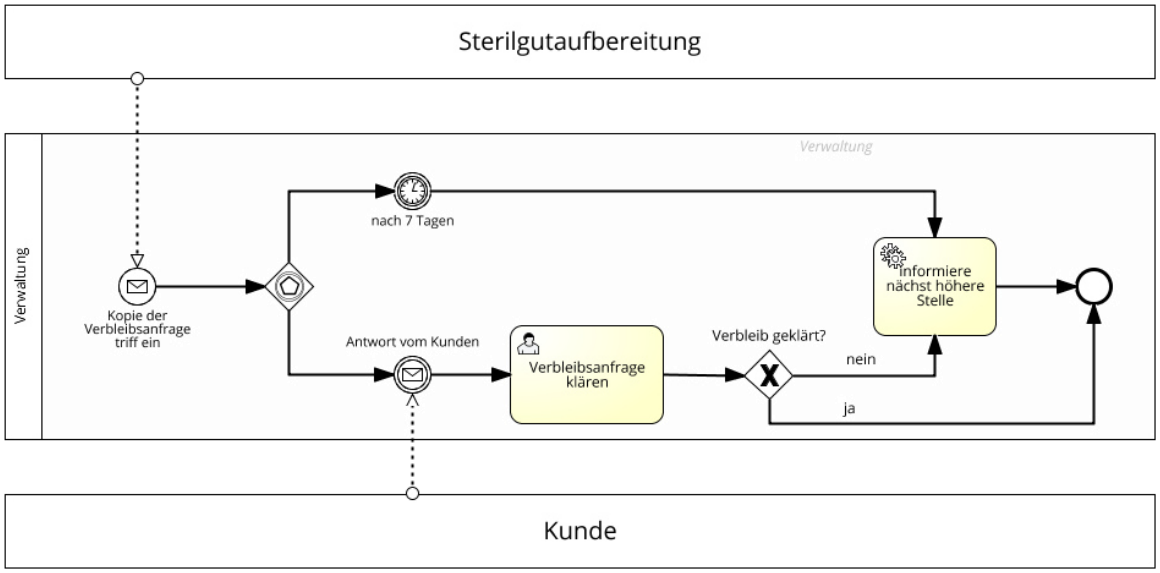


Abbildung 20: Prozessdiagramm Verbleibsanfrage
Quelle: Eigene Darstellung

Der Vorgang der Verbleibsanfrage ist in Abbildung 20 abgebildet. In der Aufgabenstellung stellt sich Verbleibsanfrage wie folgt dar. „Weitere Aufgabe der Verwaltung ist die Fristenkontrolle von Verbleibsanfragen. Diese entstehen, wenn bei der Vollständigkeitskontrolle auf der unreinen Seite, das Fehlen von Instrumenten festgestellt wird. Die Verbleibsanfrage gehen an den Kunden und in Kopie an die Verwaltung. Sollte nicht innerhalb von 7 Tagen eine Antwort vorliegen, dann wird die nächst höherer Ebene informiert. Dies erfolgt auch, wenn der Verbleib trotz einer Antwort nicht eindeutig geklärt werden konnte“ [20]. Sobald die Kopie der Verbleibsanfrage eintrifft wird der Prozess gestartet. Danach wird am Exklusiven Ereignis-basierten Gateway gewartet bis eines der darauf folgenden Ereignisse eintritt. Das Exklusive Ereignis-basierte Gateway wartet, im Vergleich zum Parallelen Ereignis-basierten Gateway, nur auf das Eintreten eines einzigen der nachfolgenden Ereignisse. Entweder erhält die Verwaltung eine Antwort vom Kunden oder das Timer Ereignis tritt nach 7 Tagen ein.

Der Task *Verbleibsanfrage klären* ist als Benutzertask konzipiert worden, da ein Mitarbeiter des Sterilgutaufbereiters die Antwort des Kunden bearbeiten muss. Des Weiteren ist die Form der Kundenantwort nicht genauer spezifiziert worden und somit sind sowohl E-Mails, schriftliche Antworten per Post oder auch telefonische Antworten denkbar. Diese Vielfalt schließt eine automatisierte Auswertung der Kundenantwort aus.

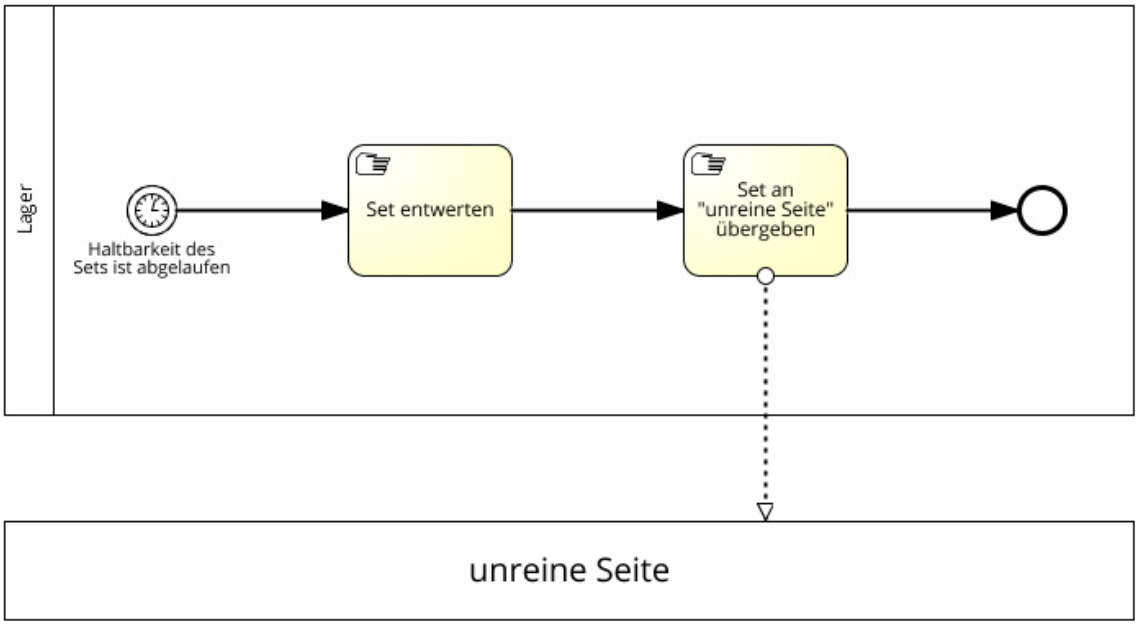


Abbildung 21: Prozessdiagramm Haltbarkeit des Instrumentensets abgelaufen
Quelle: Eigene Darstellung

Was passiert wenn die Haltbarkeit eines Instrumentensets abläuft wird in Abbildung 21 gezeigt. In der Aufgabenstellung heißt es dazu: „Falls die Haltbarkeit der Sterilisation eines gelagerten Sets abläuft, wird dieses entwertet und geht wie ein benutztes Set wieder in den Aufbereitungsprozess ein“ [20] Die Tasks *Set entwerten* und *Set an unreine Seite übergeben* sind als manuelle Tasks definiert. Dieser Prozess wird gestartet sobald das Ereignis *Haltbarkeit des Sets ist abgelaufen* eintritt. In der Aufgabenstellung ist keine Haltbarkeitsdauer definiert. Man kann in der der BPMN den Zeitpunkt wann ein Timer-Ereignis eintritt auch verbal formulieren. Diese Art der Modellierung eignet sich nur für eine fachliche Modellierung, da eine Process Engine diese Definition nicht auswerten kann. Um eine Automatisierung dieses Teilprozesses zu erreichen müsste der Zeitpunkt genauer definiert werden. Da dieser Teilprozess aber nicht umgesetzt werden soll spielt das in diesem Fall keine Rolle.

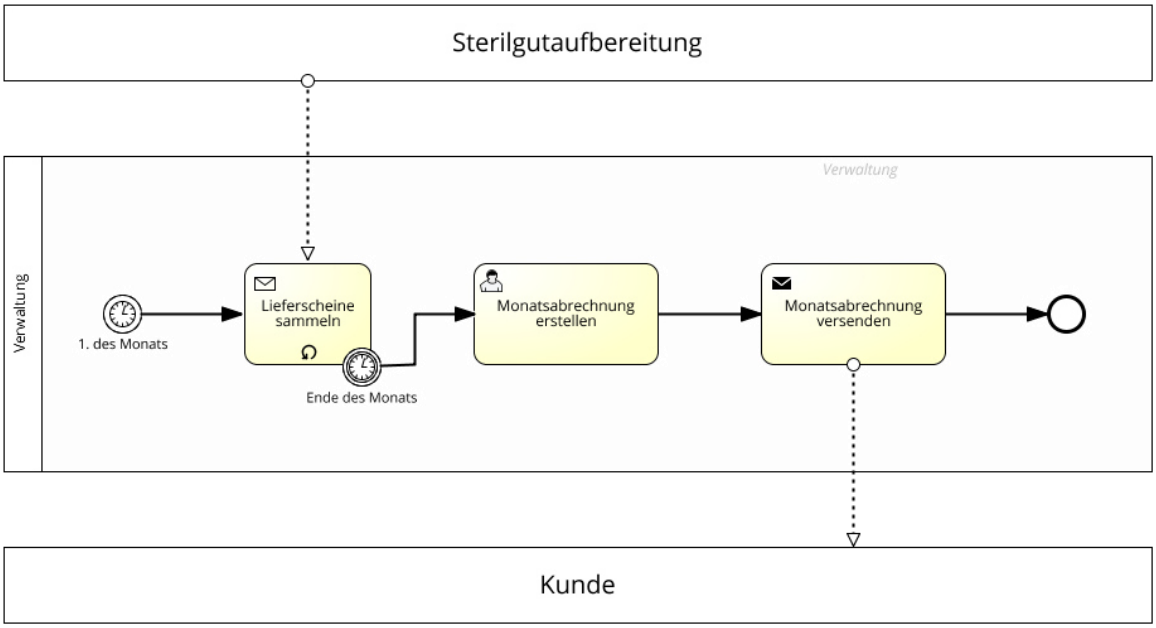


Abbildung 22: Prozessdiagramm Monatsabrechnung
Quelle: Eigene Darstellung

Ein weiterer Aspekt des Beispielszenarios ist die Abrechnung. „Die Abrechnung erfolgt über die Zentral-Verwaltung des Klinikverbundes. Diese erhält dazu Kopien der Lieferscheine und erstellt daraus eine Monatsabrechnung für den jeweiligen Kunden“ [20]. Am ersten Kalendertag des Monats tritt das Timer-Startereignis ein und startet eine Instanz des Prozess *Monatsabrechnung*. Der Task *Lieferschein sammeln* wartet auf die eintreffenden Lieferscheine aus der Sterilgutaufbereitung. Da normalerweise ein Kunde pro Monat nicht nur eine Bestellung tätigt und damit nur einen Lieferschein generiert, ist der Task als Schleife deklariert. Der Task wird wiederholt ausgeführt bis das unterbrechende Timer-Ereignis *Ende des Monats* eintritt. Danach wird der Task *Monatsabrechnung erstellen* ausgeführt. Sobald die Monatsabrechnung erstellt wurde wird der Task *Monatsabrechnung versenden* ausgeführt und die Abrechnung wird an den Kunden verschickt.

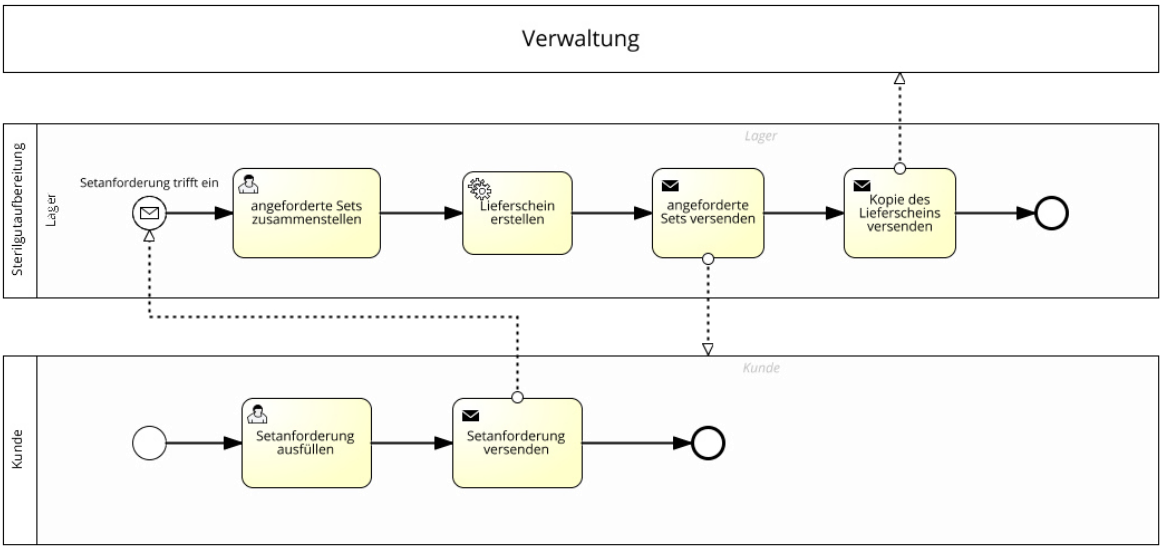


Abbildung 23: Prozessdiagramm Setanforderung durch den Kunden
Quelle: Eigene Darstellung

In der Beschreibung des Beispielszenarios wird eine Setanforderung wie folgt beschrieben. „Bei Sterilgutbedarf füllt ein Kunde eine Setanforderung aus und schickt dieses ans Lager. Dort wird die Lieferung zusammengestellt und an den Kunden versandt“ [20]. Daraus wurde ein Prozess, welcher in Abbildung 23 zu sehen ist, modelliert. Der Prozess beginnt im Pool *Kunde*. Nachdem der Benutzertask *Setanforderung ausfüllen* und der sendende Task *Setanforderung versenden* ausgeführt wurden, wird eine Nachricht an das Message-Startereignis im Pool *Sterilgutaufbereitung* verschickt. Sobald die Nachricht dort eintrifft wird als nächstes der Task *angeforderte Sets zusammenstellen* ausgeführt. Nachdem die Sets zusammengestellt wurden, wird durch den folgenden Servicetask ein dazugehöriger Lieferschein erstellt. Dieser Lieferschein ist enorm wichtig, da er die Grundlage für die monatliche Abrechnung ist. Im darauf folgenden sendenden Task werden die angeforderten Instrumentensets an den Kunden verschickt. Anschließend wird noch eine Kopie des Lieferscheins, für die monatliche Abrechnung an die Verwaltung versendet.

4.2 Umsetzung

Nach der Modellierung der einzelnen Prozesse aus dem Beispielszenario mit Hilfe des Signavio Editors, wird nun versucht die erfolgreiche Ausführung der Prozesse mit Hilfe der Camunda BPM zu erreichen. Die Vorstellung, dass man einfach die modellierten Prozessdiagramme nimmt und sie in einer beliebigen Process Engine ausführen kann erscheint sehr reizvoll, doch davon ist der aktuelle Stand der Technik noch weit entfernt. Die Diagramme müssen noch mit zusätzlichen Informationen angereichert werden. Jede Process Engine verlangt dabei eine andere Form der Informationen um das BPMN 2.0 Diagramm auszuführen. Für die Camunda Process Engine gibt es den Camunda Modeler, um BPMN Diagramme mit den notwendigen Informationen zu ergänzen. Der Camunda Modeler ist als Plugin für die Eclipse Entwicklungsumgebung erhältlich.

Zu Beginn der Arbeit wurde mit der Camunda BPM Version 7.1-final gearbeitet. Da die Camunda BPM zurzeit ein sich sehr schnell weiterentwickelndes Projekt ist wurde nach kurzer Zeit auf die aktuellste Entwicklungsversion gewechselt. Zum Zeitpunkt dieser Arbeit ist die aktuellste Version 7.2-alpha5. Das Release der finalen Version von 7.2-final steht kurz bevor. Der Wechsel auf die neuste Version verspricht einige Vorteile. Zum einen werden jetzt komplexe Datentypen als Prozessvariablen unterstützt. Zum anderen werden mit der AngularJS Integration die Möglichkeiten zur dynamischen Erstellung und Gestaltung von Task Forms enorm erweitert[5].

