Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

LUCAS LÖFFLER
MASTER THESIS

EINSATZ EINER WORKFLOW-ENGINE IN DER FINANZINDUSTRIE

Eingereicht am 24. Mai 2018

Betreuer: Prof. Dr. Dirk Riehle, M.B.A.
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Erlangen, 24. Mai 2018

License

Erlangen, 24. Mai 2018

Abstract

Various kinds of business processes play a major role in the context of financial industry. *Business Process Management Systems* or also called *Workflow Engines* are therefore powerful software tools for controlling and automating business processes. More and more companies are interested and willing to use them. This thesis considers a technical excerpt of the IT landscape of a German financial institution and presents a possible prototypical implementation of a workflow management system, which offers the possibility to automate various business processes, previously modeled in a technical notation.

Zusammenfassung

Verschiedenartige Geschäftsprozesse spielen in der Finanzindustrie eine große Rolle. *Business Process Management Systeme* bzw. *Workflow-Engines* sind mächtige Software-Werkzeuge zur technischen Steuerung und Automatisierung von Geschäftsprozessen. Diese Arbeit betrachtet einen fachlichen Ausschnitt der IT-Landschaft eines deutschen Finanzdienstleisters und stellt eine mögliche prototypische Implementierung eines Workflow-Management-Systems vor, das die Möglichkeit zur Automatisierung verschiedener Geschäftsprozesse bietet, die zuvor in einer fachlichen Notation modelliert wurden.

Inhaltsverzeichnis

1 Einleitung

Unternehmen sind bestrebt, die Effizienz und Qualität ihrer Geschäftsprozesse zu optimieren. Aus diesem Grund ist das Geschäftsprozessmanagement, auch als Business Process Management bezeichnet, ein kontinuierlich anhaltender Trend, der in den letzten Jahren durch immer fortschrittlichere Softwaresysteme in diesem Bereich sogar noch verstärkt wurde.

Die Grundidee des Prozessmanagement ist es, mithilfe standardisierter, ausdrucksstarker Notationen Prozessmodelle zu erstellen, die einen Geschäftsprozess zunächst aus einer fachlichen Sicht beschreiben. Technische Details, die eine Rolle bei der Prozessautomatisierung spielen, werden nachträglich durch Entwickler hinzugefügt, die neben der fachlichen Sicht auch das Wissen über die IT-Infrastruktur besitzen. Diese vorgesehene Trennung der abstrakteren Prozessmodellierung von der technischen Spezifizierung bringt den Vorteil, dass sich Mitarbeiter in Unternehmen auch abteilungsübergreifend über Prozessmodelle verständigen und abstimmen können und es im besten Fall zu weniger Missverständnissen und Verständnisschwierigkeiten untereinander kommt. Auch können mögliche Fehler in Geschäftsprozessen so früher erkannt und korrigiert werden (Komus, 2011, S. 3ff.)

Auch in der Finanzindustrie und im Bankenwesen spielen Geschäftsprozesse und deren Verbesserung eine immer wichtige Rolle. Laut einer Banken-IT-Studie aus dem Jahr 2009 (Schaffry, 2009) planten schon im Jahr 2009 mehr als die Hälfte aller in der Studie befragten Banken umfangreiche Investitionen im Bereich Geschäftsprozess-Optimierung. Dabei erhoffen sich die Banken insbesondere Verbesserungen für die Prozesse im Vertrieb und im Kundenmanagement. Für viele Banken sind darüber hinaus auch Maßnahmen im Kreditgeschäft sowie beim Risikomanagement denkbar und somit also in entscheidenden Kernbereichen des Bankengeschäfts. Schließlich kommt die Studie auch zu dem Ergebnis, dass es den Banken auch darum geht, ihre zum Teil sehr heterogenen IT-Landschaften zu vereinheitlichen. 40 Prozent der Banken hielten es schon zum damaligen Zeitpunkt für möglich, BPM-Systeme einzusetzen. Angesichts des Alters der Studie und des schnellen Fortschritts in diesem Bereich kann davon ausgegangen werden, dass sich dieser Wert seit 2009 klar erhöht hat und dass viele Banken bereits

Software zur Steuerung ihrer Geschäftsprozesse im Einsatz haben.

1.1 Zielsetzung

Bei der Einführung eines BPM-Systems bei einem Unternehmen kann es hilfreich sein, die generelle Machbarkeit und die Vorteile einer umfangreicheren Umwälzung der IT-Landschaft zunächst einmal explorativ im Rahmen kleinerer Projekte zu untersuchen. Dies soll im Rahmen dieser Arbeit geschehen. Es soll ein Ausschnitt des IT-Systems eines deutschen Finanzinstituts betrachtet werden und dessen Optimierungsmöglichkeit hinsichtlich der Verwendung eines BPM-Systems untersucht werden. Prototypisch soll zudem eine solche BPM-Lösung entworfen und implementiert werden und gemäß zuvor definierter Anforderungen bewertet werden.

1.2 Aufbau der Arbeit

Im nun Folgenden Kapitel werden zunächst einige Grundbegriffe im Bereich des Geschäftsprozessmanagement geklärt und es wird kurz das Camunda BPM-System vorgestellt, das im Rahmen der Implementierung später verwendet wird.
In Kapitel 3 wird der fachliche Bereich im Kontext der Bank beschrieben, den das zu entwickelnde System mithilfe einer Workflow-Engine unterstützen soll. In diesem Kapitel werden auch die Anforderungen an die Implementierung gestellt, unterteilt in funktionale und nicht-funktionale Anforderungen.
In Kapitel wird die Architektur der Camunda-Workflow-Engine erläutert und es wird näher auf einige Design-Entscheidungen eingegangen, die bei der Implementierung von Relevanz waren.
In Kapitel 5 werden die wichtigsten Implementierungsaspekte gezeigt sowie auf Besonderheiten bei der Implementierung eingegangen.
In Kapitel 6 wird eine Bewertung der Implementierung hinsichtlich der zuvor gestellten Anforderungen vorgenommen.
In Kapitel 7 wird schließlich ein kurzes Fazit gezogen und ein Ausblick auf weitere mögliche Schritte zur Etablierung einer Workflow-Engine im Unternehmen gegeben.

2 Grundlagen

2.1 Grundlegende Begriffe

Die folgenden Abschnitte dienen der Klärung einiger grundlegender Begriffe im Bereich Prozessmanagement.

2.1.1 BPM

Das Geschäftsprozessmanagement oder englisch Business Process Management beschäftigt sich mit der Modellierung, Ausführung und Verwaltung operationaler Geschäftsprozesse. (Van der Aalst, 2013)

2.1.2 BPMS

Ähnlich wie Datenbanksysteme zur Speicherung und Verwaltung von Daten können Prozessmanagementsysteme (englisch: Business Process Management Systems, kurz BPMS) zu den Standard-Typen im Bereich der Software-Systeme gezählt werden. (Dumas, La Rosa, Mendling & Reijers, 2013) Der Fokus dieser Systeme liegt dabei auf der Automatisierung von Geschäftsprozessen auf der Grundlage eines formalen Prozessmodells. Van der Aalst (2013) definiert BPMS als Software-Systeme, die durch ein eindeutiges Prozessdesign in der Lage sind, operationale Geschäftsprozesse auszuführen und zu verwalten.
Dabei unterscheiden sich die Systeme der einzelnen Hersteller vor allem darin, in welchem Umfang sie auch über die reine Prozessautomatisierung hinaus eine Unterstützung für den Anwender bieten, beispielsweise in Form von Prozess-Monitoring oder Prozess-Mining.
Systeme, deren Funktionalität sich auf die Prozessautomatisierung beschränkt, wurden vor allem früher auch als Workflow Management Systeme (WFMs) bezeichnet, für die wiederum im englischen Sprachraum häufig der prägnante Begriff

Workflow Engine verwendet wird, der sich auch im Titel dieser Arbeit wiederfindet. Alle Begriffe werden höchst uneinheitlich verwendet (Allweyer, 2014). Wenn in dieser Arbeit von Workflow Engine die Rede ist, so sind damit Systeme im Sinne vollständiger BPMS gemeint.

