School of Management and Law

Business Process as a Service: Konzeption und Realisierung

Master Thesis

Zürcher Hochschule für Angewandte Wissenschaften

Autor:
Micha Sebastian Boller
micha.boller@gmail.com

Erstgutachter:
Prof. Dr. Thomas Keller

Zweitgutachter:
Björn Scheppler

Wahrheitserklärung

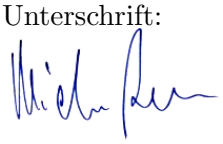
Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig, ohne Mithilfe Dritter und nur unter Benützung der angegebenen Quellen verfasst habe und dass ich ohne schriftliche Zustimmung der Studiengangleitung keine Kopien dieser Arbeit an Dritte aushändigen werde.

Gleichzeitig werden sämtliche Rechte am Werk an die Zürcher Hochschule für Angewandte Wissenschaften (ZHAW) abgetreten. Das Recht auf Nennung der Urheberschaft bleibt davon unberührt.

Vorname, Name Student:
Micha Sebastian Boller

Ort, Datum:
Zürich, 31. 05. 2017

Unterschrift:



Management Summary

Nebst strukturierten Business Prozessen existieren in Unternehmen auch unstrukturierte Business Prozesse. Solche unstrukturierten Business Prozesse sind schwierig zu modellieren, weil viele Eventualitäten auftreten können. Der Verlauf wird durch einen Knowledge Worker bestimmt und ist somit nur teilweise vorhersehbar. Ein Beispiel eines unstrukturierten Business Prozesses ist die Behandlung eines Patienten durch einen Arzt.

Zur Unterstützung unstrukturierter Business Prozesse können keine Process Engines genutzt werden, weil dazu ein Prozess Modell vorliegen muss. Das ist einer der Gründe, dass sich Unternehmen zur Unterstützung von unstrukturierten Business Prozessen zum Beispiel E-Mail Programmen und Dokumenten bedienen. Die Nachvollziehbarkeit wie auch die Übersicht über die laufenden Business Prozesse und die Automatisierbarkeit sind so eingeschränkt.

Diese Master Thesis erarbeitet ein Konzept und ein Prototyp einer Business Process as a Service (BPaaS) Plattform. Diese Plattform wird in der Cloud betrieben und unterstützt Unternehmen bei der Abwicklung von unstrukturierten Business Prozessen. Zentrales Artefakt ist der Ad Hoc Prozess. Der Ad Hoc Prozess ist ein Ansatz, wie unstrukturierte Business Prozesse ohne vorangehende Prozess Modellierung unterstützt werden können. Dabei bedient sich dieser Ansatz so genannten Activity Patterns, welche wiederkehrende Muster in Business Prozessen darstellen. Beispiele von Activity Patterns sind die Ausführung einer Aufgabe oder die Herbeiführung einer Entscheidung. Zudem wird eine Implementierung des vorgeschlagenen Ansatzes in Form eines lauffähigen Prototyps vorgelegt. Dieser Prototyp basiert auf der Camunda BPM Plattform. Nebst der Abwicklung von unstrukturierten Business Prozessen unterstützt der Prototyp auch das automatisierte On-Boarding (Erstellen eines Zugangs) eines Unternehmens auf die BPaaS Plattform.

Die Resultate dieser Master Thesis zeigen auf, dass unstrukturierte Business Prozesse mit dem erarbeiteten Ad Hoc Prozess abgewickelt werden können und damit Unternehmen eine Hilfestellung bei der Ausführung von unstrukturierten Business Prozessen geboten werden kann. Ob sich der erwartete Nutzen durch die Benutzung der BPaaS Plattform für ein Unternehmen einstellt, kann in weiteren Forschungsarbeiten ermittelt werden. Als Basis dazu dienen die erarbeiteten Resultate.

Danksagungen

Ich möchte mich bei meinem Betreuer und Erstgutachter Prof. Dr. Thomas Keller für die wertvollen Diskussionen sowie die immer vorhandene Hilfsbereitschaft bei Fragen und Unklarheiten bedanken. Zudem möchte ich mich bei meinem Zweitgutachter, Björn Scheppler, für die Betreuung dieser Arbeit bedanken.

Weiter bedanke ich mich bei der Camunda Community, welche einige Fragen beantwortet hat, die während der Erarbeitung dieser Master Thesis aufgetaucht sind.

Nicht zuletzt möchte ich mich auch bei meiner Frau, Eliane Boller, für die Unterstützung und das Verständnis für die durchgearbeiteten Abende und Wochenenden sowie bei Selina Graham und Ursula Boller für das Korrekturlesen bedanken.

Der Autor, Zürich, 31. 05. 2017

Inhaltsverzeichnis

Kapitel 1

Einleitung

Nebst strukturierten Business Prozessen, welche immer wieder gleich ablaufen, existieren in Unternehmen unstrukturierte Business Prozesse. Nach [1] haben gemäss einer Umfrage der Association for Information and Image Management (AIIM) 51% der befragten Unternehmen ausgesagt, dass mehr als die Hälfte ihrer Business Prozesse unstrukturiert verlaufen [2]. Die unstrukturierten Business Prozesse sind wissensintensiv, dynamisch und laufen situationsabhängig auf unterschiedliche Art und Weise ab [3]. Die als nächstes zu tätigenden Aufgaben sind abhängig von den Resultaten und Erkenntnissen, welche entlang des unstrukturierten Business Prozesses durch ein Knowledge Worker erarbeitet werden [4].

Viele dieser unstrukturierten Business Prozesse werden nach [5] lediglich mit Hilfe von E-Mail Programmen und Dokumenten abgewickelt. Dabei handelt sich ein Unternehmen Nachteile ein, wie zum Beispiel fehlende Transparenz über die laufenden Business Prozesse, fehlende Kontrollen und fehlende Rechenschaftspflicht.

Die in dieser Master Thesis zu konzipierende und realisierende Business Process as a Service (im Folgenden: BPaaS) Plattform soll solche unstrukturierten Business Prozesse unterstützen und Transparenz über die ablaufenden unstrukturierten Business Prozesse schaffen. Das Zielpublikum sind Unternehmen aus verschiedenen Branchen.

BPaaS baut auf dem Cloud Stack auf und ist nach [6] die nächst höhere Abstraktionsschicht, welche auf dem Software as a Service (SaaS) Layer aufsetzt. BPaaS bedeutet nicht nur, dass dem Benutzer Software als ein Service angeboten wird, sondern auch, dass dem Benutzer Business Prozesse und Kontrollfluss Logik als Service zur Verfügung

gestellt werden. Durch das Anbieten als Service muss sich ein Unternehmen nicht mehr selbst um den Betrieb und die Wartung eines Business Process Management Systems (im Folgenden: BPMS) kümmern.

1.1 Problemstellung

Unstrukturierte Business Prozesse sind eine Sammlung von Business Aktivitäten, deren Ausprägung und Reihenfolge von Fall zu Fall unterschiedlich ist [7]. Nach [7] sind das vor allem wissensintensive Tasks, wie zum Beispiel Incident Management oder Consulting Dienstleistungen.

In [8] wird erwähnt, dass viele Business Prozesse wegen ihrer Ad Hoc Charakteristik schwierig zu modellieren sind. Deshalb kommen für unstrukturierte Business Prozesse traditionelle Process Engines nicht in Frage, weil hier ein Prozess Modell Voraussetzung ist, damit ein Business Prozess abgewickelt werden kann [4].

Die zu konzipierende und realisierende BPaaS Plattform soll unstrukturierte Business Prozesse unterstützen, indem ein Ansatz gewählt wird, bei welchem kein Business Prozess Modell vorab erstellt werden muss. Durch das Anbieten von so genannten Activity Patterns kann ein Business Prozess ohne vorgängige Modellierung abgewickelt werden. Activity Patterns sind Muster von Business Funktionen, welche häufig in Business Prozessen gefunden werden [9]. Die zu unterstützenden Activity Patterns sind bei der Analyse von Business Prozessen durch Keller et al. [10] als wiederkehrende Muster aufgefallen. Die BPaaS Plattform soll dabei von mehreren Unternehmen (oder Mandanten) gleichzeitig genutzt werden können.

Der Vorteil vom Activity Pattern Ansatz (oder Allgemein von der BPaaS Plattform) ist, dass grundsätzlich beliebige Business Prozesse unterstützt werden können, ohne dass der Business Prozess im Voraus modelliert werden muss (so lange es Kombinationen von Activity Patterns sind). Es kann jederzeit ein beliebiger Business Prozess durch den Knowledge Worker gestartet werden. Dies führt dazu, dass im Idealfall für die Unterstützung von unstrukturierten Business Prozessen weder Prozess Analysten (mit hohem Modellierungswissen) noch IT Fachleute (mit Programmierkenntnissen) notwendig sind.

Die BPaaS Plattform ist für Unternehmen von Nutzen, welche keine Toolunterstützung für die Abwicklung unstrukturierter Business Prozesse besitzen. Business Prozesse können ohne grossen Vorbereitungsaufwand auf der BPaaS Plattform abgewickelt werden. Ein weiterer Nutzen resultiert aus dem Cloud Ansatz: Ein Unternehmen muss keine eigene IT Infrastruktur für den Betrieb einer Process Engine aufbauen und warten. Auch muss die BPMS Software nicht selber installiert, konfiguriert und gewartet werden. Dadurch, dass die Business Prozesse auf dem Cloud Stack abgewickelt werden, können hohe Lasten aufgefangen werden respektive bei hoher Last die zur Verfügung stehende Performance hochskaliert werden. Ein weiterer Vorteil ist, dass vorab für ein Unternehmen keine grossen Investitionen notwendig sind, um sich mit dem Thema BPaaS auseinander zu setzen. Das Risiko reduziert sich so für ein Unternehmen [11].

Eine BPaaS Plattform, welche die Ausführung von unstrukturierten Business Prozessen mittels Activity Patterns anbietet, existiert nach heutigem Stand des Wissens nicht (vergleiche Kapitel 2.1). Somit fehlt es Unternehmen an einer Umgebung, über welche unstrukturierte Business Prozesse abgewickelt werden können, ohne vorgängig entsprechende Prozess Modelle zu entwickeln und technische (ausführbare) Modelle davon abzuleiten (vergleiche [10]).

1.2 Zielsetzungen

Das Ziel dieser Master Thesis ist die Entwicklung und Evaluierung von zwei Artefakten:

1. Es soll ein Konzept für eine Cloud-basierte BPaaS Plattform erarbeitet werden, über welche unstrukturierte Business Prozesse unter Verwendung von Activity Patterns abgewickelt werden können. Die Plattform soll auf dem Cloud Stack aufbauen und die Camunda Process Engine [12] für die Ausführung der Business Prozesse als Basis verwenden.

2. Die BPaaS Plattform soll prototypisch implementiert werden, gegen Kriterien evaluiert werden und die Implementation über einen Cloud Dienstleister (z.B. Amazon Web Services [13]) veröffentlich werden. Dazu gehört auch eine Webseite, über welche ein Mandant eröffnet und gelöscht werden kann.

Durch die Entwicklung und Evaluierung dieser Artefakte soll neues Wissen entstehen, um welches die Knowledge Base erweitert werden kann (siehe Kapitel 3 Forschungsmethode).

1.3 Forschungsfragen

Die Hauptforschungsfrage und die dazugehörigen Unterforschungsfragen leiten sich aus den Zielsetzungen im Kapitel 1.2 ab.

1.3.1 Hauptforschungsfrage

HF: *Wie kann eine BPaaS Plattform realisiert werden, welche unstrukturierte Business Prozesse mittels Activity Patterns auf Basis von Camunda als Service anbietet?*

1.3.2 Unterforschungsfragen

UF 1: *Welche Arten von Zusammenarbeit zwischen verschiedenen organisatorischen Rollen sollen durch die BPaaS Plattform unterstützt werden?*

UF 2: *Wie können die Activity Patterns auf der BPaaS Plattform angeboten werden?*

UF 3: *Was muss pro Activity Pattern konfigurierbar und parametrierbar sein, damit die Activity Patterns Mandanten-spezifisch genutzt werden können?*

UF 4: *Welche Regeln bezüglich Activity Pattern Reihenfolge müssen auf der BPaaS Plattform abgebildet werden?*

UF 5: *Wie können beliebig anfallende Prozess Daten zur Verfügung gestellt und persistiert werden?*

UF 6: *Welche Kriterien sollen herangezogen werden, um zu entscheiden, welche Multi-Tenancy Architektur umgesetzt wird?*

UF 7: *Wie können Benutzer- und Gruppen Informationen, z.B. ab einem LDAP beim Mandanten, in die BPaaS Plattform integriert werden?*

1.4 Umfang und Abgrenzung

Vonseiten des Erstgutachters wurde die Anforderung an diese Master Thesis herangetragen, dass sich die zu unterstützenden unstrukturierten Business Prozesse aus einer vorgegebenen Sammlung an Activity Patterns zusammensetzen sollen [10]. Des Weiteren wurde vorgegeben, dass die BPaaS Plattform auf dem Camunda BPM Framework aufbauen soll. Camunda ist ein Open-Source, Java-basiertes Framework, welches Business Process Model and Notation (im Folgenden: BPMN) unterstützt [12]. BPMN 2.0 ist ein ISO Standard für die Business Prozess Modellierung [14]. Im Kern ist Camunda eine Modell Ausführungs-Engine, welche nebst anderen Notationen auch BPMN 2.0 ausführen kann. Die Modell Ausführungs-Engine bildet das Herzstück von der BPaaS Plattform. Sie führt die Business Prozess Instanzen aus und überwacht deren Status. Die in dieser Master Thesis verwendete Version ist Camunda 7.6.0 [15].

Weiter existieren grundsätzliche Sachverhalte, welche gemäss Aufgabenstellung vom Autor als gegeben angenommen werden. Die Relevanz einer BPaaS Plattform wird als gegeben betrachtet und nicht hinterfragt. Es werden demnach im Rahmen dieser Master Thesis keine Abklärungen gemacht, ob ein Bedürfnis an einer BPaaS Plattform besteht. Zudem wird es als gegeben betrachtet, dass Knowledge Worker ihre Arbeit mittels Activity Patterns organisieren können. Es wird weiter auch nicht hinterfragt, ob es von Unternehmen gewünscht ist, dass die BPaaS Plattform in der Cloud betrieben wird und somit die Prozess Daten nicht unter Kontrolle der Unternehmen sind.

1.5 Aufbau der Master Thesis

Diese Master Thesis ist folgendermassen gegliedert:

Kapitel 1 führt den Leser in das Thema und die Problemstellung ein. Weiter werden die Ziele und die zu beantwortenden Forschungsfragen aufgezeigt. Kapitel 2 erläutert den State-of-the-Art und vertieft weitere Aspekte des Themengebiets. In Kapitel 3 wird die Forschungsmethode erläutert. Die Forschungsmethode beschreibt die Vorgehensweise

der Erkenntnisgewinnung mit dem Ziel, die Forschungsfragen beantworten zu können. Zudem wird beschrieben, wie und anhand welcher Kriterien die Artefakte evaluiert werden. In Kapitel 4 wird das zentrale Artefakt dieser Master Thesis, der Ad Hoc Prozess, erarbeitet. Die Benutzeroberfläche von der BPaaS Plattform wird in Kapitel 5 entworfen. In Kapitel 6 wird aufgezeigt, wie ein BPaaS Plattform Zugang für ein Unternehmen automatisiert eingerichtet werden kann. Kapitel 7 beschreibt, welche Herausforderungen und Lösungen bezüglich der Zurverfügungstellung und Speicherung von Prozess Daten existieren. Kapitel 8 zeigt auf, wie die Arbeitsergebnisse dieser Master Thesis entsprechend der Forschungsmethode evaluiert werden.

Zum Abschluss werden in Kapitel 9 die Resultate dieser Master Thesis zusammengefasst, die Forschungsfragen beantwortet, die Vorgehensweise und Resultate dieser Master Thesis kritisch gewürdigt sowie einen Ausblick gegeben.

Die Anhänge A und B beschreiben die Produktiv- sowie die Entwicklungsumgebung und geben eine Anleitung zur Installation und Konfiguration dieser Umgebungen.

Kapitel 2

Hintergrund

2.1 BPaaS Plattform

Häufig lassen sich Business Prozesse nicht über ein Business Prozess Modell formalisieren, weil zu viele Eventualitäten auftreten können und der Verlauf vom Business Prozess abhängig von Interpretationen und Entscheidungen eines Knowledge Workers ist [7]. Viele Business Prozesse, vor allem wissensintensive Business Prozesse, starten Ad Hoc ohne im Voraus geplant zu werden [7]. Ein Prozess Modell würde unter der Berücksichtigung aller Ausnahmefälle sehr schnell auf ein Ausmass anwachsen, welches nicht mehr handhabbar und verständlich ist [3, 7].

Die Basis von Business Prozess Management Systemen (im Folgenden: BPMS) ist ein Business Prozess Modell, welches zuerst modelliert und dann ausgeführt wird. Entsprechend können unstrukturierte Prozesse auf einem BPMS nicht ausgeführt werden, weil kein Business Prozess Modell vorliegt [7].

Process Engines, welche Business Process Model and Notation (BPMN) oder auch Case Management Model and Notation (CMMN) Modelle ausführen können, gibt es viele am Markt. Ein Beispiel ist Camunda [12]. Weiter lassen sich im Internet auch Plattformen finden, welche unter dem Begriff BPaaS angeboten werden. Zwei davon sind die BPaaS Plattformen der Scheer Group [16] sowie von Tata Consultancy Services (TCS) [17]. Beide Plattformen scheinen aber nur strukturierte Prozesse zu unterstützen.

Zu Plattformen, welche unstrukturierte Business Prozesse unterstützen, findet man wenig Literatur. Eine BPaaS Plattform, welche das Ausführen von unstrukturierten Business Prozessen als Service anbietet, wurde bei der Recherche nicht gefunden.

2.2 Activity Patterns

Wird im Business Prozess Bereich von Patterns gesprochen, dann sind meist so genannte Workflow Patterns gemeint [18]. Workflow Patterns bieten eine Vorlage für wiederkehrende Abläufe in Workflows. In [18] wird erläutert, dass solche Workflow Patterns die Modellierungsphase (Design Time) erheblich beschleunigen können.

Die in dieser Master Thesis zu verwendenden Activity Patterns stellen nach [9] eine Business Funktion dar, welche häufig in Business Prozessen gefunden werden kann. In [10] wurden die Activity Patterns entwickelt, um ein weiteres Problem zu lösen: Es soll eine automatisierte Ableitung vom technischen (ausführbaren) Modell ermöglicht werden, indem in der Modellierungsphase diese Activity Patterns verwendet werden. Bei diesem sonst manuellen Vorgang treten häufig Probleme auf, welche umgangen werden können, indem die Prozess Modelle aus Activity Patterns aufgebaut werden.

Im Rahmen dieser Master Thesis werden die Activity Patterns nicht bei der Modellierung von Business Prozessen eingesetzt, sondern erst zur Laufzeit. Wie bereits im Kapitel 1 erläutert, soll ein Ansatz gesucht werden, welcher ohne vorangehende Prozess Modellierung auskommt. Dieser Ansatz wird im nächsten Abschnitt 2.3 beschrieben. In [10] ist erwähnt, dass die nachfolgenden zehn Activity Patterns (siehe Tabelle 2.1) ausgereicht haben, einfache Business Prozesse bei zwei Automatisierungsprojekten im öffentlichen Bereich abzubilden. Im Rahmen dieser Master Thesis soll versucht werden, unstrukturierte Business Prozesse auf Basis von diesen Activity Patterns anzubieten. So kann auf eine Business Prozess Modellierung verzichtet werden.

Activity Pattern	**Beschreibung**
Eröffnen von Business Case	Damit wird eine eindeutige ID für den Business Case erzeugt und die Prozess Daten instanziiert.
Schliessen von Business Case	Damit wird der Business Case abgeschlossen und die Prozess Daten archiviert.

Editieren von Prozess Daten	Das Ziel von jeder Aktivität ist es, die Prozess Daten zu erfassen, zu ändern oder zu löschen.
Verfeinern von Prozess Daten	Hier geht es darum, bestehende Daten miteinander zu verknüpfen und zu verfeinern. Es werden keine Daten geändert, erfasst oder gelöscht.
Exportieren von Prozess Daten	Beim Exportieren von Prozess Daten werden Daten an externe Umsysteme exportiert oder an Prozessteilnehmer verschickt (z.B. per E-Mail).
Importieren von Prozess Daten	Beim Importieren von Prozess Daten werden Daten von externen Umsystemen empfangen und in den Prozess Daten abgelegt.
Entscheidung	Es wird aufgrund von den Prozess Daten eine Entscheidung gefällt.
Ausführen von Task	Damit wird eine Aufgabe abgebildet, welche ein Benutzer z.B. in einem externen System durchführen muss und mitteilen muss, dass die Aufgabe durchgeführt wurde.
Event absetzen	Es wird damit ein Ereignis abgesetzt.
Event empfangen	Es wird damit auf ein Ereignis gewartet.

Tabelle 2.1: Activity Patterns [10]

2.3 Ad Hoc Prozess

Die BPaaS Plattform soll einem Knowledge Worker ermöglichen, die nächsten auszuführenden Aktionen in Form von Activity Patterns auszuwählen und diese in Auftrag zu geben. Sobald diese Aktionen durchgeführt wurden, soll der Knowledge Worker die nächsten Aktionen auswählen können. Dieser Ablauf wiederholt sich so lange, bis das Ziel vom Knowledge Worker erreicht ist. Entsprechend handelt es sich um einen Ablauf, welcher aus zwei Aktivitäten besteht (Activity Patterns auswählen und Activity

Patterns ausführen) und beliebig oft wiederholt werden kann. Dieser Ablauf wird vom Autor Ad Hoc Prozess genannt, weil ein solcher Business Prozess sich jederzeit starten lässt und der Ablauf Ad Hoc zur Laufzeit in Form von Activity Patterns bestimmt wird.

In [19] wurde ein ähnliches Konstrukt als Proof of Concept implementiert. Der erläuterte Ansatz ist vom Prinzip her derselbe wie der im Rahmen dieser Master Thesis umzusetzende Ad Hoc Prozess. Dabei werden aber in [19] nicht einzelne Aktivitäten (Activity Patterns) ausgewählt und ausgeführt, sondern ganze Subprozesse, welche in Form von BPMN Prozess Modellen vorliegen.

2.4 Prozess Daten

In [20] sind einige Eigenschaften und Charakteristiken von unstrukturierten Business Prozessen beschrieben. Unter anderem ist erwähnt, dass unstrukturierte Business Prozesse inhaltsintensiv, analysegetrieben und dynamisch sind. Unter Berücksichtigung dieser Eigenschaften wird klar, dass sämtliche Informationen und Dokumente (so genannte Prozess Daten), welche während der Ausführung eines unstrukturierten Business Prozesses erarbeitet und bearbeitet wurden, jederzeit dem ausführenden Knowledge Worker zur Verfügung gestellt werden müssen.

2.5 On-Boarding

Die BPaaS Plattform soll von mehreren Unternehmen parallel genutzt werden können. Damit ein Unternehmen die BPaaS Plattform nutzen kann, muss für das Unternehmen eine entsprechende Umgebung aufgebaut werden (so genanntes On-Boarding). Das On-Boarding soll automatisiert ablaufen, damit für den Betreiber der BPaaS Plattform diesbezüglich keinen Aufwand anfällt.