Die in Signavio erstellten Diagramme werden im Camunda Modeler mit Informationen angereichert. Dazu müssen die Diagramme in den Eclipse Workspace kopiert werden. Wie bereits in Kapitel 2.3.5 Cycle beschrieben, bietet Camunda mit dem Camunda Cycle eine Software an, die die Diagramme aus Signavio und dem Eclipse Workspace konsistent halten soll. Dieser Ansatz hat sich allerdings als nicht praktikabel erweisen. Der Camunda Modeler besitzt ein anderes Layout Verhalten als der Signavio Editor. Diagramme besitzen in Signavio ein anderes Layout als im Camunda Modeler. Das hat zur Folge, dass bei jedem Roundtrip das Layout der Diagramme verändert wird. Dies führt zu unübersichtlichen und teilweise unleserlichen Diagrammen und muss nach jedem Roundtrip manuell korrigiert werden. Um diese Probleme zu vermeiden wird im Rahmen dieser Arbeit auf den Roundtrip mit dem Camunda Cycle verzichtet. Die Diagramme werden aus dem Siganvio Editor mit dessen Exportfunktion in BPMN 2.0 konforme XML Dateien exportiert.

[5] Die Auflistung der neuen Feature ist nicht vollständig. Für eine vollständige Liste der neuen Features die Camunda Homepage aufsuchen.

Danach wird ein neues Maven[6] Projekt in Eclipse angelegt und das Projekt für die Erstellung einer Process Application vorbereitet. Die exportierten Diagramme werden dann von Hand in den dafür vorgesehenen Ordner im Eclipse Workspace kopiert. Jetzt erfolgt die Bearbeitung mit dem Camunda Modeler. Anschließend wird die Process Application auf einem Webserver mit Webcontainer deployed. Für dieses Szenario wurde der Apache Tomcat 7 gewählt. Camunda selbst bietet die Camunda BPM als Bundle mit weiteren Webservern auf ihrer Homepage an. Neben Apache Tomcat 7 werden noch Bundles mit JBoss AS 7, WildFly und GlassFish 3.1.x angeboten. Nachdem die Process Application erfolgreich deployed wurde, kann über einen Browser mit dem Prozess interagiert werden. Camunda bietet von Haus aus mit der Tasklist und dem Cockpit zwei Werkzeug an um die Prozesse auf dem Webserver zu administrieren.

4.2.1 Datenbankkonzept

Für die Umsetzung der Diagramme wird eine Datenbank zur Datenhaltung benötigt. Die Datenbank soll alle vorhandenen Instrumente und Instrumentensets speichern. Sie soll auch zu jedem Instrument und Instrumenten die aktuelle Position im Prozess speichern. Camunda liefert von Haus aus keine eigene Datenbank mit. Im Rahmen dieser Arbeit wird die H2 Datenbank verwendet. Die H2 Datenbank bietet sich für ein Testsetup an, da sie eine leichtgewichtige in-Memory Datenbank mit JDBC API ist (vgl. [21]).

Das verwendete Datenbankschema ist in Abbildung 24 abgebildet. Die Datenbank besteht aus zwei Tabellen. Die Tabellen sind mit einer 1:n Beziehung miteinander verknüpft. Jedes Instrument gehört entweder genau einem oder keinem Instrumentenset an. Und jedes Instrumentenset besitzt keines bis unendlich viele Instrumente.

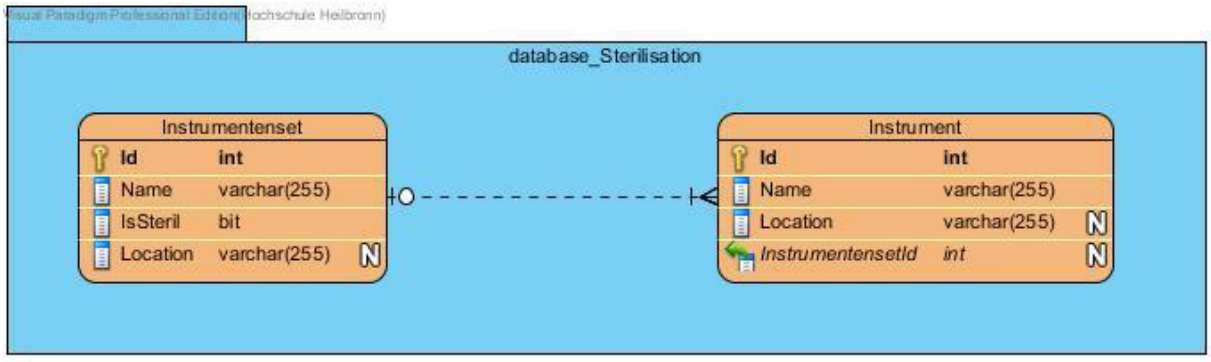


Abbildung 24: Datenbankschema
Quelle: Eigene Darstellung

[6] Apache Maven ist ein Software Projekt Management Werkzeug. Siehe http://maven.apache.org/

Für die Kommunikation mit der Datenbank gibt es die Klasse *DatabaseManager*. Diese Klasse implementiert das Singleton Pattern und initialisiert die Datenbank bei der Erzeugung der Instanz. Bei der Initialisierung werden alle vorhandenen Daten aus der Datenbank entfernt und die Datenbank anschließend neu befüllt. Dieses Vorgehen stellt sicher, dass bei jedem Durchlauf immer auf den gleichen Daten gearbeitet wird. Das Klassendiagramm des *DatabaseManager* ist in Abbildung 25 zu sehen. Es wird eine Connection Pool[7] verwendet um die Connections zur Datenbank zu verwalten.

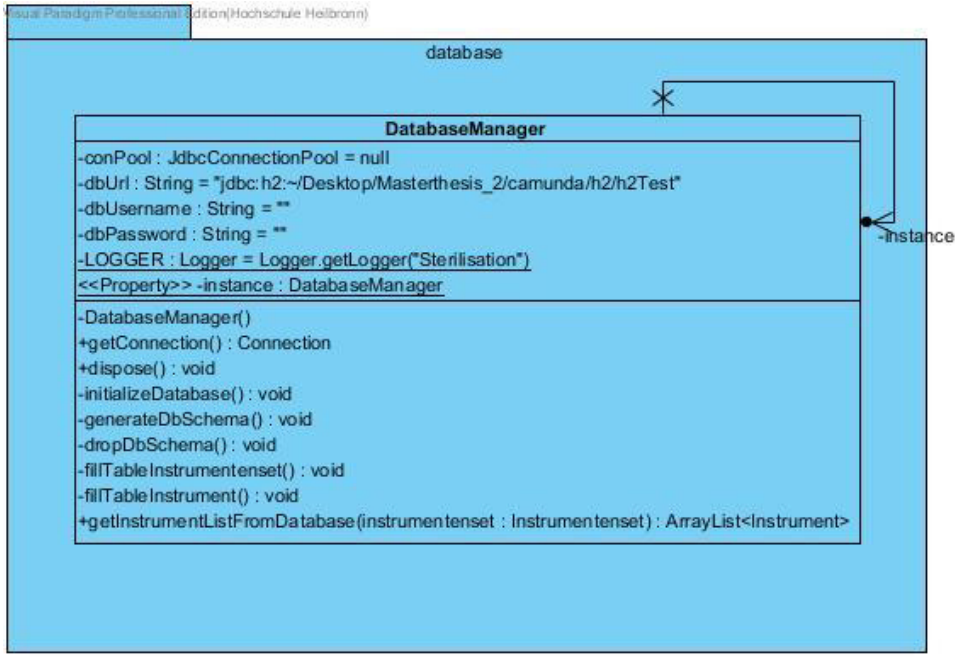


Abbildung 25: Klassendiagramm DatabaseManager
Quelle: Eigene Darstellung

4.2.2 Instrument und Instrumentenset

Im Rahmen dieser Arbeit soll der Prozess der Sterilgutaufbereitung umgesetzt werden. Da nicht mit echtem Sterilgut gearbeitet werden kann wurden die Klassen Instrument.java und Instrumenteset.java für die Umsetzung implementiert. Sie simulieren bei der Ausführung ein Instrumentenset bzw. dessen Instrumente. In Abbildung 26 ist das zugehörige Klassendiagramm zu sehen. Die Klasse *Instrument* besitzt die Datenfelder *id* und *name,* welche das Instrument eindeutig identifizieren. Die Klasse *Instrumentenset* besitzt neben den Datenfeldern *id* und *name* auch eine Liste mit allen enthaltenen Objekten der Klasse *Instrument*.

[7] Beim Connection Pooling werden Connections zur Datenbank in einem sogenannten Connection Pool vorgehalten. Bei Bedarf kann ein Client sich eine Connection reservieren und verwenden. Wird die Connection nicht mehr gebraucht kann sie an den Connection Pool zurückgegeben werden (vgl. [9]).

Später bei der Umsetzung wird ein Objekt der Klasse *Instrumentenset* beim Start des Prozesses *Sterilisation* als Prozessvariable übergeben. Diese Prozessvariable entspricht einem unreinen Instrumentenset, welches aufbereitet werden soll.

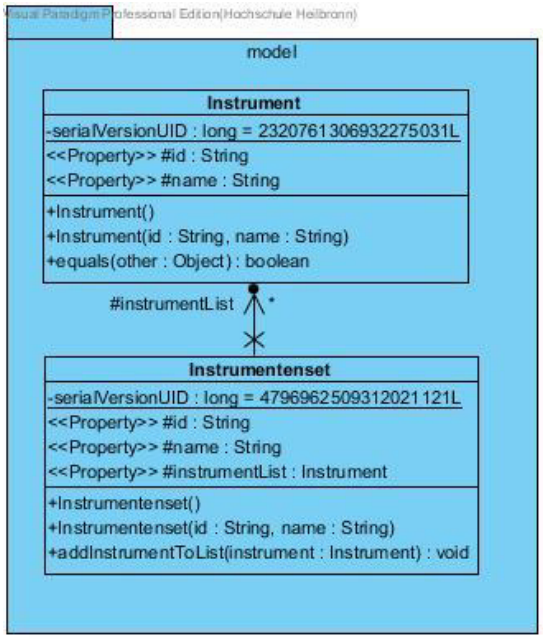


Abbildung 26: Klassendiagramm Instrument und Instrumentenset
Quelle: Eigene Darstellung

4.2.3 Process Application

Eine Process Application beinhaltet die Prozessdefinitionen und wird auf der Process Engine ausgeführt. Für die Erstellung der Process Application sind die nachfolgenden Schritte notwendig.

4.2.3.1 pom.xml

Die Camunda BPM nutzt Maven als Grundlage zur Erstellung einer Process Application. Wie bei jedem Maven Projekt wird eine Datei mit dem Namen pom.xml benötigt. Diese Datei definiert unter anderem die verwendeten Libraries und deren Versionen. Es wird auch das Format, in welchem die Process Application gespeichert wird definiert.

```
<packaging>war</packaging>
<dependencies>
	<dependency>
		<groupId>com.h2database</groupId>
		<artifactId>h2</artifactId>
		<version>1.4.182</version>
	</dependency>
	<dependency>
		<groupId>org.camunda.bpm</groupId>
		<artifactId>camunda-engine</artifactId>
		<version>7.2.0-alpha5</version>
		<scope>provided</scope>
	</dependency>
	<dependency>
		<groupId>javax.servlet</groupId>
		<artifactId>javax.servlet-api</artifactId>
		<version>3.0.1</version>
		<scope>provided</scope>
	</dependency>
</dependencies>
```

Listing 1: Auszug aus der pom.xml
Quelle: Eigene Darstellung

Listing 1 zeigt einen Ausschnitt aus der verwendeten pom.xml. Das Format der Process Application ist als war Container definiert. Dieses Format wird benötigt um die Process Application auf einem Webserver deployen zu können. Es sind die camunda-engine und die javax.servlet-api als Dependencies eingetragen. Beide Abhängigkeiten werden fürs Kompilieren der Process Application benötigt.

4.2.3.2 Process Application Class

Nach der Konfiguration von Maven muss nun die Process Application konfiguriert werden. Dafür muss eine Process Application Class definiert werden. Die Process Application Class dient als Interface zwischen der Process Engine und der eigenen Process Application.

Die Process Application Class erbt von der Klasse ServletProcessApplication. Die Annotation *@ProcessApplcation* stellt sicher, dass die Process Engine diese Klasse als Process Application Class erkennt.

Dabei kann auch ein Name für die Process Application vergeben werden. Die Process Application Class kann auch benutzt werden um Prozessinstanzen automatisch zu erzeugen.

In Listing 2 wird die verwendete Process Application Class dargestellt. Es wird die Methode *startFirstProcess* definiert, welche nach dem Deployment des Prozessarchivs aufgerufen wird. Dies geschieht durch die Annotation *@PostDeploy*. Die Methode *startFirstProcess* erstellt eine neue HashMap mit dem Namen *variables*. Es wird ein Objekt der Klasse *Instrumentenset* mit verschiedenen *Instrumenten* erzeugt. Das Instrumentenset wird unter dem String *instrumentenset* in die HashMap eingefügt. Zum Schluss wird eine neue Prozessinstanz der Prozessdefinition mit der *Process ID Sterilisationsprozess* erzeugt und die HashMap als Sammlung von Prozessvariablen übergeben.

Das als Prozessvariable übergebene Instrumentenset simuliert in dieser Arbeit das unreine Instrumentenset, welches vom Sterilgutdienstleister aufbereitet wird.

```
@ProcessApplication("Sterilisation")
public class SterilizationApplication extends ServletProcessApplication {

	@PostDeploy
	public void startFirstProcess(ProcessEngine processEngine) {

		// Declare Variables
		Map<String, Object> variables = new HashMap<String, Object>();
		Instrumentenset instrumentenset = new
		Instrumentenset("ID1001","Appendixset 1");
		instrumentenset.addInstrumentToList(new Instrument("ID501", "Pinzette
		klein"));
		instrumentenset.addInstrumentToList(new Instrument("ID506", "Schere
		(Mayo)"));
		instrumentenset.addInstrumentToList(new Instrument("ID511", "Schere
		(Metzenbaum)"));
		instrumentenset.addInstrumentToList(new Instrument("ID516",
		"Arterienklemme"));
//		instrumentenset.addInstrumentToList(new Instrument("ID521",
		"Skalpell"));
		instrumentenset.addInstrumentToList(new Instrument("ID666", "Evil
		Scissors"));

		variables.put("instrumentenset", instrumentenset);

		// Start Process Instance and pass variables
		processEngine.getRuntimeService().startProcessInstanceByKey(
		"Sterilisationsprozess", variables);
	}

}
```

Listing 2: Auszug aus SterilizationApplication.java
Quelle: Eigene Darstellung

4.2.3.3 processes.xml

Ein weiterer zentraler Punkt in der Erstellung einer Process Application ist die Datei processes.xml. Sie befindet sich im Normalfall im Ordner src/main/ressources/META-INF des Projekts. Die Datei beschreibt die Prozessarchive der Process Application. Es werden die Eigenschaften des Prozessarchivs definiert. Es kann unter anderem definiert werden welche Process Engine verwendet wird[8]. Außerdem kann das Verhalten der Process Engine beim Deployment bzw. Undeployment des Prozessarchivs definiert werden.