2.1.3 Vorteile des Einsatzes von Workflow-Engines

Für Unternehmen ergeben sich verschiedene Vorteile aus dem Einsatz von einer Workflow-Engine (Dumas et al., 2013). Dazu zählen:

a) Reduktion des Arbeitsaufwands

b) Flexible Systemintegration

c) Transparenz bei der Ausführung

d) Striktere Durchsetzung von Regeln

2.1.4 Herausforderungen beim Einsatz von Workflow-Engines

Auf der anderen Seite ergeben sich auch Herausforderungen aus dem Einsatz einer Workflow-Engine, die berücksichtigt werden sollten und gegeüber anderen Möglichkeiten der Prozessunterstützung abgewogen werden sollten (Allweyer, 2014). Dazu zählen:

a) Hohe Gesamtkomplexität durch eine weitere Schicht in der Architektur

b) Hohe Ansprüche an die Entwickler

c) Fehlende Datenintegration

d) Tendenziell geringere Performance

e) Eventuell geringe Benutzerakzeptanz innerhalb des Unternehmens

2.2 Prozess-Modellierung mit BPMN

BPMN steht für *Business Processing Modeling Notation* und ist eine standardisierte, grafische Prozessnotation. Seit 2005 wird die BPMN von der Object Management Group (OMG), die auch für den Modellierungsstandard im Softwaredesign UML bekannt ist, übernommen. Im Februar 2011 wurde die aktuell geltende BPMN-Version 2.0 verabschiedet und diese im September 2013 auch offiziell als ISO-Standard publiziert. Der große Vorzug von BPMN gegenüber anderen vergleichbaren Sprachen ist, dass es direkt in einer Workflow-Engine zur

Ausführung gebracht werden kann, ohne zuvor in ein anderes Format konvertiert werden zu müssen, was unglückliche Mappings zur Folge hatte. Ein weiterer wichtiger Vorteil ist die bereits oben genannte Standardisierung. Eine Übersicht zum Thema BPMN 2.0 findet sich in Freund und Rücker (2016).

2.3 Camunda BPM

Die in diesem Abschnitt genannten Informationen finden sich in der Dokumentation von Camunda[1] sowie in Bischoff und van Dinther (2016). In dieser Arbeit soll als Workflow-Engine Camunda BPM, das Business-Process-Management-System der Camunda Services GmbH genutzt werden. Das Unternehmen mit Sitz in Berlin arbeitete ab 2008 auch an der Definition des BPMN 2.0-Standards mit und entwickelt Camunda als quelloffenes Softwareprodukt unter der Apache Lizenz 2.0. Die Camunda Workflow-Engine kann daher kostenlos privat und in Unternehmen verwendet werden. Darüber hinaus bietet Camunda auch eine Enterprise Edition seiner BPM-Software an, die neben einem umfangreicheren Support für Unternehmen auch weitere Funktionalitäten der Software bietet.
Camunda bietet eine gute Unterstützung für die Entwicklung im Java-Umfeld und lässt sich gut mit Frameworks wie Spring oder Spring Boot integrieren. Im Folgenden soll kurz auf die wichtigsten Komponenten von Camunda BPM eingegangen werden.

Workflow Engine

Die Workflow Engine oder auch Process Engine, die für die Abarbeitung der Prozesse zuständig ist. Details zum Aufbau und zur Architektur der Workflow Engine von Camunda folgen in Kapitel 4.

Camunda Cockpit

Beim Camunda Cockpit handelt es sich um eine Webanwendung für Administratoren. Es können damit die laufenden Prozessinstanzen eingesehen und verwaltet werden. Die modulare Architektur des Cockpits erlaubt es, dieses in seiner Funktionalität durch Plugins zu erweitern.

Camunda Tasklist

Die Camunda Tasklist ist ebenfalls eine Webanwendung, die die Basisfunktionalität bereit stellt, um User Tasks zu bearbeiten. Dabei sind verschiedenen Filter-Einstellungen möglich und es lassen sich auch hier individuelle Anpas-

[1] https://docs.camunda.org/manual/7.8/

sungen vornehmen. Die Tasklist kann als eine Art Demo-Applikation angesehen werden. In der Regel (so auch im Rahmen dieser Arbeit) wird sie durch ein eigens entwickeltes Frontend ersetzt, was sich besser in den unternehmenseigenen Kontext einfügt und an die Design-Vorgaben geknüpft ist.

Camunda Modeler

Der Camunda Modeler ist eine Desktop-Anwendung zur Modellierung von BPMN 2.0-Workflows. Die damit erstellten Modelle lassen sich im Format .bpm speichern und sind auf einer Workflow-Engine, wie zum Beispiel Camunda, deployfähig. Der Modeler unterstützt darüber hinaus auch andere Sprachkonstrukte, wie die Standards CMMN 1.1 und DMN 1.1.

2.4 Vergleichbare Arbeiten

Vergleichbare Ansätze zur Integration von BPM-Anwendungen im weitesten Sinne in eine komplexe Geschäftsanwendung finden sich in zahlreichen weiteren Studien- und Abschlussarbeiten, so zum Beispiel Zahn (2013), Süß (2012) sowie Lotz (2015).

3 Anforderungen

In diesem Abschnitt wird zunächst das betrachtete IT-System und dessen Problemdomäne im Kontext der Finanzindustrie aus fachlicher Sicht beschrieben. Anschließend wird kurz auf die technische Realisierung des aktuellen Systems eingegangen sowie auf dessen Schwächen. Beide Abschnitt dienen dem besseren Verständnis für die Anforderungen an den Entwurf und an die Implementierung, die in Abschnitt 3.3 gestellt werden. Dabei wird eine Unterscheidung zwischen funktionalen und nicht-funktionalen Anforderungen vorgenommen.

3.1 Fachliche Beschreibung des Systems

Die IT-Landschaft von Unternehmen im Finanzbereich kann im Allgemeinen als sehr komplex bezeichnet werden. Vor allem die Heterogenität verschiedener in sich greifender Einzelanwendungen und Subsysteme spielt dabei eine große Rolle, aber auch der hohe fachliche Anspruch der in der Software realisierten Geschäftsprozesse selbst.
Im Rahmen dieser Arbeit kann daher nur ein Teilbereich der IT-Infrastruktur der betroffenen Bank betrachtet werden. Bei der Auswahl dieses Bereichs unter den zahlreichen möglichen wurde vor allem auf seine exemplarische Relevanz Wert gelegt sowie auf die Ganzheitlichkeit der dort auftretenden Geschäftsvorfälle. So gibt es im Finanzwesen typischerweise viele Geschäftsprozesse, die in Dunkelverarbeitung ablaufen. Solche Prozesse sind zwar durchaus relevant im Bezug auf die Automatisierung, können jedoch das Spektrum der Möglichkeiten eines BPM-Systems nicht ausreichend abbilden. Andere Geschäftsbereiche, die das Kernbankengeschäft unmittelbar betreffen, wären zwar interessant für eine Betrachtung, konnten aber aufgrund hoher betrieblicher Hürden (missionskritische Software-Bestandteile) nicht ohne Weiteres für „Experimente“ freigegeben werden.
Die Entscheidung für einen Teilbereich fiel letztendlich auf ein System, das im Folgenden als *Dokumenten-Workflow-Archiv (kurz: DWA)* bezeichnet wird. Das DWA ist für all jenen Geschäftsprozesse zuständig, die sich aus Dokumenten er-

geben, die dem System geliefert werden. Dokumente können dabei entweder in Form von eingescannten Papierdokumenten oder als Fax direkt an das DWA übermittelt werden. Jedem Dokument ist ein eindeutiger Dokumenttyp zugeordnet, woraus das DWA selbstständig den daraus hervorgehenden Geschäftsprozess, der mit dem Dokument verknüpft ist, ableitet. Der Namensbestandteil Archiv weist darauf hin, dass es sich beim DWA zudem um ein Archivsystem handelt, das die ankommenden Dokumente nicht nur verarbeitet, sondern auch archiviert.