2.5.1 Multi-Tenancy

Camunda ist mandantenfähig [21]. Multi-Tenancy kann in Camunda auf verschiedene Arten umgesetzt werden. In der Camunda Dokumentation ist beschrieben, welche Multi-Tenancy Ansätze unterstützt werden [21]:

- Jedem Mandanten (Unternehmen) wird eine eigene Process Engine zur Verfügung gestellt. Dabei können die Daten der verschiedenen Mandanten in einer Datenbank pro Mandant oder in einer globalen, Mandanten-übergreifenden Datenbank gespeichert werden.
- Alle Mandanten werden auf einer globalen, Mandanten-übergreifenden Process Engine und Datenbank betrieben. Dabei müssen die Mandanten über einen Mandanten Identifier identifiziert werden.

Des Weiteren kann die Multi-Tenancy auch auf einem tieferen Architektur Layer realisiert werden, indem dem Mandanten eine eigene virtuelle Maschine zur Verfügung gestellt wird und darauf entsprechend eine Process Engine mit Datenbank betrieben wird.

Diese Ansätze sollen nach gewissen Kriterien bewertet werden, damit eine optimale Wahl für die BPaaS Plattform getroffen werden kann. Mögliche Kriterien sind in [22] und [23] erwähnt.

2.5.2 Benutzer und Gruppen

Zum On-Boarding gehört auch der Import von Benutzer- und Gruppen Informationen z.B. ab einem Lightweight Directory Access Protocol (LDAP) dazu. Grundsätzlich erlaubt es Camunda, ein LDAP einzubinden [24]. Die Schwierigkeit besteht darin, dass dieses LDAP in der Regel in der Netzwerkumgebung beim Kunden betrieben wird. Wenn möglich sollte das LDAP transparent integriert werden, damit die Benutzer- und Gruppen Informationen nicht doppelt gepflegt werden müssen. Eine andere Möglichkeit wäre ein periodischer, automatischer oder manueller Abgleich zwischen dem LDAP des Kunden und der BPaaS Plattform. Einfachste Möglichkeit ist die einmalige Übermittlung von Benutzer- und Gruppen Informationen (z.B. mit einer Datei) beim On-Boarding. Nachträgliche Ergänzungen müssen dann manuell auf der BPaaS Plattform gepflegt werden.

2.6 Anwendungsfälle

In diesem Abschnitt wird erläutert, wie die zu erschaffende BPaaS Plattform genutzt werden soll. Dazu wird in einem UML (Unified Modeling Language) Anwendungsfall Diagramm (Abbildung 2.1) aufgezeigt, welche Akteure welche Anwendungsfälle ausführen.

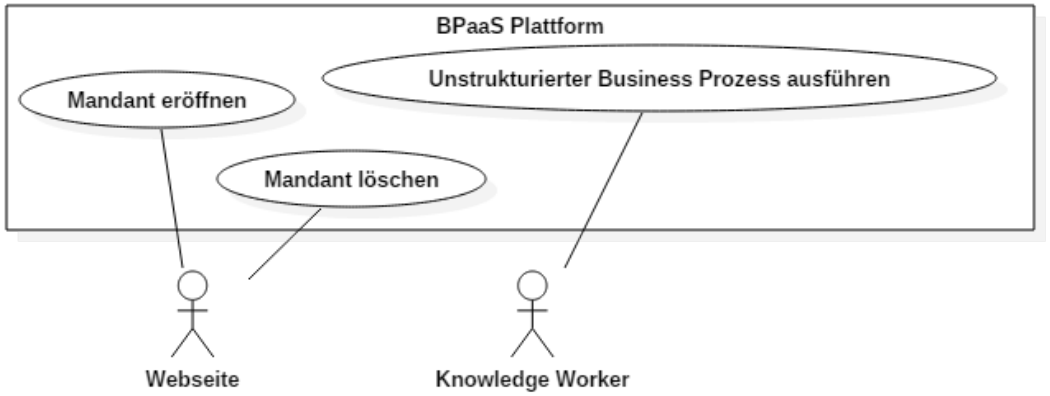


Abbildung 2.1: Anwendungsfall Diagramm BPaaS

2.6.1 Mandant eröffnen

Über eine Webseite soll es möglich sein, einen Mandanten für ein Unternehmen auf der BPaaS Plattform zu eröffnen (Anwendungsfall *Mandant eröffnen*). Dabei soll der Anwendungsfall vom interessierten Unternehmen selber angestossen werden können. Der Anwendungsfall soll automatisch, ohne manuelle Tätigkeiten vom BPaaS Plattform Betreiber ablaufen.

2.6.2 Mandant löschen

Gleich wie *Mandant eröffnen* soll auch *Mandant löschen* über eine Webseite und ohne manuelle Tätigkeiten vom BPaaS Plattform Betreiber ausgeführt werden können.

2.6.3 Unstrukturierter Business Prozess ausführen

Der zentrale Anwendungsfall, welcher die BPaaS Plattform unterstützt, ist *Unstrukturierter Business Prozess ausführen*. Der Anwendungsfall wird von einem registrierten Unternehmen angestossen, für welches zuvor ein Mandant eröffnet wurde. Bei der ausführenden Person wird davon ausgegangen, dass es sich um eine Person aus dem Business handelt (ein Knowledge Worker).

Kapitel 3

Forschungsmethode

Bei dieser Master Thesis kommt ein Design Science Ansatz nach Hevner [25] zum Einsatz, weil das primäre Ziel die Entwicklung eines neuen Artefakts ist. Durch die Entwicklung und Evaluierung des Artefakts gelangt man zu Erkenntnissen und Wissen. Design Science ist ein Paradigma, welches seit der Publikation von [25] in der Information Systems (IS) Forschung immer mehr Aufmerksamkeit erhält. Die Motivation von Design Science ist nach [26] der Wunsch, das Umfeld durch die Einführung von neuen und innovativen Artefakten zu verbessern.

Diese Master Thesis besteht aus zwei Artefakten. Zum einen aus einem Konzept, welches die BPaaS Plattform konzeptionell beschreibt und zum anderen aus einem Prototyp, welcher die BPaaS Plattform prototypisch gemäss Konzept implementiert. Der Hauptaspekt vom Konzept ist der Ad Hoc Prozess, welcher aufzeigt, welche Abläufe durch die BPaaS Plattform angeboten werden. Beide Artefakte adressieren das in der Einleitung (vergleiche Kapitel 1.1) aufgezeigte Problem.

3.1 Research Cycles

In [26] werden als Erweiterung zu [25] die drei grundlegenden Research Cycles erläutert, welche die Basis vom Design Science Ansatz darstellen und eine Hilfestellung bei der Ausführung von Design Science bieten. Das Ziel von der Anwendung der Research Cycles ist es, erstens ein Ergebnis zu erhalten, welches Relevanz hat und zweitens die Sicherstellung der wissenschaftlichen Rigorosität.

In der nachfolgenden Abbildung 3.1 ist das Design Science Research Modell nach [25] dargestellt und die drei Research Cycles aus [26] darübergelegt. Die Research Cycles werden in den folgenden Abschnitten erläutert und auf diese Master Thesis angewendet.

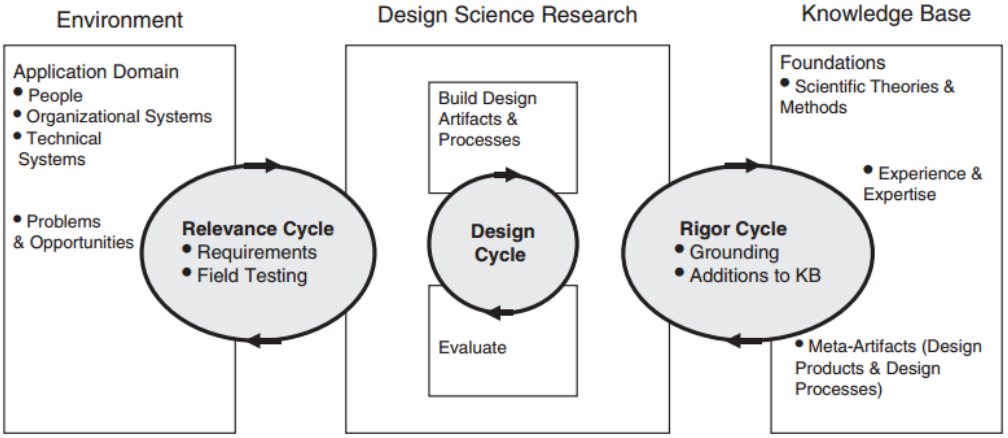


Abbildung 3.1: Design Science Research Cycles [27]

3.1.1 Relevance Cycle

Gemäss Hevner [26] beginnt gute Forschung nach dem Design Science Ansatz mit dem Relevance Cycle. Es werden Möglichkeiten und Probleme in der Umgebung (Environment) identifiziert. Aus dieser Umgebung werden die Anforderungen abgeleitet. Wie bereits in der Einleitung erwähnt (vergleiche Kapitel 1.4), wird die Relevanz der BPaaS Plattform nicht hinterfragt resp. untersucht, sondern wird als gegeben betrachtet. Auch werden im Rahmen der Master Thesis keine Feldtests mit Benutzern aus der Umgebung (Environment) gemacht. Dieser Schritt kann Teil von nachfolgenden Forschungsarbeiten sein (vergleiche Kapitel 9).

3.1.2 Rigor Cycle

Durch den Rigor Cycle wird sichergestellt, dass vorhandenes Wissen berücksichtigt wird und durch die Forschungsarbeit somit Innovation entsteht. Dies wird durch die Erarbeitung vom State-of-the-Art, also dem Studium von der vorhandenen Literatur erreicht. Das im Rahmen der Vorstudie und der Master Thesis erarbeitete Wissen ist im Kapitel Hintergrund aufbereitet (siehe Kapitel 2). Zudem soll das während der Master

Thesis erarbeitete Wissen wieder zurück in die Knowledge Base fliessen. Durch die Implementierung und Evaluierung der eingangs erwähnten Artefakte sollen die im Kapitel 1.3 aufgestellten Forschungsfragen beantwortet werden können. Die beantworteten Forschungsfragen werden im Rahmen der Master Thesis publiziert und bieten so einen Beitrag zur Knowledge Base.

3.1.3 Design Cycle

Der Design Cycle ist der zentrale Cycle vom Design Science Ansatz, bei welchem das Artefakt entwickelt wird. Im Design Cycle werden die zwei Phasen *Build* und *Evaluate* durchlaufen. Nach [25] wird dabei ein iteratives Vorgehen angewendet:

In der Build Phase wird das Artefakt oder ein Teil davon auf Basis der identifizierten Probleme und Anforderungen (Relevance Cycle) sowie unter Berücksichtigung von vorhandenem Wissen (Rigor Cycle) entwickelt.

In der Evaluate Phase wird das Artefakt gegen Kriterien evaluiert. Verschiedene Evaluierungsmethoden werden unter anderem in [25] und [28] erläutert.

3.2 Forschungsprozess

Nach Peffers et al. [29] fehlt in der Design Science Forschung eine Methodologie, welche bei der Ausführung von Design Science befolgt werden kann und eine akzeptierte Vorgehensweise repräsentiert. Deshalb erarbeiteten Peffers et al. in ihrem Paper die Design Science Research Methodology (vergleiche Abbildung 3.2), welche bei der Forschung nach dem Design Science Ansatz eine Hilfestellung bietet und einen Konsens über die bereits zum Thema Design Science publizierte Literatur bilden soll. Diese Vorgehensweise wird im Rahmen dieser Master Thesis angewendet. Die nachfolgend erläuterten Schritte von der Design Science Research Methodology decken sich mit den im Kapitel 3.1 aufgezeigten Research Cycles.

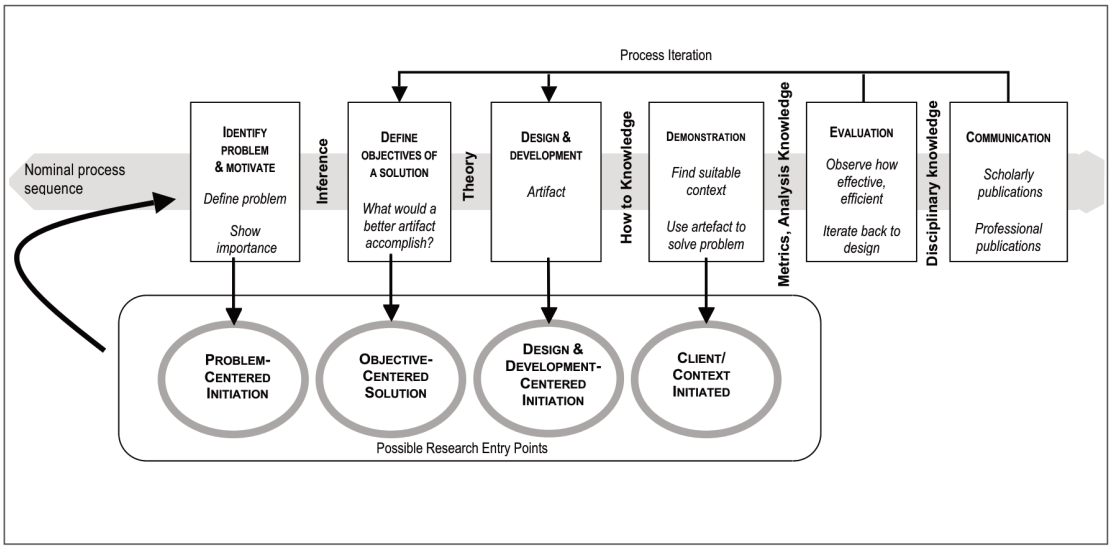


Abbildung 3.2: Design Science Research Methodology [29]

3.2.1 Problem Identifikation und Motivation

Im ersten Schritt wird das Forschungsproblem identifiziert und die Bedeutung des Problems nachgewiesen. Das im Rahmen dieser Master Thesis zu lösende Problem sowie dessen Relevanz wird, wie oben erläutert, gemäss Aufgabenstellung als gegeben betrachtet und nicht weiter erforscht. Entsprechend finden keine Aktivitäten diesbezüglich statt. Das Forschungsproblem, die Motivation sowie der Nutzen vom Resultat dieser Master Thesis werden im Kapitel 1 aufgezeigt.

3.2.2 Ziele und Anforderungen

Im zweiten Schritt geht es darum, die Ziele und Anforderungen zu beschreiben, welche die Lösung erfüllen soll. Dazu hat sich der Autor gemeinsam mit dem Erstgutachter für folgende Vorgehensweise entschieden: Aus der Literatur werden mögliche, konkrete Anwendungsfälle hergeleitet und aufgezeigt, wie die BPaaS Plattform genutzt werden kann und so in den Anwendungsfällen Unterstützung bietet. Aus diesen Anwendungsfällen werden schliesslich Anforderungen und Regeln an den Ad Hoc Prozess abgeleitet. Anhand dieser Regeln wird ein Ad Hoc Prozess Modell erarbeitet und so eine Arbeitshypothese gebildet. Die Resultate dieser Aktivität sowie die Arbeitshypothese finden sich im Kapitel 4.

3.2.3 Design und Entwicklung

Im dritten Schritt wird die Lösung entworfen und umgesetzt. Der Entwurf der Lösung ist in Kapitel 4 und 5 aufgezeigt, wo das zentrale Artefakt, das Ad Hoc Prozess Modell auf Basis vorhandener Forschungsergebnisse erarbeitet wird sowie User Interface Mockups beschrieben werden. Durch die Integration in das Camunda Framework ergeben sich trotz Entwicklung eines Software Prototyps keine weiteren, relevanten Software Design Aktivitäten. Das Ad Hoc Prozess Modell dient als Arbeitshypothese, welches im Rahmen eines Prototyps umgesetzt wird. Die Umsetzung der Lösung findet iterativ statt, sprich die Funktionalitäten werden in handhabbare Teilfunktionalitäten aufgeteilt, diese entwickelt und mit diesen die nachfolgenden Evaluierungsaktivitäten durchlaufen (Demonstration, Evaluation). Danach wird erst mit der Umsetzung der nächsten Teilfunktionalität begonnen. Die Aufteilung in Teilfunktionalitäten orientiert sich an den umzusetzenden User Interface Masken und den dazugehörigen Funktionalitäten.

3.2.4 Demonstration

In [29] wird vor der Evaluation eine Demonstration Aktivität durchgeführt. Nach [28] wird die Demonstration Aktivität als frühe Evaluationsphase verstanden. Im Rahmen der Master Thesis wird zur Demonstration die Teilfunktionalität des Prototyps zur Ausführung gebracht und explorativ getestet.

Das BPaaS Plattform Konzept, welches ebenfalls ein Artefakt nach dem Design Science Ansatz darstellt, wird keiner Demonstration unterzogen.

3.2.5 Evaluierung

In diesem Schritt geht es darum, das Artefakt zu evaluieren. Dabei sollen begründete Evaluierungskriterien und -methoden zum Einsatz kommen.

In Prat et al. [28] ist erwähnt, dass die Kriterien und Methoden zur Evaluierung über die Design Science Literatur hinweg verteilt und unvollständig sind. In [28] werden deshalb die Kriterien und Methoden mittels einer Literaturanalyse strukturiert aufgearbeitet.

Diese Kriterien und Methoden bilden die Basis für die Definition der Evaluierung dieser Master Thesis. Die Kriterien zur Evaluierung werden in [28] anhand von fünf Dimensionen organisiert. Dabei wird erwähnt, welche Kriterien für welche Design Science Artefakttypen (Constructs, Models, Methods, Instantiations) eingesetzt werden.

Der Autor entscheidet sich aufgrund von der Aufgabenstellung für das Kriterium Efficacy (Wirksamkeit) aus der Goal (Ziel) Dimension. Als Evaluierungsmethode soll die in [28] erwähnte Methode *Demonstration of the use of the artifact with one or several examples* zum Einsatz kommen. Die Methode wird gewählt, weil sie zur Vorgehensweise passt, die bei der Erarbeitung vom Ad Hoc Prozess Modell angewendet wird. Hier werden anhand von Beispielabläufen die Anforderungen an den Ad Hoc Prozess abgeleitet. Die Beispielabläufe sollen mit dem Prototyp durchgespielt werden und so die Wirksamkeit respektive den gewünschten Effekt vom Prototyp aufgezeigt werden. So wird der Prototyp in einem realen Kontext getestet.

Die Evaluierungsmethode soll auf verschiedenen Layern (Business Logik und User Interface) angewendet werden. Zum einen soll die Business Logik evaluiert werden. Hier sollen mittels Unit Tests die Beispielabläufe programmatisch abgedeckt werden. Und zum anderen sollen die Beispielabläufe auch von Hand direkt im Prototyp durchgespielt werden (User Interface Layer).

Das BPaaS Plattform Konzept wird keiner Evaluierung unterzogen.

3.2.6 Kommunikation

Im letzten Schritt geht es um die Kommunikation vom Problem, der Relevanz vom Problem sowie dem Nutzen und Neuartigkeit vom umgesetzten Artefakt. Die Kommunikation wird durch die Erstellung von diesem Master Thesis Bericht und gegebenenfalls der Publikation der Resultate dieser Master Thesis sichergestellt.

3.3 Fazit

Ein weitreichenderer Ansatz zur Erarbeitung der Anforderungen an den Ad Hoc Prozess wäre die Beobachtung von ablaufenden unstrukturierten Business Prozessen im Feld oder die Befragung von Experten (Knowledge Workern) gewesen. Weil aber aus der

Aufgabenstellung der Fokus auf der Erkenntnisgewinnung durch die Entwicklung von einem Prototyp lag, wurde darauf verzichtet.

Es kann kritisiert werden, dass das Konzept, insbesondere der Ad Hoc Prozess, nicht in den Aktivitäten *Demonstration* und *Evaluierung* überprüft wird. Der Autor ist überzeugt, dass die Befragung von Experten zum entwickelten Ad Hoc Prozess Modell und den damit unterstützten Abläufen sicherlich Erkenntnisse hervorgebracht hätten, die das Prozess Modell weiter verfeinert und der Nutzen der Lösung erhöht hätten.

Weiter kann kritisiert werden, dass der Prototyp nicht bezüglich seines Nutzens (Utility) für Organisationen und Benutzer (Knowledge Worker) evaluiert wird, denn gemäss [28] ist der Nutzen von einem Artefakt das primäre Ziel von Design Science Research. Dazu wäre eine Befragung von Knowledge Workern notwendig gewesen. Dies kann in nachfolgenden Forschungsarbeiten geschehen (vergleiche Kapitel 9).

Kapitel 4

Ad Hoc Prozess

Dieses Kapitel beschreibt den Ad Hoc Prozess. Der Ad Hoc Prozess ist das zentrale Artefakt dieser Master Thesis. Entsprechend wird ab hier der State-of-the-Art verlassen und eine Arbeitshypothese gebildet.

Der Ad Hoc Prozess definiert, wie auf der BPaaS Plattform mit den Activity Patterns gearbeitet werden kann. Der Ad Hoc Prozess wird als BPMN Prozess Modell entwickelt, welches auf der Camunda Plattform ausgeführt werden kann.

Die Vorgehensweise zur Erarbeitung vom Ad Hoc Prozess Modell ist im vorangehenden Kapitel 3 beschrieben. Es sollen unstrukturierte Business Prozesse aus wissenschaftlichen Quellen erarbeitet werden und anhand dieser aufgezeigt werden, wie über die BPaaS Plattform diese unstrukturierten Business Prozesse abgewickelt werden könnten. Daraus sollen Erkenntnisse gewonnen werden, welche Arbeitsabläufe durch den Ad Hoc Prozess unterstützt werden müssen.

4.1 Verwendung der BPaaS Plattform

In diesem Abschnitt werden Beispiele von Business Prozessen näher erläutert, welche unstrukturiert verlaufen und sinnvollerweise über eine BPaaS Plattform abgewickelt werden.

4.1.1 Behandlung eines Patienten

Als erstes Beispiel soll die Behandlung eines Patienten durch einen Arzt aufgegriffen werden [30]. Die Behandlung ist ein wissensintensiver Prozess, welcher je nach Zustand des Patienten, Charakteristiken vom Patienten oder gestellter Diagnose anders verläuft.

Der nachfolgende Abschnitt soll aufzeigen, wie ein Arzt die BPaaS Plattform nutzbringend während der Behandlung eines Patienten einsetzen kann. Dazu wird anhand eines beispielhaften Diagnose- und Behandlungsablaufs die Verwendung der BPaaS Plattform sowie von den Activity Patterns (siehe Tabelle 2.1) erläutert. Die nachfolgenden Erläuterungen basieren auf einer Feldstudie in der Notfallaufnahme des Klinikums der Johannes Gutenberg-Universität Mainz [31]. Im erwähnten Paper werden konkrete Prozess Instanzen von Behandlungen dokumentiert. Vom Autor wurde absichtlich ein Notfall ausgewählt (akuter Herzinfarkt), weil hier die Abläufe nicht immer der gleichen Prozedur folgen. Der untenstehende Prozessablauf ist ein durch den Autor zusammengestellter Ablauf, welcher an die in [31] dokumentierte Prozessinstanz *P 01* angelehnt ist.