Die Eigenschaft *isDeleteUponUndeploy* definiert ob die Prozesse im Prozessarchiv beim Undeployment aus der Process Engine entfernt werden. In einer Test- bzw. Entwicklungsumgebung ist dies normalerweise die gewünschte Verhaltensweise. Dadurch werden bei der Aktualisierung der Process Application die vorhandenen Versionen der alten Prozessdefinitionen verworfen. Bei der Umsetzung von neuen Prozessdiagrammen können mitunter mehrere Dutzend Versuche notwendig sein, um die Prozessanforderungen zu erfüllen. Dies hätte mehrere Dutzend Versionen desselben Prozesses zur Folge. Camunda bietet zum aktuellen Zeitpunkt keine Möglichkeit alte Versionen der Prozesse von Hand wieder zu entfernen. Es bleibt nur der Umweg über eine neue Webserverinstanz. Für den Produktiveinsatz sollte diese Einstellung auf false gesetzt werden, sonst werden beim Deployment einer aktuelleren Version der Prozesse im Prozessarchiv alle noch aktiven Instanzen der alten Prozessdefinition entfernt. Im Falle eines Bestellprozesses, mit noch aktiven Instanzen würden alle noch nicht abgeschlossenen Instanzen auf der Stelle gelöscht werden. Die Eigenschaft *isScanForProcessDefinitions* definiert ob die Prozess Engine beim Deployment des Prozessarchivs automatisch alle vorhandenen Prozessdefinitionen deployen soll. Dabei durchsucht die Process Engine das Prozessarchiv nach Dateien mit der Endung .bpmn20.xml und .bpmn (vgl. [18]). Alle gefundenen Dateien mit diesen Endungen werden automatisch deployed und stehen, falls sie so definiert wurden, zur Ausführung bereit. Ist diese Einstellung auf false gesetzt werden nach dem Deployment keine Prozessdefinitionen gesucht. Die im Prozessarchiv vorhandenen Prozessdefinitionen können dann nicht ausgeführt werden. Die Datei kann auch leer gelassen werden. In diesem Fall werden die default Werte verwendet.

Listing 3 zeigt die im Rahmen der Arbeit verwendete processes.xml. Es wird die default Process Engine für das Prozessarchiv verwendet. Die Eigenschaft *isDeleteUponUndeploy* ist auf true gesetzt, da immer nur die aktuellste Version der Prozessdefinitionen verwendet wird.

[8] Es können verschiedene Process Engines mit verschiedenen Einstellungen innerhalb eines Webservers vorhanden sein (vgl. [4]).

Um die automatische Erkennung der Prozessdefinitionen einzuschalten, ist die Eigenschaft *isScanForProcessDefinitions* auf true gesetzt.

```
<?xml version="1.0" encoding="UTF-8" ?>

<process-application
    xmlns="http://www.camunda.org/schema/1.0/ProcessApplication"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <process-archive>
    <process-engine>default</process-engine>
    <properties>
      <property name="isDeleteUponUndeploy">true</property>
      <property name="isScanForProcessDefinitions">true</property>
    </properties>
  </process-archive>
</process-application>
```

Listing 3: processes.xml
Quelle: Eigene Darstellung

4.2.4 Bearbeitung der Prozessdiagramme

Nachdem die Process Application konfiguriert wurde, werden jetzt die einzelnen BPMN 2.0 Diagramme mit den notwendigen Informationen für die Ausführung angereichert. Die Diagramme werden dazu im Camunda Modeler geöffnet und bearbeitet. Die folgende Auflistung gibt einen Überblick über die Schritte, die benötigt werden um die erforderlichen Informationen in die Diagramme einzufügen. Die hier aufgeführten Schritte beziehen sich auf das Beispielszenario und sind als Dokumentation des Arbeitsverlaufs zu verstehen. In anderen Szenarien sind eventuell zusätzliche Schritte notwendig oder es können Schritte ausgelassen werden.

- Prozessdiagramme ausführbar deklarieren
- Call Activities referenzieren
- User Tasks definieren
    - Task Form erstellen und dem User Task zuweisen
    - User Task einem User zuweisen
- Service Tasks definieren
    - Delegate Klasse definieren
    - Delegate Klasse dem Service Task zuweisen
- Gateways definieren
- Error Events definieren
- Nachrichtenaustausch
    - Send Tasks mit entsprechenden Messages korrelierenden
    - Messages mit den entsprechenden empfangenden Events korrelieren
- Timer Events definieren
- Multiple Instances definieren

Jeder der hier aufgeführten Schritte wird nachfolgend anhand eines Beispiels dargestellt.

Prozessdiagramme ausführbar deklarieren

Um Prozess überhaupt ausführen zu können muss man der Camunda Engine mitteilen, welche Prozesse ausführbar sind. Dies geschieht entweder direkt auf einer Lane, falls vorhanden, oder auf dem gesamten Prozess, wenn keine Lane vorhanden ist. Dazu wird die entsprechende Lane ausgewählt und im Properties Panel die Checkbox *Is Executable* ausgewählt. Soll der Prozess später als Call Activity referenziert werden sollte noch eine sprechende *Process Id* vergeben werden. Abbildung 27 zeigt die entsprechenden Einstellungen im Property Panel der Lane *Verwaltung* aus dem Prozess *Verbleibsanfrage*.

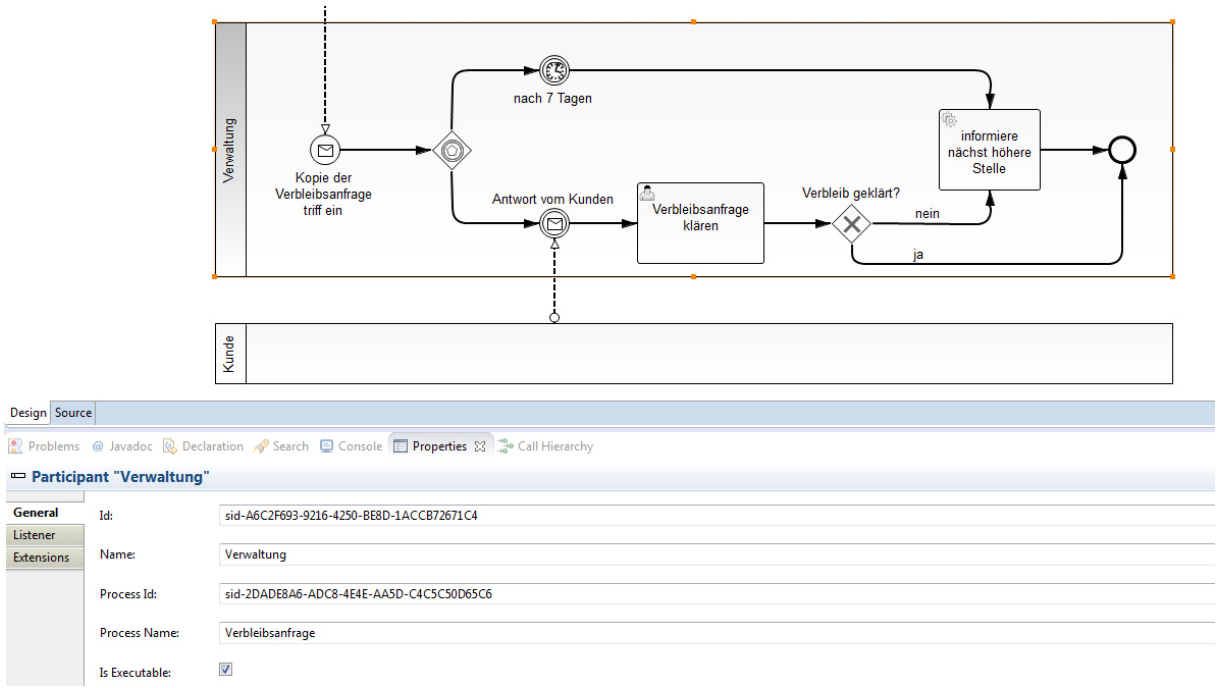


Abbildung 27: Benutzeroberfläche Properties Panel Lane
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

Call Activities referenzieren

Als Call Activities können alle Prozesse, welche zuvor als ausführbar deklariert wurden, referenziert werden. Im Property Panel muss beim Punkt *Called Element* die Prozess Id des Prozesses angegeben werden, der aufgerufen werden soll. Soll ein Austausch der Prozessvariablen der beiden Prozesse möglich sein, dann muss das Mapping für Prozessvariablen aktiviert werden. Beim Mapping wird zwischen zwei Richtungen unterschieden. Dem Mapping der Prozessvariablen in die Call Activity und dem Mapping aus der Call Activity heraus in den aktuellen Prozess. Dabei wird für jede Richtung getrennt festgelegt welche Prozessvariablen ausgetauscht werden sollen. Mit dem Aktivieren der Checkbox *All Variables* werden alle Prozessvariablen an den Sub- bzw. Mainprozess weitergegeben. In Abbildung 28 ist der entsprechende Ausschnitt aus dem Property Panel der Call Activity *Instrumentenset registrieren* abgebildet.

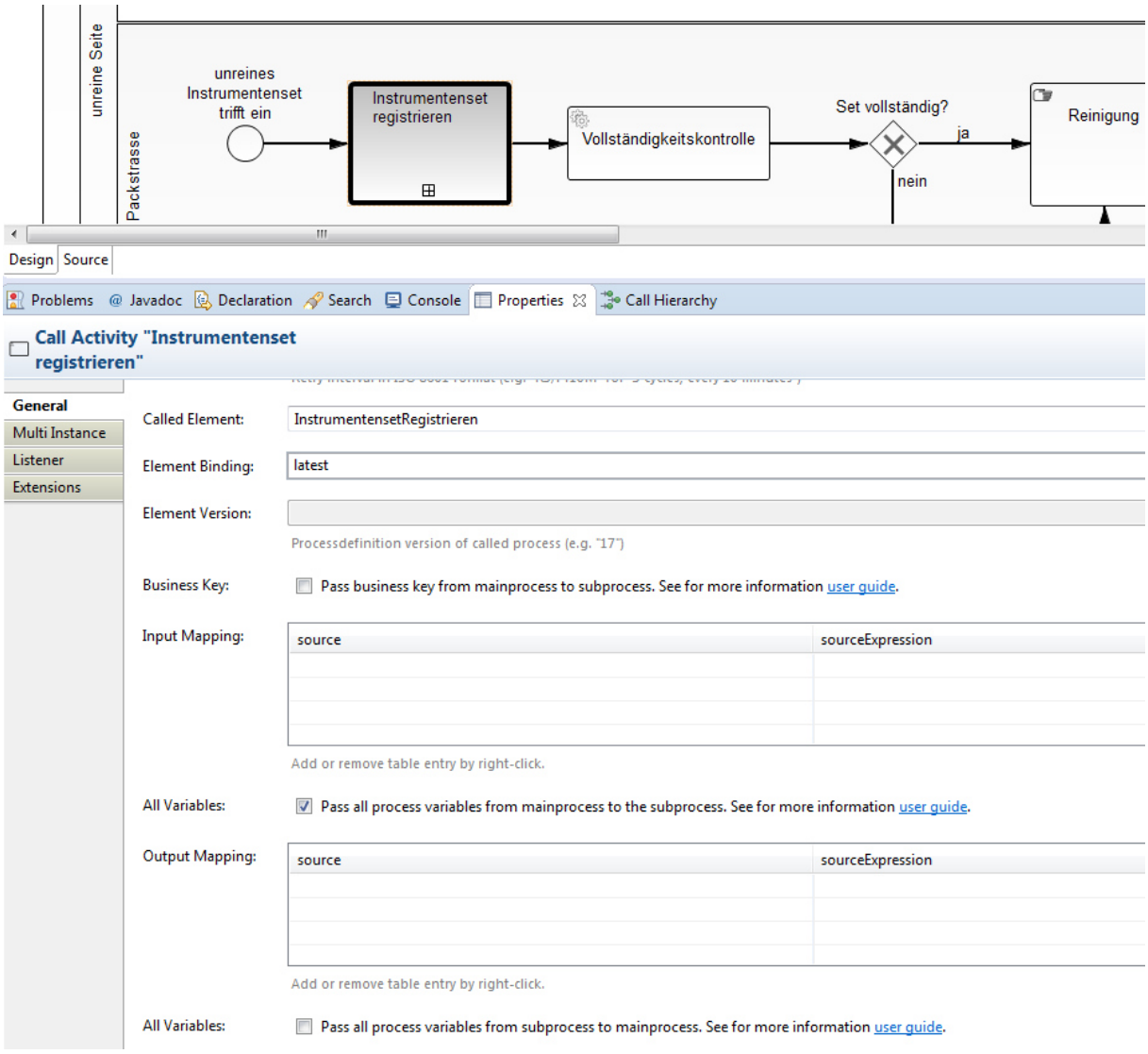


Abbildung 28: Benutzeroberfläche Properties Panel Call Activity
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

User Tasks

Wie bereits in Kapitel 2.3.3 Tasklist erwähnt wurde, werden Task Forms verwendet um mit Benutzern zu interagieren. Dazu muss eine Task Form erzeugt werden. Task Forms werden im Verzeichnis src/main/webapp/forms angelegt. In Listing 4 ist der Code, der Task Form für die Funktionskontrolle, abgebildet. Mit Hilfe von JavaScript wird die Prozessvariable *instrument* geladen. Wenn diese geladen ist, wird sie einer lokalen Variable mit demselben Namen zugewiesen. Jetzt kann man die Id und den Namen des aktuellen Instruments abfragen und darstellen. Es wird ein Eingabefeld definiert, in welchem der Benutzer angeben muss ob das Instrument die Funktionskontrolle bestanden hat oder nicht. Aus dieser Eingabe wird eine neue Variable mit dem Namen *functiontestInstrumentSuccessful* vom Typ boolean erzeugt[9].

```
<form class="form-horizontal" role="form">

	<!-- fetch process variable "instrumentenset" -->
	<script cam-script type="text/form-script">
	camForm.on('form-loaded', function() {
		camForm.variableManager.fetchVariable('instrument');
	});
	camForm.on('variables-fetched', function() {
		$scope.instrument =
		camForm.variableManager.variableValue('instrument');
	});
	</script>


	<div class="form-group">
		<label for="instrumentensetId-field">ID:
			{{instrument.id}}</label>
	</div>
	<div class="form-group">
		<label for="instrumentensetName-field">Name:
			{{instrument.name}}</label>
	</div>

	<div class="form-group">
		<label>Hiermit versichere ich, dass das vorliegende Instrument die
		erforderlichen Anforderungen erf&uuml;llt hat.</label>
		<br/>
		<input type="text"
			cam-variable-name="functiontestInstrumentSuccessful"
			cam-variable-type="Boolean"
			required/>
	</div>

</form>
```

Listing 4: Embedded Task Form formFunktionskontrolle.html
Quelle: Eigene Darstellung

[9] Die Eingabe der Variable erfolgt in einem Textfeld, da zum Zeitpunkt der Umsetzung noch keine Checkboxen implementiert sind. In der finalen Version von 7.2 sollen diese jedoch vorhanden sein.

In Abbildung 29 ist die korrespondierende Task Form in der Camunda Task List zu sehen. Um eine Task Form einem User Task zuzuweisen wird der Eintrag *Form Key* im Properties Panel verwendet. Dort gibt man den Pfad zur entsprechenden Task Form an. Handelt es sich um eine Embedded Task Form muss vor dem Pfad zur Task Form *embedded:app:* angegeben werden. Bei External Task Forms reicht die Angabe von *app:* vor dem Pfad aus.