3.2 Aktuelles System

Abbildung 3.1 zeigt schematisch den Aufbau des aktuellen DWA-Systems. Auf den Eingang von Dokumenten durch einen externen Dienstleister erfolgt eine Archivierung der Dokumente in einem Dokumentmanagementsystem. Hierfür wird das IBM FileNet Content Management System verwendet. Über eine eigene Zuordnungslogik im Auftragszuordnungssystem wird anschließend abhängig vom Dokumenttyp und weiteren Metadaten des Dokuments eine Postkorb-Zuordnung vorgenommen. Postkörbe sind dabei im aktuellen System eine wichtige Komponente und müssen verstanden werden als Sammlung aller Vorgänge zu einem Geschäftsvorfall. Sachbearbeiter erhalten über den FileNet Workplace je nach Berechtigung eine Einsicht in bestimmte Postkörbe und können die Vorgänge innerhalb ihrer Postkörbe bearbeiten.

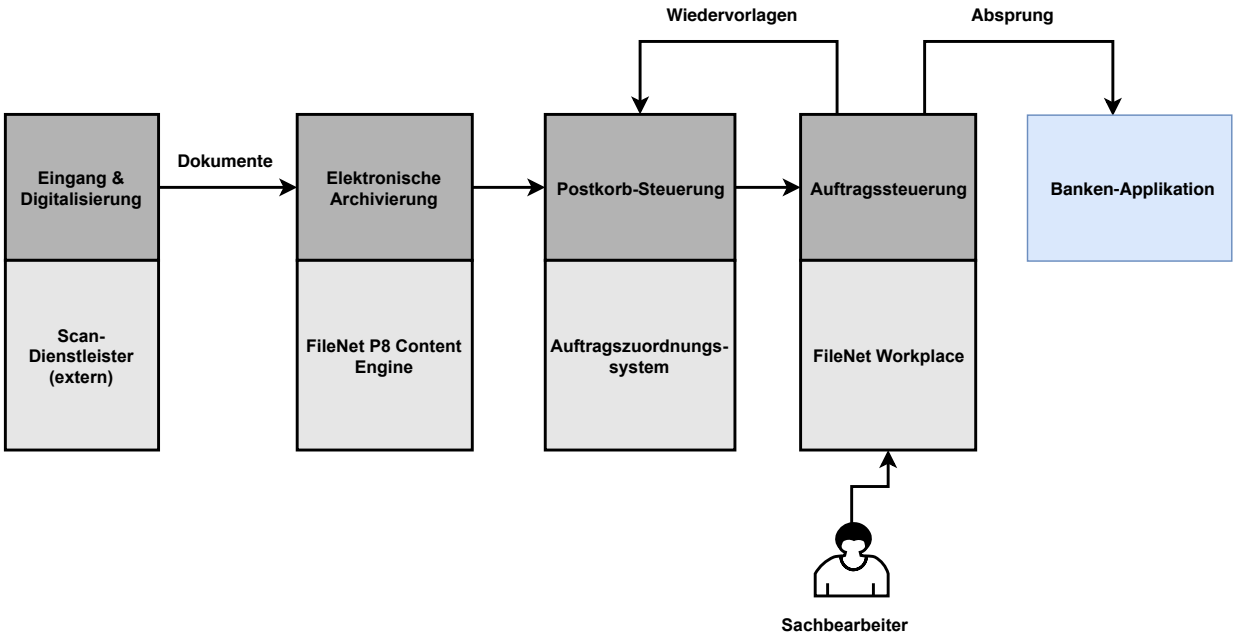

Abbildung 3.1: DWA-System (schematisch)

3.2.1 Schwächen des aktuellen Systems

Hardcodierte Prozessmodellierung

Die meisten der Prozess-Abläufe im aktuellen DWA-System sind hardcodiert und fest im Code verdrahtet.

Keine nachvollziehbaren Prozess-Instanzen

Wie oben erwähnt, kann ein Vorgang über Wiedervorlagen eine Reihe weiterer Vorgänge erzeugen. Die Einheit eines Vorgangs als zusammenhängender Geschäftsvorfall geht dabei sowohl in der Wahrnehmung für die Sacharbeiter als auch im System selbst verloren.

Kaum Monitoring-Möglichkeiten

Das jetzige DWA-System bietet kaum Möglichkeiten zur systematischen Auswertung von Prozess-Statistiken. Schwachstellen im Prozess-Ablauf oder proaktive Optimierungen aufgrund von Engpässen sind somit nicht umsetzbar.

Wenig Flexibilität

Allgemein bietet die momentane Lösung wenig Flexibilität in Bezug auf Austauschbarkeit einzelner Komponenten sowie die Anpassbarkeit des Gesamtsystems an neue Anforderungen.

3.3 Anforderungen

Die oben erläuterten Schwächen des aktuellen DWA-Systems zusammen mit dem geplanten Einsatz einer Workflow-Engine führen zu einer Reihe an Anforderungen an ein neues System *DWA-Neu*. Dabei wird wie üblich zwischen funktionalen Anforderungen, die die gewünschten Funktionalitäten beschreiben und nichtfunktionalen Anforderungen, die die Anforderungen an die Qualität beschreiben, unterschieden.

3.3.1 Funktionale Anforderungen

Fachliche Prozessmodellierung

Workflows sollen nicht mehr hartcodiert im System verankert werden, stattdeesen soll das System die Möglichkeit bieten, fachliche Prozessmodelle zu verwerten und auszuführen. Dabei sollen Prozessmodelle im BPMN 2.0-Standard akzeptiert werden.

Benutzeroberfläche zur Prozesssteuerung

Ähnlich wie das aktuelle *DWA*-System soll auch das neue System eine Benutzeroberfläche bieten, die es ermöglicht, Vorgänge zu bearbeiten und die dazu gehörigen Dokumente zu einzusehen. Die logische Aufteilung der Vorgänge auf virtuelle Postkörbe sollte dabei beibehalten werden.

Monitoring-Funktionen

Ein Geschäftsprozess-Monitoring (GPM) des bestehenden Systems soll auch im neuen System integriert werden. Die Monitoring-Möglichkeiten des verwendeten BPM-Systems sollen soweit wie möglich ebenfalls integriert und genutzt werden.

Vermeidung von Redundanz bei Prozessmodellen

Verschiedene Geschäftsprozesse, die sich nur in wenigen Einzelschritten und Untervorgängen unterscheiden, sollen nicht einzeln in separaten Prozessmodellen behandelt werden. Stattdessen soll es möglich sein, die Unterschiede der einzelnen Geschäftsprozesse in möglichst wenig Modellen zusammen zu fassen und so die Anzahl notwendiger Prozessmodelle zu minimieren.

Transaktionssicherheit innerhalb von Prozessen

Prozesse sollen transaktionssicher umgesetzt werden. Das heißt, Prozessinstanzen dürfen unter keinen Umständen im Laufe ihres Lebenszyklus „verloren“ gehen. Bei Fehlern soll die laufende Prozessinstanz zurückgerollt werden.

Testabdeckung

Die Funktionalität des Systems soll durch Unit-Tests sowie Integrationstests sichergestellt werden.

3.3.2 Nicht-funktionale Anforderungen

Möglichkeit zur Integration in bestehende Infrastruktur

Das System sollte die Möglichkeit bieten, in die bestehende IT-Infrastruktur eingebunden werden.

Erhöhung der Effizienz

Bei allen Schritten, die einen manuellen Eingriff von Sachbearbeitern erfordern (Human Tasks) soll das System zu einer Erhöhung der Effizienz bei der Erledigung der Aufgaben durch die Sachbearbeiter führen.

Geringere Lizenzkosten

Das System soll soweit möglich Lizenzkosten-Ersparnisse erbringen.