1. Der Patient tritt mit Verdacht auf einen Herzinfarkt in die Notfallaufnahme ein.
2. Der nächste freie Pflegemitarbeiter übernimmt den Patienten. Der Pflegemitarbeiter startet einen neuen Ad Hoc Prozess.
3. Als nächstes befragt der Pflegemitarbeiter den Patienten zu den Personalien. Dazu wählt der Pflegemitarbeiter das Activity Pattern *Editieren von Prozess Daten* und hält die Patientendaten fest.
4. Der Pflegemitarbeiter übergibt den Ad Hoc Prozess an die Rolle Arzt.
5. Der nächste freie Arzt übernimmt den Ad Hoc Prozess. Der Arzt befragt und untersucht den Patienten. Dabei füllt der Arzt einen Verdachtsdiagnosebogen, einen Anordnungsbogen sowie einen Anamnesebogen elektronisch aus. Der Arzt wählt das Activity Pattern *Editieren von Prozess Daten* und hängt die Dokumente dem Business Case an.
6. Der Arzt veranlasst die Erstellung von einem Elektrokardiogramm (im Folgenden: EKG) sowie von einem Blutbild. Dazu erfasst der Arzt je ein Activity Pattern vom Typ *Ausführen von Task* und gibt an, dass aus den Activity Patterns neue Ad

Hoc Prozesse gestartet werden können. Das Activity Pattern für die Durchführung vom EKG weist der Arzt der Rolle Kardiologe zu. Das Activity Pattern für die Erstellung vom Blutbild weist der Arzt der Rolle Labormitarbeiter zu.

7. Zudem veranlasst der Arzt die Abgabe von Medikamenten. Dazu erfasst der Arzt ein Activity Pattern vom Typ *Ausführen vom Task*. Der Arzt gibt an, dass nicht auf die Ausführung von diesem Activity Pattern gewartet werden muss. Sprich, sobald das EGK und das Blutbild erstellt sind, kommt der Business Case wieder zurück zum Arzt, unabhängig davon, ob die Medikamente bereits abgegeben wurden. Das Activity Pattern weist der Arzt der Rolle Pflegemitarbeiter zu.

8. Der nächste freie Pflegemitarbeiter sieht das auszuführende Activity Pattern auf der BPaaS Plattform. Der Pflegemitarbeiter verabreicht dem Patienten die Medikamente und schliesst Activity Pattern ab.

9. Der nächste freie Labormitarbeiter sieht das auszuführende Activity Pattern auf der BPaaS Plattform. Da die Erstellung vom Blutbild wiederum aus mehreren Schritten besteht, startet der Labormitarbeiter aus dem Activity Pattern einen neuen Ad Hoc Prozess, um die Aufgabe durchzuführen. Der Labormitarbeiter erfasst ein Activity Pattern vom Typ *Ausführen von Task*, um das Blut abzunehmen. Zudem erfasst der Labormitarbeiter ein zweites Activity Pattern vom Typ *Editieren von Prozess Daten*, um das Resultat des Blutbilds zu dokumentieren. Der Labormitarbeiter nimmt dem Patienten Blut ab und führt die Laboranalyse durch. Der Labormitarbeiter dokumentiert das Ergebnis im Activity Pattern *Editieren von Prozess Daten*. Danach schliesst der Labormitarbeiter die Activity Patterns ab.

10. Parallel dazu sieht der nächste freie Kardiologe das auszuführende Activity Pattern auf der BPaaS Plattform. Da die Erstellung vom EKG wiederum aus mehreren Schritten besteht, startet der Kardiologe aus dem Activity Pattern einen neuen Ad Hoc Prozess, um die Aufgabe durchzuführen. Der Kardiologe erfasst ein Activity Pattern vom Typ *Ausführen von Task*, um das EKG durchzuführen. Zudem erfasst der Kardiologe ein zweites Activity Pattern vom Typ *Editieren von Prozess Daten*, um das Resultat vom EKG zu dokumentieren. Der Kardiologe führt das EKG durch. Der Kardiologe dokumentiert das Ergebnis vom EGK im Activity Pattern

Editieren von Prozess Daten. Danach schliesst der Kardiologe die Activity Patterns ab.

11. Der Arzt sieht auf der BPaaS Plattform, dass der Business Case wieder bei ihm liegt (Aktivität *Activity Patterns auswählen*). Der Arzt studiert die Resultate von der Blut- und EKG Untersuchung und beantragt eine Herzkatheteruntersuchung in der Kardiologie. Die Entscheidung für eine solche Untersuchung muss durch den Oberarzt freigegeben werden. Der Arzt wählt dazu das Activity Pattern *Entscheidung* und dokumentiert dort den Behandlungsvorschlag. Das Activity Pattern weist der Arzt der Rolle Oberarzt zu.

12. Der nächste freie Oberarzt sieht das Activity Pattern und analysiert die vorhandenen Informationen auf dem Business Case sowie die bereits getätigten Schritte (Activity Patterns). Der Oberarzt akzeptiert den Behandlungsvorschlag und schliesst das Activity Pattern ab.

13. Der Arzt sieht auf der BPaaS Plattform, dass der Business Case wieder bei ihm liegt. Der Patient muss nun in die Intensivstation aufgenommen werden. Dazu wählt der Arzt ein Activity Pattern *Ausführen von Task* und dokumentiert, dass der Patient in die Intensivstation aufgenommen werden muss. Damit endet der Notfallprozess. Dabei übergibt der Arzt die Verantwortung für den Business Case der Rolle Pflegemitarbeiter, indem der Arzt den Business Case auf der BPaaS Plattform delegiert.

14. Der nächste freie Pflegemitarbeiter sieht das auszuführende Activity Pattern. Der Pflegemitarbeiter teilt dem Patienten ein Zimmer auf der Intensivstation zu. Danach schliesst der Pflegemitarbeiter das Activity Pattern und den Business Case ab.

Der obenstehende Ablauf zeigt auf, dass der Arzt verantwortlich ist, die richtigen Entscheidungen zu treffen. So entscheidet der Arzt über die jeweils nächsten Behandlungsschritte und hält die Erkenntnisse fest. Der Arzt kann jederzeit die Verantwortlichkeit für einen Patienten an einen nächsten Arzt übergeben. Dieser muss in der Lage sein, alle bereits getätigten Schritte sowie Entscheidungen einzusehen und muss auf sämtliche bereits erfassten Informationen und Daten zugreifen können. Über die BPaaS Plattform

ist ersichtlich, welche Abklärungen und Diagnosen für welche Patienten bereits getätigt wurden und in welchem Status sich die Behandlung des Patienten befindet.

Folgende Anforderungen an den Ad Hoc Prozess und die Activity Patterns leitet der Autor aus obenstehendem Ablauf ab (siehe Tabelle 4.1).

Anforderung
Es muss jederzeit ein neuer Ad Hoc Prozess gestartet werden können.
Es muss eines oder mehrere Activity Patterns gewählt werden können, die auszuführen sind (Parallelität von Activity Patterns).
Pro Activity Pattern muss definiert werden können, welche Rolle oder Person das Activity Pattern auszuführen hat.
Pro Activity Pattern muss definiert werden können, ob die ausführende Rolle zur Abarbeitung des Activity Patterns einen weiteren Ad Hoc Prozess starten darf (Rekursion).
Pro Activity Pattern muss definiert werden können, ob auf das Resultat vom Activity Pattern gewartet werden soll (synchron) oder nicht (asynchron).
Der Ad Hoc Prozess wartet, bis alle synchronen Activity Patterns ausgeführt wurden.
Der Ad Hoc Prozess kehrt sofort zur Aktivität *Activity Patterns auswählen* zurück, wenn asynchrone Activity Patterns ausgewählt wurden.
Die Verantwortung für den Ad Hoc Prozess muss einer Person oder Rolle übergeben werden können.
Es müssen jederzeit die aktuellen Prozess Daten (z.B. Patientendaten, Laborwerte, etc.) sowie der Ad Hoc Prozess Ablauf (welche Schritte sind durch wen getätigt worden oder noch zu tätigen) angezeigt werden können.

Tabelle 4.1: Anforderungen an den Ad Hoc Prozess am Beispiel Behandlung eines Patienten

4.1.2 Umsetzung eines Projekts

Als zweites Beispiel soll die Umsetzung eines Projekts aufgegriffen werden. Sobald die Geschäftsleitung eines Unternehmens eine Entscheidung gefällt hat (z.B. für die Umsetzung von strategischen Massnahmen), sollen Projekte anlaufen, um diese Entscheidung umzusetzen [5]. Über die BPaaS Plattform kann eine mit dem Projekt beauftragte Person einen Ad Hoc Prozess starten und so das Projekt abwickeln. Grosse Projekte werden oft in Teilprojekte aufgeteilt [32], welche an weitere Personen delegiert werden können. Durch die Abwicklung über die BPaaS Plattform behält das Unternehmen die Übersicht über sämtliche Projekte und Teilprojekte und deren aktuellen Stand. Dabei kommen wie bei einem Projekt üblich zeitliche Faktoren in Spiel. Teilaufgaben im Projekt (Activity Patterns) sollen eine zeitliche Limite haben, bis wann der Projektleiter mit dem Abschluss vom Activity Pattern rechnen kann.

Der nachfolgende Abschnitt soll aufzeigen, wie ein Projektleiter die BPaaS Plattform einsetzen könnte:

1. Einem Projektleiter in einem Unternehmen wird ein zu erledigendes Projekt übergeben.
2. Dazu startet der Projektleiter einen Ad Hoc Prozess, welcher das umzusetzende Projekt repräsentiert.
3. Der Projektleiter studiert die Projektziele. Daraus resultieren zu erledigende Abklärungen.
4. Pro Abklärung eröffnet der Projektleiter ein Activity Pattern *Ausführen von Task* und dokumentiert den Sachverhalt, welchen es abzuklären gilt. Die Activity Patterns weist er den entsprechenden Experten zu. Zudem gibt der Projektleiter an, bis wann das Activity Pattern abgeschlossen werden soll. Der Ad Hoc Prozess soll zum Projektleiter zurückkommen, wenn diese Abklärungen gemacht sind. Der Projektleiter gibt an, dass ein neuer Ad Hoc Prozess aus den Activity Patterns gestartet werden kann.
5. Die verschiedenen Experten sehen ihre jeweiligen auszuführenden Activity Patterns auf der BPaaS Plattform. Dort wo eine Abklärung wiederum aus mehreren

Schritten besteht, startet der Experte einen neuen Ad Hoc Prozess, um die Abklärung durchzuführen.

6. Die Experten führen die Abklärungen durch und dokumentieren ihre Erkenntnisse. Dann schliessen sie die eröffneten Activity Patterns ab.

7. Sobald sämtliche vom Projektleiter erfassten Activity Patterns abgeschlossen sind, kommt der Ad Hoc Prozess zurück zum Projektleiter. Der Projektleiter muss nun wieder entscheiden können, was als nächstes zu tun ist (indem er die nächsten auszuführenden Activity Patterns wählen kann).

8. Die Abklärungen zeigen auf, wie das Projekt in Teilprojekte aufzuteilen ist.

9. Pro Teilprojekt eröffnet der Projektleiter ein Activity Pattern *Ausführen von Task* und beschreibt das Teilprojekt. Dabei soll der Teilprojektleiter wiederum das Activity Pattern (das Teilprojekt) als neuen Ad Hoc Prozess starten können, damit das Teilprojekt weiter unterteilt werden kann.

10. Das einzelne Teilprojekt kann als eigenständiges Projekt angesehen werden. Dabei kann mit dem Teilprojekt wieder bei Punkt 1 begonnen werden usw.

Aus dem oben aufgezeigten Ablauf leitet der Autor folgende Anforderungen ab (siehe Tabelle 4.2). Bereits in Tabelle 4.1 berücksichtigte Anforderungen werden nicht erneut aufgelistet.

Anforderung
Pro Activity Pattern muss definiert werden können, bis wann das Activity Pattern auszuführen ist.
Der Projektleiter benötigt eine Übersicht über alle zum Projekt dazugehörigen, laufenden Ad Hoc Prozesse, welche Activity Patterns welchen Personen und Rollen zugeordnet sind und in welchem Status diese sind. So behält der Projektleiter die Übersicht über das Projekt. Diese Übersicht muss vom Projektleiter ausserhalb vom Ad Hoc Prozess aufgerufen werden können. Des Weiteren müssen auch die Prozess Daten angezeigt werden können.

Tabelle 4.2: Anforderungen an den Ad Hoc Prozess am Beispiel Umsetzung eines Projekts

4.2 Prozess Modell

Ein BPMN 2.0 Prozess Modell, welches die erkannten Anforderungen abdeckt und von der Camunda Process Engine ausgeführt werden kann wird in Abbildung 4.1 dargestellt.

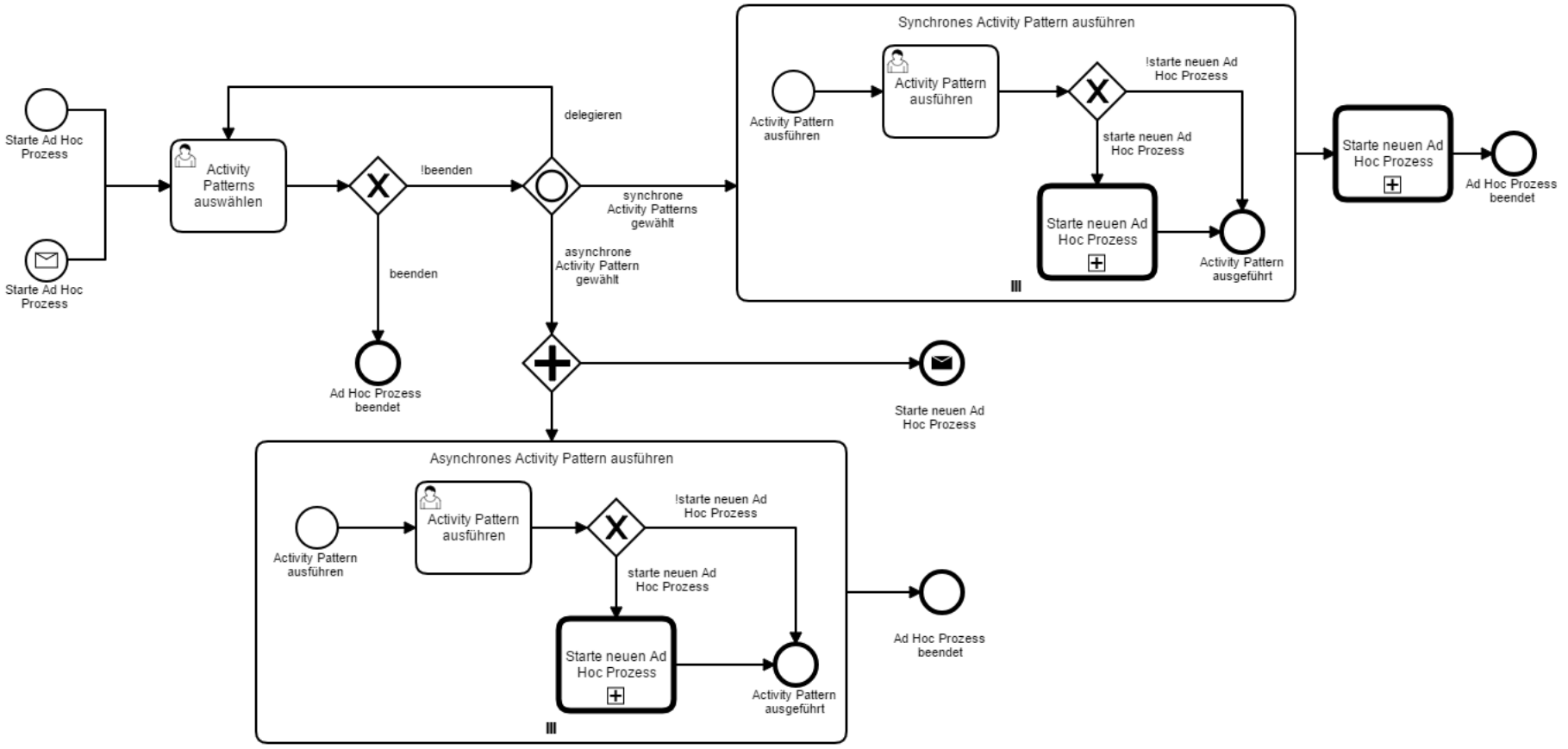


Abbildung 4.1: Ad Hoc Prozess

4.2.1 Erläuterungen

In diesem Abschnitt wird das Prozess Modell aus Abbildung 4.1 erläutert. Sämtliche BPMN Modellelemente beziehen sich auf [14] und [33].

Der Ad Hoc Prozess wird jeweils von einem Knowledge Worker gestartet (Blanko Startereignis *Starte Ad Hoc Prozess*). Die erste Aktivität *Activity Patterns auswählen* erlaubt das Auswählen der auszuführenden Activity Patterns. Bei diesem Schritt kann der Knowledge Worker auch entscheiden, dass der Ad Hoc Prozess beendet werden soll und ob die Verantwortlichkeit des Ad Hoc Prozesses an eine andere Rolle übergeben werden soll. Werden synchron auszuführende Activity Patterns gewählt, startet pro gewähltem synchronen Activity Pattern ein Subprozess Instanz *Synchrones Activity Pattern ausführen*. Dabei handelt es sich um einen Parallel Multi Instance Subprocess. Gleich verhält es sich auch für die gewählten asynchronen Activity Patterns.

Das den beiden Parallel Multi Instance Subprocesses vorangehende inklusive Gateway kann eines oder mehrere Pfade nehmen und sorgt dafür, dass je nach Auswahl vom Knowledge Worker in der Aktivität *Activity Patterns auswählen* Tokens auf den entsprechenden ausgehenden Branches abgegeben werden. Die Logik dafür ist direkt im Prozess Modell als Expression (vergleiche [34]) auf den Branches hinterlegt.

Die beiden Parallel Multi Instance Subprocesses *Synchrones Activity Pattern ausführen* sowie *Asynchrones Activity Pattern ausführen* verlaufen grundsätzlich gleich: Der Knowledge Worker, welchem das Activity Pattern zugewiesen ist, erhält ein entsprechender Eintrag in der Camunda Tasklist [35]. Der Knowledge Worker hat dann je nach Activity Pattern und je nach Vorgabe des Knowledge Workers, welcher das Activity Pattern erfasst hat, die Möglichkeit, ein neuer Ad Hoc Prozess zu starten (über die Call Activity *Starte neuen Ad Hoc Prozess*). Ist diese Möglichkeit nicht gegeben oder entscheidet sich der Knowledge Worker gegen das Starten eines neuen Ad Hoc Prozesses, dann führt der Knowledge Worker das Activity Pattern aus und der Subprozess endet mit dem Blanko Endereignis *Activity Pattern ausgeführt*.

Wird aufgrund eines Activity Patterns ein neuer Ad Hoc Prozess gestartet, dann passiert das wie oben erwähnt über eine Call Activity. Dabei wartet das Token, bis die über die Call Activity gestartete Prozess Instanz beendet ist. Sobald die über die Call

Activity gestartete Prozess Instanz beendet ist, fährt das Token weiter. So sind beliebige Verschachtelungen von neu gestarteten Ad Hoc Prozessen möglich.

Im Falle von synchron auszuführenden Activity Patterns wird über die dem Subprozess nachgelagerte Call Activity eine neue Ad Hoc Prozess Instanz gestartet. Bei den asynchronen Activity Patterns wird durch das parallele Gateway (UND Gateway) immer gleich ein neuer Ad Hoc Prozess gestartet. So erhält der Knowledge Worker die Möglichkeit, weitere Activity Patterns zu erfassen, ohne auf das Ende der asynchronen Activity Patterns zu warten. Das Token, welches im Subprozess *Asynchrones Activity Pattern ausführen* wartet, endet nachdem alle asynchronen Activity Patterns ausgeführt wurden (respektive den daraus neu gestarteten Ad Hoc Prozessen).

4.2.2 Laufzeit Modell

Dieser Abschnitt zeigt anhand des in Abschnitt 4.1 erläuterten Ablaufs *Behandlung eines Patienten* auf, wie sich der Ad Hoc Prozess zur Laufzeit entwickelt (vergleiche Abbildung 4.2).

Die verwendeten Symbole halten sich nicht an eine Notation, sind aber an BPMN angelehnt. Der Ablauf beschreibt einen Business Case (Behandlung eines Patienten), welcher von oben nach unten verläuft. Der Ad Hoc Prozess startet mit dem Kreis Symbol. Die Rechtecke repräsentieren Aktivitäten. Die drei Buchstaben sind Abkürzungen, welche aussagen, um welche Aktivität es sich handelt: Activity Patterns wählen (APW), Editieren von Prozess Daten (EPD), Ausführen von Task (AVT), Entscheidung (ENT). Rechtecke, welche Activity Patterns darstellen, sind mit einem Symbol ergänzt. Ein synchron ablaufendes Activity Pattern wird mit einer Sanduhr dargestellt (es wird auf das Resultat gewartet), asynchron ablaufende Activity Patterns werden mit zu einem Kreis geformten Pfeilen dargestellt (es wird nicht auf das Resultat gewartet). Bei der Aktivität *Activity Patterns auswählen* hat der Benutzer die Möglichkeit, den Ad Hoc Prozess abschliessen. Macht der Benutzer davon Gebrauch, wird dies durch einen doppelten Kreis dargestellt.

Bereits dieses relativ einfache Szenario repräsentiert sich als komplexes Gebilde. Bedenkt man, dass jedes einzelne Activity Pattern durch das Starten eines neues Ad Hoc Prozesses (Rekursion) sich wieder in einem solchen Gebilde manifestieren kann, erkennt

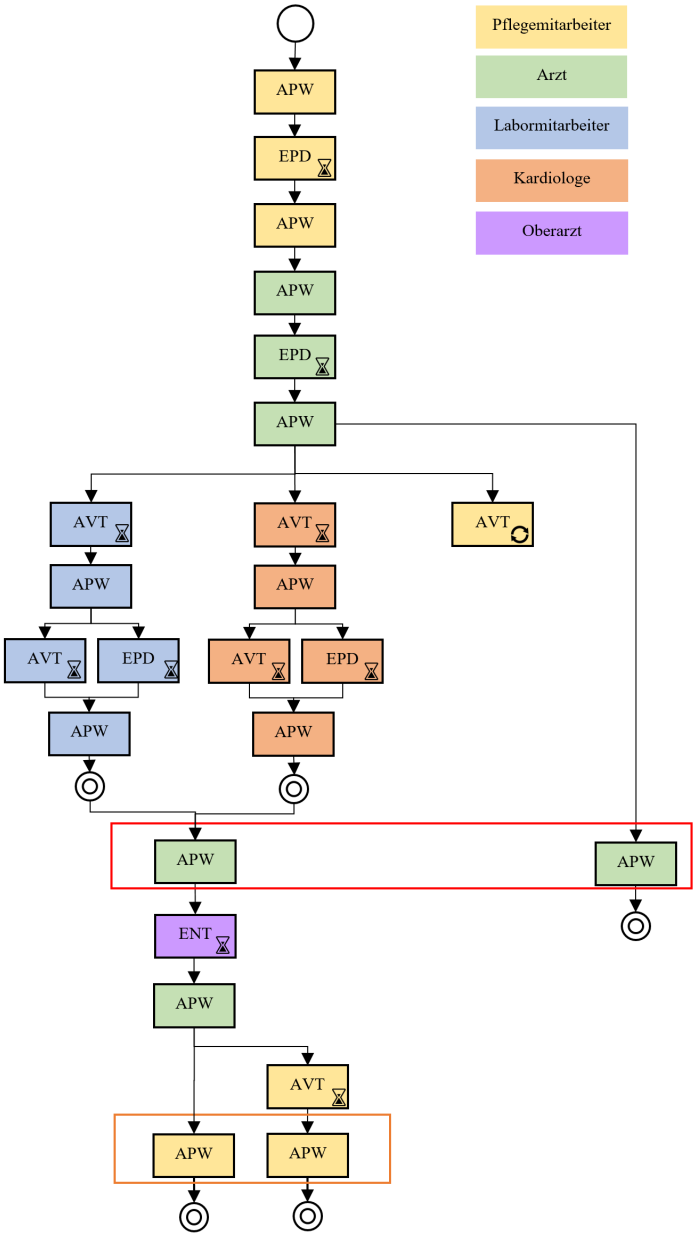


Abbildung 4.2: Ad Hoc Prozess zur Laufzeit

man die mögliche Komplexität eines solchen Ablaufs. Entsprechend stellt die Darstellung und Nachvollziehbarkeit vom Prozess Ablauf auf der Benutzeroberfläche der BPaaS Plattform eine besondere Herausforderung dar (vergleiche Kapitel 5).