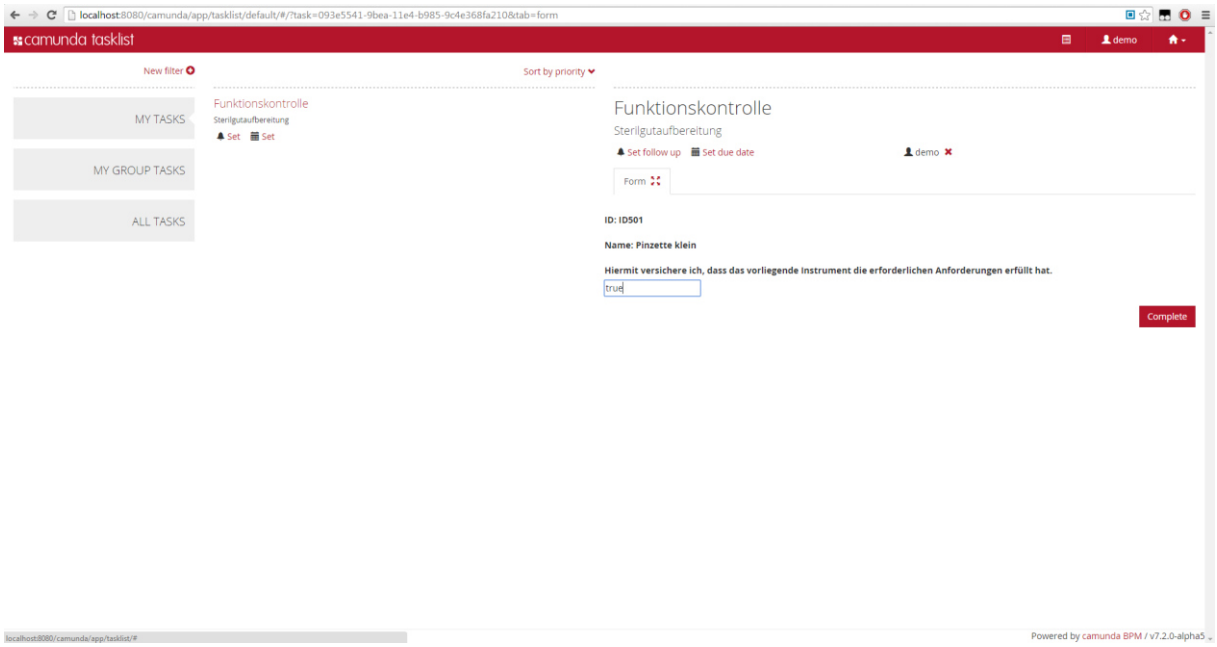


Abbildung 29: Benutzeroberfläche Task Form Funktionskontrolle
Quelle: Screenshot Camunda Tasklist Version 7.2-alpha5

User Tasks müssen einem User zugewiesen werden, um in dessen Tasklist sichtbar zu sein. Das entsprechende Feld im Properties Panel ist *Assignee*. Dort wird der Benutzername eines Benutzers eingetragen. In der Tasklist dieses Benutzers erscheint ein neuer Task sobald der User Task aufgerufen wird. Diese Zuweisung kann auch während der Ausführung des Prozesses dynamisch erfolgen. Es ist auch möglich einen Task mehreren Benutzern zuzuweisen. Dies erfolgt über das Feld *Candidate Users*. Der Task erscheint in der Tasklist aller beteiligten Benutzer. Wenn der Task von einem dieser Benutzer abgearbeitet wurde, verschwindet er auch aus der Tasklist der anderen. Um nicht alle möglichen Benutzer einzeln aufzählen zu müssen gibt es mit dem Feld *Candidate Groups* die Möglichkeit ganze Benutzergruppen zu deklarieren. In Abbildung 30 wird der entsprechende Ausschnitt des Property Panels für User Tasks gezeigt. Für die Umsetzung des Beispielszenarios steht ist der Benutzer *demo* angelegt worden. Aus diesem Grund werden alle User Tasks fest diesem Nutzer zugeordnet.

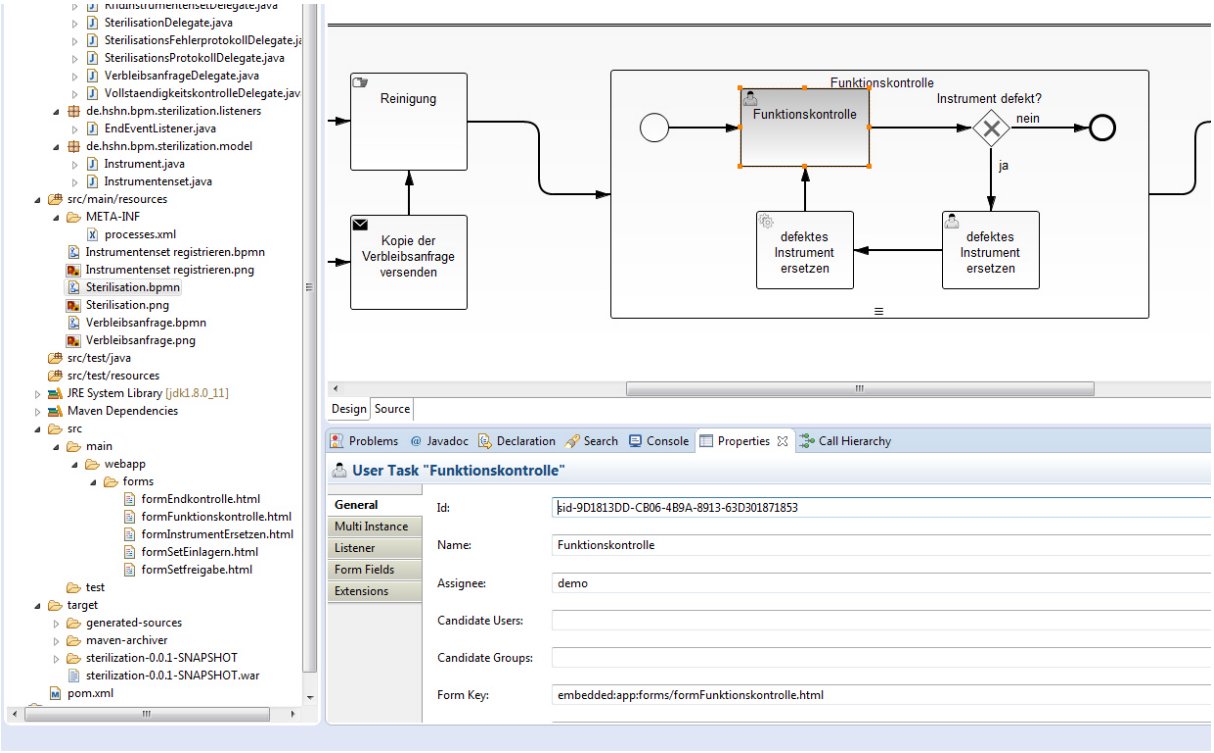


Abbildung 30: Benutzeroberfläche Properties Panel User Task
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

Service Tasks definieren

Service Tasks ermöglichen das Aufrufen von Java Code aus dem Prozess heraus. Sie bilden damit eine Schnittstelle zwischen Prozessen und Computerprogrammen. Um einen Service Task auszuführen wird ein Java Delegate benötigt. Java Delegates sind Java Klassen, welche das JavaDelegate Interface von Camunda implementieren. Das Interface schreibt eine Methode mit dem Namen *execute* vor, welche beim Ausführen des Service Tasks aufgerufen wird. Innerhalb dieser Klassen kann ganz gewöhnlicher Java Code verwendet werden. Dadurch werden Service Tasks zu einer sehr eleganten und mächtigen Methode eigenen Programmcode auszuführen. In Listing 5 wird ein Auszug aus dem JavaDelegate der Vollständigkeitskontrolle aufgeführt. Hier wird ersichtlich wie ein JavaDelegate aussehen kann. In der execute Methode wird überprüft welche Instrumente fehlen. Dazu wird eine Anfrage an die Datenbank gestellt und abgefragt, welche Instrumente vorhanden sein sollen. Danach wird eine Methode mit dem Namen *compareInstruments* aufgerufen. Diese Methode liefert eine ArrayList zurück in der die fehlenden Instrumente aufgeführt werden. Wenn in der Liste keine Elemente vorhanden sind, ist das vorliegende Instrumentenset vollständig. In dem Fall wird die Prozessvariable *complete* auf den Wert true gesetzt. Das bedeutet, dass das Instrumentenset die Vollständigkeitsprüfung bestanden hat. Befinden sich jedoch Instrumente in der zurückgelieferten Liste, so fehlen diese im vorliegenden Instrumentenset.

In dem Fall wird die Prozessvariable *complete* auf den Wert false gesetzt und es wird eine weitere Prozessvariable erzeugt, die eine Liste mit den fehlenden Instrumenten beinhaltet. Diese Liste wird zu einem späteren Zeitpunkt mit der Verbleibsanfrage an den Kunden und die Verwaltung weitergegeben.

```
…
public class VollstaendigkeitskontrolleDelegate implements JavaDelegate {
	private final static Logger LOGGER = Logger.getLogger("Sterilisation");
	public void execute(DelegateExecution execution) throws Exception {
		ProcessEngine processEngine =
		ProcessEngines.getDefaultProcessEngine();
		RuntimeService runtimeService = processEngine.getRuntimeService();
		Instrumentenset instrumentenset = (Instrumentenset) runtimeService
				.getVariable(execution.getId(), "instrumentenset");

		if (instrumentenset != null) {
			ArrayList<Instrument> instrumentListFromDatabase =
			DatabaseManager.getInstance().
			getInstrumentListFromDatabase(instrumentenset);
			ArrayList<Instrument> instrumentListFromSet =
			instrumentenset.getInstrumentList();

			// Determine missing instruments
			ArrayList<Instrument> missingInstruments =
			compareInstruments(instrumentListFromDatabase,
			instrumentListFromSet);

			// Add variable "complete" and "missingInstruments" if
			"complete" is false
			if (missingInstruments.size() == 0) {
				LOGGER.info("Instrumentenset vollstaendig.");
				runtimeService.setVariable(execution.getId(),
				"complete", true);
			} else {
				String message = "Es fehlen folgende Instrumente:\r\n";
				for (Instrument instrument : missingInstruments) {
					message += "ID: " + instrument.getId() + ", Name:"
							+ instrument.getName() + "\r\n";
				}
				LOGGER.info(message);
				runtimeService.setVariable(execution.getId(),
				"complete", false);
				Instrumentenset instrumentensetMissingInstruments = new
				Instrumentenset();
				instrumentensetMissingInstruments.
				setInstrumentList(missingInstruments);
				runtimeService.setVariable(execution.getId(),
				"instrumentensetMissingInstruments",
				instrumentensetMissingInstruments);
			}
…
```

Listing 5: Auszug aus VollstaendigkeitskontrolleDelegate.java
Quelle: Eigene Darstellung

Nachdem das JavaDelegate erstellt ist wird es nun dem Service Task zugewiesen. Dazu wird im Properties Panel beim Eintrag *Class:* der vollständige Klassenname angegeben. In Abbildung 31 ist der entsprechen Ausschnitt aus dem Properties Panel zu sehen.

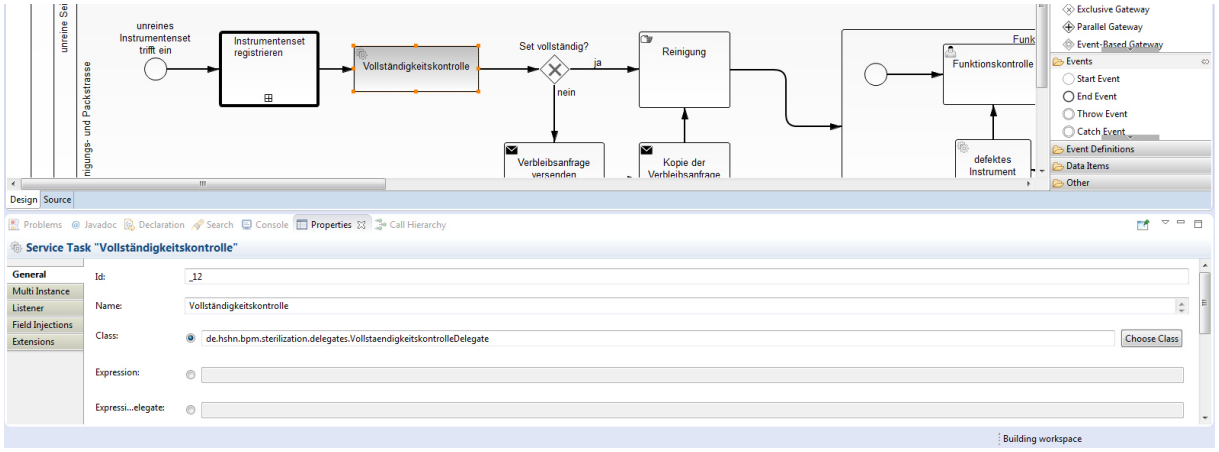


Abbildung 31: Benutzeroberfläche Properties Panel Service Task
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

Um Fehler zu vermeiden kann mit Hilfe des Modelers die entsprechende Java Delegate Klasse ausgewählt werden. Um diese Funktion zu benutzen klickt man auf den Button *Choose Class* auf der rechten Seite. Danach erscheint ein Dialog mit Suchfunktion. Dieser Dialog ist in Abbildung 32 zu sehen. Mit Hilfe der Suchfunktion lässt sich die gewünschte JavaDelegate Klasse schnell finden. Nach dem Bestätigen des Dialogs wird die ausgewählte Klasse automatisch eingetragen.

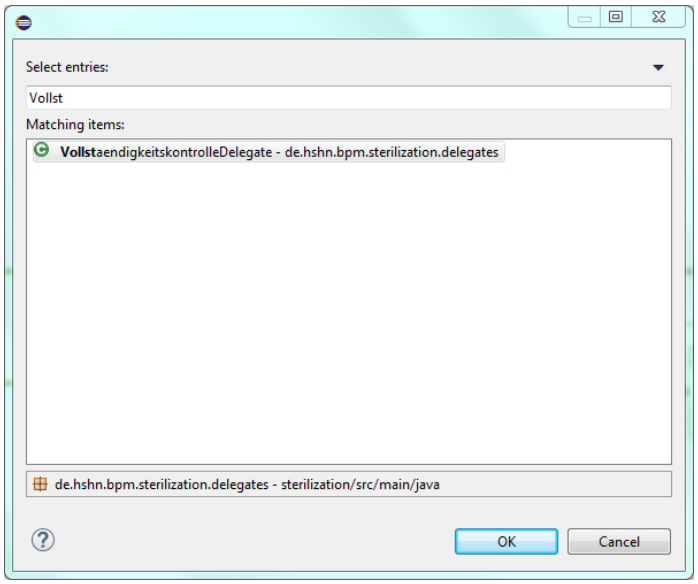


Abbildung 32: Benutzeroberfläche Choose Java Delegate Class Dialog
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

Gateways definieren

Die BPMN 2.0 spezifiziert mehrere verschiedene Gateways. Im Rahmen der Modellierung wurden davon das Parallele-, das Exklusive- und das Ereignis-basierte Gateway verwendet. Beim Parallelen Gateway bedarf es keiner zusätzlichen Informationen für die Ausführung, da für alle ausgehenden Sequenzflüsse ein neues Teiltoken erzeugt wird. Gleiches gilt für Ereignis-basierte Gateways. Hier wird die Entscheidung auf welchem Sequenzfluss das Token weiterwandert anhand des zeitlichen Eintretens der nachfolgenden Ereignisse gefällt. Das Ereignis, das zuerst eintritt wird mit dem Token versorgt und alle anderen Ereignisse werden fortlaufend ignoriert.

Beim Exklusiven Gateway ist eine Bearbeitung im Modeler notwendig. Die Process Engine kann sonst nicht entscheiden, auf welchen Sequenzfluss ein Token geleitet wird. Exklusive Gateways entscheiden anhand von Prozessvariablen den weiteren Sequenzfluss des Tokens. Die Bedingungen werden entgegen der Intuition nicht im Gateway selbst sondern an den ausgehenden Sequenzflüssen hinterlegt. Die Bedingungen werden mit Hilfe der Java Unified Expression Language (JUEL) beschrieben. In Abbildung 33 wird der entsprechende Ausschnitt aus dem Properties Panel für einen Sequenzfluss gezeigt. Es wird die Prozessvariable *complete* auf den Wert true getestet. Im Modeler findet keine Validierung der eingegebenen Bedingungen statt. Semantische oder Syntaktische Fehler werden erst beim Deployment sichtbar. Tritt der Fall ein, dass Bedingungen mehrerer Sequenzflüsse zutreffen, so wird das Token trotzdem nur auf einen Sequenzfluss weitergeleitet. Die Entscheidung wird anhand der Reihenfolge der ausgehenden Sequenzflüsse im XML Format gefällt. Die erste Bedingung in der XML Datei die zutrifft wird ausgewählt.