4 Architektur und Design

Im nachfolgenden Kapitel wird der Entwurf für die im Rahmen dieser Arbeit entwickelten Einsatz einer Workflow-Engine vorgestellt. Zunächst werden in Kapitel 4.1 einige grundlegende Komponenten von IT-Architekturen im Banken-Sekttor beschrieben, anschließend werden in Kapitel 4.2 der Aufbau und die Komponenten der Camunda Workflow-Engine erläutert. Die beiden verwendeten Prozessmodelle und deren Einbettung in die Gesamtarchitektur werden in den Kapiteln 4.3 bzw. 4.4 beschrieben.

4.1 Banken-IT-Architektur

Die IT-Infrastruktur von Banken und Finanzinstituten besteht klassischerweise aus dem Kernbanksystem, das sämtliche Abläufe auf der Ebene der einzelnen Konten verarbeitet, sowie einer Reihe auf dem Kernbanksystem aufgesetzter eigenentwickelte Individualsysteme. (Moormann, 2018)

4.2 Camunda-Architektur

Camunda BPM ist ein Java-basiertes Framework, bei dem die zentralen Komponenten selbst in Java geschrieben sind. Abbildung 4.1 zeigt den grundlegenden Aufbau der Architektur der Camunda Workflow Engine.

4.2.1 Public API

Die API erlaubt es, verschiedenen Programmen mit der Process Engine zu kommunizieren und diese zu steuern. Verschiedene Teilbereiche wie zum Beispiel Prozess- oder Aufgabenverwaltung sind dabei in jeweils eigene Services ausgelagert.

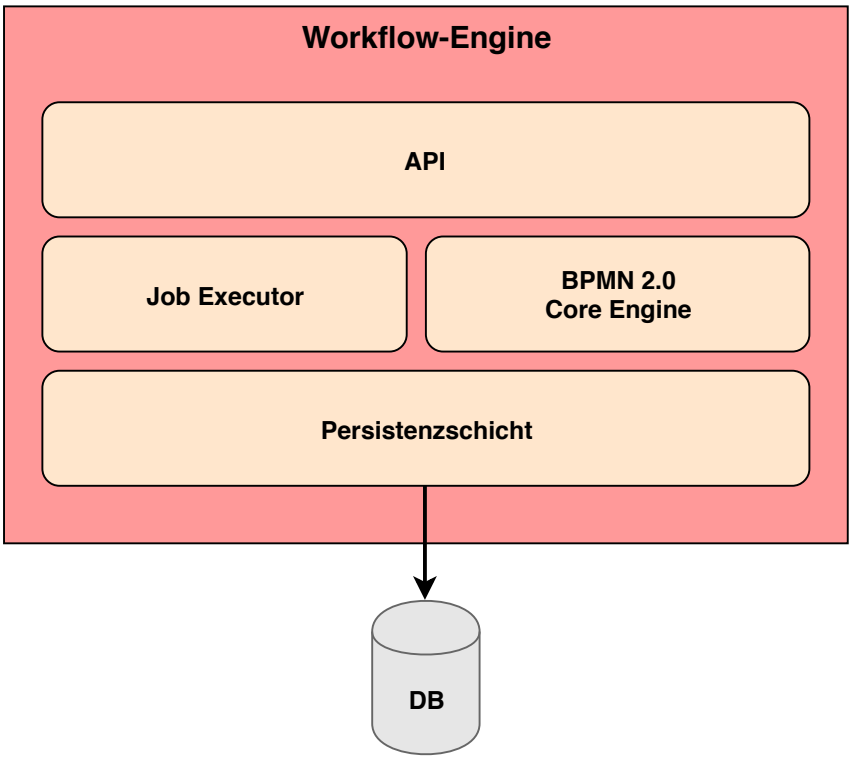

Abbildung 4.1: Architektur der Camunda Workflow Engine

4.2.2 BPMN 2.0 Core Engine

Die Core Engine ist das Herzstück der Workflow Engine. Sie besteht aus einer Ausführungs-Engine, einem BPMN 2.0 Parser, der die übergebenen BPMN 2.0 Dateien in Java-Objekte übersetzt sowie den Implementierungen zahlreicher spezifischer BPMN 2.0 Sprachkonstrukte, wie zum Beispiel Gateways oder Service Tasks.

4.2.3 Job Executor

Die als *Job Executor* bezeichnete Komponente der Workflow Engine dient der Ausführung und Koordination asynchroner Hintergrundprozesse, wie zum Beispiel Timern oder asynchron laufenden Verzweigungen in Prozessen.

4.2.4 Persistenzschicht

Die Workflow Engine enthält eine Persistenzschicht zur Speicherung von Zuständen einzelner Prozessinstanzen in einer relationalen Datenbank. Technisch

wird das objektrelationale Mapping der Java-Objekte auf die Datenbank-Tabellen durch das Open-Source-Persistenz-Framework *MyBatis* realisiert.

4.3 Einbettungsmöglichkeiten von Camunda

Camunda bietet verschiedene Möglichkeiten der Einbettung der Workflow Engine in eine bestehende Anwendung, die im Folgenden kurz diskutiert und abgewogen werden sollen.

4.3.1 Eingebettete Workflow Engine

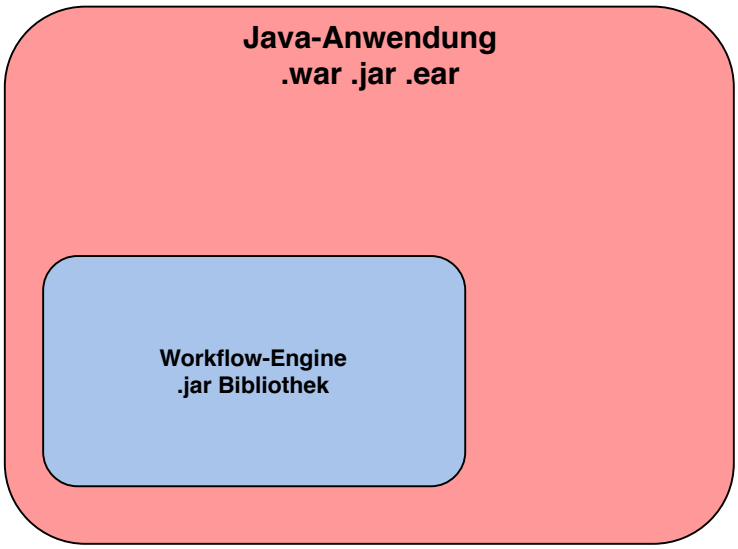

Abbildung 4.2: Architektur mit eingebetteter Workflow-Engine

Abbildung 4.2 zeigt die Möglichkeit der Einbettung der Workflow Engine in eine bestehende Java-Anwendung durch das Hinzufügen der Workflow-Engine als Java-Bibliothek. Dabei läuft die Workflow Engine nicht parallel neben der Anwendung, sondern ist in deren Kontext eingebettet und wird ebenso beendet, wenn die Anwendung beendet wird. Die Kommunikation zwischen der Anwendung und der Workflow Engine erfolgt dabei über eine Java API.

4.3.2 Geteilte Workflow-Engine

Wie in Abbildung 4.3 zu sehen ist, wird die Workflow Engine in diesem Fall auf dem selben Runtime Containers gestartet, auf dem auch die Hauptanwendung

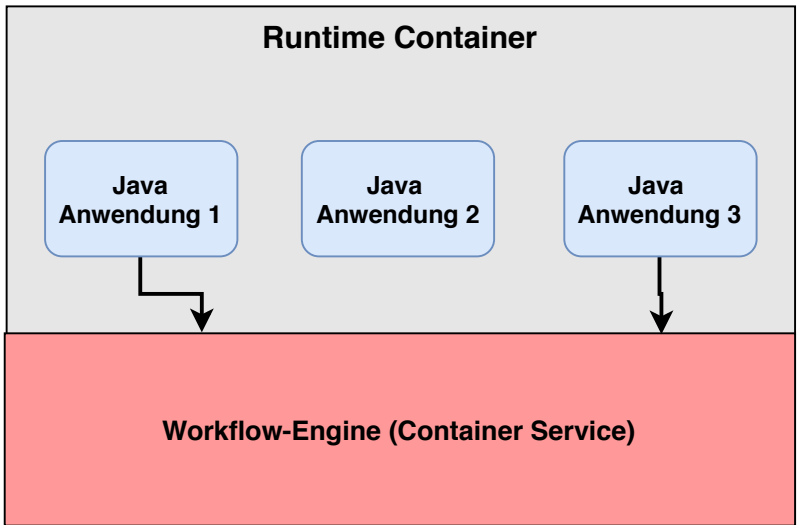

Abbildung 4.3: Architektur mit geteilter Workflow-Engine

(oder mehrere andere Anwendungen) laufen. Die Workflow Engine läuft dabei als *Container Service* und kann so durch alle Anwendungen auf dem Container genutzt werden. Dabei weiß die Workflow Engine über die Prozess-Definitionen, die von einer Anwendung genutzt werden, Bescheid.