Die Abbildung 4.2 zeigt auch auf, dass Aktivitäten, je nach zeitlichem Ablauf, in mehrfacher Ausführung in der Tasklist einer Rolle erscheinen können. Diese Fälle sind mit orangen und roten Rechtecken markiert.

4.3 Implementierung

Die zentralen Artefakte von der Implementierung sind das in BPMN modellierte Ad Hoc Prozess Modell und die in Java implementierten Listener Klassen. Camunda erlaubt die Ausführung von Java Code, wenn gewisse Ereignisse in einer Prozess Instanz auftreten (zum Beispiel Starten und Beenden einer Prozess Instanz, Beginn und Abschluss eines User Tasks, usw.) [36]. Durch diese Listener werden entsprechende Daten Modell Modifikationen vorgenommen sowie Prozess Variablen modifiziert und so der Ad Hoc Prozess Ablauf gesteuert.

4.3.1 Daten Modell

Dem Ad Hoc Prozess Modell liegt ein Daten Modell zu Grunde. Dieses Daten Modell wird zur Laufzeit erstellt und modifiziert. Auf das Daten Modell wird über Expressions, welche direkt im Prozess Modell hinterlegt sind, zugegriffen (vergleiche [34]). In nachfolgender Abbildung 4.3 ist das Daten Modell als UML Klassendiagramm dargestellt. Auf die Auflistung von getter und setter Methoden wird verzichtet:

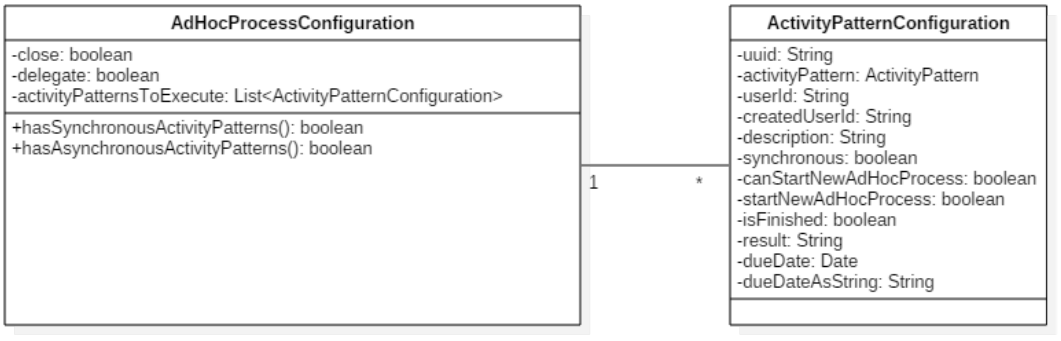


Abbildung 4.3: Daten Modell vom Ad Hoc Prozess

Pro Ad Hoc Prozess, welcher startet, wird in eine Prozess Variable [37] `adHocProcessConfiguration` ein Objekt vom Typ `AdHocProcessConfiguration` gespeichert. Darüber wird der Ablauf vom Ad Hoc Prozess gesteuert und Informationen über den Ad Hoc Prozess abgelegt.

Das Objekt besitzt folgende Eigenschaften:

- *close*: Ob der Ad Hoc Prozess abzuschliessen ist.

- *delegate*: Ob der Ad Hoc Prozess zu delegieren ist.
- *activityPatternsToExecute*: Die im Ad Hoc Prozess auszuführenden Activity Patterns.

Pro im Schritt *Activity Pattern auswählen* gewählten Activity Pattern wird ein Objekt vom Typ `ActivityPatternConfiguration` angelegt und in die Collection `activityPatternsToExecute` von der `AdHocProcessConfiguration` abgelegt. Das Objekt `AdHocProcessConfiguration` besitzt Methoden, über welche auf die synchron und asynchron auszuführenden Activity Patterns zugegriffen werden kann. Direkt im Prozess Modell hinterlegte Expressions [34] nutzen diese Methoden, um auf die Activity Patterns zuzugreifen.

Das Objekt besitzt folgende Eigenschaften:

- *uuid*: Eine eindeutige Identifikation der Activity Pattern Instanz in Form eines Universal Unique Identifiers [38]. Über die UUID kann ein Ad Hoc Prozess, welcher durch ein Activity Pattern gestartet wurde, dem Activity Pattern zugeordnet werden.
- *activityPattern*: Der Typ des auszuführenden Activity Patterns.
- *userId*: Der verantwortliche Benutzer.
- *createdUserId*: Der Benutzer, welcher das Activity Pattern erfasst hat.
- *description*: Die textuelle Beschreibung davon, was zu tun ist.
- *synchronous*: Ob das Activity Pattern synchron oder asynchron ausgeführt werden soll.
- *canStartNewAdHocProcess*: Ob ein neuer Ad Hoc Prozess gestartet werden kann.
- *startNewAdHocProcess*: Ob der Knowledge Worker, welchem das Activity Pattern zugeteilt ist, ein neuer Ad Hoc Prozess gestartet hat.
- *isFinished*: Ob das Activity Pattern ausgeführt wurde.
- *result*: Eine textuelle Beschreibung vom Resultat vom Activity Pattern (z.B. Task ausgeführt oder Entscheidung akzeptiert).

- *dueDate*: Das Datum, bis wann das Activity Pattern auszuführen ist.
- *dueDateAsString*: Text Repräsentation vom Due Date für die Anzeige auf der Benutzeroberfläche.

Beim Start eines neuen Ad Hoc Prozesses wird eine neue Business Case Id generiert (realisiert als UUID). Diese Business Case Id wird allen aus diesem Ad Hoc Prozess aufgerufenen weiteren Ad Hoc Prozesse als Prozess Variable weitergegeben. So können zusammengehörige Prozess Instanzen identifiziert werden.

Wie im Daten Modell ersichtlich, besitzt jedes erfasste Activity Pattern eine eindeutige Id (ebenfalls realisiert als UUID). Wird aus einem Activity Pattern ein neuer Ad Hoc Prozess gestartet, so wird diese UUID dem neu gestarteten Ad Hoc Prozess als Prozess Variable mitgegeben. Damit kann festgestellt werden, dass erstens ein Ad Hoc Prozess Instanz aus einem Activity Pattern gestartet wurde und zweitens aus welchem Activity Pattern die Ad Hoc Prozess Instanz entstanden ist.

4.4 Fazit

Die Entwicklung vom Prozess Modell erfolgte gemäss Forschungsmethode in mehreren Zyklen (in mehreren Build- und Evaluate Zyklen, vergleiche Abschnitt 3.1.3). Die Demonstration (vergleiche Abschnitt 3.2.4) erfolgte, indem das entwickelte BPMN Prozess Modell in ein Java Projekt integriert und als WAR Archiv auf die Camunda Plattform deployed wurde. So wurden kleine Veränderungen direkt auf der Camunda Plattform getestet und das Verhalten konnte über das Camunda Frontend durchgespielt und überprüft werden.

Eine erste Version vom Prozess Modell arbeitete mit Sequenzflüssen statt mit Call Activities. Aus Subprozessen können keine Sequenzflüsse ein- resp. ausgehen. Daran wurde bei der ersten Skizzierung von möglichen Lösungsansätzen, welche auf Papier erfolgte, nicht gedacht. Deshalb wurden Call Activities eingeführt. Weiter würde die Verwendung von Sequenzflüssen dazu führen, dass in der gleichen Prozess Instanz gleichzeitig mehrere Tokens auf der Aktivität *Activity Patterns auswählen* existieren könnten. Die Benutzeroberfläche nutzt Camunda Controls, welche an Camunda Prozess Variablen gebunden sind. Bei gleichzeitiger Ausführung der Aktivität würde es zu Problemen kommen, weil

dieselben Prozess Variablen unter Umständen gleichzeitig modifiziert würden [39]. Auch deshalb musste dieser Ansatz verworfen und für jeden neuen Durchlauf eine neue Prozess Instanz gestartet werden.

Das in Abbildung 4.1 aufgezeigte Prozess Modell ist eine Arbeitshypothese des Autors, aus welcher sich folgende Konsequenzen, offene Fragen und Verbesserungsmöglichkeiten ergeben:

- Weil über die Call Activities jeweils neue Prozess Instanzen gestartet werden, besteht ein Business Case aus mehreren Ad Hoc Prozess Instanzen. Das macht die Nachvollziehung vom Ablauf eines Business Cases aufwändiger und komplexer.
- Dadurch, dass erstens mehrere Prozess Instanzen vorhanden sein können und zweitens innerhalb von einer Prozess Instanz mehrere Activity Patterns vom Typ *Editieren von Prozess Daten* gleichzeitig ausgeführt werden können, müssen die Prozess Daten zwischen mehreren Prozess Instanzen geteilt werden. Das führt dazu, dass die Verwaltung der Prozess Daten komplexer wird (vergleiche Kapitel 7).
- Durch die Modellierung mit Call Activities bleibt bei der Ausführung von synchronen Activity Patterns die Prozess Instanz am Laufen, welche die Call Activity aufruft und das Token wartet dort, bis die gestartete Prozess Instanz beendet ist. Dadurch kann es zu vielen laufenden Prozess Instanzen kommen, was wiederum Arbeitsspeicher in Anspruch nimmt.
- Die Aktivität *Activity Pattern ausführen* ist als User Task modelliert. Grund dafür ist, dass im Prototyp lediglich die Activity Patterns *Ausführen von Task*, *Entscheidung* sowie *Editieren von Prozess Daten* berücksichtigt sind. Kommen weitere Activity Patterns dazu (wie z.B. *Event absetzen* oder *Event empfangen*) so müssen Activity Patterns ausgeführt werden, die in BPMN als Service Tasks modelliert würden. Dazu ist eine Erweiterung vom Prozess Modell notwendig.
- Es fragt sich, ob bei auszuführenden asynchronen Activity Patterns in jedem Fall über das parallele Gateway ein neuer Ad Hoc Prozess gestartet werden soll. Sind gleichzeitig noch synchrone Activity Patterns gewählt worden, so erhält der Knowledge Worker den Ad Hoc Prozess nach Beendigung der synchronen Activity Patterns zurück. Nach Beendigung der synchronen Activity Patterns besitzt

der Knowledge Worker zwei Activities vom Typ *Activity Patterns auswählen* (vergleiche dazu die rote Markierung in Abbildung 4.2). Das könnte den Knowledge Worker verwirren.

- Weiter kommt es beim Delegieren eines Ad Hoc Prozesses an eine Rolle unter gleichzeitiger Erfassung von einem synchronen Activity Pattern für die gleiche Rolle zu der Situation, dass die Rolle nach Abschluss vom Activity Pattern zwei Aktivitäten vom Typ *Activity Patterns auswählen* in der Task Liste sieht (vergleiche dazu die orange Markierung in Abbildung 4.2). Auch das könnte den Knowledge Worker verwirren.

- Es fragt sich, ob der Ad Hoc Prozess abgebrochen werden können soll. Sprich Tokens in allen Ad Hoc Prozess Instanzen, welche zum Business Case gehören, werden konsumiert.

- Es kann zur Situation kommen, dass ein Knowledge Worker ein Activity Pattern ausführt und abschliesst und dann merkt, dass noch weitere Schritte notwendig gewesen wären. In dieser Situation müsste der Knowledge Worker auf eine Person zugehen, welche Activity Patterns erfassen kann.

- Das Modell erlaubt es, dass mehrere parallele Ad Hoc Prozesse (welche alle zum gleichen Business Case gehören) am Laufen sind. So kann es dazu kommen, dass gleiche auszuführende Tätigkeiten mehrfach durch verschiedene Knowledge Worker erfasst werden. Es ist durch das auch schwierig nachzuvollziehen, welcher Knowledge Worker für einen Business Case im Lead ist.

- Es fragt sich, ob der Knowledge Worker, welcher den Ad Hoc Prozess initial gestartet hat, den Business Case zurückerhalten soll, bevor die letzte Prozess Instanz vom Business Case endet. Oder eventuell muss der Knowledge Worker, welcher die letzte Prozess Instanz von einem Business Case beendet, darauf aufmerksam gemacht werden, dass mit diesem Schritt der Business Case abgeschlossen wird.

- Es fragt sich, anhand von welchen Kriterien ein Knowledge Worker entscheidet, ob aus einem Activity Pattern ein Ad Hoc Prozess gestartet werden darf (Rekursion). Diese Funktionalität wird im Prototyp nur für *Ausführen von Task* angeboten. Eventuell sollten alle Activity Patterns von diesem Typ einen neuen Ad Hoc Prozess starten können.

Die Vorgehensweise zur Erarbeitung von Anforderungen an den Ad Hoc Prozess muss in folgenden Punkten kritisch hinterfragt werden:

- Das Erarbeiten von Anforderungen an den Ad Hoc Prozess sowie das Schliessen von den erkannten Anforderungen auf das Prozess Modell im Abschnitt 4.2 basiert auf ein paar wenigen Quellen. Hier hätten durchaus auch noch mehr Quellen eingearbeitet werden können und so das Prozess Modell noch weiter verfeinert werden können. Noch besser wären Beobachtungen vorgenommen worden, um daraus eine Arbeitshypothese zu bilden. Diese Arbeitshypothese hätte dann mittels Expertenbefragungen validiert werden können.

- Die abgeleiteten Anforderungen beziehen sich nur auf den Ad Hoc Prozess, aber nicht auf die einzelnen Activity Patterns. So fehlt eine Erhebung der Anforderungen bezüglich der einzelnen Activity Patterns.

- Mit dem gewählten Ansatz zur Erkennung von Anforderungen war es nicht möglich, nicht funktionale Anforderungen (z.B. Performance) abzuleiten.

- Ein anderer Ansatz für die Unterstützung von unstrukturierten Business Prozessen ist das Case Management [3]. Dazu gibt es ebenfalls eine Notation mit der Bezeichnung Case Management Model and Notation (CMMN) [40]. Camunda unterstützt die Ausführung von CMMN Modellen [41]. Es fragt sich, ob eine Modellierung der Problemstellung auch mittels CMMN möglich gewesen wäre und welche Vor- und Nachteile sich daraus für den Anwender als auch für die Umsetzung in Camunda ergeben hätten.

Kapitel 5

Benutzeroberfläche

Dieses Kapitel beschreibt die Benutzeroberflächen, welche die BPaaS Plattform zur Verfügung stellt.

5.1 Design

Die auf einem Teil der nachfolgenden Mockups ersichtlichen Buttons *Save* und *Complete* werden durch Camunda zur Verfügung gestellt.

5.1.1 Ad Hoc Prozess starten

Beim Starten eines Ad Hoc Prozesses soll eine optionale Beschreibung eingegeben werden können, welche etwas über den Ad Hoc Prozess aussagt. Ein Mockup dazu ist in Abbildung 5.1 dargestellt.

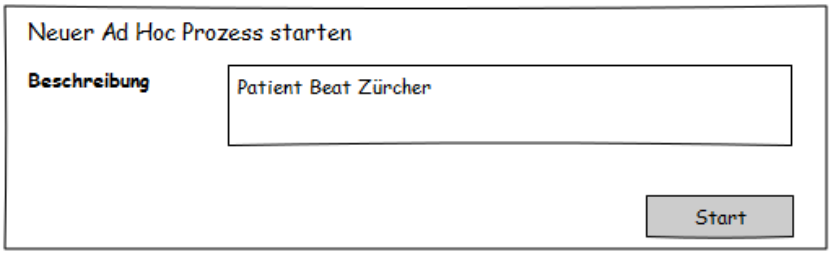


Abbildung 5.1: Mockup für Start von Ad Hoc Prozess

5.1.2 Activity Patterns auswählen

Die Aktivität *Activity Patterns auswählen* erlaubt es dem ausführenden Knowledge Worker, eines oder mehrere Activity Patterns auszuwählen, welche als nächstes auszuführen sind. Der Knowledge Worker kann die Verantwortlichkeit für das Activity Pattern festlegen. Weil die Activity Patterns generisch sind, muss durch den Knowledge Worker zu jedem gewählten Activity Pattern eine Beschreibung, respektive Arbeitsanweisungen angegeben werden. Zu diesem Zeitpunkt kann der Knowledge Worker auch entscheiden, dass keine weiteren Activity Patterns notwendig sind und der Ad Hoc Prozess abgeschlossen werden soll. Der Knowledge Worker kann die Verantwortlichkeit für den Ad Hoc Prozess auch an eine weitere Person oder Rolle übertragen. Ein Mockup zu dieser Aktivität ist in Abbildung 5.2 dargestellt.

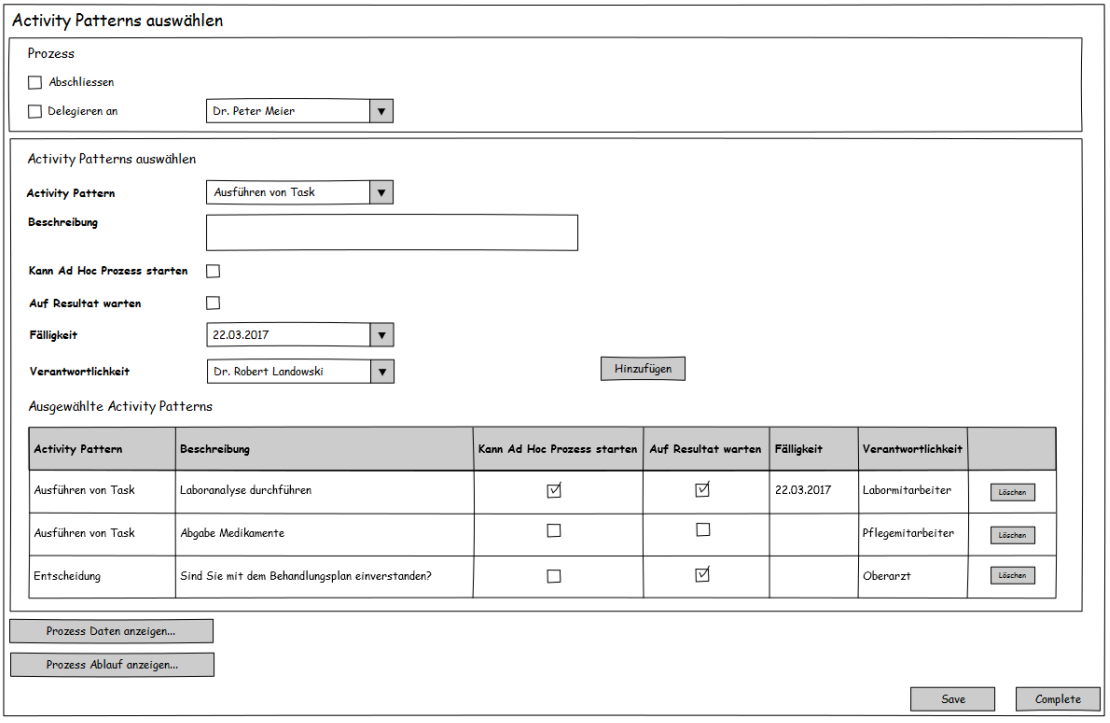


Abbildung 5.2: Mockup für Aktivität Activity Pattern auswählen

5.1.3 Activity Pattern ausführen

Die Aktivität *Activity Patterns ausführen* zeigt abhängig vom gewählten Activity Pattern ein entsprechendes User Interface an. Jedes zur Ausführung erfasste Activity Pattern äussert sich auf der BPaaS Plattform durch eine Maske, über welche das Activity

Pattern abgewickelt wird. In den nachfolgenden Abschnitten werden diese Masken erarbeitet.

5.1.4 Editieren von Prozess Daten

Das User Interface muss es erlauben, Prozess Daten zu erfassen und bestehende Prozess Daten zu bearbeiten oder zu löschen. Dabei kann es sich um Daten in Form von Text, um Bilder oder um Dokumente handeln (vergleiche die erläuterten Abläufe in Kapitel 4.1). Es wird vom Autor als Vereinfachung angenommen, dass jede Person oder Rolle jede Information einsehen und bearbeiten darf. Ein Mockup zu diesem Activity Pattern ist in Abbildung 5.3 dargestellt.

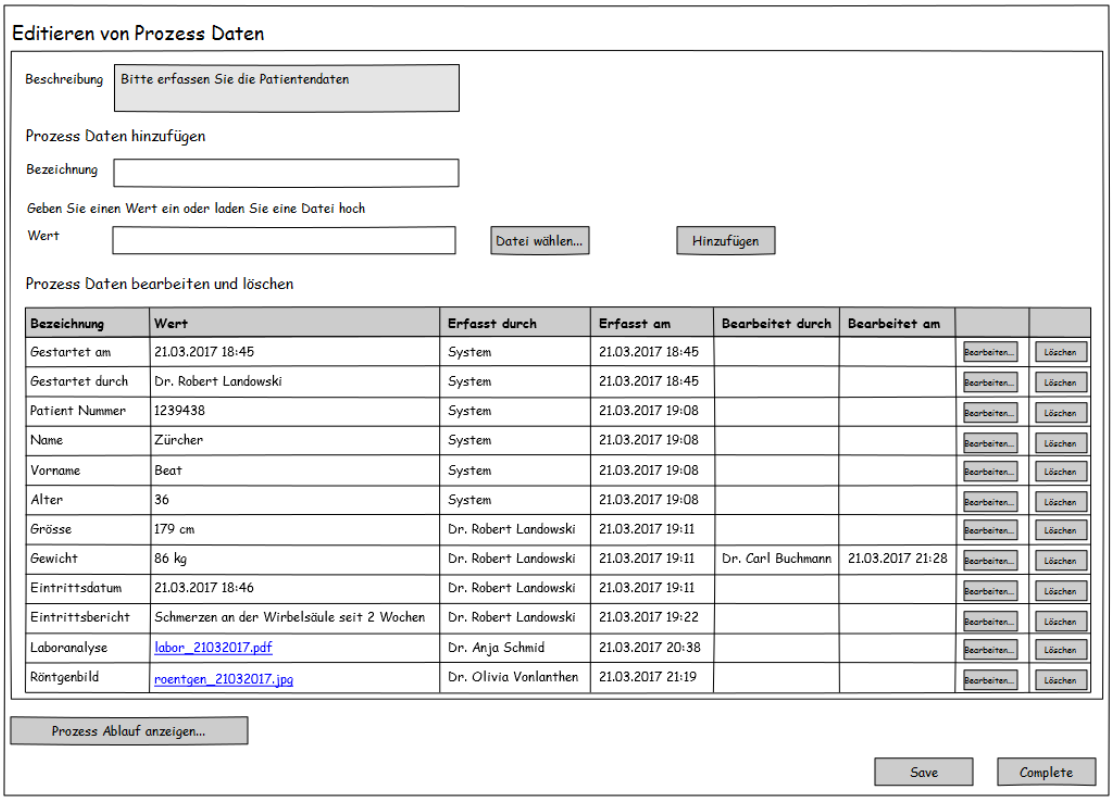


Abbildung 5.3: Mockup für Activity Pattern Editieren von Prozess Daten

5.1.5 Entscheidung

Das User Interface für das Activity Pattern *Entscheidung* hält sich relativ simpel. Es muss eine Entscheidung in Form von der Akzeptanz oder der Ablehnung eines Sachverhalts herbeigeführt werden. Dabei wird angenommen, dass der Knowledge Worker,

welcher das Activity Pattern erfasst, die zu treffende Entscheidung aufgrund von einer textuellen Beschreibung erfasst. Ein Mockup zu diesem Activity Pattern ist in Abbildung 5.4 dargestellt.