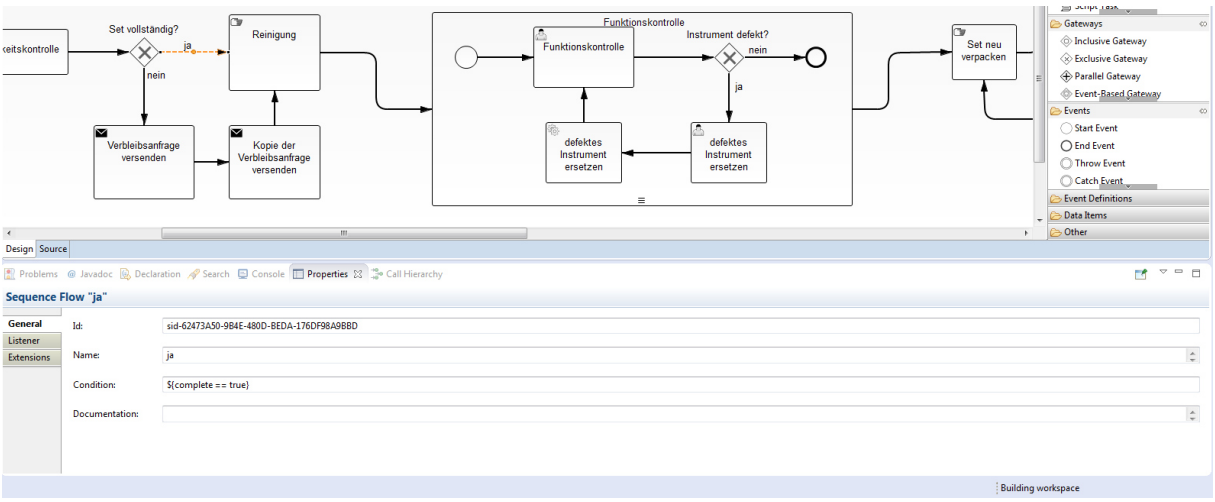


Abbildung 33: Benutzeroberfläche Properties Panel Sequenzfluss
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

Im Gateway kann über ein Drop Down Menu ein Default Flow festgelegt werden, welcher beschritten wird, falls keine Bedingung der übrigen Sequenzflüsse zutrifft. Trifft keine Bedingung zu und ist kein Default Sequenzfluss definiert, gibt es einen Fehler und die Instanz der Prozessdefinition wird beendet.

Error Events definieren

Für die Umsetzung von Error Events in Camunda muss ein BPMN Error definiert und zugewiesen werden. Die Definition von Error Events kann im Modeler geschehen. Dazu wird im Properties Panel des Error Events die Registerkarte Event ausgewählt. Abbildung 34 zeigt die Registerkarte Event des Error Events. Unter dem Punkt *Errors:* kann man alle definierten BPMN Errors sehen. Über den Punkt *Add* im Kontextmenü können neue BPMN Errors definiert werden. Die Definition beinhaltet einen Error Code und den Namen des BPMN Errors. Beide können frei gewählt werden. Nach der Definition wird der BPMN Error dem Error Event zugewiesen. Das Dropdown Menü von *Error:* liefert eine Auswahl an verfügbaren BPMN Errors.

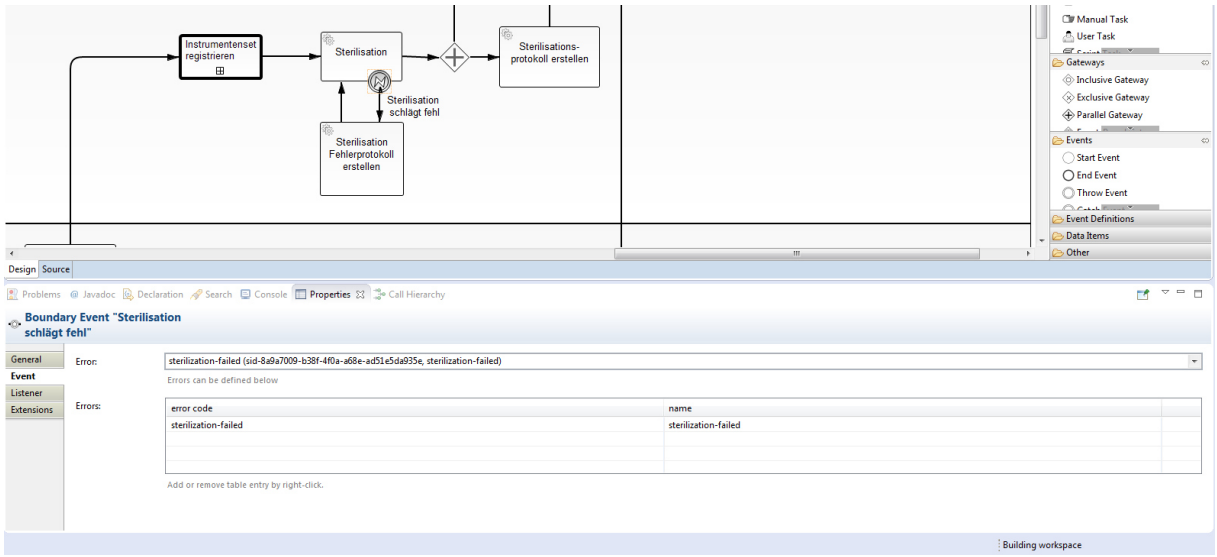


Abbildung 34: Benutzeroberfläche Properties Panel Error Event
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

Um das Error Event auszulösen muss im Service Task ein BPMN Error geworfen werden. Der BPMN Error muss denselben Error Code enthalten, wie das auszulösende Error Event. Listing 6 zeigt einen Ausschnitt des Java Delegates, der beim Ausführen des Tasks *Sterilisation* aufgerufen wird. Die Klasse wirft einen BPMN Error, der das entsprechende Error Event auslöst. Das Token wird dann auf den am Error Event angehängten Sequenzfluss geleitet.

```
public class SterilisationDelegate implements JavaDelegate {

	private final static Logger LOGGER = Logger.getLogger("Sterilisation");

	public void execute(DelegateExecution execution) throws Exception {
		LOGGER.info("Sterilisation.");

		Random random = new Random();
		int randomNumber = random.nextInt(2);

		if (randomNumber == 1) {
			LOGGER.info("Sterilisationerror thrown.");
			throw new BpmnError("sterilization-failed");
		}
	}

}
```

Listing 6: Ausschnitt aus SterilisationDelegate.java
Quelle: Eigene Darstellung

Nachrichtenaustausch definieren

Beim Nachrichtenaustausch müssen die Nachricht, das empfangende Ereignis und der Versand für die Process Engine genauer spezifiziert werden. In Abbildung 35 ist das Property Panel eines Message Events zu sehen. Dort können im Feld *Messages:* neue Messages definiert werden. Eine Message besteht nur aus einem Namen. Wählt man den Punkt *Add* im Kontextmenü kann ein neuer Name vergeben werden. Anschließend steht die Message im Dropdown Menü des Punkts *Message:* zur Auswahl. Dort wird die Nachricht, auf die das Message Event reagieren soll, ausgewählt.

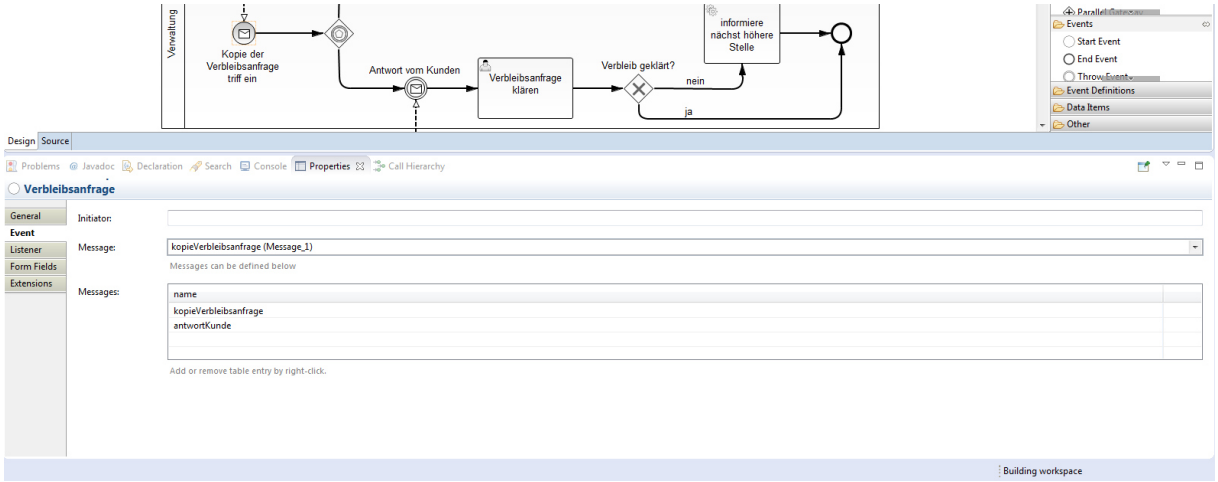


Abbildung 35: Benutzeroberfläche Properties Panel Message Event
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

Um eine entsprechende Nachricht zu verschicken wird ein neuer MessageCorrelationBuilder mit dem entsprechenden Namen der Nachricht erzeugt. Der Nachricht können auch Parameter mitgegeben werden. Diese werden als HashMap an die Nachricht angehängt.

Um die Nachricht zu versenden wird die Methode *correlate()* aufgerufen. Bei dem Beispiel in Listing 7 wird die Nachricht *kopieVerbleibsanfrage* erzeugt und ihr die fehlenden Instrumente mit übergeben.

```
public class KopieVerbleibsanfrageDelegate implements JavaDelegate {

	private final static Logger LOGGER = Logger.getLogger("Sterilisation");

	@Override
	public void execute(DelegateExecution execution) throws Exception {

		// Get variable instrumentensetMissingInstruments
		Instrumentenset instrumentensetMissingInstruments = (Instrumentenset)
		execution.getProcessEngineServices().getRuntimeService().
		(execution.getId(), "instrumentensetMissingInstruments");

		// Put variable instrumentensetMissingInstruments into variables
		Map<String, Object> variables = new HashMap<String, Object>();
		variables.put("instrumentensetMissingInstruments",
		instrumentensetMissingInstruments);

		/**
		 * Send message to administration. They'll handle the contact
		with the customer.
		 */

		MessageCorrelationBuilder messageCorrelation =
		execution.getProcessEngineServices().getRuntimeService().
		createMessageCorrelation("kopieVerbleibsanfrage");
		messageCorrelation.setVariables(variables).correlate();
		LOGGER.info("Kopie der Verbleibsanfrage an die Verwaltung
		verschickt.");
	}
}
```

Listing 7: Ausschnitt aus KopieVerbleibsanfrageDelegate.java
Quelle: Eigene Darstellung

Timer Events definieren

Timer Events werden mit Zeit- bzw. Datumsangaben, welche im ISO 8601 Format angegeben werden, definiert Die Angaben können in der Registerkarte *Event* eingetragen werden. Der Camunda Modeler unterstützt dabei drei verschiedene Arten von Zeit- bzw. Datumsangaben für Timer Events. Abbildung 36 zeigt das dazu entsprechende Properties Panel. Timer Events können an einem bestimmten Zeitpunkt ausgelöst werden. Dazu wird im Feld *Date:* ein festgelegtes Datum mit Uhrzeit eingetragen. Soll das Timer Event wie ein Countdown fungieren, wird eine Zeitperiode im Feld *Duration:* eingetragen. Ein so definiertes Timer Event wird nur genau einmal ausgeführt. Soll ein Timer Event wiederholt ausgeführt werden muss eine Angabe im Feld *Cycle* erfolgen.

Dabei kann auch die Anzahl der Wiederholungen angegeben werden. Das in Abbildung 35 gezeigte Beispiel erzeugt einen Timer, der nach sieben Sekunden ausgeführt wird[10].

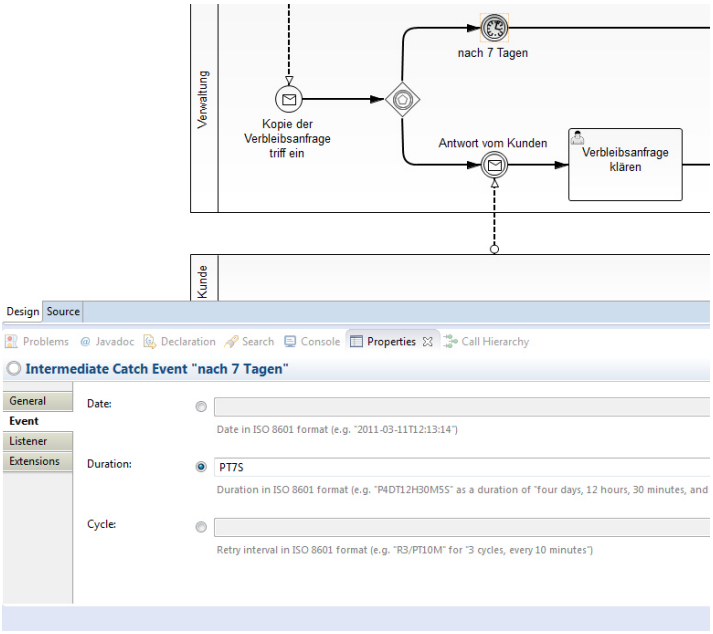


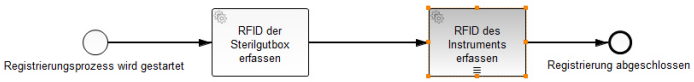
Abbildung 36: Benutzeroberfläche Properties Panel Timer Event
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

Multiple Instances definieren

Die Informationen für Multiple Instance Tasks werden in der Registerkarte *Multi Instance* des Property Panels vorgenommen. Abbildung 37 zeigt den entsprechenden Ausschnitt des Property Panels. Es muss die Checkbox *Is Multi Instance:* aktiviert werden. Über die Checkbox *Is Sequential:* kann angegeben werden, ob die Ausführung sequentiell oder parallel erfolgen soll. Der Process Engine muss mitgeteilt werden wie oft der Task ausgeführt werden soll. Die Angabe kann entweder als Zahl oder als Expression erfolgen. Dazu wird die entsprechende Zahl oder Expression in das Feld *Loop Cardinality:* eingetragen. Alternativ kann auch über eine Liste, welche als Prozessvariable definiert ist, iteriert werden. Im Feld *Collection* wird die entsprechende Variable mit der JUEL Syntax eingetragen. Dabei wird für jedes Element in der Liste eine neue Instanz erzeugt. Möchte man auf das entsprechende Element innerhalb der Instanz zugreifen, kann man im Feld *Element Variable:* einen Namen für eine Prozessvariable vergeben. Diese Prozessvariable existiert nur innerhalb der Instanzen des Multi Instance Tasks und kann wie eine gewöhnliche Prozessvariable abgefragt werden.

[10] Aus Zeitgründen ist der Timer mit sieben Sekunden anstatt, wie im Diagramm, mit sieben Tagen Wartezeit definiert worden.

In diesem Beispiel wird die Liste der Instrumente aus der Prozessvariable *instrumentenset* abgefragt und über diese iteriert. Das jeweilige Element der Liste steht in den Instanzen als Prozessvariable *instrument* zur Verfügung.



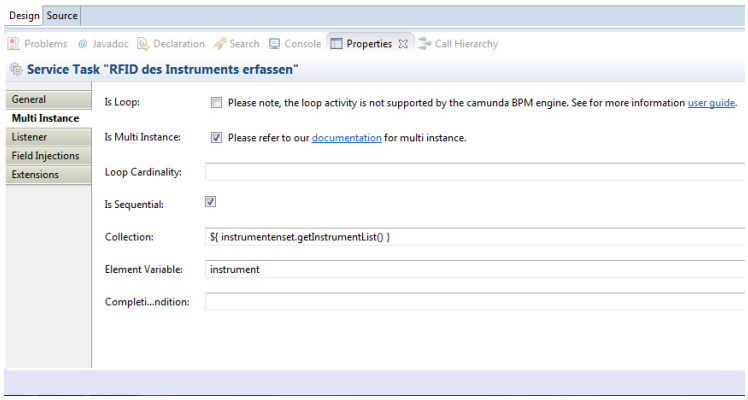


Abbildung 37: Benutzeroberfläche Properties Panel Multi Instance
Quelle: Screenshot Camunda Modeler für Eclipse Version 2.60

4.2.5 Diagramme nach der Umsetzung

Bei der Umsetzung der einzelnen Diagramme mussten einige Änderungen vorgenommen werden um diese erfolgreich auf der Camunda Process Engine auszuführen. Diese Änderungen wurden bedingt durch Einschränkungen seitens der Process Engine gegenüber dem BPMN 2.0 Standard. Nachfolgend werden die umgesetzten Diagramme im Einzelnen vorgestellt und mit den ursprünglichen Diagrammen verglichen. Die notwendigen Änderungen werden dargestellt und jeweils begründet.