4.3.3 Eigenständiger Workflow-Engine-Server

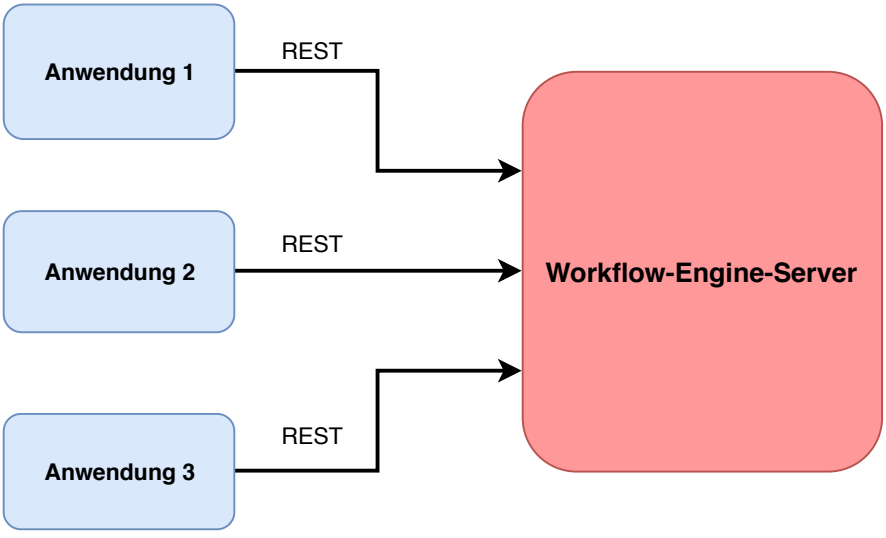

Abbildung 4.4: Architektur mit eigenständigem Workflow-Engine-Server

Bei diesem Architekturmodell (siehe Abbildung 4.4) wird die Workflow Engine unabhängig von der Hauptanwendung auf einem eigenen Server gestartet und

stellt über einen Netzwerk-Service eine Schnittstelle nach außen zur Verfügung. Dadurch können verschiedenen Anwendungen über Webservices mit der Workflow Engine kommunizieren. Zur Realisierung dieses Ansatzes kann die in Camunda integrierte REST API, die Teil des in Abschnitt 4.2.1 beschriebenen Public APIs ist, verwendet werden. Auch eine Kommunikation mit der Workflow Engine über SOAP oder JMS ist denkbar.

4.3.4 Weitere Modelle

Die Camunda Workflow-Engine kann auch über ein Cluster-Modell mit mehreren Knoten und einer geteilten Datenbank oder über ein *Multi-Tenancy*-Modell betrieben werden. Da diese Lösungen aber nicht in Frage kommen oder aufgrund technischer Restriktionen nicht realisiert werden können, wird hier nicht näher darauf eingegangen.

4.3.5 Diskussion zur Einbettung im Kontext

Die dargestellten Einbettungsmöglichkeiten führen zur Frage nach deren Vor- und Nachteilen und darüber hinaus, welche der vorgestellten Modelle sich im Fall der in dieser Arbeit vorgestellten Lösung am besten eignet.
Sowohl die eingebettete Workflow-Engine als auch die geteilte Workflow-Engine bieten den Vorteil einer einfachen Anwendung durch die Integration in eine bestehende Java-Anwendung. Auch lassen sich hier aufgrund der Modularität Unit-Tests gut umsetzen und die Einführung eines eventuell zusätzlichen Transaktions-Managements stellt kein Problem dar. Der Nachteil beider Ansätze ist die starke Bindung an die Java-Welt, sodass in Fällen, bei denen zwingend Services, die nicht in Java geschrieben sind, mit der Workflow-Engine kommunizieren müssen, auf den eigenständigen Workflow-Engine-Server (siehe 4.3.3) zurückgegriffen werden muss.
Im Fall der in dieser Arbeit entwickelten Anwendung stellt die Bindung an die Java-Welt kein Problem dar, da auch im Kontext der Finanzanwendung bei der betrachteten Bank Java in Form von Java EE Containern verwendet wird. Die Entscheidung zwischen der eingebetteten Workflow-Engine und der geteilten Workflow Engine auf einem Application Server ist vor allem eine Frage, die von Details der Hauptanwendung abhängt. Die eingebettete Workflow-Engine sorgt für eine sehr enge Kopplung und macht den Lebenszyklus der Workflow-Engine abhängig von der Anwendung. In Szenarien mit mehreren parallelen Anwendungen, die auf die Workflow-Engine zugreifen, ist daher die geteilte Workflow-Engine mit einer Orchestrierung durch den Container Service die bessere Lösung.
Bei der Anwendung im Rahmen dieser Arbeit musste auch berücksichtigt werden, dass die von Camunda bereitgestellte Webapp-Frontend durch eine eigene

GUI ergänzt bzw. ersetzt werden sollte (siehe Anforderungen). Zur Separierung einer modularen GraphQL-Schicht für das Frontend, die im nächsten Abschnitt beschrieben wird, wurde daher der Ansatz einer geteilten Workflow-Engine gewählt.