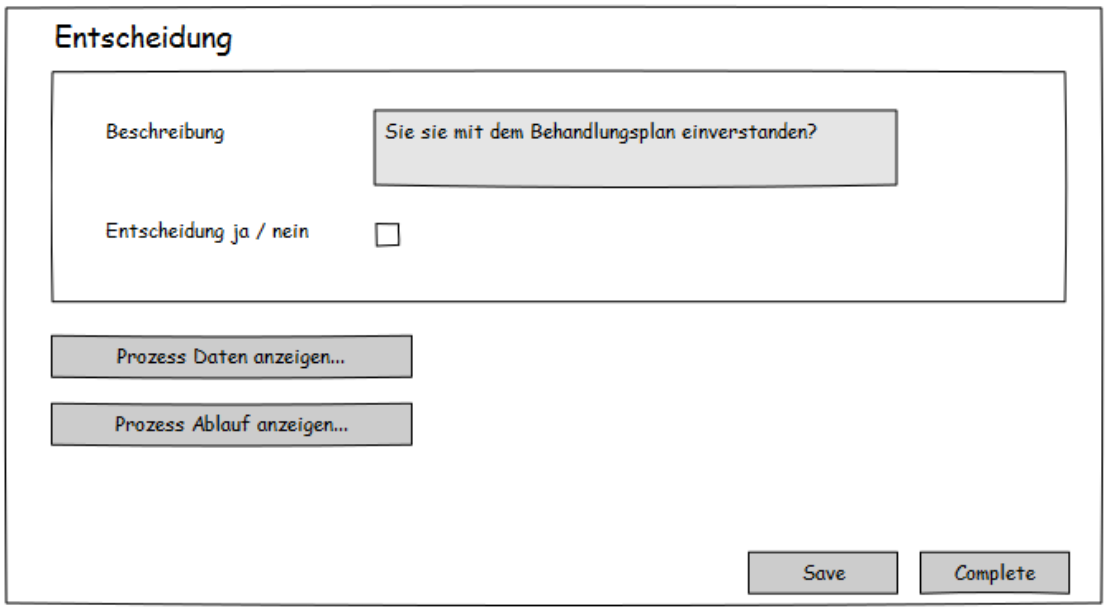


Abbildung 5.4: Mockup für Activity Pattern Entscheidung

5.1.6 Ausführen von Task

Bei diesem Activity Pattern geht es darum, dass eine Person oder Rolle eine Tätigkeit ausführt, welche sich ausserhalb von der Process Engine abspielt. Beispiele dazu sind das Abnehmen von Blut, die Abgabe von Medikamenten etc. Was zu tun ist, wird in Form einer textuellen Beschreibung durch den Knowledge Worker angegeben, während das Activity Pattern erfasst wird. Sobald die Tätigkeit ausgeführt wurde, kann dies über das Setzen des Häkchens auf dem User Interface bestätigt werden. Ein Mockup zu diesem Activity Pattern ist in Abbildung 5.5 dargestellt.

5.1.7 Prozess Daten und Prozess Ablauf anzeigen

Von jedem Activity Pattern, respektive von der Aktivität *Activity Patterns auswählen* aus, müssen sowohl die aktuell erfassten Prozess Daten sowie der Prozess Ablauf vom Business Case (in Form bereits ausgeführter und noch auszuführenden Activity Patterns) angezeigt werden können. Es wird davon ausgegangen, dass zur Entscheidung

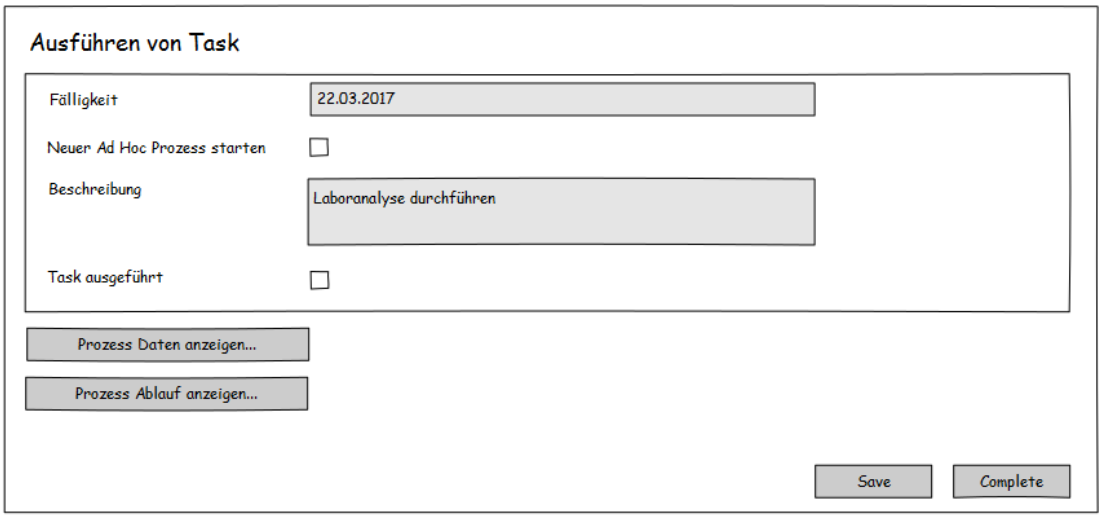


Abbildung 5.5: Mockup für Activity Pattern Ausführen von Task

über die nächsten Schritte und zur Ausführung von einem Activity Pattern diese Informationen wertvoll sind. Ein Mockup von den User Interfaces, welche diese Informationen darstellen, findet man in den Abbildungen 5.6 und 5.7.

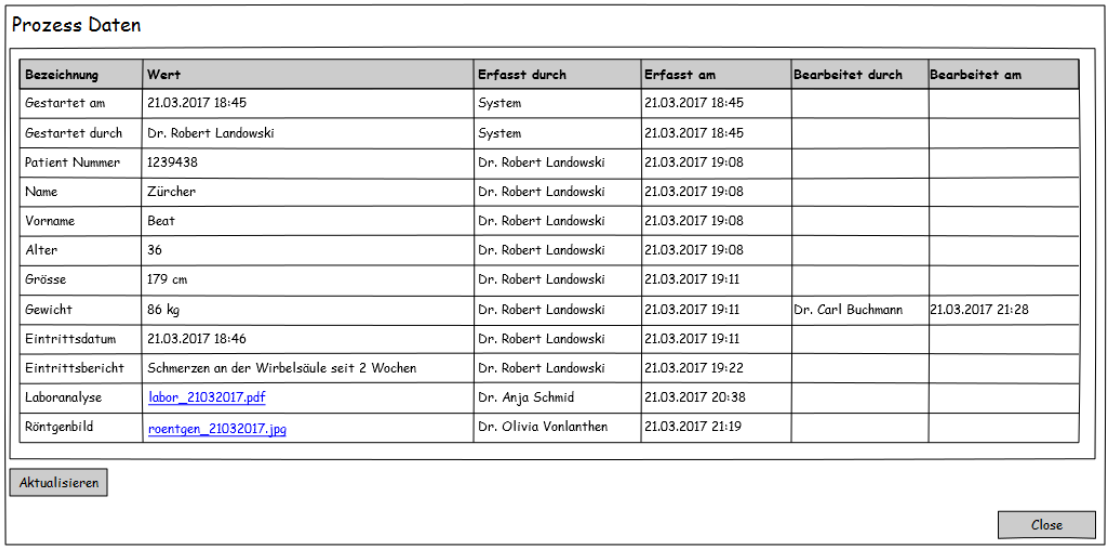


Abbildung 5.6: Mockup für das Anzeigen von den Prozess Daten

Weil ein Ad Hoc Prozess aus mehreren Durchläufen besteht (Activity Patterns auswählen und ausführen), sind diese Durchläufe im Prozess Ablauf als Teilprozesse auf der Benutzeroberfläche kenntlich gemacht (vergleiche Abbildung 5.7).

Des Weiteren sollen die Prozess Daten und der Prozess Ablauf (wie in den Abbildungen 5.6 und 5.7 aufgezeigt) auch aus einer Übersicht über alle Ad Hoc Prozesse angezeigt werden können. Eine solche Übersicht wird in Abbildung 5.8 dargestellt. Klickt der

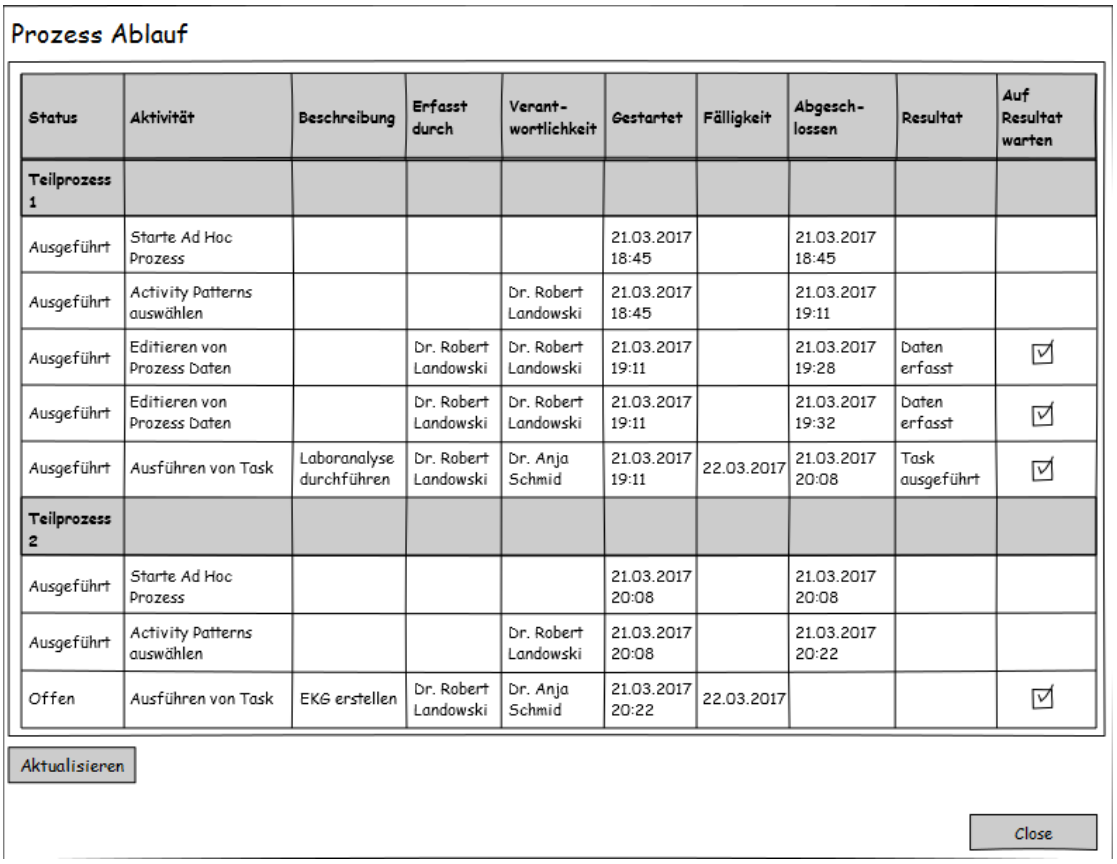


Abbildung 5.7: Mockup für das Anzeigen vom Prozess Ablauf

Benutzer auf *Prozess Eigenschaften anzeigen*, so werden die Prozess Daten und der Prozess Ablauf des entsprechenden Ad Hoc Prozesses angezeigt.

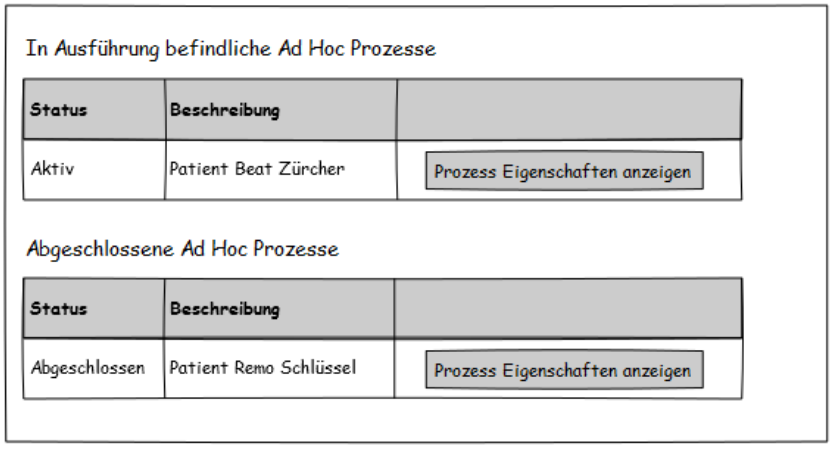


Abbildung 5.8: Mockup für Ad Hoc Prozess Übersicht

5.2 Implementierung

Die Benutzeroberflächen von der BPaaS Plattform äussern sich im Prototyp durch User Tasks, welche mittels Camunda Embedded Task Forms umgesetzt werden [42].

Grundsätzlich werden durch das Embedded Tasks Forms Framework Input Felder angeboten, welche an Camunda Prozess Variablen [37] gebunden werden können. Über diese Prozess Variablen kann im Listener (vergleiche Kapitel 4.3), welcher auf den Abschluss vom User Task wartet, auf die durch den Benutzer eingegebenen Werte zugegriffen werden.

Es hat sich gezeigt, dass die Anforderungen an die Benutzeroberflächen nicht ohne Weiteres, mittels den durch das Embedded Tasks Forms Framework angebotenen Mechanismen, sich umsetzen lassen. Die Benutzeroberfläche für die Auswahl der Activity Patterns setzt voraus, dass eine dynamische Anzahl an Activity Patterns erfasst werden können. Entsprechend kann dies nicht mit vordefinierten Input Felder realisiert werden. Dazu mussten via Java Script dynamisch Prozess Variablen erstellt werden, welche die durch den Benutzer erfassten Activity Pattern Informationen abbilden. Das gleiche gilt für die Benutzeroberfläche *Editieren von Prozess Daten*. So entstand teilweise viel clientseitiger Java Script Code.

Damit ein Knowledge Worker sich jederzeit eine Übersicht über die laufenden sowie abgeschlossenen Ad Hoc Prozesse verschaffen kann, wurde ein Camunda Cockpit Plugin [43] implementiert, über welches von jedem Ad Hoc Prozess der Prozess Ablauf sowie die Prozess Daten angezeigt werden können (vergleiche Abbildung 5.8). Für die Anzeige vom Prozess Ablauf und den Prozess Daten wurde mittels Java Script auf die Camunda REST APIs zugegriffen und die entsprechenden Informationen zusammengetragen und aufbereitet. Durch den Ansatz mit dem clientseitigen REST API Zugriff kann auch der Anforderung entsprochen werden, dass die Informationen auf der Benutzeroberfläche aktualisiert werden können.

Zu Beginn wurde ein Ansatz verfolgt, welcher die Erweiterung des Camunda REST APIs vorsah, damit das Zusammentragen der Informationen zum Prozess Ablauf und zu den Prozess Daten auf der Serverseite passieren konnte. Dazu ist aber wenig Dokumentation vorhanden, deshalb wurde der Ansatz nicht weiterverfolgt [42].

Beim Zugriff auf das REST API muss der Process Engine Name als URL (Uniform Resource Locator) Parameter mitgegeben werden (derjenige des Mandanten). Die Schritte zur Ermittlung des Process Engine Namens mussten im Camunda Forum erfragt werden [44].

Teilweise sind Informationen, welche auf der der Benutzeroberfläche darzustellen sind, in Java Objekten abgelegt. Damit diese Java Objekte sehr einfach auf dem User Interface geladen und angezeigt werden können, werden sämtliche Java Objekte mittels JSON [45, 46] serialisiert gespeichert. Über eine entsprechende Einstellung in der Process Engines Konfigurationsdatei `bpm-platform.xml` kann Camunda angewiesen werden, dass die Prozess Variablen generell im JSON Format abgelegt werden sollen:

```
<property name="defaultSerializationFormat">application/json</property>
```

5.3 Fazit

Die User Interface Mockups sowie die Implementierung müssen in folgenden Punkten kritisch hinterfragt werden:

- Die in diesem Kapitel vorgestellten User Interface Mockups wurden nicht validiert, sprich einem Experten zur Durchsicht vorgelegt.
- Bei der Implementierung entstand viel Java Script Code, welcher zum Teil auch doppelt in den verschiedenen Benutzeroberflächen geführt wird und schwierig automatisiert zu testen ist.
- Der Prozess Ablauf kann sehr komplex und lang werden. Das Problem wird in Kapitel 4.2.2 aufgezeigt. Entsprechend braucht es diesbezüglich eine intuitive und benutzerfreundliche Darstellung der Informationen, welche auch mobilen Geräten gerecht wird. Das ist nicht ein triviales Problem, welches in weiteren Forschungsarbeiten gelöst werden muss (vergleiche Kapitel 9).
- Die Tabellen, welche den Prozess Ablauf sowie die Prozess Daten darstellen, werden zum Teil so breit, dass auf mobilen Geräten gescrollt werden muss. Dies muss bei einer Überarbeitung der Benutzeroberflächen berücksichtigt werden.

- Camunda zeigt oben rechts auf der Benutzeroberfläche eine Dropdown Box mit allen verfügbaren Process Engines an. Die Mandanten auf der BPaaS Plattform sollten nichts voneinander mitbekommen. Entsprechend sollte eine solche Dropdown Box nicht sichtbar sein. Diesbezüglich wurde eine Frage im Camunda Forum gestellt [47]. Die Dropdown Box liesse sich mit CSS Manipulationen ausblenden.

- Die User Interface Masken, welche vom Ad Hoc Prozess angeboten werden, besitzen keine oder nur sehr rudimentäre clientseitige Validierungen der Benutzereingaben.

- User Interface Texte und Fehlermeldungen sind hartcodiert auf Deutsch im Source Code hinterlegt. Somit können andere Sprachen im Moment nicht auf einfache Art und Weise unterstützt werden.

- Das REST API von Camunda wurde im Prototyp nicht mittels Authentifizierung geschützt [48]. So können mit entsprechenden Kenntnissen auf die Daten aller Mandanten zugegriffen werden. Der gegenseitige Datenzugriff von verschiedenen Mandanten über das REST API ist somit möglich.

- Mit der vorgeschlagenen Lösung ist für den Knowledge Worker ein Mehraufwand zu erwarten, weil nur generische Activity Patterns unterstützt werden und keine Aktivitäten vorausgewählt werden können. Eine Parametrierung von den Activity Patterns auf die Business Domain wäre sinnvoll. Im erwähnten Beispiel (Kapitel 4) vom Arzt in der Notfallaufnahme sollte der Arzt zum Beispiel aus einer Dropdownbox auswählen können, welche Aufgabe im Activity Pattern *Ausführen von Task* auszuführen ist, statt dies von Hand eingeben zu müssen. Gerade in einer Notsituation spart das Zeit.

- Der Prototyp bietet für das Activity Pattern *Editieren von Prozess Daten* nicht die komplette Funktionalität. Prozess Daten können nicht bearbeitet werden sondern nur erfasst und gelöscht werden. Zudem können Dateien nicht als Prozess Daten erfasst werden.

- Die von Camunda angebotene *Save* Funktionalität wird nicht unterstützt. Werden Activity Patterns in der Aktivität *Activity Patterns auswählen* erfasst, auf *Save* gedrückt und dann die Seite neu geladen, so sind die erfassten Activity Patterns nicht mehr sichtbar. Dazu ist ein Refactoring notwendig, welches im Code vermerkt ist.

Kapitel 6

On-Boarding

Dieses Kapitel beschreibt den On-Boarding Prozess. Dabei geht es um das automatisierte Erstellen einer Umgebung für ein Unternehmen, welches die BPaaS Plattform nutzen möchte.

6.1 Lösungsansätze

Je nach gewähltem Multi-Tenancy Ansatz verläuft das On-Boarding anders. Entsprechend muss als erstes der umzusetzende Multi-Tenancy Ansatz definiert werden. Die Wahl vom Multi-Tenancy Ansatz soll auf begründeten Kriterien basieren.

Kabbedijk et al. haben entlang der beiden Dimensionen *Application Layer* und *Data Layer* 12 verschiedene Multi-Tenant Architekturen (MTA), so genannte MTA Patterns, abgeleitet [22]. Diese werden gegeneinander verglichen und so wird ein Hilfsmittel geboten, die ideale Multi-Tenancy Architektur zu identifizieren.

Im Rahmen der Literaturanalyse (vergleiche Kapitel 2.5) wurden die in Camunda möglichen Multi-Tenancy Ansätze [21] bereits erarbeitet. Ein weiterer Ansatz wird in Kapitel 2.5 erwähnt. Nachfolgend eine Auflistung der möglichen Ansätze, welche zu bewerten sind. Zu jedem der Ansätze wird angegeben, welchem MTA Pattern aus [22] der Ansatz entspricht, damit später die Ansätze mit dem Hilfsmittel aus [22] bewertet werden können.

- Eine geteilte Process Engine, auf welchem alle Mandanten gleichzeitig laufen. Die Daten sind dabei in einer geteilten Datenbank gespeichert. Dies entspricht einem AI, DC Pattern.
- Pro Mandant eine eigene Process Engine. Dazu eine geteilte Datenbank mit einem Satz an Tabellen pro Mandant. Dies entspricht einem AS, DB Pattern.
- Pro Mandant eine eigene Process Engine und Datenbank. Dies entspricht einem AS, DS Pattern.
- Pro Mandant eine eigene Virtual Machine. Entsprechend besitzt jeder Mandant eine eigene Process Engine (sowie Application Server) und Datenbank (sowie Datenbank Server). Dies entspricht einem AD, DD Pattern.

In [22] werden die Multi-Tenant Architekturen anhand von 17 Kriterien bewertet. Die Kriterien wurden durch die Autoren Kabbedijk et al. mittels einer strukturierten Literaturanalyse erarbeitet. Erklärungen zu den einzelnen Kriterien findet man in [49]. Aus diesen 17 Kriterien sollen die für die BPaaS Plattform relevanten Kriterien abgeleitet werden. Nicht alle von den 17 Kriterien, welche in [22] aufgeführt sind, sind aus Sicht des Autors relevant oder gleich relevant für die BPaaS Plattform.

Die Kriterien Portability (Übertragbarkeit vom System auf eine andere Laufzeitumgebung), Variability (Unterstützung von individuellen Lösungen oder Konfigurationen pro Mandant) sowie Diverse SLA (Unterstützung von verschiedenen SLAs für verschiedene Mandanten) werden vom Autor als unwichtig eingestuft (Gewichtungsfaktor 0.1). Die restlichen Kriterien werden alle als wichtig eingestuft und erhalten das Gewicht 1.0.