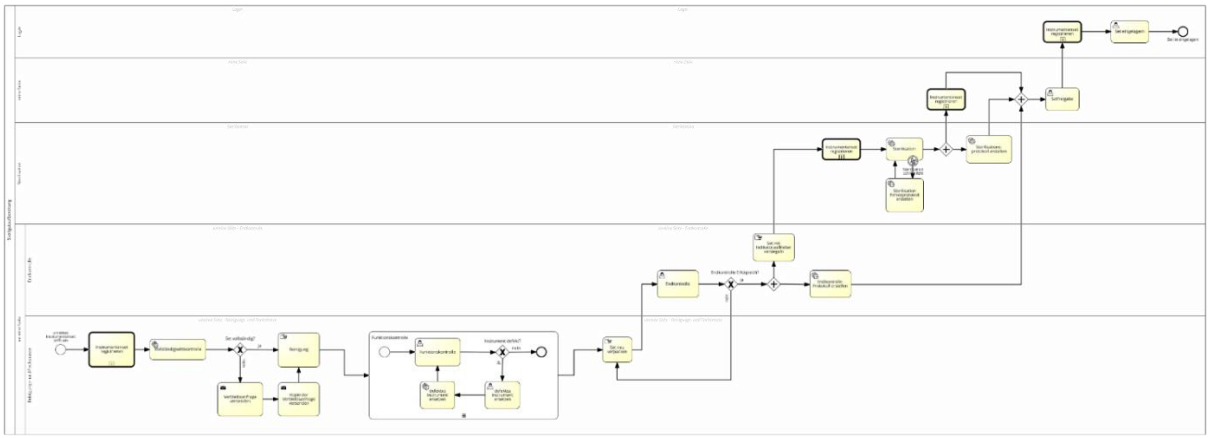


Abbildung 38: Prozessdiagramm Sterilisation nach der Umsetzung
Quelle: Eigene Darstellung siehe Anhang A.3

In Abbildung 38 wird der ausführbare Sterilisationsprozess dargestellt[11]. Im Vergleich zur Abbildung 17, welche den modellierten Prozess vor der Umsetzung darstellt, werden einige Anpassungen deutlich, auf die im Folgenden genauer eingegangen wird.

Die Call Activity *Funktionskontrolle* ist durch einen eingebetteten Subprocess ersetzt worden. Diese Änderung ist notwendig, da Änderungen von komplexen Prozessvariablen innerhalb einer Call Activity nicht an übergeordnete Prozessinstanzen weitergegeben werden. Die Änderung ist damit nur innerhalb der Call Activity sichtbar. Bei eingebetteten Subprozessen befinden sich alle Prozessvariablen im selben Geltungsbereich des übergeordneten Prozesses. Die Änderungen an Prozessvariablen innerhalb des Subprozesses sind damit auch außerhalb des Subprozesses wirksam.

Im Folgenden wird der Subprocess *Funktionskontrolle* genauer untersucht.

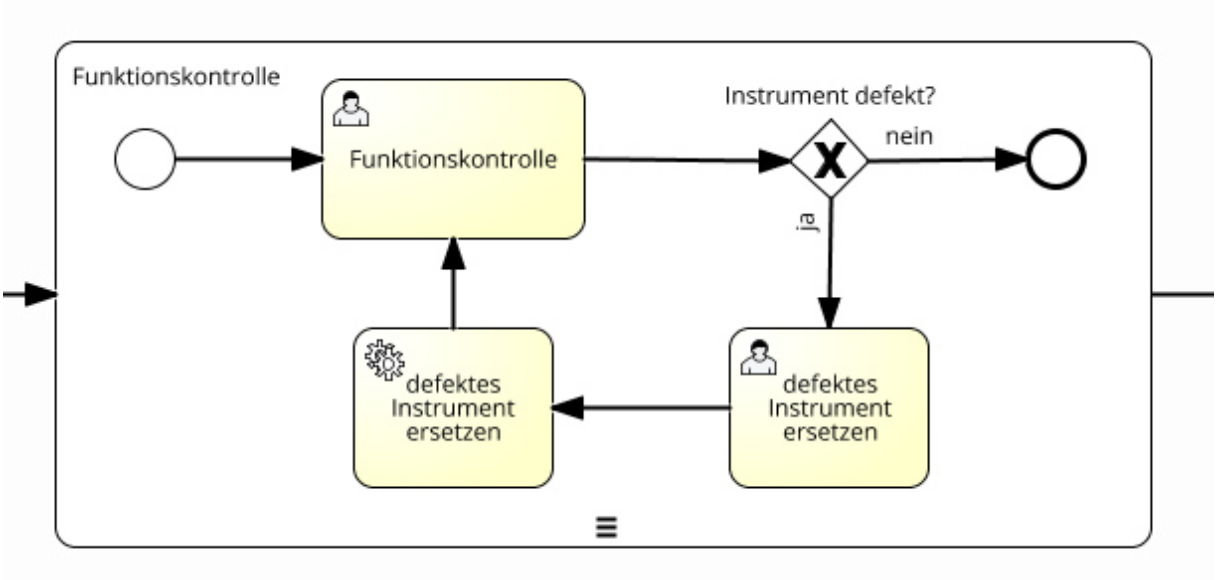


Abbildung 39: Prozessdiagramm Subprozess Funktionskontrolle nach der Umsetzung
Quelle: Eigene Darstellung

Im Vergleich zu Abbildung 19 werden die Unterschiede deutlich. Die Aufgaben-Typen von *Funktionskontrolle* und *defektes Instrument ersetzen* sind jetzt Benutzeraufgaben. Diese Änderungen sind notwendig, da die Camunda BPMN 2.0 Process Engine manuelle Tasks nicht bei der Ausführung berücksichtigt. Wenn ein Task von der Process Engine nicht berücksichtigt wird, kann auch keine Prozessvariable gesetzt werden. Prozessvariablen dienen unter anderem als Entscheidungsgrundlage für Gateways. Dadurch entsteht am exklusiven Gateway *Instrument defekt?* ein Entscheidungsproblem. Die Process Engine kann nicht entscheiden auf welchen Sequenzfluss das Token weitergeleitet wird.

[11]Für eine größere Darstellung der Abbildung siehe Anhang A.3

Der Servicetask *defektes Instrument ersetzen* ist notwendig um den Zugriff auf die Datenbank zu ermöglichen. Die *InstrumentensetId* des defekten Instruments wird in der Datenbank auf den Wert -1[12] gesetzt und das neue Instrument wird dem aktuell vorliegenden Instrumentenset zugeordnet. Es wird auch das aktuelle Instrumentenset, welches in der Prozessvariable *instrumentenset* vorliegt, aktualisiert.

Nach der Betrachtung des Subprocesses ist die nächste Stelle an der eine Änderung vorgenommen werden musste der User Task *Endkontrolle*. Ursprünglich war an diesen Task ein unterbrechendes Fehlerereignis angehängt. Dieses Fehlerereignis musste durch ein exklusives Gateway ersetzt werden, da es nicht möglich ist, dass ein Benutzer ein Fehlerereignis auslösen kann. Am Gateway *Endkontrolle Erfolgreich* wird jetzt die Prozessvariable *endkontrolleSuccessful* ausgewertet. Falls diese vom Mitarbeiter der *Endkontrolle* mit false deklariert wurde, geht das Instrumentenset wieder zurück und muss neu gepackt werden. Der Rest des Diagrammes konnte ohne Veränderungen umgesetzt werden.

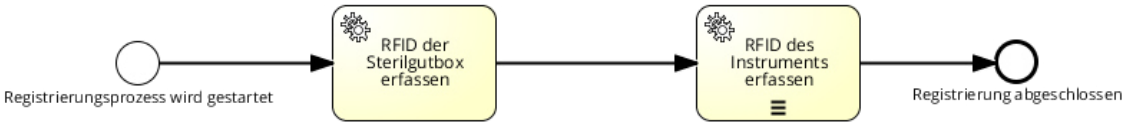


Abbildung 40: Prozessdiagramm Instrumentenset registrieren nach der Umsetzung
Quelle: Eigene Darstellung

In Abbildung 40 wird die Call Activity *Instrumentenset registrieren* nach der Umsetzung dargestellt. Im Vergleich zu Abbildung 18 waren keine Änderungen für die Umsetzung notwendig. Auch der Prozess *Verbleibsanfrage*, welcher in Abbildung 41 *zu* sehen ist, konnte ohne Änderung umgesetzt werden.

[12] Der Wert -1 gibt an, dass das Instrument als defekt gekennzeichnet wurde

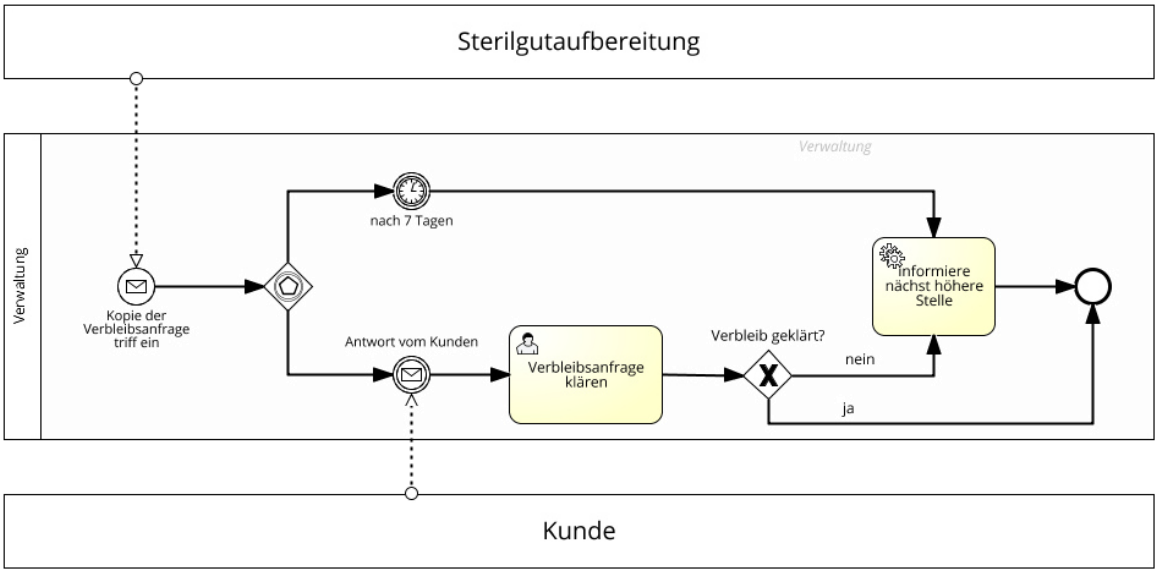


Abbildung 41: Prozessdiagramm Verbleibsanfrage nach der Umsetzung
Quelle: Eigene Darstellung

4.3 Evaluation

Auf Basis der Umsetzung der modellierten Diagramme erfolgt die Evaluation. Zuerst wird das Evaluationskonzept vorgestellt. Das Evaluationskonzept entspricht der Methodik der Evaluation und gibt deren Rahmenbedingungen vor. Im Anschluss daran werden die Ergebnisse der Evaluation anhand der Kriterien präsentiert und mit Beispielen aus der Umsetzung belegt.

4.3.1 Das Evaluationskonzept

Die Evaluation ist eine Machbarkeitsanalyse und soll klären ob die BPMN 2.0 sich als Modellierungssprache für Geschäftsprozesse im klinischen Umfeld eignet. Des Weiteren geklärt werden inwiefern Signavio und die Camunda BPM-Plattform geeignete Werkzeuge zur Prozessautomatisierung von BPMN 2.0 Prozessen sind. Der Schwerpunkt der Evaluation liegt dabei auf der Camunda BPM-Plattform. Es werden die Prozessdiagramme der Modellierung und des ausführbaren Prozesses betrachtet. Die Evaluation erfolgt empirisch, d.h. anhand der Überführung des Beispielszenarios in einen automatisierten Prozess mit Hilfe von Signavio und der Camunda BPM-Plattform. Die Bewertung richtet sich nach der Unterstützung von typischen Elementen der Beispielprozesse seitens der Camunda BPM.

Als typische Elemente in diesem Beispiel sind zu nennen:

- Manuelle Tasks / User Tasks – Interaktion mit dem Prozess
- Error Events - Fehlerhandling
- Servicetasks - Anbindung von Maschinen / Programmen
- Messages - Kommunikation zwischen Beteiligten (Nachrichtenaustausch)
- Call Activity / Subprocess - wiederverwendbare Teilprozesse
- Multiple Instances - Mehrfachausführungen
- Gateways - Verzweigungen (parallel / exklusiv / ereignis-basiert)
- Timer Events - zeitabhängige Ereignisse

Es wird zwischen drei möglichen Ausprägungen unterschieden:

- unterstützt
- unterstützt (mit Einschränkungen)
- nicht unterstützt

Die Ausprägung *unterstützt* steht für eine, im Rahmen des Beispielszenarios, vollständige Unterstützung seitens der Camunda BPM-Plattform. *Unterstützt (mit Einschränkungen)* bezeichnet eine eingeschränkte Unterstützung, welche sich z.B. darin äußert, dass das typische Element zwar generell unterstützt wird, aber die Unterstützung im Rahmen des Beispielszenarios sich als nicht ausreichend erwiesen hat und das BPMN Modell angepasst werden muss um eine ausreichende Unterstützung zu erhalten. *Nicht unterstützt* findet Anwendung, wenn ein Element gar nicht von der Camunda BPM unterstützt wird. Für einen Erfolg muss mindestens die Ausprägung *unterstützt (mit Einschränkungen)* erreicht werden.

4.3.2 Unterstützung der typischen Elemente durch die Camunda BPM

Manueller Task

Manuelle Tasks werden von der Camunda Process Engine nicht unterstützt. Die Process Engine ignoriert diese bei der Ausführung. Diese Tatsache muss beachtet werden, wenn auf Grund von manuellen Tasks eine Entscheidung getroffen werden soll.

User Task

User Tasks werden generell von der Process Engine unterstützt. Es gilt allerdings ein paar Einschränkungen zu beachten. User Tasks dienen der Präsentation und der Interaktion mit Prozessvariablen. Es ist nicht möglich ein Error Event aus einem User Task heraus anzustoßen. Komplexe Prozessvariablen werden in Embedded Task Forms nicht angezeigt, wenn diese das Serializable Interface implementieren.

Das Verhalten der Process Engine wird dadurch nicht beeinflusst. Der Zugriff auf diese Prozessvariablen ist jederzeit möglich, jedoch lassen sich die Datenfelder nicht in Embedded Task Forms darstellen. Das Serializable Interface ist notwendig um komplexe Prozessvariablen zu verwenden. Wird das Interface nicht implementiert treten an verschiedenen Stellen Fehler auf, welche die Process Engine zum Absturz bringen.

Error Event

Error Events werden unterstützt. Hierbei gilt es jedoch auch wieder eine Einschränkung zu beachten. Laut BPMN 2.0 Standard ist es möglich Error Events an User Tasks anzuheften. In der Camunda Process Engine gibt es allerdings keine Möglichkeit dieses Error Event auszulösen. Error Events können nur auf zwei Arten ausgelöst werden. Entweder über ein Error End Event, welches explizit im BPMN Diagramm modelliert ist, oder über Java Delegation Code.

Service Task

Die Camunda Process Engine bietet eine uneingeschränkte Unterstützung von Service Tasks, im Rahmen des Beispielszenarios. Alle modellierten Service Tasks konnten ohne Probleme umgesetzt werden. Die Anbindung von Java Code war jederzeit und uneingeschränkt möglich.

Message

Der Nachrichtenaustausch zwischen zwei Beteiligten konnte ohne Probleme realisiert werden. Der Nachrichtenaustausch ist auch über Diagrammgrenzen hinweg möglich. Den Messages kann zusätzliche Information mitgegeben werden.

Call Activities

Call Activities werden nur eingeschränkt unterstützt. Bei der Verwendung von komplexen Prozessvariablen lassen diese sich an die Call Activity übergeben und verwenden. Änderungen, die innerhalb der Call Activity an komplexen Prozessvariablen stattfinden, können nicht zurück an den Hauptprozess übergeben werden.

Subprocesses

Die Camunda Process Engine unterstützt die Verwendung von Subprozessen uneingeschränkt. Komplexe Prozessvariablen können verwendet und verändert werden. Ein Mapping von Prozessvariablen ist nicht erforderlich, da derselbe Scope wie im Hauptprozess vorliegt.