4.4 Gesamt-Architektur

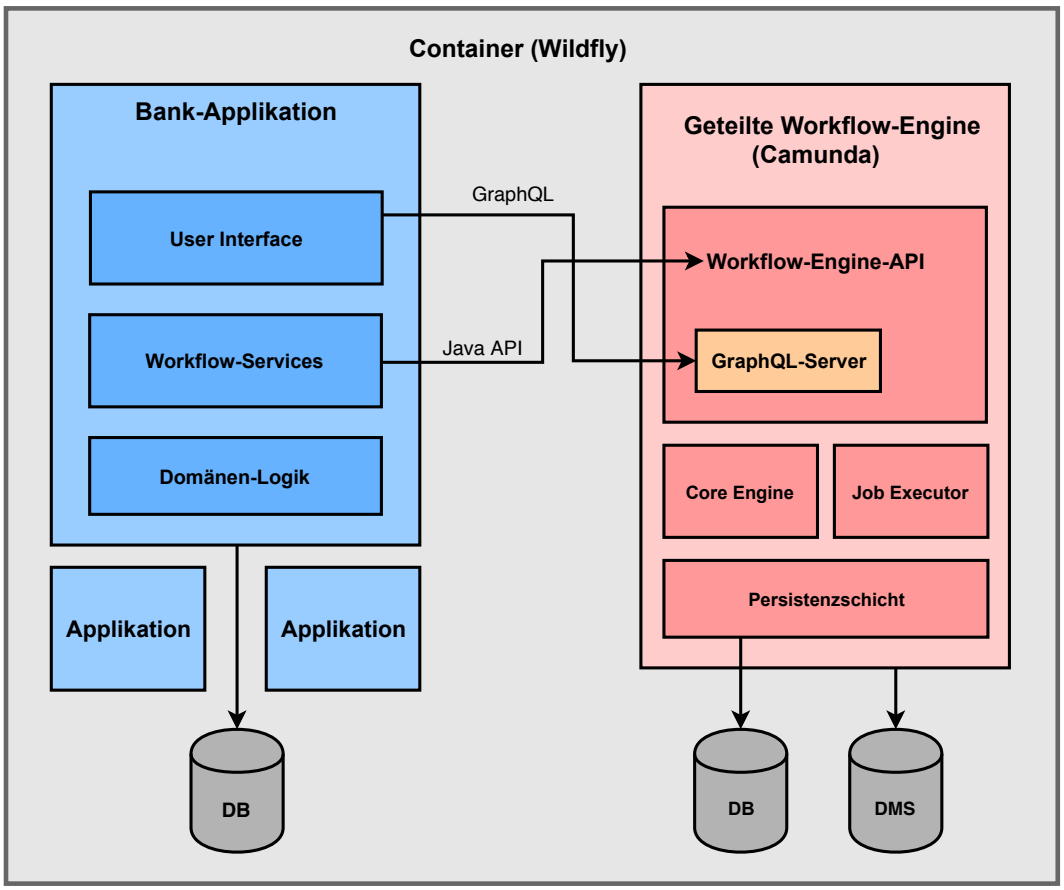

Abbildung 4.5: Gesamt-Architektur

Die in Abbildung 4.5 gezeigte Darstellung bietet eine Übersicht über die Gesamt-Architektur des im Rahmen dieser Arbeit implementierten Systems. Die Anwendung besteht aus zwei Teilen. Der bereits im Einsatz befindliche Anwendungs-Container soll um die Workflow-Engine-Komponente mit den oben genanten Bausteinen ergänzt werden. Die Workflow-Engine ist mit einer Datenbank sowie einem Dokumentmanagementsystem verbunden.
Die Bank-Applikation kann weitgehend unangetastet bleiben, sie wird jedoch um eine Komponente für die Workflow-Services ergänzt, die mit der Workflow-

Engine über die Java-API kommuniziert. Weitere Applikationen im Container können parallel laufen.

4.5 Prozessmodelle

Da bei der Verwendung von Workflow-Engines die Prozessmodellierung ein zentraler Baustein des Entwurfs ist, sollen das entworfene Prozessmodell, das später bei der Implementierung verwendet werden soll, an dieser Stelle erläutert werden.

4.5.1 Bestellungsprozess

Abbildung 4.6 zeigt ein BPMN 2.0-Modell des Bestellprozesses. Ein Bestellprozess ist dabei beispielsweise als Kreditantrag eines Neukunden zu verstehen. Der Antrag wird an das System übermittelt und das System erkennt, dass es sich um einen Neuantrag handelt. Erst hier ist das Startereignis des modellierten Prozesses. Das Dokument wird anschließend einer genauen Prüfung (Valutierung durch einen Sachbearbeiter unterzogen, wozu im Modell ein Human Task verwendet wurde. Bestätigt der Sachbearbeiter die Bestellung, so kommt es zum Ereignis *Bestellung freigegeben*, wodurch weitere Aktionen ausgelöst werden. In diesem Fall wird durch die Banken-Anwendung zunächst ein Konto angelegt und die Auszahlung des Vertrags anschließend veranlasst. Zwischen beiden Schritten liegen zahlreiche Detailschritte in der Banken-Anwendungen, die in diesem Modell nicht relevant sind. Über einen weiteren (automatisierten) Task wird im Bestätigungsfall das Versenden eines Bestätigungsschreibens an den betreffenden Kunden veranlasst.
Im Falle der Ablehnung des Antrags durch den Sachbearbeiter, wird das Ereignis *Bestellung abgelehnt* ausgelöst. Es können im weiteren Verlauf dann eventuell weitere Unterlagen vom betroffenen Kunden angefordert werden und der Prozess gegebenenfalls erneut durchlaufen werden.

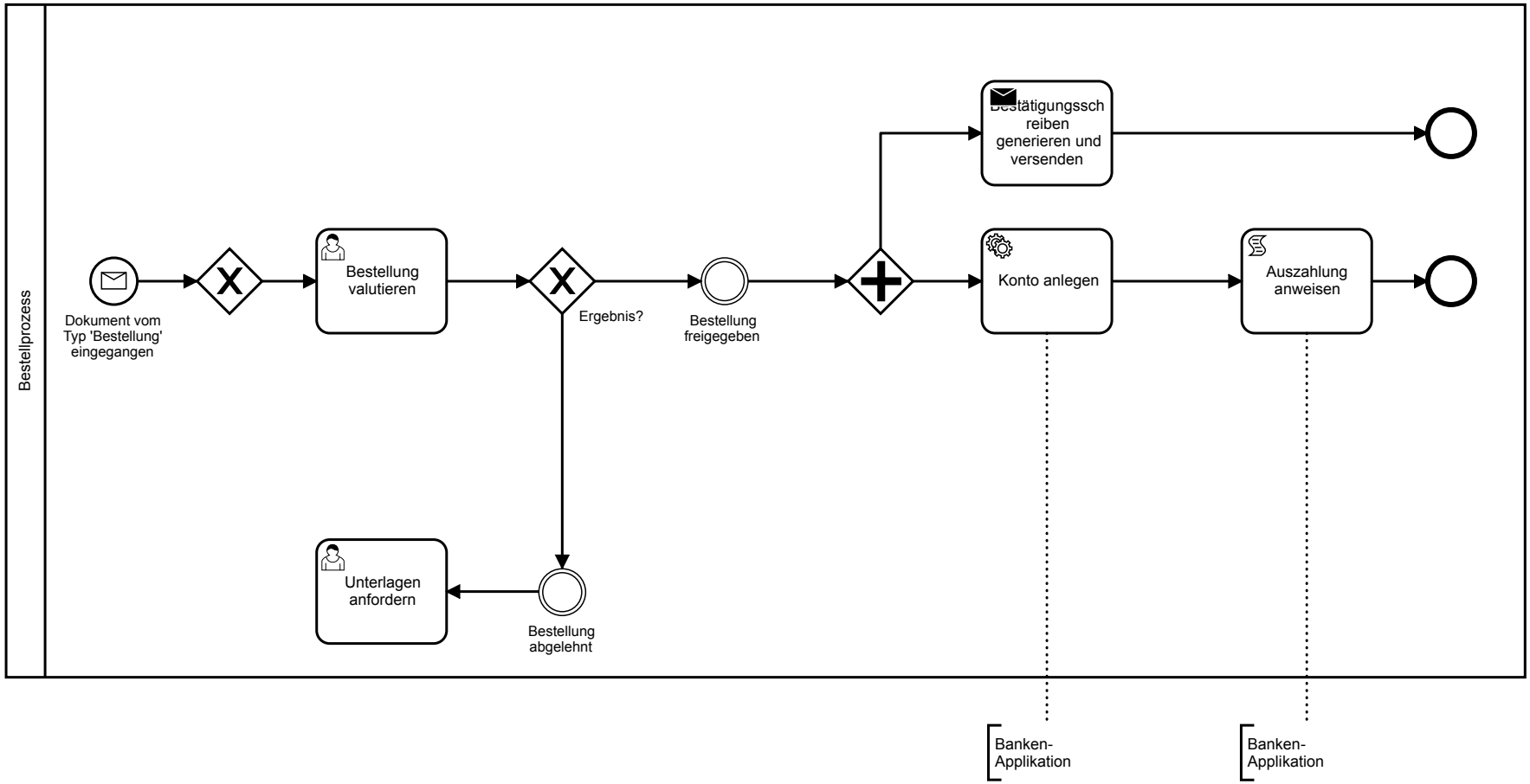

Abbildung 4.6: Prozessmodell für die Neubestellung

5 Implementierung

In diesem Kapitel werden die wichtigsten Implementierungsaspekte des realisierten Systems *DWA-Neu* erläutert und auf Besonderheiten sowie Schwierigkeiten bei der Umsetzung eingegangen.