Abbildung 6.1 zeigt zusammenfassend die Kriterien aus [22]. Die MTA Patterns, welche durch Camunda unterstützt sind und entsprechend bewertet werden, sind blau markiert:

6.2 Entscheid

Die Entscheidung für einen Multi-Tenancy Ansatz soll mittels einer Nutzwertanalyse [50] herbeigeführt werden. Wie oben bereits erläutert, erhalten die aus Sicht des Autors wichtigen Kriterien ein Gewicht von 1.0 und die unwichtigen Kriterien ein Gewicht von 0.1 (siehe nachfolgende Tabelle 6.1). Die Bewertung der einzelnen Kriterien pro Ansatz

Decision Criterion	$\langle AD, DD \rangle$	$\langle AS, DD \rangle$	$\langle AI, DD \rangle$	$\langle \bar{AD}, \bar{DS} \rangle$	$\langle AS, DS \rangle$	$\langle AI, DS \rangle$	$\langle \bar{AD}, \bar{DB} \rangle$	$\langle AS, DB \rangle$	$\langle AI, DB \rangle$	$\langle \bar{AD}, \bar{DC} \rangle$	$\langle AS, DC \rangle$	$\langle AI, DC \rangle$	Dist. Factor (σ^2)
Time Behavior	5.0	4.0	4.0	4.0	4.0	3.0	4.0	3.5	3.0	3.5	3.0	2.0	0.6
Resource Utilization	2.5	2.5	3.0	2.5	3.0	3.0	3.0	3.0	4.0	3.0	3.0	4.5	0.4
Throughput	4.5	3.0	3.0	4.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	0.2
Number of Tenants	1.0	3.0	3.0	3.0	3.5	4.0	3.0	4.0	4.0	3.0	4.0	5.0	1.0
Number of End-Users	2.5	3.5	3.0	3.0	3.5	3.5	3.0	3.5	4.0	3.5	4.0	4.0	0.2
Availability	4.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0	0.1
Recoverability	5.0	4.5	4.5	4.0	4.0	4.0	3.0	3.0	3.0	2.0	2.0	2.0	1.1
Confidentiality	5.0	4.5	4.0	4.0	4.0	4.0	3.5	3.0	3.0	2.0	2.0	2.0	1.0
Integrity	4.5	4.0	3.0	4.0	3.5	3.0	3.5	3.0	3.0	3.0	2.5	2.5	0.4
Authenticity	4.5	3.5	3.0	3.5	3.0	3.0	4.0	3.0	3.0	3.0	3.0	3.0	0.2
Maintainability	1.5	2.5	3.0	2.0	3.0	3.5	2.5	4.0	4.0	3.0	4.0	5.0	1.0
Portability	5.0	5.0	5.0	4.5	4.5	4.5	4.0	4.0	4.0	3.0	3.0	3.0	0.6
Deployment Time	1.5	3.0	3.0	2.5	3.5	4.0	3.0	4.0	4.0	3.0	4.0	5.0	0.8
Variability	5.0	4.0	2.0	5.0	4.0	2.0	4.5	3.5	2.0	2.5	2.0	1.0	1.9
Diverse SLA	5.0	4.0	3.0	4.0	3.5	2.5	4.0	3.0	3.0	3.0	2.5	2.0	0.7
Software Complexity	5.0	4.5	4.0	4.5	4.5	3.5	4.0	4.0	3.0	2.5	2.5	2.0	0.9
Monitoring	1.0	2.5	3.0	2.5	3.0	3.0	3.0	4.0	4.0	3.0	4.0	5.0	1.0

Abbildung 6.1: Multi-Tenant Architekturen [22]

wurde aus [22] übernommen. Jede Zelle besteht aus zwei Werten. Die Zahl links ist die Bewertung, welche aus [22] übernommen wurde. Die Zahl rechts ist die gewichtete Bewertung (Gewicht multipliziert mit der Bewertung).

Aufgrund der Nutzwertanalyse in Tabelle 6.1 wird ein AS, DS Ansatz verfolgt. Gemäss [22] bedeutet das folgendes:

1. Es existiert ein Application Server, welcher von allen Mandanten geteilt wird. Pro Mandant existiert eine Applikationsinstanz von Camunda (Process Engine). Es wird vom Autor angenommen, dass eine Applikationsinstanz im Fall von Camunda mit einer Process Engine gleichgesetzt wird.

2. Es existiert ein Datenbank Server, welcher von allen Mandanten geteilt wird. Pro Mandant existiert eine Datenbank.

Kriterium	Gewicht	AD,DD		AS,DS		AS,DB		AI,DC	
Time Behavior	1.0	5.0	5.0	4.0	4.0	3.5	3.5	2.0	2.0
Resource Utilization	1.0	2.5	2.5	3.0	3.0	3.0	3.0	4.5	4.5
Throughput	1.0	4.5	4.5	3.0	3.0	3.0	3.0	3.0	3.0
Number of Tenants	1.0	1.0	1.0	3.5	3.5	4.0	4.0	5.0	5.0
Number of End-Users	1.0	2.5	2.5	3.5	3.5	3.5	3.5	4.0	4.0
Availability	1.0	4.0	4.0	3.0	3.0	3.0	3.0	3.0	3.0
Recoverability	1.0	5.0	5.0	4.0	4.0	3.0	3.0	2.0	2.0
Confidentiality	1.0	5.0	5.0	4.0	4.0	3.0	3.0	2.0	2.0
Integrity	1.0	4.5	4.5	3.5	3.5	3.0	3.0	2.5	2.5
Authenticity	1.0	4.5	4.5	3.0	3.0	3.0	3.0	3.0	3.0
Maintainability	1.0	1.5	1.5	3.0	3.0	4.0	4.0	5.0	5.0
Portability	0.1	5.0	0.5	4.5	0.45	4.0	0.4	3.0	0.3
Deployment Time	1.0	1.5	1.5	3.5	3.5	4.0	4.0	5.0	5.0
Variability	0.1	5.0	0.5	4.0	0.4	3.5	0.35	1.0	0.1
Diverse SLA	0.1	5.0	0.5	3.5	0.35	3.0	0.3	2.0	0.2
Software Complexity	1.0	5.0	5.0	4.5	4.5	4.0	4.0	2.0	2.0
Monitoring	1.0	1.0	1.0	3.0	3.0	4.0	4.0	5.0	5.0
Summe Gewichtet			49		**49.7**		49.05		48.6

Tabelle 6.1: Bewertung von Multi-Tenant Ansätzen [22]

6.3 Implementierung

Jede Process Engine muss bei Camunda in eine Datei namens `bpm-platform.xml` eingetragen werden [51]. Des Weiteren muss pro Mandant eine Java Database Connectivity (JDBC) Ressource in der Tomcat Datei `server.xml` registriert werden [52]. Dem gewählten Ansatz entsprechend bedeutet das, dass für jeden Mandanten einen Eintrag in diesen Dateien erfolgen muss. Damit die Änderungen sichtbar werden, muss Camunda (respektive der darunterliegende Application Server) neu gestartet werden [53].

Ein Neustart vom Application Server ist bei der BPaaS Plattform nicht denkbar, weil das On-Boarding eines neuen Mandanten nicht zu einem Unterbruch bei den bereits vorhandenen Mandanten führen darf. Entsprechend musste ein dynamischer Ablauf vom On-Boarding gesucht werden, zu welchem auch eine Frage im Camunda Forum gestellt wurde [53]. Mit dem Bootstrapping API von Camunda konnte diese Dynamik erreicht werden (vergleiche [54]).

Der Ablauf für das On-Boarding eines neuen Mandanten sieht folgendermassen aus:

1. Erstellen einer MySQL Datenbank auf dem Datenbank Server.
2. Ausführen von den Datenbank Scripts (`mysql_engine_7.6.0.sql` sowie `mysql_identity_7.6.0.sql`), welche die Camunda Tabellen und Indizes erstellen (enthalten in der Camunda Distribution).
3. Erstellen und Registrieren einer Process Engine über die Bootstrapping Java API (vergleiche [54]). Das Verwenden der Bootstrapping Java API verhindert einen Neustart von Tomcat.
4. Erstellen einer JDBC Ressource in der Datei `server.xml` von Tomcat.
5. Registrieren der Process Engine in der Datei `bpm-platform.xml` von Camunda. So steht die Process Engine auch nach einem Restart von Tomcat zur Verfügung.
6. Erstellen eines Camunda Administrator Benutzers mit entsprechenden Berechtigungen.
7. Erstellen von weiteren Camunda Benutzern mit entsprechenden Berechtigungen. Die Benutzerinformationen können beim On-Boarding Prozess über eine CSV Datei mitgegebenen werden.
8. Deployen des BPaaS Ad Hoc Prozesses in die erstellte Process Engine (Details dazu in Abschnitt 6.3.1).
9. Versenden einer Benachrichtigung per E-Mail an das Unternehmen, sobald der On-Boarding Prozess abgeschlossen ist. In dieser E-Mail Benachrichtigung ist ein Link enthalten, welchen das Unternehmen direkt auf die neu erstellte Process Engine führt.
10. Eintragen des Mandanten in die BPaaS Admin Datenbank (vergleiche Abschnitt 6.3.2).

Ein Unternehmen kann über eine Web-Applikation selbständig einen Mandanten eröffnen und erhält so Zugriff auf die BPaaS Plattform. Dazu wurde eine Servlet / JSP (HTML, Bootstrap) Web-Applikation erstellt. Um einen Mandanten zu eröffnen muss das Unternehmen ein Formular ausfüllen. Dabei kann auch eine CSV Datei mitgegeben

werden, welche Benutzerinformationen enthält. Diese Benutzer werden durch das On-Boarding auf der Process Engine erstellt. Die Benutzer haben Zugriff auf die Tasklist sowie das Cockpit und können Ad Hoc Prozesse starten.

Des Weiteren wurde eine Administrations Webseite realisiert, über welche auf die Process Engines der registrierten Mandanten zugegriffen werden kann sowie die Process Engines von registrierten Mandanten gelöscht werden können. Weil keine Dokumentation bezüglich dem Löschen einer Process Engine gefunden wurde, wurde eine Frage im Camunda Forum gestellt [55].

Der erwähnte On-Boarding Ablauf ist in der Klasse `ch.boller.bpaas.registration.service.TenantService` in der Methode `create` abgebildet. Weil zum Teil exklusiv auf Ressourcen zugegriffen wird (z.B. die Datei `bpm-platform.xml` oder `server.xml`), mussten die Methoden `create` und `delete` threadsicher implementiert werden, damit sich immer nur ein Thread in der Methode befinden kann. Der implementierte On-Boarding Ablauf ist an [56] angelehnt.

6.3.1 Ad Hoc Process Deployment

Wie im Ablauf oben beschrieben, muss nach dem Bootstrapping der Process Engine der BPaaS Ad Hoc Process in die neu erstellte Process Engine deployed werden, damit das Prozess Modell für den Mandanten zur Verfügung steht. Dafür existiert kein Java API (vergleiche [57]). Deshalb musste ein Umweg über die WAR Datei gemacht werden. In der WAR Datei befindet sich die Datei `processes.xml`. In dieser Datei muss für jeden Mandanten ein *process-archive* eingetragen werden [58] und darin der Name der Process Engine angegeben werden (diejenige des neuen Mandanten). Dazu sind Datei Manipulationen notwendig (Bearbeiten der Datei `processes.xml`, Ersetzen von der bearbeiteten Datei in der Ad Hoc Prozess WAR Datei). Diese Datei Manipulationen finden direkt im webapps Verzeichnis von Tomcat statt.

6.3.2 Admin Datenbank

Die `bpaas_admin` Datenbank befindet sich auf dem MySQL Server und beherbergt Metadaten zur BPaaS Plattform:

Alle Angaben zu den registrierten Mandanten werden in der Tabelle *tenant* geführt. Dazu gehört eine Identifikation der Process Engine vom Mandanten, der Name vom Unternehmen, die E-Mail Adresse sowie wann der Mandant erstellt wurde.

Damit für das On-Boarding eine gewisse Flexibilität bezüglich Konfiguration besteht (z.B. unterschiedliche Konfigurationen auf Entwicklungs- und Produktivumgebung), wurde eine Tabelle *settings* eingerichtet, welche Key-Value Paare enthält. Die On-Boarding Web-Applikation liest diese Konfiguration aus und verwendet diese zur Laufzeit. Folgende Konfigurationen sind möglich:

- Pfadangabe vom Tomcat Installationsverzeichnis.
- Camunda Datenbank Name Präfix von den Datenbanken, welche für die Mandanten erstellt werden (Default: `camunda_`).
- Name von der Tomcat Datasources Datei (Default: `bpmn-platform.xml`).
- Name von der Camunda Process Engine Datei (Default: `server.xml`).
- Link auf die Camunda Process Engine. Dieser Link wird in die E-Mail Benachrichtigung eingefügt, welche nach erfolgreichem On-Boarding versendet wird.
- Betreff der E-Mail Benachrichtigung, welche nach erfolgreichem On-Boarding versendet wird.
- Text der E-Mail Benachrichtigung, welche nach erfolgreichem On-Boarding versendet wird.
- Name der Ad Hoc Prozess WAR Datei, welche während dem On-Boarding verändert (siehe Abschnitt 6.3.1) und deployed wird (Default: `adhocprocess.war`).

Alle diese Key-Value Paare müssen vorhanden sein und einen Wert enthalten, sonst wirft die On-Boarding Web-Applikation eine Exception vom Typ `ch.boller.bpaas.registration.exception.InitializationException`.

6.4 Fazit

Das umgesetzte On-Boarding erfüllt den Zweck, dass sich ein Unternehmen selbständig eine Umgebung auf der BPaaS Plattform einrichten kann. Leider ist das Camunda Java

API noch nicht durchgängig für das On-Boarding nutzbar, so dass ein paar Schritte vom On-Boarding Prozess auf andere Art und Weise umgesetzt werden mussten (z.B. durch Datei Manipulationen, vergleiche Kapitel 6.3.1). Zudem ermöglicht die On-Boarding Web-Applikation potentiell Denial of Service Attacken, weil durch das Ausfüllen vom Registrierungsformular Ressourcen im Hintergrund angelegt werden. Das duzendfache Ausfüllen des Formulars braucht die verfügbaren Ressourcen auf dem Server auf (insbesondere Arbeitsspeicher).

Die On-Boarding Implementierung ist in folgenden Punkten noch von einer sauberen und produktiv einsetzbaren Lösung entfernt.

- Das On-Boarding ist dahingehend zu verfeinern, dass der On-Boarding Ablauf, welcher aus mehreren Schritten besteht, komplett durchläuft oder im Falle eines Fehlers sämtliche bereits getätigten Schritte rückgängig gemacht werden.
- Es sollte einen Kundenbereich geben, wo ein Unternehmen ein Account eröffnen kann und dort die BPaaS Plattform Zugänge verwalten kann. Dort könnten auch verschiedene Performance Packages gewählt werden (Arbeitsspeicher, CPU etc.) Dieser Kundenbereich existiert noch nicht.

Weiter muss der über die Nutzwertanalyse herbeigeführte On-Boarding Ansatz Entscheid kritisch hinterfragt werden. Die erreichten Punktzahlen der einzelnen Ansätze liegen zum Teil sehr nahe beieinander und durch eine minimale Veränderung der Gewichtungen würde eine andere Lösung präferiert werden. Im Idealfall wäre die Gewichtung der einzelnen Kriterien, welche die Kunden betreffen, durch Expertenbefragungen ermittelt worden.

Kapitel 7

Prozess Daten

Die Prozess Daten werden durch das Activity Pattern *Editieren von Prozess Daten* erstellt, modifiziert und gelöscht. Prozess Daten sind eine Sammlung an Datensätzen, wobei jeder Datensatz aus einem Schlüssel und einem Wert besteht. Der Schlüssel identifiziert und beschreibt den Datensatz, der Wert beinhaltet die eigentliche Information. Als Wert sollen beliebige Datentypen unterstützt werden. Beispiele davon können aus den erarbeiteten Anwendungsfällen (vergleiche Abschnitt 4.1) abgeleitet werden: Texte (z.B. Name des Patienten), Zahlen (z.B. Patientennummer), Bilder, Dokumente (z.B. Anamnesebogen) als PDF, Word, Excel.

Für die Ablage von Daten zu einer Prozess Instanz sind bei Camunda so genannte Prozess Variablen vorgesehen [37].

Grundsätzlich können die Prozess Daten gleichzeitig von mehreren Personen und Rollen bearbeitet werden, weil mehrere Activity Patterns vom Typ *Editieren von Prozess Daten* erfasst und gleichzeitig in Ausführung sein können. Dieser Umstand macht die Umsetzung von der Funktionalität komplex, weil z.B. eine Person den Datensatz Patientennummer bearbeiten kann, währenddem eine andere Person diesen Datensatz löscht. Weitere Parallelität kann durch das Activity Pattern *Importieren von Prozess Daten* dazu kommen, welches ebenfalls auf die Prozess Daten zugreift und diese bearbeitet.

Für diese Art von Problem implementiert Camunda den Optimistic Locking Mechanismus: Jeder Datensatz erhält eine Version. Bei einer Modifikation wird diese Version um eins erhöht. Wird nun versucht, die Daten zu bearbeiten oder löschen, dann wird immer die Version versucht zu bearbeiten oder löschen. Wenn keinen Datensatz mit

der Version gefunden wurde, dann erkennt Camunda das als Konflikt. Es muss also in der Zwischenzeit eine Bearbeitung des Datensatzes durch einen anderen Benutzer stattgefunden haben. Beim Vorliegen einer solchen Situation wird durch Camunda eine *OptimisticLockingException* geworfen [59].

Könnten die Prozess Daten des Ad Hoc Prozesses in Prozess Variablen abgelegt werden, so wäre das oben erwähnte Problem bereits durch den Optimistic Locking Mechanismus vom Camunda behandelt.

Durch das gewählte Ad Hoc Prozess Design besteht ein Ad Hoc Prozess aus mehreren Prozess Instanzen (vergleiche Kapitel 4). Deshalb kommen Prozess Variablen, welche pro Prozess Instanz gelten, für die Ablage und Verwaltung der Prozess Daten nicht in Frage. Entsprechend muss für dieses Problem ein anderer Lösungsansatz gesucht werden, welcher es erlaubt, die Prozess Daten zwischen mehreren Prozess Instanzen zu teilen.

7.1 Lösungsansätze

Eine Möglichkeit, die Prozess Daten aus mehreren Prozess Instanzen zugreifbar zu machen und zwischen Prozess Instanzen zu teilen, ist die Ablage der Prozess Daten in eine Datenbank. Zum einen könnten die Prozess Daten in die bereits pro Mandant vorhandene Camunda Datenbank abgelegt werden (vergleiche Kapitel 6). Dazu müsste das Camunda Datenbankschema um Tabellen ergänzt werden, damit die Prozess Daten verwaltet werden können.

Eine weitere Möglichkeit ist es, pro Mandant eine eigene Datenbank anzulegen, worin die Prozess Daten abgelegt werden. Eine Mandanten-übergreifende Datenbank für die Prozess Daten würde dem im Kapitel 6 herbeigeführten Entscheid einer eigenen Camunda Datenbank pro Mandant widersprechen. Dieser Ansatz hat den Vorteil, dass andere Datenbanktypen verwendet werden könnten als eine relationale Datenbank (die Camunda Daten sind im Prototyp pro Mandant in einer MySQL Datenbank abgelegt, vergleiche Kapitel 6). Unter Umständen wären andere Datenbank Typen [60] für die Ablage von den Prozess Daten besser geeignet (z.B. könnte eine MongoDB Document Database genutzt werden [61]).

Der Aufbau von eigenen Datenbankstrukturen für die Verwaltung der Prozess Daten ist ein Lösungsansatz, welcher aus Sicht des Autors für eine produktiv einsetzbare Lösung umgesetzt werden sollte. So lassen sich Berechtigungen abbilden und die Konsistenz der Daten ist durch die Datenbank sichergestellt.

7.2 Entscheid

Aus Sicht des Autors reicht für einen Prototyp ein Ansatz aus, welcher die Prozess Daten mittels eines Code Konstrukts verwaltet, auf welches alle Prozess Instanzen Zugriff haben.

7.3 Implementierung

In eine Datenstruktur wird pro Business Case Id ein Objekt vom Typ `AdHocProcessData` abgelegt, worin die Prozess Daten verwaltet werden. Jede Prozess Instanz kennt über eine entsprechende Prozess Variable die Business Case Id, welche beschreibt, welchem Business Case die Prozess Instanz angehört. So hat jede Prozess Instanz Zugriff auf das `AdHocProcessData` Objekt. Jedes `AdHocProcessData` Objekt besitzt eine Version, welche bei einer Modifikation (Erfassen, Bearbeiten, Löschen) inkrementiert wird. Bevor die Prozess Daten bearbeitet werden, wird die Version von den aktuellsten Ad Hoc Prozess Daten gegen die Version verglichen, als die Ad Hoc Prozess Daten gelesen wurden. Stimmt die Version nicht überein, dann hat in der Zwischenzeit eine Modifikation der Prozess Daten stattgefunden und es wird eine entsprechende Exception geworfen, welche den Benutzer auf die Ausnahmesituation hinweist. Dieses umgesetzte Muster entspricht dem Optimistic Locking Mechanismus.

Damit sind die Prozess Daten aber noch nicht persistiert. Damit die Prozess Daten persistent abgelegt werden, führt jede Modifikation der Prozess Daten dazu, dass der aktuelle Stand der Prozess Daten (das `AdHocProcessData` Objekt) in eine neue Prozess Variable abgelegt wird. Die Prozess Variable wird unter einem Schlüssel abgelegt, welcher die Version mitbeinhaltet. So kann es nicht zu einem Prozess Variable Namenskonflikt kommen. Über Camunda APIs (REST API) kann nach allen Prozess Variablen gesucht werden, welche zu einer Business Case Id gehören und es kann durch die Version im

Variable Name die Prozess Variable ermittelt werden, welche den aktuellsten Stand der Prozess Daten enthält (entspricht der höchsten Version Id). Dies ist zum Beispiel notwendig, wenn die aktuellsten Prozess Daten auf dem User Interface angezeigt werden sollen. Der Vorteil von diesem Ansatz ist, dass keine Datenbank Erweiterungen gemacht werden müssen. Dafür sind die Prozess Daten mehrfach in der Camunda Datenbank abgespeichert.

7.3.1 Daten Modell

Dieser Abschnitt beschreibt das Daten Modell der Prozess Daten. In nachfolgender Abbildung 7.1 ist das Daten Modell als UML Klassendiagramm dargestellt. Auf die Auflistung von getter und setter Methoden wird verzichtet:

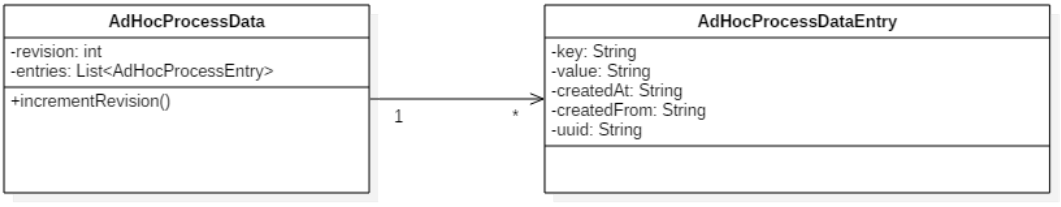


Abbildung 7.1: Daten Modell der Prozess Daten

Pro Business Case existiert ein Objekt vom Typ `AdHocProcessData`. Ein einzelner Wert wird durch ein Key und ein Value repräsentiert und in ein Objekt vom Typ `AdHocProcessDataEntry` abgelegt. Bei jeder Modifikation der Prozess Daten durch das Activity Pattern *Editieren von Prozess Daten* wird die Version auf dem Objekt `AdHocProcessData` erhöht.

Das Objekt besitzt folgende Eigenschaften:

- *revision*: Die Versionsnummer, welche den Stand identifiziert. Bei jeder Modifikation der Prozess Daten wird die Versionsnummer inkrementiert.
- *entries*: Die Sammlung an Prozess Daten.

Ein einzelner Wert in den Prozess Daten wird in ein Objekt vom Typ `AdHocProcessDataEntry` abgelegt.