Multiple Instances

Multiple Instance Tasks werden im Rahmen des Beispielszenarios vollständig unterstützt. Der Modeler erlaubt die Modellierung von sequenziellen und parallelen Multiple Instance Tasks.

Die Anzahl der auszuführenden Instanzen lässt sich numerisch oder durch Prozessvariablen festlegen. Bei einer Iteration über eine Collection ist ein Zugriff auf das aktuelle Element möglich.

Gateways

Gateways werden von der Camunda Process Engine uneingeschränkt unterstützt. Komplexe und Parallele Gateways benötigen keine weiteren Informationen um ausgeführt zu werden. Exklusive Gateways benötigen eine Modellierung der Bedingung. Falls ein Default Sequenzfluss vorhanden ist, muss dieser für alle drei Arten von Gateways definiert werden.

Timer Events

Timer Events werden von der Camunda Process Engine uneingeschränkt unterstützt. Die Camunda BPM unterstützt sowohl einmal auftretende als auch sich wiederholende Timer Events. Einmal auftretende Events, können zu einem bestimmten Zeitpunkt oder als eine Art Countdown definieret werden. Sich wiederholende Events lassen sich über ein Intervall definieren. Die Anzahl der Wiederholungen kann festgelegt werden.

Die Ergebnisse der Evaluation werden abschließend in Tabelle 1 in vereinfachter Form zusammenfassend dargestellt.

	unterstützt	unterstützt (mit Einschränkungen)	nicht unterstützt
manuelle Tasks			x
User Tasks		x	
Error Events		x	
Servicetasks	x		
Messages	x		
Call Activity		x	
Subprocess	x		
Multiple Instances	x		
Gateways	x		
Timer Events	x		

Tabelle 1: Ergebnisse der Evaluation
Quelle: Eigene Darstellung

4.3.3 Ergebnis der Evaluation

Die Camunda BPM eignet sich als Process Engine für BPMN 2.0 Modelle. Es wurde gezeigt, dass ein Großteil der BPMN 2.0 Elemente unterstützt wird. Einige Elemente werden allerdings nicht vollständig oder gar nicht unterstützt. Die vorhandenen Defizite konnten im Beispielszenario durch eine andere Modellierung der Prozesse ausgeglichen werden.

5 Diskussion

In diesem Kapitel werden die Ergebnisse bewertet. Es wird dabei die Aussagekraft der Ergebnisse beurteilt. Danach werden alternative Modellierungsmöglichkeiten vorgestellt und deren Auswirkungen auf die Umsetzung und anschließende Evaluation diskutiert. Zum Schluss werden in einem Ausblick die Möglichkeiten und Fragestellungen, welche sich aus den Ergebnissen ableiten lassen, dargestellt.

5.1 Beurteilung der Ergebnisse

Im Rahmen der Arbeit hat sich gezeigt, dass sich die BPMN 2.0 grundsätzlich als Modellierungssprache für Prozesse im klinischen Umfeld eignet. Die Prozesse müssen dazu standardisiert und strukturiert sein, damit sich eine Modellierung mit Hilfe der BPMN 2.0 realisieren lässt. Die BPMN 2.0 unterstützt eine Vielzahl an verschiedenen Modellierungselementen. Dadurch lassen sich auch komplexe Zusammenhänge in Prozessen modellieren. Diagramme lassen sich über Nachrichten und Call Activities miteinander verbinden. Durch diese Möglichkeit der Kapselung von Teilprozessen wird die Komplexität der Diagramme reduziert.

Der Signavio Editor eignet sich sehr gut für die Modellierung von BPMN 2.0 konformen Diagrammen. Es werden alle Elemente der BPMN 2.0 unterstützt. Der Editor verfügt über eine Syntaxvalidierung und gibt Hinweise für eine übersichtliche Modellierung. Die Exportfunktion erlaubt es die modellierten Prozesse in einem BPMN 2.0 konformen XML Dateiformat zu speichern. Die Diagramme lassen sich so in Programmen, die den BPMN 2.0 XML Standard unterstützten, weiterverarbeiten.

Die Camunda BPM ist generell als Process Engine für BPMN 2.0 Diagramme aus dem klinischen Umfeld geeignet. Bei der Umsetzung hat sich gezeigt, dass die Camunda BPM einen Großteil der BPMN 2.0 abdeckt. Dennoch bestehen zum jetzigen Zeitpunkt einige Defizite. Die Defizite konnten größtenteils durch eine andere Modellierung der Prozesse ausgeglichen werden. Das einzige Problem, welches nicht gelöst werden konnte, ist die Darstellung von komplexen Prozessvariablen in User Tasks. Der Inhalt dieser Prozessvariablen konnte nicht angezeigt werden, obwohl die Prozessvariablen nachweislich vorhanden und richtig initialisiert sind.

Im Zusammenspiel von BPMN 2.0, Signavio Process Editor und Camunda BPM hat sich gezeigt, dass diese Kombination ein guter Ansatz ist um reale Prozesse zu Erfassen und Automatisieren. Das schwächste Glied der Kette ist bisher noch die Camunda BPM, da die Modellierung der Prozesse geändert werden muss um eine Ausführung zu erreichen. Trotz Änderungen an den Modellen gibt es Probleme bei der Darstellung von komplexen Prozessvariablen, die sich auch nicht lösen ließen. Die Camunda BPM wird laufend weiterentwickelt und das Projekt macht rasante Fortschritte. Die Geschwindigkeit der Weiterentwicklung von Camunda lässt darauf hoffen, dass das in naher Zukunft die noch vorhandenen Probleme minimiert werden.

Die Aussagekraft dieser Evaluation ist allerdings beschränkt, da bei der Modellierung des Beispielszenarios nicht die gesamt BPMN 2.0 abgedeckt wurde. Viele Elemente, darunter Inklusive Gateways, Komplexe Gateways, Terminierung, Kompensationsereignisse, Abbruchereignisse, Geschäftsregeln und viele mehr wurden nicht verwendet. Die Evaluation bezieht sich nur auf die verwendeten Elemente der BPMN und deren Unterstützung bei der Ausführung seitens der Camunda BPM. Deshalb ist die Aussagekraft der Evaluation auf diese Elemente beschränkt. Die in diesem Beispielszenario verwendeten Elemente sind sicherlich die meist benutzten, aber es gibt sehr wohl Situationen in denen auch seltener verwendete Elemente umgesetzt werden müssen und sich die Modellierung nicht so ohne weiteres ändern lässt.

5.2 Alternative Modellierungsmöglichkeiten

Durch die große Vielfalt der BPMN 2.0 gibt es, bei der Modellierung von Geschäftsprozessen, verschiedene Möglichkeiten ein und denselben Sachverhalt darzustellen. Daraus ergeben sich verschiedene Auswirkungen auf die Prozessmodelle bei der Ausführung. Diese Auswirkungen sind nicht immer unmittelbar sichtbar.

In diesem Abschnitt werden zwei Stellen des Szenarios noch einmal gesondert dargestellt. Das Hauptaugenmerk liegt hier vor allem auf den unterschiedlichen Möglichkeiten der Modellierung. Während der Modellierung des Beispielszenarios gab es immer wieder Stellen an denen nicht eindeutig war wie diese am besten zu modellieren sind. Dieser Abschnitt soll verdeutlichen, dass unterschiedliche Modellierungen, die denselben Sachverhalt darstellen sollen völlig unterschiedliche Auswirkungen bei der Ausführung haben.

5.2.1 Endkontrolle

Die Rollenverteilung bei der Endkontrolle ist ein wichtiger Beitrag zur Qualitätssicherung des Prozesses. „Alle Arbeitsschritte werden von Mitarbeitern durchgeführt, wobei die Endkontrolle grundsätzlich von einem anderen Mitarbeiter durchgeführt werden muss als das Packen“ [20]. Für die Modellierung der Rollenverteilung gibt es verschiedene Möglichkeiten, von denen hier drei genauer beleuchtet werden.

1. Möglichkeit: Rollenverteilung über Assoziation

In der BPMN können mit Hilfe von Assoziationen einzelne Objekte verbunden werden, um eine Zugehörigkeit auszudrücken. Benutzt man diese Möglichkeit für die Situation der Endkontrolle sieht das Diagramm aus wie in Abbildung 42. Hier werden über Assoziationen einzelnen Tasks verschiedene Benutzerrollen zugeordnet. Diese Modellierung erschließt sich dem Betrachter und erscheint auf den ersten Blick als ausreichende Lösung.

Allerdings gibt es ein Problem bei der Umsetzung des Diagramms. Bei dieser Form der Modellierung kann die Process Engine die Einhaltung der Bedingung nicht sicherstellen, da Assoziationen keine Ausführungssemantik besitzen. Es gibt keinen Mechanismus, der bei einer Verletzung dieser Bedingung eingreift.

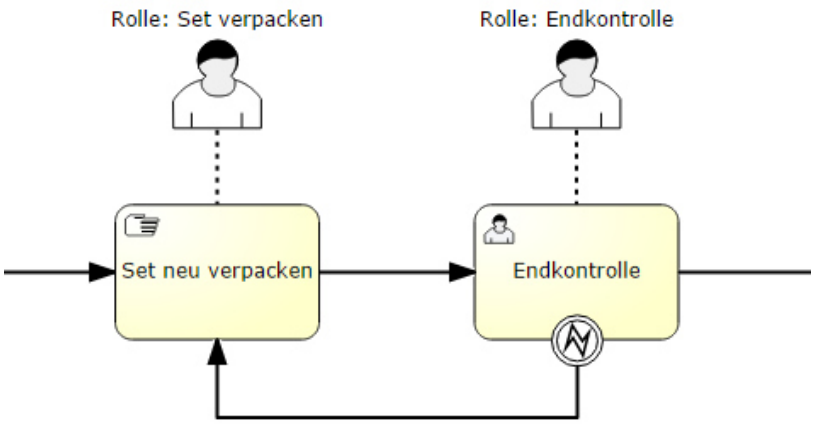


Abbildung 42: Prozessdiagramm Rollenverteilung über Assoziationen
Quelle: Eigene Darstellung

2. Möglichkeit: Rollentrennung über Lanes

In Abbildung 43 wird die Rollenverteilung über Lanes modelliert. Bei dieser Art der Modellierung ist die Rollenverteilung für den Betrachter immer noch deutlich zu erkennen. Die Process Engine kann jetzt jedoch überprüfen, sofern sie es unterstützt, ob die Rollenverteilung eingehalten wird. Die Camunda BPM unterstützt, in der aktuellen Version, keine Zuordnung von Lanes zu einzelnen Benutzerrollen. Die Festlegung der Rolle eines Tasks wird bei dem jeweiligen Task selbst definiert. Bei der Umsetzung erfolgt die Rollentrennung damit bei jedem Task einzeln. Deshalb bietet diese Form der Modellierung bei der Ausführung in der Camunda Process Engine gegenüber der Rollenverteilung über Assoziation keinen Vorteil.

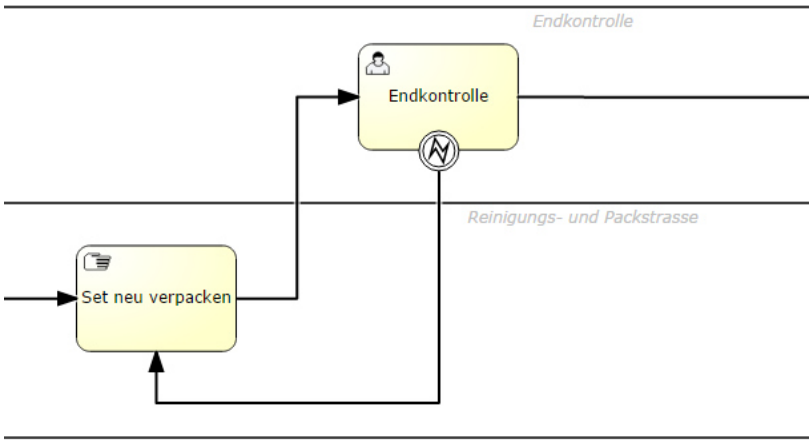


Abbildung 43: Prozessdiagramm Rollenverteilung über Lanes
Quelle: Eigene Darstellung

3. Möglichkeit: Rollenauswahl mit Hilfe einer Rules Engine

Bei der dritten Möglichkeit wird die Rollenauswahl mit Hilfe einer Rule Engine getroffen. Rule Engines werden verwendet um Geschäftsregeln von der Prozesslogik zu trennen. Geschäftsregeln können sich verändern während der Prozess an sich dennoch gleich bleibt (vgl. [22]). In Abbildung 44 ist die Rollenauswahl mit Hilfe einer Rule Engine abgebildet. Die Regeln zur Auswahl der richtigen Rolle werden in der Rule Engine hinterlegt. Diese kümmert sich um die Auswahl der richtigen Rolle für die Endkontrolle. So kann sichergestellt werden, dass die Endkontrolle nicht vom selben Mitarbeiter wie das Packen des Instrumentensets durchgeführt wird. Dieser Mechanismus greift auch dann, wenn das Instrumentenset von einem anderen Mitarbeiter als üblich gepackt wird, da die Auswahl der Rolle dynamisch zur Laufzeit erfolgt.

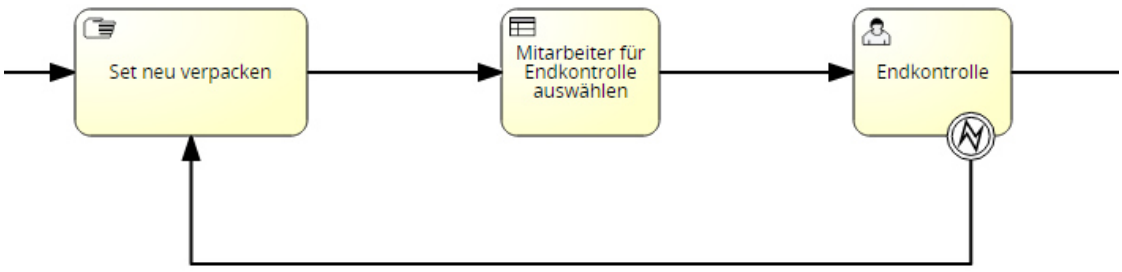


Abbildung 44: Prozessdiagramm Rollenverteilung mit Hilfe einer Rules Engine
Quelle: Eigene Darstellung

5.2.2 Haltbarkeit des Instrumentensets abgelaufen

„Falls die Haltbarkeit der Sterilisation eines gelagerten Sets abläuft, wird dieses entwertet und geht wie ein benutztes Set wieder in den Aufbereitungsprozess ein“ [20]. Auch hier gibt es verschiedene Möglichkeiten der Modellierung.

Die BPMN erlaubt es Tasks mit Hilfe von angehängten Events zu unterbrechen. In Abbildung 45 ist dargestellt, wie die Haltbarkeit eines Instrumentensets mit Hilfe eines attached Timer Events modelliert werden kann. Daraus ergibt sich dann, dass die Anforderung durch einen Kunden auch als attached Event modelliert werden muss, da das Token im Task *Set einlagern* stehen bleibt bis entweder die Haltbarkeit abgelaufen ist oder ein Kunde das Set anfordert. Diese Modellierung ist vollkommen syntaxkonform besitzt jedoch einen gravierenden Nachteil. Der Prozess kann nur beendet werden wenn ein Kunde das Set anfordert. Es wird also für jedes Instrumentenset, welches aufbereitet aber nicht wieder durch einen Kunden angefordert wurde, eine Instanz des Prozesses vorgehalten. Jede Instanz verbraucht Ressourcen. Bei wenigen Instanzen merkt man keine Einschränkung. Werden jedoch pro Tag mehrere tausend Instrumentensets aufbereitet fällt der Ressourcenverbrauch doch sehr deutlich aus.