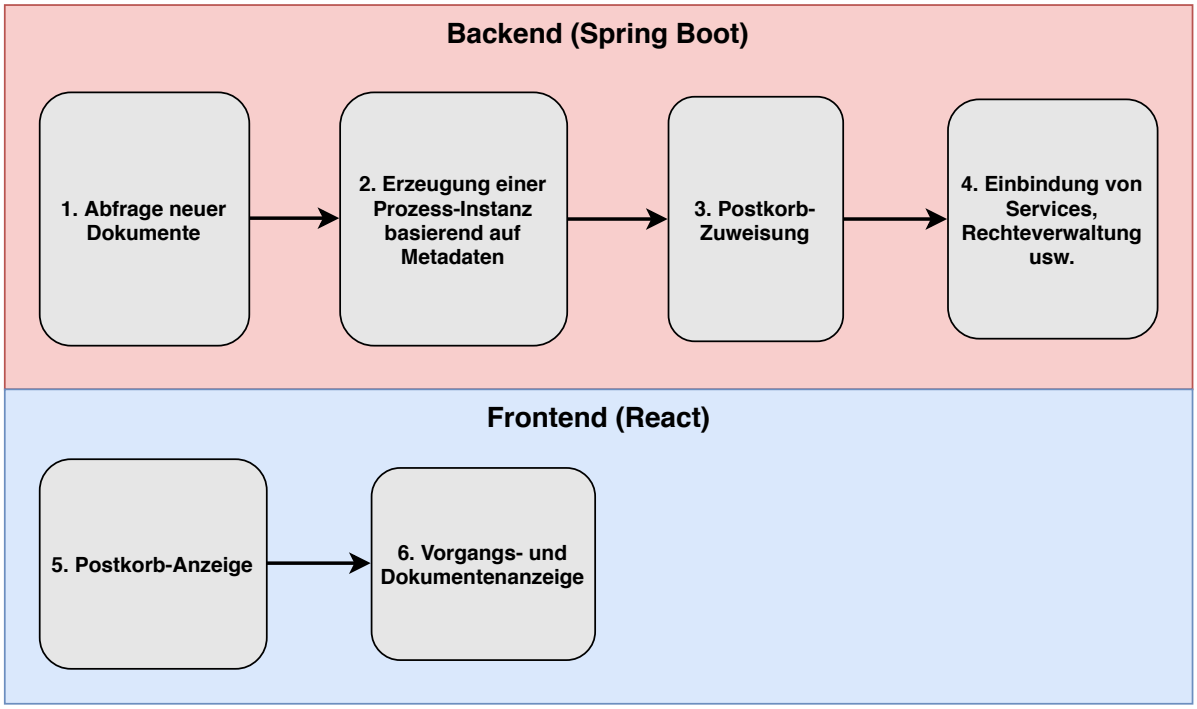

Abbildung 5.1: Schritte der Implementierung

Abbildung 5.1 zeigt die allgemeine Vorgehensweise bei der Implementierung. Zunächst wurde die Abfrage für neue Dokumente implementiert. Dabei musste eine flexible Lösung für die Anbindung eines Dokumentmanagementsystems gefunden werden, weil das sich eigentlich im Einsatz befindliche IBM FileNet P8-System aus lizenztechnischen Gründen nicht für die Entwicklung zur Verfügung stand. Anschließend konnte die Erzeugung von Prozess-Instanzen anhand

der Definition des Prozesses aus Abschnitt 4.5.1 implementiert werden. Im Backend wurde auch die Realisierung der Postkorb-Zuordnung, die auch ein Teil des bisherigen *DWA*-Systems ist, realisiert. Zudem wurde die Einbindung von Services zur Realisierung der automatisierten Service-Tasks aus dem Prozessmodell ermöglicht.
Im Frontend waren vor allem die Anzeige der Postkörbe sowie die sich daraus ergebenden Funktionalitäten für die Sachbearbeiter entscheidend. Außerdem mussten hier bestimmte Grundfunktionen, die in der Camunda Tasklist bereits enhalten sind, noch einmal „nachgebaut“ werden, um beispielsweise die Sicht auf Prozessdefinitionen und Vorgänge zu ermöglichen.

5.1 Backend-Implementierung

Das Backend wurde als Spring Boot-Applikation realisiert. Die Einstiegsklasse bildet dabei die Klasse `ProcessApplication`, die in Anhang A vollständig angefügt ist. Die `@SpringBootApplication`-Annotation sorgt dabei für die Auto-Konfiguration der Anwendung, wobei die Packages, die nach annotierten Klassen durchsucht werden, übergeben werden.
Die `@EnableProcessApplication`-Annotation ist eine camunda-spezifische Annotation, die dem Entwickler die Konfiguration der Camunda Workflow-Engine über eine separate Konfigurationsklasse erspart. Stattdessen kann so über die dritte Annotation `EnableConfigurationProperties` die Unterstützung für `ConfigurationProperties` aktiviert werden. Damit kann eine einzige .yml-Konfigurationsdatei für die Konfiguration der gesamten Spring-Boot-Anwendung genutzt werden.

5.1.1 Workflow-Engine

Mit dem Hochfahren der oben genannten Spring Boot-Anwendung wird auch die Workflow-Engine gestartet.

5.1.2 Dokumentenabfrage

Als Dokumentenmanagementsystem wurde die Community Edition von Alfresco in der Version 5.0.1 benutzt. Alfresco ist ein leistungsfähiges DMS, das in Form von freier Software als Alternative zur proprietären Software vieler Hersteller dienen kann. Alle modernen DMS und ECM-Systeme bieten den Vorteil, dass sie sich über den *Content Management Interoperability Services (kurz: CMIS)*-Standard an andere Softwaresysteme herstellerunabhängig anbinden lassen. Der Standard

beschreibt dazu eine Abstraktionsschicht für Webservices auf Basis von SOAP und REST, mithilfe derer dann mit dem jeweiligen Repository kommuniziert werden kann. (Thiede, 2010)

Für die Dokumentenabfrage muss zunächst eine Verbindung Die Abfrage neuer Dokumente vom Dokumentmanagementsystem wurde durch die Implementierung einer Task-Klasse vom Typ `Runnable` realisiert.

Listing 5.1: Implementierung der `run`-Methode für die Task-Klasse

```
@Override
public void run() {
    LOGGER.info("Task run...");
    try {
        List<Document> docs = cmisRepository.getLatestDocuments();
        for (Document doc : docs) {
            LOGGER.info("Property: " +
                doc.getProperty("finance:dokumentart"));
            NewCmisDocumentEvent evt = new
                NewCmisDocumentEvent(this, doc);
            eventPublisher.publishEvent(evt);
        }
    } catch (CmisConnectionException e) {
        LOGGER.error(e.getMessage());
    }
}
```

5.1.3 Erzeugung von Prozess-Instanzen

Für die Erzeugung von Prozess-Instanzen beim Entdecken neuer Dokumente wurde die Klasse `NewDocumentListener` entwickelt. Diese Klasse implementiert einen `ApplicationListener` für die oben genannten `NewDocument`-Events.

Listing 5.2: Ausschnitt der Klasse `NewDocumentListener`

```
@Override
public void onApplicationEvent(NewCmisDocumentEvent event) {
    LOGGER.info("Event fired!");

    Map<String, Object> variables = new HashMap<>();
    variables.put("documentName", event.getDocumentName());
    variables.put("documentLink", event.getDocumentLink());
    variables.put("documentType", event.getDocumentType());

    // Start process for every document that has been uploaded
```

```
        runtimeService.startProcessInstanceByKey("documentWorkflow",
            variables);
    }
```

5.1.4 Postkorb-Steuerung

Ein aufwändiges Postkorb-Routing wie beim aktuellen *DWA*-System ist in dieser Lösung nicht mehr von Nöten. Postkörbe können eher als Filtersichten auf Tasklists angesehen werden und werden dementsprechend auch realisiert.