Das Objekt besitzt folgende Eigenschaften:

- *key*: Schlüssel des Wert, welcher den Wert beschreibt (z.B. Patientenname).
- *value*: Der eigentliche Wert als String.
- *createdAt*: Wann der Wert erfasst wurde.
- *createdFrom*: Der Benutzer, welcher den Wert erfasst hat.
- *uuid*: Zur eindeutigen Identifikation eines Werts.

7.4 Fazit

Der in diesem Kapitel erarbeitete Ansatz zur Verwaltung der Prozess Daten ist aus Sicht des Autors für einen Prototyp ausreichend. Für einen produktiven Einsatz ist dieser Ansatz aber ungeeignet und muss in folgenden Punkten kritisch hinterfragt werden:

- Die Prozess Daten werden mehrfach in Prozess Variablen gespeichert (Redundanz).
- Bei der Darstellung der Prozess Daten muss immer zuerst die Prozess Variable ermittelt werden, welche die aktuellsten Prozess Daten beinhaltet.
- Der implementierte Ansatz erlaubt es nicht, Berechtigungsmechanismen für den Zugriff (Lesen, Schreiben) auf die Prozess Daten zu implementieren.
- Aktuell unterstützt werden nur Prozess Daten in Form von Zeichenketten (Strings).

Kapitel 8

Evaluierung

Wie in der Forschungsmethode erläutert (vergleiche Kapitel 3), soll der entwickelte Prototyp bezüglich der Wirksamkeit evaluiert werden. Dabei fanden Evaluationen auf dem Business Logik Layer und dem User Interface Layer statt.

8.1 Business Logik

Auf dem Business Logik Layer wurden die in Kapitel 4 vorgestellten Beispielabläufe automatisiert getestet. Camunda bietet zur Automatisierung von Abläufen entsprechende APIs an [62].

Die erarbeiteten Abläufe (Behandlung eines Patienten, Umsetzung eines Projekts) sind in den Test Klassen
`ch.boller.bpaas.adhocprocess.ScenarioBehandlungEinesPatientenTest` sowie
`ch.boller.bpaas.adhocprocess.ScenarioUmsetzungEinesProjektsTest` umgesetzt.

8.2 User Interfaces

Auch das User Interface sollte bei der Evaluierung zur Ausführung kommen, stellt es doch die Schnittstelle zwischen BPaaS Plattform und Benutzer dar. Auch hier kamen die im Kapitel 4 vorgestellten Beispielabläufe zur Anwendung. Die Beispielabläufe wurden

direkt auf der BPaaS Plattform manuell durchgeführt. Als Testumgebung wurde die produktive Amazon AWS Umgebung genutzt (vergleiche Anhang A).

Der Beispielablauf *Umsetzung eines Projekts* liesse sich beliebig weit treiben, weil das Beispiel rekursiv aufgebaut ist. Im Rahmen von der Evaluierung wurde ein Projekt in zwei Teilprojekte aufgeteilt, welche nicht weiter aufgeteilt wurden.

Den Beispielabläufen vorangehängt wurde beim Testen der On-Boarding Prozess, welcher über die On-Boarding Web-Applikation durchgeführt wurde. Zudem wurde während der Ausführung der Beispielabläufe immer mal wieder das Cockpit Plugin geöffnet, welches den Prozess Ablauf und die Prozess Daten aller Ad Hoc Prozesse anzeigt.

Zu Abweichungen gemäss den dokumentierten Beispielabläufen kam es bei der Durchführung der Evaluierung bei folgenden Punkten:

- Es können beim Durchführen vom Activity Pattern *Editieren von Prozess Daten* keine Dateien auf die BPaaS Plattform heraufgeladen werden, weil der Prototyp dies nicht unterstützt. Entsprechend wurden anstatt Dateien einfach Textwerte eingegeben.
- Anders als in den Beispielabläufen erwähnt, unterstützt der Prototyp keine Auswahl und Zuteilung von Activity Patterns zu Rollen. Entsprechend wurden die Activity Patterns direkt Benutzern zugeordnet.

Kapitel 9

Diskussion

9.1 Resultat

Diese Master Thesis legt als Resultat ein Konzept sowie ein Software Prototyp vor.

9.1.1 Konzept

Zur Lösung von der Problemstellung wurde ein Ad Hoc Prozess erarbeitet, welcher als BPMN Prozess Modell modelliert wurde. Der Ad Hoc Prozess beschreibt, wie unstrukturierte Business Prozesse mittels Activity Patterns auf der BPaaS Plattform unterstützt werden können. Weiter zeigt das Konzept User Interface Mockups auf. Diese beschreiben die Benutzeroberfläche, welche den Ad Hoc Prozess für den Endbenutzer ausführbar und steuerbar macht. Des Weiteren wurden Lösungen erarbeitet für die Unterstützung von Prozess Daten sowie für das automatisierte On-Boarding von Unternehmen auf die BPaaS Plattform.

9.1.2 Prototyp

Der umgesetzte Prototyp nutzt das im Konzept erarbeitete BPMN Prozess Modell und erlaubt es, die Activity Patterns *Ausführen von Task*, *Entscheidung* sowie *Editieren von Prozess Daten* auszuführen. Der Prototyp umfasst des Weiteren ein Camunda Cockpit Plugin, über welches der Status der sich in Ausführung befindlichen und beendeten Ad Hoc Prozesse eingesehen werden kann. Zudem bietet der Prototyp eine On-Boarding

Web-Applikation, worüber automatisiert eine Umgebung für ein Unternehmen aufgebaut werden kann.

Die weiteren Activity Patterns (vergleiche Tabelle 2.1) wurden entweder bewusst oder aus zeitlichen Gründen nicht umgesetzt. Bewusst nicht umgesetzt wurden die Activity Patterns *Eröffnen von Business Case* sowie *Schliessen von Business Case*. Diese Activity Patterns sind durch das Starten eines Ad Hoc Prozesses sowie durch das Abschliessen vom Ad Hoc Prozess implizit vorhanden. Dabei wird eine neue Business Case Id generiert und die Prozess Daten initialisiert. Für das Activity Pattern *Verfeinern von Prozess Daten* ergab sich aus den erarbeiteten Anwendungsfällen (vergleiche Kapitel 4.1) kein Szenario, welches das Activity Pattern notwendig machte. Die weiteren Activity Patterns wurden aus zeitlichen Gründen nicht umgesetzt und hätten sowohl die Erweiterung vom Ad Hoc Prozess Modell als auch die Implementierung von weiteren User Interfaces zur Folge gehabt.

Mit der Evaluierung (vergleiche Kapitel 8) wurde dargelegt, dass typische unstrukturierte Business Prozesse durch den implementierten Software Prototyp unterstützt werden. Die Wirksamkeit vom Software Prototyp wurde so aufgezeigt. Auch wenn der umgesetzte Prototyp noch einige Schwachstellen besitzt und Workarounds implementiert, kann der Prototyp nach Meinung des Autors die Basis für weitere Forschungsarbeiten bilden.

9.2 Beantwortung der Forschungsfragen

9.2.1 Hauptforschungsfrage

HF: *Wie kann eine BPaaS Plattform realisiert werden, welche unstrukturierte Business Prozesse mittels Activity Patterns auf Basis von Camunda als Service anbietet?*

Das zentrale Artefakt dieser Master Thesis ist der Ad Hoc Prozess. Das Artefakt beschreibt einen Ansatz, wie unstrukturierte Business Prozesse unterstützt werden können. Der Ad Hoc Prozess wurde als BPMN Prozess Modell entwickelt und stellt so ein generisches Konstrukt dar, welches grundsätzlich auf beliebigen BPMS ausführbar ist. Der Ad Hoc Prozess erlaubt es dem Knowledge Worker die jeweils nächsten auszuführenden

Activity Patterns auszuwählen und in Auftrag zu geben. Sind die Activity Patterns ausgeführt, so gelangt der Ad Hoc Prozess wieder zurück zum Knowledge Worker und die nächsten auszuführenden Activity Patterns können gewählt werden.

Es wurden verschiedene voneinander unabhängige zu unterstützende Dimensionen identifiziert, über welche der Ablauf von einem Ad Hoc Prozess gesteuert werden kann. Diese Dimensionen sind die Synchronität / Asynchronität, Rekursion und Parallelität (vergleiche UF 1 in Abschnitt 9.2.2). Nach dem aktuellen Wissensstand sind für die Unterstützung von unstrukturierten Business Prozessen keine weiteren Dimensionen notwendig. Definitiv ausschliessen lässt sich das aber nicht.

Durch das Deployment der BPaaS Plattform auf einen Amazon Cloud basierten Windows Rechner sowie das Anbieten eines On-Boarding Prozesses kann die Umgebung für den Kunden als Service angeboten werden. Der Kunde kann selbständig einen Zugang zur BPaaS Plattform beantragen und sofort mit der Ausführung von unstrukturierten Business Prozessen beginnen.

9.2.2 Unterforschungsfragen

UF 1: *Welche Arten von Zusammenarbeit zwischen verschiedenen organisatorischen Rollen sollen durch die BPaaS Plattform unterstützt werden?*

Die Anforderungen an den Ad Hoc Prozess wurden aus beispielhaften unstrukturierten Business Prozessen abgeleitet, welche wiederum teilweise aus der Literatur abgeleitet wurden. Bei diesem Vorgang hat sich gezeigt, dass der Ad Hoc Prozess mehrere voneinander unabhängige Dimensionen unterstützen soll: Es soll möglich sein, Activity Patterns synchron sowie asynchron auszuführen. Das heisst, der Knowledge Worker kann entscheiden, ob auf das Resultat vom Activity Pattern gewartet werden soll (synchron) oder nicht (asynchron). Eine zweite Dimension stellt das rekursive Starten von einem Ad Hoc Prozess aus einem Activity Pattern dar. Es soll entschieden werden können, ob aus einem Activity Pattern ein neuer Ad Hoc Prozess gestartet werden können soll. So kann ein Knowledge Worker z.B. für die Ausführung eines Tasks die Arbeit auch wieder in Form von Activity Patterns organisieren. Eine dritte Dimension stellt die Parallelität dar. Der Knowledge Worker kann eines oder mehrere Activity Patterns parallel in Auftrag geben.

UF 2: *Wie können die Activity Patterns auf der BPaaS Plattform angeboten werden?*

Bei den umgesetzten Activity Patterns handelt es sich um BPMN User Tasks, sprich der Task wird durch einen Benutzer auf der BPaaS Plattform ausgeführt. Dabei wird dem Benutzer ein User Interface angeboten. Pro umgesetztem Activity Pattern wurde ein User Interface Mockup erstellt (vergleiche Kapitel 5). Diese User Interface Mockups zeigen wie die Activity Patterns auf der BPaaS Plattform angeboten werden könnten. Die im Rahmen vom entwickelten Prototyp erstellten User Interfaces sind an diese Mockups angelehnt.

Allen Activity Patterns ist gemeinsam, dass der Knowledge Worker beim Erfassen vom Activity Pattern eine textuelle Beschreibung eingeben muss, um die auszuführende Tätigkeit zu beschreiben.

UF 3: *Was muss pro Activity Pattern konfigurierbar und parametrierbar sein, damit die Activity Patterns Mandanten-spezifisch genutzt werden können?*

Diese Unterforschungsfrage zielte vor allem auf die Activity Patterns ab, welche als Service Task ausgeführt würden (*Importieren und Exportieren von Prozess Daten* sowie *Absetzen und Empfangen von einem Event*). Wie bereits im Abschnitt 9.1.2 erwähnt, wurden diese Activity Patterns nicht umgesetzt. Entsprechend kann diese Unterforschungsfrage nicht beantwortet werden.

UF 4: *Welche Regeln bezüglich Activity Pattern Reihenfolge müssen auf der BPaaS Plattform abgebildet werden?*

Diese Unterforschungsfrage hat sich im Verlauf der Master Thesis als nicht relevant erwiesen. Die im Rahmen dieser Master Thesis umgesetzten Activity Patterns (*Ausführen von Task*, *Entscheidung* sowie *Editieren von Prozess Daten*) setzen nach Meinung des Autors keine Regeln bezüglich Reihenfolge voraus.

UF 5: *Wie können beliebig anfallende Prozess Daten zur Verfügung gestellt und persistiert werden?*

Camunda bietet Prozess Variablen an, welche es erlauben, beliebige Informationen zu einer Prozess Instanz abzulegen, die im Laufe einer Prozessausführung anfallen. Auch

Java Objekte lassen sich ablegen. Camunda kümmert sich dabei um die Persistenz der Daten. Dieser Mechanismus kann nicht verwendet werden, weil die Prozess Variablen pro Prozess Instanz gelten, ein Ad Hoc Prozess aber durch das gewählte Design aus mehreren Prozess Instanzen besteht, welche die Prozess Daten untereinander teilen müssen. Es wurden verschiedene Lösungsansätze erarbeitet, aus welchen ein Lösungsansatz durch den Autor ausgewählt und im Prototyp umgesetzt wurde. Die Prozess Daten werden zur Laufzeit in einer Datenstruktur verwaltet und pro Business Case Id abgelegt. So haben alle Prozess Instanzen Zugriff auf die jeweiligen Prozess Daten. Es wurde ein Optimistic Locking Mechanismus umgesetzt, um Konflikte bei der Bearbeitung der Prozess Daten zu erkennen. Dabei erhalten die Prozess Daten eine Version, welche beim Bearbeiten der Prozess Daten noch dieselbe sein muss, wie die Version, als die Prozess Daten gelesen wurden. Ansonsten liegt ein Konflikt vor und die Bearbeitung der Prozess Daten wird nicht fortgesetzt. Nach der Bearbeitung der Prozess Daten wird die Version inkrementiert. Die Prozess Daten werden in einem Java Objekt abgelegt, welches beliebig erweitert werden kann, um weitere Datentypen zu unterstützen. Aktuell unterstützt werden nur Prozess Daten, welche als Strings (Zeichenfolgen) vorliegen.

UF 6: *Welche Kriterien sollen herangezogen werden, um zu entscheiden, welche Multi-Tenancy Architektur umgesetzt wird?*

Die aus Sicht des Autors relevanten Kriterien wurden in Kapitel 6 erarbeitet. Dazu gehören zeitliches Verhalten, Ressourcenverbrauch, Durchsatz, Skalierung mit Anzahl Mandanten und Endbenutzer, Verfügbarkeit, Wiederherstellbarkeit, Vertraulichkeit, Integrität, Authentizität (Echtheit der Daten), Wartbarkeit, Dauer für das Deployment, Softwarekomplexität sowie Überwachungsmöglichkeit. Als weniger relevant eingestuft wurden aus Sicht des Autors die Kriterien Übertragbarkeit vom System auf eine andere BPMS Plattform und die Möglichkeit haben, an verschiedene Mandanten unterschiedliche Funktionalitäten sowie Service Level Agreements anbieten zu können. In [22] wurden anhand der erwähnten Kriterien verschiedene Multi-Tenancy Architekturen bewertet, woraus durch den Autor die optimale Multi-Tenancy Architektur für die BPaaS Plattform abgeleitet wurde.

UF 7: *Wie können Benutzer- und Gruppen Informationen, z.B. ab einem LDAP beim Mandanten, in die BPaaS Plattform integriert werden?*

Diese Forschungsfrage wurde aus Zeitgründen nicht berücksichtigt. Die Relevanz der Erkenntnisse durch die Beantwortung dieser Forschungsfrage wurde durch den Autor als geringer eingestuft als die anderen Forschungsfragen. Damit aber beim On-Boarding Benutzer auf der BPaaS Plattform automatisiert erstellt werden können, wurde eine CSV Dateischnittstelle implementiert, welcher Benutzerinformationen strukturiert mitgegeben werden können.

9.3 Kritische Reflektion

Die einzelnen Kapitel des Hauptteils dieser Master Thesis (Kapitel 3 - 7) enthalten jeweils als Abschluss ein Unterkapitel *Fazit*, welches die erarbeiteten Resultate sowie die Vorgehensweise hinterfragt und kritisch würdigt. Auf diese Punkte wird an dieser Stelle nicht noch ein zweites Mal eingegangen.

Grundsätzlich fehlt nach Abschluss dieser Master Thesis das Wissen darüber, ob es für Knowledge Worker vorstellbar ist, nach dem im Rahmen dieser Master Thesis postulierten Ansatz (Ad Hoc Prozess zusammen mit den Activity Patterns) zu arbeiten und ob der erwartete Nutzen sich daraus einstellt. Der im Kapitel 3 vorgestellte Relevance Cycle wurde entsprechend nicht geschlossen, weil keine Feldtests oder Befragungen bezüglich der Resultate durchgeführt wurden. Die im Rahmen dieser Master Thesis durchgeführten Evaluationen zielten nicht auf den Nutzen (Utility) ab.

Weiter ergaben sich aus dem erarbeiteten Ad Hoc Prozess einige Fragestellungen, welche als offene Punkte in Kapitel 4 dokumentiert sind. Hier wären Experteninterviews z.B. mit Ärzten oder eine Beobachtung im Feld [63] sicher hilfreich gewesen, den erarbeiten Ad Hoc Prozess Ansatz zu validieren, Antworten auf die Fragen zu erhalten und den Lösungsansatz weiter zu verfeinern.

Dem Betrieb der BPaaS Plattform auf einer Cloud Plattform steht der Autor kritisch gegenüber, weil gerade das Beispiel eines Patienten in der Notfallaufnahme (vergleiche Abschnitt 4.1) aufzeigt, was für heikle und besonders schützenswerte Daten [64] auf der BPaaS Plattform anfallen können. Sollten tatsächlich solche Daten Eingang auf die

BPaaS Plattform finden, müsste ein Datensicherheitskonzept unter Berücksichtigung der gesetzlichen Grundlagen ausgearbeitet werden und dieses in der BPaaS Plattform umgesetzt werden.

9.4 Ausblick und zukünftige Forschung

Durch diese Master Thesis wurden drei Activity Patterns aus [10] umgesetzt. Weitere Activity Patterns können durch zukünftige Forschungsarbeiten in den Prototyp integriert werden. Wie in Kapitel 4 erwähnt, setzt das Anbieten weiterer Activity Patterns eine Erweiterung vom BPMN Ad Hoc Prozess Modell voraus, weil aktuell nur User Tasks und keine Service Tasks unterstützt werden.

Des Weiteren existiert im User Interface Bereich Verbesserungspotential, welches durch weitere Forschungsarbeiten erschlossen werden kann. Der Prozess Ablauf zeigt auf, welche Activity Patterns erfasst wurden und in welchem Status sich diese befinden. Diese Information ist zwingend notwendig, kann aber durch die Dimensionen der Synchronität / Asynchronität, Rekursion sowie Parallelität sehr komplex und schwierig erfassbar werden. Auch kann der Prozess Ablauf sehr lang werden und somit kann die Aussagekraft und Übersichtlichkeit der im Rahmen dieser Master Thesis umgesetzten Lösung bemängelt werden. Hier fragt sich, wie diese Informationen intuitiv und eventuell verdichtet dargestellt werden können. Dabei muss daran gedacht werden, dass mit der BPaaS Plattform sehr wahrscheinlich auch mit mobilen Geräten interagiert wird. Kapitel 4.2 zeigt in einer Netzwerkstruktur auf, wie ein Ad Hoc Prozess sich zur Laufzeit entwickeln kann. Eventuell liesse sich anhand dieser Information eine übersichtliche Darstellung vom Prozess Ablauf ableiten.

Interessant wäre zu wissen, ob sich Knowledge Worker vorstellen könnten, durch eine BPaaS Plattform mit den vorgeschlagenen Abläufen unterstützt zu werden und ob sich dabei der erwartete Nutzen einstellt. Mit diesen Erkenntnissen lassen sich das ausgearbeitete Konzept sowie der Prototyp weiter verfeinern. Der Autor ist überzeugt, dass mit den aus dieser Master Thesis hervorgegangenen Resultaten weitere Forschungsarbeiten in diesem Bereich unterstützt werden.

Literaturverzeichnis

[1] Z. A. Bukhsh, ,,Master Thesis BPMN Plus: A Modelling Language for Unstructured Business Processes," Ph.D. dissertation, 2015.

[2] AIIM, ,,Case Management and Smart Process Applications," 2016. Aufgerufen von http://www.aiim.org/Resources/Research/Industry-Watches/2014/2014_July_Case-Management

[3] Y. Stavenko, N. Kazantsev, und A. Gromoff, ,,Business Process Model Reasoning: From Workflow to Case Management," *Procedia Technology*, Vol. 9, S. 806–811, 2013.

[4] Ian James, ,,What are Unstructured Business Processes? - The Process Consultant," 2013. Aufgerufen von http://theprocessconsultant.com/what-are-unstructured-business-processes-2/

[5] J. Ukelson, ,,Five ways to discover unstructured processes - TechRepublic," 2009. Aufgerufen von http://www.techrepublic.com/blog/tech-decision-maker/five-ways-to-discover-unstructured-processes/

[6] T. Barton, C. Seel, und A. Lurzenhof, ,,Business process as a service-status and architecture." in *EMISA*, 2014, S. 145–158.

[7] M. Böhringer, ,,Emergent case management for ad-hoc processes: A solution based on microblogging and activity streams," *Lecture Notes in Business Information Processing*, Vol. 66 LNBIP, S. 384–395, 2011.

[8] O. Marjanovic, ,,Towards IS supported coordination in emergent business processes," *Business Process Management Journal*, Vol. 11, Nr. 5, S. 476, 2005.

[9] L. Thom, M. Reichert, und C. Iochpe, ,,Activity patterns in process-aware information systems: basic concepts and empirical evidence," *International Journal of Business Process Integration and Management*, Vol. 4, Nr. 2, S. 93–110, 2009.

[10] T. Keller und D. Grünert, ,,Business Process Modeling using Activity Patterns," Zürcher Hochschule für Angewandte Wissenschaften, Tech. Rep., 2014.

[11] T. Baeyens, ,,BPM in the Cloud," S. 1–3, 2010. Aufgerufen von http://www.prozesse.org/newsletter/GP-Newsletter04-10IBM.pdf

[12] camunda, ,,Camunda Home," 2016. Aufgerufen von https://camunda.com/de

[13] Amazon, ,,Amazon Web Services AWS - Server Hosting & Cloud Services," 2016. Aufgerufen von https://aws.amazon.com/de/

[14] OMG, ,,BPMN 2.0," 2011. Aufgerufen von http://www.omg.org/spec/BPMN/2.0/

[15] camunda, ,,Camunda BPM 7.6.0 Released," 2017. Aufgerufen von https://blog.camunda.org/post/2016/11/camunda-bpm-760-released/

[16] Scheer, ,,Scheer BPaaS - Die Process2Application Plattform," 2016. Aufgerufen von https://www.scheer-group.com/scheer-bpaas-process2application-user-guide

[17] Tata Consultancy Services, ,,Business-Process-as-a-Service - BPaaS," 2016. Aufgerufen von http://www.tcs.com/business-process-services/Pages/platform-solutions.aspx

[18] L. H. Thom, C. Lochpe, und M. Reichert, ,,Workflow Patterns for Business Process Modeling," *Proceedings of Workshops and Doctoral Consortium of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007), Trondheim, Norway*, Vol. I, S. 349–358, 2007.

[19] B. Rücker, ,,BPMN und unstrukturierte Prozessteile? ACM? Beispiel Patentantrag," 2012. Aufgerufen von http://www.bpm-guide.de/2012/11/02/bpmn-unstructured-processes-and-acm-example-patent-application/

[20] Everteam, ,,Why Unstructured Processes Are Critical to an Organization's Success," 2016. Aufgerufen von http://www.everteam.com/en/2016/04/15/why-unstructured-processes-are-critical-to-an-organizations-success/

[21] camunda, ,,Multi-Tenancy," 2016. Aufgerufen von https://docs.camunda.org/manual/7.6/user-guide/process-engine/multi-tenancy/

[22] J. Kabbedijk, M. Pors, S. Jansen, und S. Brinkkemper, ,,Multi-tenant architecture comparison," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 8627 LNCS, S. 202–209, 2014.