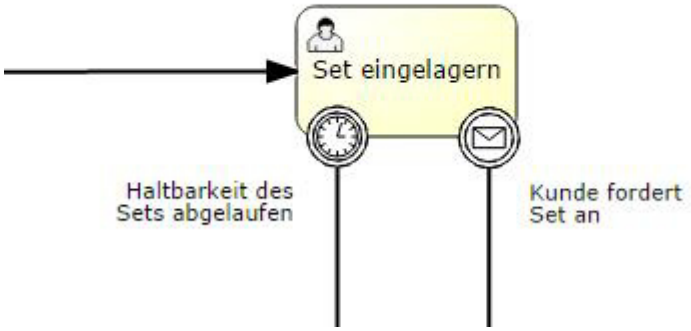


Abbildung 45: Prozessdiagramm Haltbarkeit als attached Timer Event
Quelle: Eigene Darstellung

Deutlich besser ist es die Prozessinstanz zu beenden und einen weiteren Prozess zu modellieren, welcher von einem Timer Ereignis gestartet wird. In Abbildung 46 ist diese Form der Modellierung zu sehen. Dadurch werden keine unnötigen Instanzen der Prozesse vorgehalten und der Ressourcenverbrauch gesenkt.

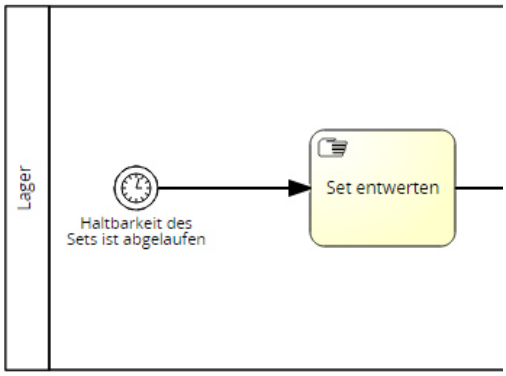


Abbildung 46: Prozessdiagramm Haltbarkeit über eine Timer Startereignis
Quelle: Eigene Darstellung

Es wird deutlich, dass die Art der Modellierung bei der Ausführung von Prozessen eine entscheidende Rolle spielt. Unterschiedliche Modellierungen wirken sich auf den Prozess bzw. die Process Engine aus.

5.3 Ausblick

Das Beispielszenario lässt sich noch weiter ausbauen, um eine größere Abdeckung der BPMN 2.0 Notation zu erhalten. Dadurch können auch Aussagen über hier nicht verwendete Elemente getroffen werden.

Es ist auch denkbar die Evaluation mit verschiedenen Szenarien durchzuführen und diese miteinander zu vergleichen. Daraus könnte sich ein standardisiertes Szenario ableiten lassen. Auf Basis eines standardisierten Szenarios können sowohl verschiedene Modellierungswerkzeuge als auch verschiedene Process Engines miteinander vergleichen werden.

Eine neue Herangehensweise von Prozessmodellierung stellt das Adaptive Case Management (ACM) dar. Im Mai 2014 wurde die Version 1.0 des CMMN (Case Management Model and Notation) Standards von der Object Management Group (OMG) verabschiedet. CMMN soll als Modellierungssprache für ACM fungieren. Gerade im medizinischen Bereich mit schwach strukturierten Prozessen erscheinen die Möglichkeiten von ACM besonders interessant (vgl. [23]). Man könnte evaluieren inwieweit sich ACM bzw. CMMN für die Modellierung von medizinischen Prozessen eignet.

Die Camunda BPM unterstützt neben der BPMN 2.0 auch die CMMN 1.0. Es kann evaluiert werden ob die Unterstützung von CMMN 1.0 seitens der Camunda BPM bereits ausreichend ist um komplexe Prozesse auszuführen.

Literaturverzeichnis

[1] P. Böcker, Qualitätsmanagement im Krankenhaus - Ein praxisorientierter Vergleich von Qualitätsmanagementsystemen und Bewertungsverfahren, Norderstedt: GRIN Verlag, 2005, p. 8.

[2] J. Becker, M. Kugeler und M. Rosemann, Prozessmanagement, Springer, 2006, p. 6.

[3] H. J. Schmelzer und W. Sesselmann, Geschäftsprozessmanagement in der Praxis, Hanser Verlag, 2010, p. 62.

[4] . T. Allweyer, Geschäftsprozessmanagement : Strategie, Entwurf, Implementierung, Controlling, W3l, 2005, p. 4.

[5] R. Lackes und M. Siepermann, „Gabler Wirtschaftslexikon," Springer Gabler Verlag, [Online]. Available: http://wirtschaftslexikon.gabler.de/Archiv/1057697/geschaeftsprozessmanagement-v4.html. [Zugriff am 15 Januar 2015].

[6] EABPM, European Association of Business Process Management, Business Process Management BPM Common Body of Knowledge - BPM CBOK, Gießen: Schmidt, 2009.

[7] S. Adam und N. Riegel, BPMN vs. EPK & Co. …oder auf was es wirklich ankommt, Kaiserslautern: Fraunhofer IESE , 2012.

[8] C. V. GEAMBAŞU, BPMN VS. UML Activity Diagram for Business Process Modeling, Bucharest, 2012.

[9] C. Pikalek, „Kampf der Notationen: BPMN vs. UML," 2011.

[10] K. Kruczynski, „Prozessmodellierung im Wettbewerb: EPK vs. BPMN," *IS Report,* pp. 30-35, 6 2008.

[11] T. van Lessen, J. Kress, B. Rücker und T. Winterberg, „BPMN ist tot, es lebe BPEL! | JAXenter.de," 28 Juni 2011. [Online]. Available: http://jaxenter.de/artikel/BPMN-ist-tot-lebe-BPEL. [Zugriff am 12 September 2014].

[12] Object Management Group, „BPMN Specification - Business Process Model and Notation," 2011.

[13] S. Adam, R. Norman, J. Thomas und K. Matthias, Studie – BPM Suites 2013, Kaiserslautern: Fraunhofer IESE, 2013.

[14] A. Gadatsch, IT-gestütztes Prozessmanagement im Gesundheitswesen, Wiesbaden: Springer Fachmedien, 2013, pp. 5 - 25.

[15] T. Allweyer, BPMN 2.0 - Business Process Model and Notation, 2., aktualisierte und erw. Aufl Hrsg., Norderstedt: Books on Demand GmbH, 2009, pp. 8ff, 168.

[16] R. Freund, Praxishandbuch BPMN 2.0, Hanser Fachbuchverlag, 2012, p. 23f.

[17] Signavio GmbH, „BPM Academic Initiative | Signavio," 2014. [Online]. Available: http://www.signavio.com/bpm-academic-initiative/. [Zugriff am 2 12 2014].

[18] Camunda, „User Guide | camunda BPM docs," 30 11 2014. [Online]. Available: http://docs.camunda.org/latest/guides/user-guide/. [Zugriff am 8 1 2015].

[19] Camunda, „BPMN Workflow Engine," 30 11 2014. [Online]. Available: http://camunda.org/. [Zugriff am 8 1 2015].

[20] C. Fegeler, *GPM-Modellierungsaufgabe Sommersemsester 2014,* Heilbronn, 2014.

[21] „H2 Database Engine," 2015. [Online]. Available: http://www.h2database.com/html/main.html. [Zugriff am 12 1 2015].

[22] A. Groenewold und M. Bartonitz, „Geschäftsregeln mit Business Rule Engines abbilden," *WissenHeute,* Nr. 5/2011, pp. 42 - 48, 2011.

[23] M. Kurz und C. Herrmann, Adaptive Case Management – Anwendung des Business Process Management 2.0-Konzepts auf wissensintensive schwach strukturierte Geschäftsprozesse, Bamberg, Erlangen-Nürnberg, Regensburg: Bayerischer Forschungsverbund forFLEX - Dienstorientierte IT-Systeme für hochflexible, 2011.

[24] dpunkt.Verlag, „Programmieren mit Java - Connection Pooling,“ 2002. [Online]. Available: http://www.dpunkt.de/java/Programmieren_mit_Java/Java_Database_Connectivity/48.html.

[25] Wikipedia, „WS-Business Process Execution Language - Wikipedia,“ [Online]. Available: http://de.wikipedia.org/wiki/WS-Business_Process_Execution_Language. [Zugriff am 25 Januar 2015].

Anhänge

A.1 Beispielszenario

GPM-Modellierungsaufgabe Sommersemsester 2014:

Sie haben im Rahmen eines Softwareentwicklungsprojektes den Auftrag erhalten für die Anforderungsanalyse den Geschäftsprozess eines Sterilgutdienstleisters mittels der BPMN Version 2.0 zu visualisieren. In der ersten Erhebungsphase wurde der Fokus auf den Ablaufprozess der Sterilgutaufbereitung gelegt, die Abläufe bei den Kunden und in der Verwaltung sollen in einer zweiten Runde detailliert werden. Deren Integration soll später ohne wesentliche Änderungen des jetzt gewählten Grundschemas möglich sein.

Ihnen stehen folgende Informationen zur Verfügung:

Beteiligte: Kunden, Sterilgutaufbereitung, Zentrale Verwaltung Ablauf:

Die Sterilgutaufbereitung ist eine Servicegesellschaft eines Klinikverbundes und beliefert sowohl Kunden im Verbund als auch außerhalb des Verbundes. Grundsätzlich werden alle Kunden gleich behandelt. Der Prozess der Sterilgutaufbereitung teilt sich in vier große Arbeitsbereiche auf. Auf der unreinen Seite gehen die benutzten Instrumenten-Sets ein und werden registriert. Es wird eine Vollständigkeitskontrolle durchgeführt, dann folgt die Reinigung. Danach wird jedes Instrument des Sets einer Funktionskontrolle unterzogen, wenn ein Defekt festgestellt wird, wird das jeweilige Instrument ersetzt. Im nächsten Schritt wird das Set neu gepackt und es folgt die Endkontrolle. Bei einem Fehler wird das gesamte Set neu gepackt. Alle Arbeitsschritte werden von Mitarbeitern durchgeführt, wobei die Endkontrolle grundsätzlich von einem anderen Mitarbeiter durchgeführt werden muss als das Packen. Alle Instrumente, die zur Mehrfachnutzung aufbereitet werden können besitzen einen RFID über den auch die Registrierung abläuft. Einmalmaterialien werden immer ersetzt. Gepackt wird ein Set in eine Sterilgutbox, auch diese hat einen RFID. Die Endkontrolle erfolgt IT-gestützt mittels Checklisten. Diese werden mit der jeweiligen Aufbereitungs-ID als Dokumentation abgelegt. Nach erfolgreicher Endkontrolle wird das Set mit einem Indikatoraufkleber versiegelt. Dieser zeigt über Indikatorfelder an, ob eine Sterilisation stattgefunden hat. Auf dem Siegel ist aufgedruckt der Set-Typ und die Aufbereitungs-ID, zusätzlich gibt es einen QR-Code.

Nun wechselt das Set in den Arbeitsbereich der eigentlichen Sterilisation. Dies ist ein vollautomatisierter Prozess. Nach erfolgter Registrierung des RFID der Sterilgutbox erfolgt diese. Dabei werden der Verlauf von Dampfdruck und Temperatur kontinuierlich mitgeloggt und unter einer Durchlauf-ID in einer Datenback des Automaten archiviert.

Weiter hin wird ein Protokoll des Sterilisationslaufes ausgedruckt. Anschließend wird das Set auf der reinen Seite erneut registriert. Für die Setfreigabe sind das Vorliegen der Protokolle der Endkontrolle und des Sterilisationslaufes zwingend erforderlich. Freigegebene Sets werden im Lager registriert und stehen zur Anforderung durch den Kunden zur Verfügung. Falls die Haltbarkeit der Sterilisation eines gelagerten Sets abläuft, wird dieses entwertet und geht wie ein benutztes Set wieder in den Aufbereitungsprozess ein.

Bei Sterilgutbedarf füllt ein Kunde eine Setanforderung aus und schickt dieses ans Lager. Dort wird die Lieferung zusammengestellt und an den Kunden versandt. Nach Benutzung durch den Kunden wird das Set zurück an die Sterilgutaufbereitung geschickt.

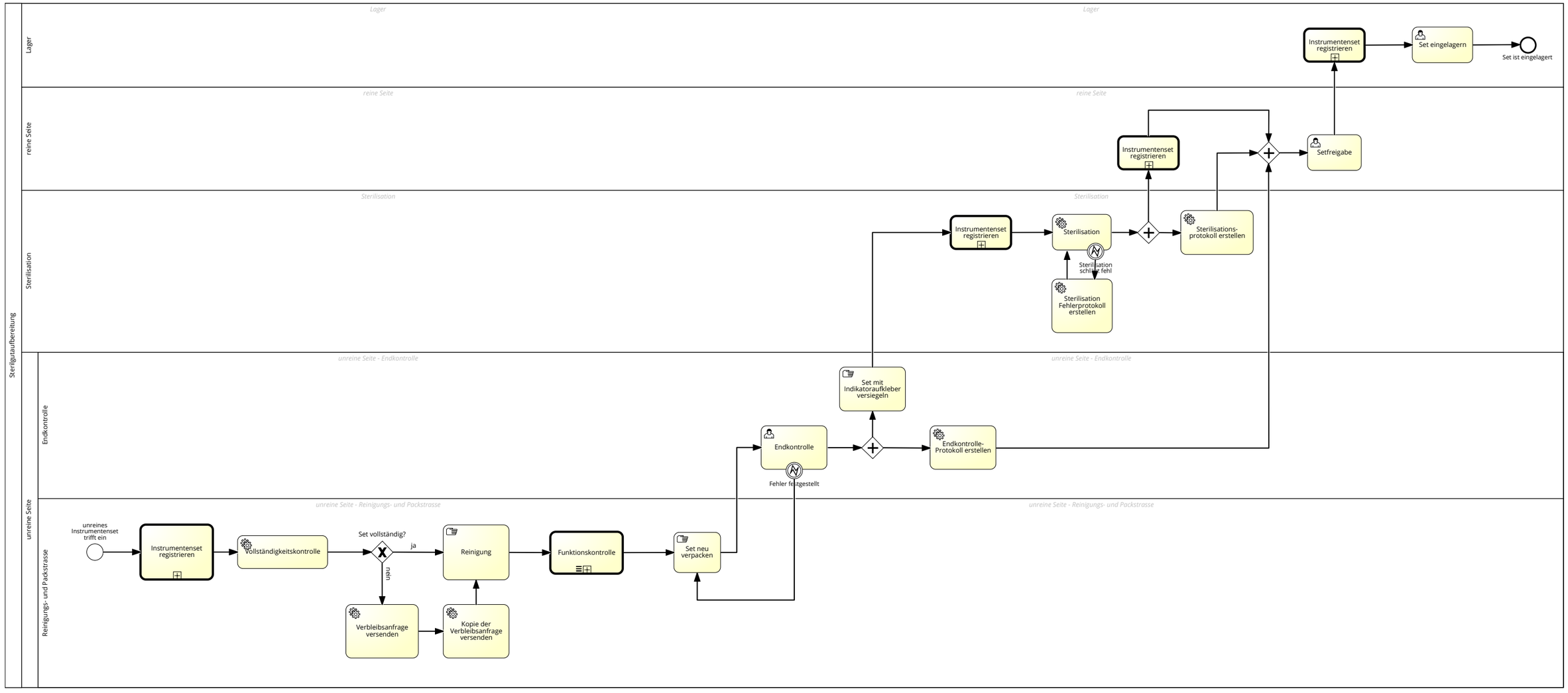
Die Abrechnung erfolgt über die Zentral-Verwaltung des Klinikverbundes. Diese erhält dazu Kopien der Lieferscheine und erstellt daraus eine Monatsabrechnung für den jeweiligen Kunden.

Weitere Aufgabe der Verwaltung ist die Fristenkontrolle von Verbleibsanfragen. Diese entstehen, wenn bei der Vollständigkeitskontrolle auf der unreinen Seite, das Fehlen von Instrumenten festgestellt wird. Die Verbleibsanfrage gehen an den Kunden und in Kopie an die Verwaltung. Sollte nicht innerhalb von 7 Tagen eine Antwort vorliegen, dann wird die nächst höherer Ebene informiert. Dies erfolgt auch, wenn der Verbleib trotz einer Antwort nicht eindeutig geklärt werden konnte.

Aufgaben:

1. Erstellen Sie ein Basismodell aus der Beschreibung
2. Zielsetzung ist es auf eine komplette digitale Dokumentation in der Sterilgutaufbereitung umzustellen. Muss der Prozess angepaßt werden? Erweitern Sie Ihr Modell entsprechend.

A.2 Prozessdiagramm Sterilisation



A.3 Prozessdiagramm Sterilisation (nach der Umsetzung)