[23] Carl Rabeler, ,,Design patterns for multitenant SaaS applications and Azure SQL Database," 2016. Aufgerufen von https://azure.microsoft.com/en-us/documentation/articles/sql-database-design-patterns-multi-tenancy-saas-applications/

[24] camunda, ,,Identity Service," 2016. Aufgerufen von https://docs.camunda.org/manual/7.6/user-guide/process-engine/identity-service/

[25] A. R. Hevner, S. T. March, J. Park, und S. Ram, ,,Design Science in Information Systems Research," *MIS Quaterly*, Vol. 28, Nr. 1, S. 75–105, 2004.

[26] A. R. Hevner, ,,A Three Cycle View of Design Science Research," *Scandinavian Journal of Information Systems*, Vol. 19, Nr. 2, S. 87–92, 2007.

[27] V. Vaishnavi und W. Kuechler, ,,Design Science Research in Information Systems," *Wwwisworldorg*, Vol. 22, Nr. 2, S. 1–16, 2004.

[28] N. Prat, I. Comyn-Wattiau, und J. Akoka, ,,Artifact Evaluation in Information Systems Design Science Research - A Holistic View," *PACIS 2014 Proceedings*, Vol. Paper 23, S. 1–16, 2014.

[29] K. Peffers, T. Tuunanen, M. A. Rothenberger, und S. Chatterjee, ,,A Design Science Research Methodology for Information Systems Research," *Source Journal of Management Information Systems*, Vol. 24, Nr. 3, S. 45–77, 2007.

[30] C. Di Ciccio, A. Marrella, und A. Russo, ,,Knowledge-intensive Processes: An overview of contemporary approaches," *CEUR Workshop Proceedings*, Vol. 861, S. 33–47, 2012.

[31] K. Sarshar, P. Loos, P. Dominitzki, und C. Reichel, ,,Krankenhausprozesse: Dokumentation erhobener Daten einer Feldstudie in einem Universitäts-Klinikum,"

S. 83, 2005. Aufgerufen von http://www.uni-saarland.de/fileadmin/user_upload/Professoren/fr13_ProfLoos/isym_paper_024.pdf

[32] Projektmanagement-Definitionen, ,,Teilprojekte - Projektmanagement: Definitionen, Einführungen und Vorlagen," 2017. Aufgerufen von http://projektmanagement-definitionen.de/glossar/teilprojekte/

[33] M. Dumas, M. La Rosa, J. Mendling, und H. A. Reijers, *Fundamentals of business process management*, 2012.

[34] camunda, ,,Expression Language," 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/user-guide/process-engine/expression-language/

[35] camunda, ,,Tasklist," 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/webapps/tasklist/

[36] camunda, ,,Delegation Code," 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/user-guide/process-engine/delegation-code/

[37] camunda, ,,Process Variables," 2016. Aufgerufen von https://docs.camunda.org/manual/7.6/user-guide/process-engine/variables/

[38] ITU, ,,Universally Unique Identifiers (UUIDs)," 2017. Aufgerufen von http://www.itu.int/en/ITU-T/asn1/Pages/UUID/uuids.aspx

[39] camunda, ,,Loop in process model: How to handle the process variables - Camunda BPM Forum," 2017. Aufgerufen von https://forum.camunda.org/t/loop-in-process-model-how-to-handle-the-process-variables/3379

[40] OMG, ,,CMMN 1.1," 2016. Aufgerufen von http://www.omg.org/spec/CMMN/

[41] camunda, ,,Introduction," 2016. Aufgerufen von https://docs.camunda.org/manual/7.6/introduction/

[42] camunda, ,,Embedded Task Form: Show information about running process instances - Camunda BPM Forum," 2017. Aufgerufen von https://forum.camunda.org/t/embedded-task-form-show-information-about-running-process-instances/3423

[43] camunda, ,,Cockpit Plugins," 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/webapps/cockpit/extend/plugins/

[44] camunda, ,,Access the process engine name - Camunda BPM Forum,” 2017. Aufgerufen von https://forum.camunda.org/t/access-the-process-engine-name/3329

[45] IETF, ,,The JavaScript Object Notation (JSON) Data Interchange Format,” 2014. Aufgerufen von https://tools.ietf.org/html/rfc7159

[46] camunda, ,,Camunda Data Format JSON,” 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/user-guide/data-formats/json/

[47] camunda, ,,Multi-Tenancy: Hide Tenant Dropdown Box - Camunda BPM Forum,” 2017. Aufgerufen von https://forum.camunda.org/t/multi-tenancy-hide-tenant-dropdown-box/3070

[48] camunda, ,,Configure Authentication,” 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/reference/rest/overview/authentication/

[49] M. Pors, L. Blom, J. Kabbedijk, und S. Jansen, ,,Sharing is Caring A Decision Support Model for Multi-Tenant Architectures,” Nr. September, 2013.

[50] C. Putzhammer, ,,Nutzwertanalyse,” 2015. Aufgerufen von https://www.controlling-wiki.com/de/index.php/Nutzwertanalyse

[51] camunda, ,,bpm-platform.xml,” 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/reference/deployment-descriptors/descriptors/bpm-platform-xml/

[52] camunda, ,,Install the Full Distribution on a Tomcat Application Server manually,” 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/installation/full/tomcat/manual/

[53] camunda, ,,Multi-Tenancy: Programmatically create tenant - Camunda BPM Forum,” 2017. Aufgerufen von https://forum.camunda.org/t/multi-tenancy-programmatically-create-tenant/2876

[54] camunda, ,,Process Engine Bootstrapping,” 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/user-guide/process-engine/process-engine-bootstrapping/

[55] camunda, ,,Unregister / Delete Process Engine - Camunda BPM Forum,” 2017. Aufgerufen von https://forum.camunda.org/t/unregister-delete-process-engine/3526

[56] D. Meyer, ,,ProcessEngineStandalone - GitHub," 2015. Aufgerufen von https://github.com/meyerdan/camunda-main/blob/master/src/main/java/org/camunda/bpm/test/ProcessEngineStandalone.java

[57] camunda, ,,Deployment through Java API - Camunda BPM Forum," 2017. Aufgerufen von https://forum.camunda.org/t/deployment-through-java-api/3224

[58] camunda, ,,Multi-Tenancy processes.xml," 2017. Aufgerufen von https://docs.camunda.org/manual/7.5/user-guide/process-engine/multi-tenancy/#specify-the-tenant-identifier-via-deployment-descriptor

[59] camunda, ,,Transactions in Processes," 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/user-guide/process-engine/transactions-in-processes/#optimistic-locking

[60] A. Meier und M. Kaufmann, *SQL- & NoSQL-Datenbanken*, 8. Aufl. Springer Vieweg, 2016.

[61] MongoDB, ,,Datenverwaltung neu erfunden," 2016. Aufgerufen von https://www.mongodb.com/de

[62] camunda, ,,Testing," 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/user-guide/testing/

[63] N. Baur und J. Blasius, *Handbuch Methoden der empirischen Sozialforschung*, 2014.

[64] Die Bundesversammlung der Schweizerischen Eidgenossenschaft, *DSG: Bundesgesetz über den Datenschutz vom 19. Juni 1992 (Stand am 1. Januar 2014), 235.1*, 2014, Vol. 1992.

[65] Amazon, ,,EC2-Instance-Typen - Amazon Web Services (AWS)," 2017. Aufgerufen von https://aws.amazon.com/de/ec2/instance-types/

[66] Oracle, ,,Java SE Runtime Environment 8 - Downloads," 2017. Aufgerufen von http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

[67] Oracle, ,,MySQL - Download MySQL Installer," 2017. Aufgerufen von https://dev.mysql.com/downloads/windows/installer/5.7.html

[68] camunda, ,,Download Camunda BPM platform," 2017. Aufgerufen von https://camunda.org/download/

[69] Maven, ,,Maven Repository: mysql - mysql-connector-java - 6.0.5," 2017. Aufgerufen von http://mvnrepository.com/artifact/mysql/mysql-connector-java/6.0.5

[70] Amazon, ,,Amazon EC2 Security Groups for Windows Instances - Amazon Elastic Compute Cloud," 2017. Aufgerufen von http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/using-network-security.html

[71] Amazon, ,,Elastic IP Addresses - Amazon Elastic Compute Cloud," 2017. Aufgerufen von http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/elastic-ip-addresses-eip.html

[72] camunda, ,,Custom Date Format," 2017. Aufgerufen von https://docs.camunda.org/manual/7.6/reference/rest/overview/date-format/

[73] eclipse, ,,Eclipse IDE for Java EE Developers," 2017. Aufgerufen von http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/neon3

[74] Maven, ,,Maven - Home," 2017. Aufgerufen von https://maven.apache.org/

Abkürzungsverzeichnis

AWS Amazon Web Services. 62, 81

BPaaS Business Process as a Service. 1–9, 11–15, 17–19, 21–24, 26, 32, 39, 40, 45, 47–49, 51, 53–55, 61–63, 65–69, 81, 83, 85

BPM Business Process Management. 5

BPMN Business Process Model and Notation. 5, 7, 10, 21, 28, 30, 31, 35, 63, 64, 69

BPMS Business Process Management System. 2, 3, 7, 64

CMMN Case Management Model and Notation. 7, 38

CSS Cascading Style Sheets. 47

CSV Comma Separated Values. 52, 68

DNS Domain Name System. 85

EKG Elektrokardiogramm. 22–24

FTP File Transfer Protocol. 82

IDE Integrated Development Environment. 87

IS Information Systems. 14

JAR Java Archive. 88

JDBC Java Database Connectivity. 51

JPDA Java Platform Debugger Architecture. 88

JSON JavaScript Object Notation. 46

JVM Java Virtual Machine. 88

LDAP Lightweight Directory Access Protocol. 11

MTA Multi-Tenancy Architektur. 48

RAM Random-Access Memory, Arbeitsspeicher. 81

SaaS Software as a Service. 1

SMTP Simple Mail Transfer Prototcol. 80, 86

UML Unified Modeling Language. 12, 33, 59

URL Uniform Resource Locator. 46, 84

UUID Universally Unique Identifier. 35

WAR Web Application Archive. 35, 53

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Produktivumgebung

Als produktive Umgebung wird ein Windows Server 2012 R2 verwendet, welche als virtuelle Instanz über Amazon Web Services (AWS) bezogen wird. Dazu wird für die Dauer der Master Thesis eine Instanz vom Typ *t2.micro* verwendet (Free Tier) [65]. Der Server besitzt 1 GB RAM (Random-Access Memory), eine 2.4 GHz Intel Xeon CPU und 30 GB Festplattenspeicher. Diese Konfiguration ist nicht dazu gedacht, die BPaaS Plattform produktiv zu betreiben, sondern dient zur Entwicklung und Test vom BPaaS Plattform Prototyp. Mit nur 1 GB RAM verschlechtert sich die Performance spürbar, wenn einige Process Engines (Mandanten) parallel betrieben werden.

Die Windows Server 2012 R2 Instanz ist unter folgenden Angaben erreichbar, sofern die Instanz am Laufen ist (vergleiche Tabelle A.1).

Public DNS Name	`ec2-35-157-223-169.eu-central-1.compute.amazonaws.com`
Benutzername	`Administrator`
Passwort	`7Lzvm-QKZh?`

Tabelle A.1: Angaben zur AWS Instanz

A.1 Installation und Konfiguration

Für die Installation und Konfiguration der produktiven Umgebung wurden auf der Server Instanz folgende Schritte durchlaufen. Als Anleitung dazu diente [52].

1. Download von Java JRE 1.8 [66].
2. Installation von Java JRE 1.8 nach `C:\Java\jre1.8`.
3. Hinzufügen von `C:\Java\jre1.8` zur `PATH` Umgebungsvariable.
4. Setzen von `JAVA_HOME` Umgebungsvariable auf den Wert `C:\Java\jre1.8`.
5. Download von MySQL Server Version 5.7.17 [67].
6. Installation von MySQL Server Version 5.7.17 und MySQL Workbench Version 6.3.8 über den Installer (siehe Abschnitt A.2).
7. Download von Camunda Distribution Version 7.6.0, welche einen Tomcat Server enthält [68].
8. Entpacken von Camunda Distribution nach `C:\bpaas\camunda-bpm-tomcat-7.6.0`.
9. Download von MySQL JDBC Treiber Version 6.0.5 (`mysql-connector-java-6.0.5.jar`) [69].
10. Kopieren vom MySQL JDBC Treiber nach `C:\bpaas\camunda-bpm-tomcat-7.6.0\server\apache-tomcat-8.0.24\lib`.
11. Übertragen vom BPaaS Ad Hoc Prozess (`adhocprocess.war`), BPaaS On-Boarding (`registration.war`) und Cockpit Plugin (`cockpitplugin.jar`) auf die AWS Instanz (z.B. per FTP).
12. Kopieren vom BPaaS Ad Hoc Prozess (`adhocprocess.war`) nach `C:\bpaas\camunda-bpm-tomcat-7.6.0\server\apache-tomcat-8.0.24\webapps`.
13. Kopieren von der BPaaS On-Boarding (`registration.war`) nach `C:\bpaas\camunda-bpm-tomcat-7.6.0\server\apache-tomcat-8.0.24\webapps`.

14. Kopieren vom BPaaS Cockpit Plugin (`cockpitplugin.jar`) nach `C:\bpaas\camunda-bpm-tomcat-7.6.0\server\apache-tomcat-8.0.24\webapps\camunda\WEB-INF\lib`.

15. Ändern vom Datumsformat, welches vom REST API genutzt und zurückgegeben wird (siehe Abschnitt A.5).

16. Einrichten von Admin Datenbank und Anpassen der On-Boarding Konfiguration (siehe Abschnitt A.3)

Nachdem diese Schritte durchgeführt wurden, kann nun Camunda aufgestartet werden. Dazu findet sich im Ordner `C:\bpaas\camunda-bpm-tomcat-7.6.0` eine Datei `start-camunda.bat`. Nun ist die On-Boarding Web-Applikation lokal unter `http://localhost:8080/registration` erreichbar. Von dort aus kann auf der BPaaS Plattform ein Mandant eröffnet werden.

A.2 MySQL Server Installation

Nachfolgend die Detailangaben zur Installation und Konfiguration des MySQL Servers (vergleiche Tabelle A.2). Gegebenenfalls müssen bei der Installation vom MySQL Server noch gewisse Vorbedingungen installiert werden (z.B. C++ 2013 Runtime). Diese Vorbedingungen können in der Regel direkt über den MySQL Server Installer installiert werden. Der admin Benutzer wird zum einen aus der On-Boarding Web-Applikation genutzt, um auf die MySQL Datenbank zu verbinden. Zum anderen wird dieser Benutzer auch benutzt, um aus den Camunda Process Engine Instanzen der verschiedenen Mandanten auf die jeweiligen Mandanten Datenbanken zu verbinden. Sollte dieser Benutzer oder das Passwort ändern, so muss in folgender Klasse dieses Änderung nachgetragen werden:
`ch.boller.bpaas.registration.service.TenantService`.

Config Type	Server Machine
MySQL Root Passwort	`!mMawdHuibG17.`
Benutzername	`admin` (DB Admin Role)

Passwort	`!aDmBpaaS17.`
Als Windows Service installieren?	Ja
Service Name	MySQL57

Tabelle A.2: Angaben zum MySQL Server

A.3 Einrichten von Admin Datenbank

Um die `bpaas_admin` Datenbank einzurichten, kann das SQL Script `bpaas_admin.sql` ausgeführt werden. Dieses Script ist im Projekt *registration* im Ordner *sql* im git Repository eingecheckt. Das Script erstellt die Datenbank und darin zwei Tabellen:

1. `settings`: In dieser Tabelle werden die Einstellungen der BPaaS On-Boarding Web-Applikation abgelegt.
2. `tenant`: In dieser Tabelle werden Metadaten zu jedem registrierten Mandanten abgelegt.

Die Werte in der Tabelle `settings` müssen nach dem Ausführen vom SQL Script auf die Umgebung angepasst werden. Die Tabelle enthält lediglich beispielhafte Default Werte. Ein besonderes Augenmerk muss auf die Einstellung `REGISTRATION_CONFIRMATION_EMAIL_PROCESS_ENGINE_URL` geworfen werden. Hier wird über die Zeichenfolge `%1$s` zur Laufzeit der Name der Process Engine des Mandanten in die URL eingesetzt. Fehlt diese Zeichenfolge in der URL, so verweist die URL nicht auf die entsprechende Process Engine.

A.4 Einrichten von Zugriff auf Tomcat

Damit der Tomcat Webapplication Server auf dem Windows Server 2012 R2 von aussen zugreifbar wird, müssen zwei Inbound Rules konfiguriert werden. Zum einen auf dem Windows 2012 R2 Server direkt in der Windows Firewall und zum anderen in den Security Group Einstellungen der AWS Instanz:

1. Neue Inbound Rule einrichten für Port 8080 in der Windows Firewall vom Windows 2012 R2 Server.

2. Neue Inbound Rule einrichten für Port 8080 in der Security Group, welche die AWS Instanz nutzt [70].

Zudem muss eine Elastic IP Adresse [71] eingerichtet werden, damit der Public DNS (Domain Name System) Name auch nach einem Restart der AWS Instanz immer derselbe bleibt und somit die BPaaS Plattform immer unter dem selben DNS Name erreichbar bleibt.

A.5 Konfiguration von REST API

Damit das REST API Datumswerte im gewohnten Format zurückgibt, muss die Datei `web.xml` um folgende Abschnitte ergänzt werden [72]. Die Abschnitte müssen innerhalb vom `web-app` Tag reinkopiert werden. Die Datei `web.xml` befindet sich im Tomcat Ordner `webapps\engine-rest\WEB-INF`.

```
<listener>
    <listener-class>
        org.camunda.bpm.engine.rest.CustomJacksonDateFormatListener
    </listener-class>
</listener>

<context-param>
    <param-name>org.camunda.bpm.engine.rest.jackson.dateFormat</param-name>
    <param-value>dd.MM.yyyy HH:mm:ss</param-value>
</context-param>
```

A.6 E-Mail Benachrichtigung beim On-Boarding

Für die Korrespondenz von der BPaaS Plattform zum Unternehmen, welches einen BPaaS Plattform Zugang beantragt, wird folgende E-Mail Adresse verwendet (vergleiche Tabelle A.3). Änderungen dieser Angaben müssen in folgender Klasse vorgenommen werden: `ch.boller.bpaas.registration.service.EmailSender`.

Absender E-Mail Adresse	`bpaas-noreply@cmdesign.ch`
Passwort	`TWPxy8VYVTtZHcQr`
Host	`w00c6bed.kasserver.com`
Port	25

Tabelle A.3: Angaben zum SMTP Server

Anhang B

Entwicklungsumgebung

Grundsätzlich können für das Aufsetzen einer Entwicklungsumgebung die gleichen Schritte durchlaufen werden wie bei der produktiven Umgebung (siehe Anhang A). Die Amazon AWS spezifischen Schritte können ausgelassen werden. Eine Entwicklungsumgebung wurde lokal auf einem Laptop (12 GB RAM, 2.1 GHz CPU) installiert.

B.1 Integrated Development Environment

Als Integrated Development Environment (IDE) wurde eclipse JEE Neon (Neon.1a) verwendet [73]. Des Weiteren wurden die Plugins *m2e* als Maven Integration und *EGit* zur Verwaltung der git Repositories eingesetzt.

B.2 Source Code

Als Source Code Repository wurde git eingesetzt. Dabei wurde ein privates (nicht öffentliches) git Repository erstellt. Das git Repository ist unter folgender URL erreichbar:

`https://gitlab.com/miguelgalaxy/bpaas-platform`

Innerhalb des oben erwähnten git Repositories existieren drei eclipse Maven Projekte. *adhocpress* enthält die Logik des Ad Hoc Prozesses, welches zu einer WAR Datei kompiliert wird, welche wiederum auf Tomcat deployed werden kann. Das Projekt *registration* enthält die On-Boarding Logik und stellt ein Servlet / JSP Web-Applikation bereit.

Auch hier wird das Projekt zu einer WAR Datei kompiliert. Das Projekt *cockpitplugin* registriert ein Plugin in das Camunda Cockpit. So können im Cockpit die Details über die Ad Hoc Prozesse in Ausführung sowie beendeten Ad Hoc Prozesse eingesehen werden. Das Projekt wird zu einer JAR Datei kompiliert. Als Build Tool kommt Maven [74] zum Einsatz.

B.2.1 Unit Tests

Der Code wurde mittels Unit Tests getestet. Die Unit Tests können aus eclipse heraus ausgeführt werden. Zudem ist durch die Verwendung von Maven sichergestellt, dass bei einem Deployment auf den Application Server die Unit Tests ausgeführt werden. Nur wenn die Unit Tests erfolgreich durchlaufen findet auch ein Deployment statt.

B.2.2 Remote Debugging

Bei der Entwicklung wurde eine vorpaketierte Distribution von Camunda (Version 7.6.0) mit integriertem Tomcat Application Server (Version 8.0.24) verwendet. Damit aus der eclipse Entwicklungsumgebung heraus der entwickelte Code debugged werden kann, muss auf Seite von Tomcat der JPDA (Java Platform Debugger Architecture) Debugger Server in der Tomcat JVM aktiviert werden. Die Batch Scripts in der Camunda Distribution enthalten bereits die entsprechenden Anweisungen. Damit der JPDA Debugger Server beim Aufstarten von Camunda mitgestartet wird, ergänzt man den Aufruf von `catalina.bat` in der Datei `startup.bat` um das Command Line Argument `jpda` (vergleiche Abbildung B.1).

B.3 Dokumentation

Auch dieses Master Thesis Dokument ist in einem privaten git Repository eingecheckt. Das git Repository ist unter folgender URL erreichbar:

`https://gitlab.com/miguelgalaxy/master-thesis`

```
rem     http://www.apache.org/licenses/LICENSE-2.0
rem
rem Unless required by applicable law or agreed to in writing, software
rem distributed under the License is distributed on an "AS IS" BASIS,
rem WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
rem See the License for the specific language governing permissions and
rem limitations under the License.

rem ---------------------------------------------------------------------------
rem Start script for the CATALINA Server
rem ---------------------------------------------------------------------------

setlocal

rem Guess CATALINA_HOME if not defined
set "CURRENT_DIR=%cd%"
if not "%CATALINA_HOME%" == "" goto gotHome
set "CATALINA_HOME=%CURRENT_DIR%"
if exist "%CATALINA_HOME%\bin\catalina.bat" goto okHome
cd ..
set "CATALINA_HOME=%cd%"
cd "%CURRENT_DIR%"
:gotHome
if exist "%CATALINA_HOME%\bin\catalina.bat" goto okHome
echo The CATALINA_HOME environment variable is not defined correctly
echo This environment variable is needed to run this program
goto end
:okHome

set "EXECUTABLE=%CATALINA_HOME%\bin\catalina.bat"

rem Check that target executable exists
if exist "%EXECUTABLE%" goto okExec
echo Cannot find "%EXECUTABLE%"
echo This file is needed to run this program
goto end
:okExec

rem Get remaining unshifted command line arguments and save them in the
set CMD_LINE_ARGS=
:setArgs
if ""%1""==""" goto doneSetArgs
set CMD_LINE_ARGS=%CMD_LINE_ARGS% %1
shift
goto setArgs
:doneSetArgs

call "%EXECUTABLE%" jpda start %CMD_LINE_ARGS%

:end
```

Abbildung B.1: Remote Debugging aktivieren