5.1.5 REST-Zugriffsschicht

Um die Implementierung eines Frontends für die entwickelte Workflow-Engine zu vereinfachen, wurde im Backend mit einer REST-Zugriffsschicht eine weitere Abstraktionsebene eingeführt. Dazu wurde ein *GraphQL*-Server auf die bereits vorhandene Workflow-Engine gesetzt. *GraphQL*[1] ist eine Abfragesprache, die es ermöglicht, auf einfache Art und Weise, auf die Daten einer API zuzugreifen und diese zu bündeln. Der Vorteil liegt darin, dass alle Anfragen vom Client an den Server an nur noch einen gemeinsamen Endpunkt gerichtet werden können und die an den Endpunkt gerichteten Anfragen sehr einfach sind.
Die Konfiguration für den *GraphQL*-Server wird der Anwendung in der Konfigurationsdatei übergeben. Die entsprechenden Angaben sind in Listing 5.3 aufgeführt.

Listing 5.3: Konfiguration für den *GraphQL*-Server

```
graphql:
    servlet:
        mapping: /graphql
        enabled: true
        corsEnabled: true

graphiql:
    enabled: true
    endpoint: /graphql
    mapping: graphiql
```

Anschließend können die Schema-Definitionen erstellt werden, mit der die Felder und Eigenschaften der Camunda-Workflow-Engine auf den *GraphQL*-Endpoint gemappt werden. Eine beispielhafte Typ-Definition für die Prozess-Definitionen von Camunda findet sich in Listing 5.4

[1] http://graphql.org/

Listing 5.4: GraphQL-Typ-Definition für Prozess-Definitionen

```
type ProcessDefinition
{
        id: String
        name: String
        description: String
        startFormKey: String
        isSuspended: Boolean
        versionTag: String
        contextPath: String
        deploymentId: String
        deploymentResourceNames: [String]
        processModel: String
}
```

Weitere Erweiterungen der `ProcessApplication`-Klasse sind in dem Fall nicht notwendig. Der implementierte *GraphQL*-Server wird beim Starten der Anwendung mit dem übergebenen Schema parallel zur Workflow-Engine hochgefahren. Anfragen an das Schema sind im lokalen Browser über das *GraphiQL*-Tool möglich[2]. Abbildung 5.2 zeigt eine schematische Darstellung des erzeugten GraphQL-Schemas, das mithilfe des Online-Tools *GraphQL-Voyager*[3] erstellt wurde.

5.2 Frontend-Implementierung

Das Frontend wurde mithilfe von React und Redux realiert. Es handelt sich um eine reine Webanwendung. Da der Frontend-Teil nicht zentral für die Themenstellung dieser Arbeit war, soll hier nur kurz darauf eingegangen werden und ein Eindruck von der entwickelten Web-Oberfläche gegeben werden. Der Screenshot in Anhang B zeigt die Aufgaben-Ansicht wie sie in der Anwendung realisiert wurde.

5.3 Besonderheiten der Implementierung

Schwierigkeiten bei der Implementierung ergaben sich vor allem dadurch, dass auf wesentliche Komponenten, die Teil des Architektur-Konzepts waren, nicht ohne Weiteres zugegriffen werden konnte, sei es auch lizenzrechtlichen Gründen oder

[2] http://localhost:8080/graphiql
[3] https://apis.guru/graphql-voyager/

aus betriebstechnischen Gründen. So mussten im Rahmen der Implementierung einige Komponenten in einer simulierten Umgebung bereitgestellt werden.

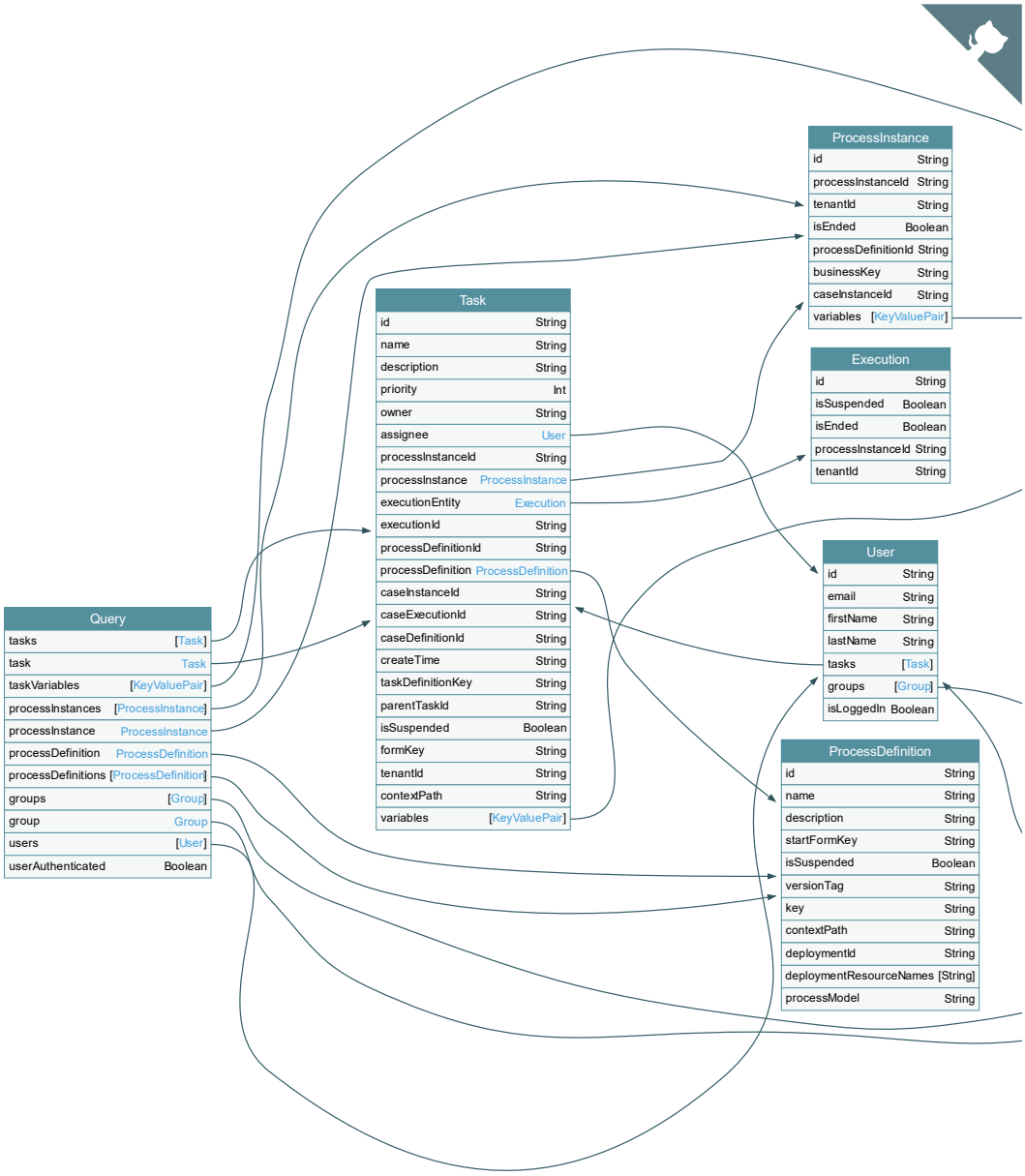

Abbildung 5.2: Visualisiertes GraphQL-Schema

6 Evaluation

In diesem Kapitel werden die funktionalen und nicht-funktionalen Anforderungen, die in Kapitel 2 festgelegt wurden, mit dem implementierten System aus Kapitel 4 verglichen.

6.1 Funktionale Anforderungen

Fachliche Prozessmodellierung

Die fachliche Prozessmodellierung wird durch das umgesetzte System ermöglicht. Die Camunda-Workflow-Engine unterstützt Prozessmodelle im BPMN 2.0-Standard.

Benutzeroberfläche zur Prozesssteuerung

Eine Benutzeroberfläche wurde wie in Abschnitt 5.2 beschrieben realisiert. Dabei wurden alle genannten Anforderungen realisiert. Die Aufteilung der Prozessinstanzen auf die Postkörbe erfolgt in *DWA-Neu* nicht mehr durch eine technische Zuweisung, sondern lediglich durch eine Filtersicht über Dokumenttypen, was eine erhebliche Vereinfachung darstellt.

Monitoring-Funktionen

Die Integration des unternehmenseigenen Geschäftsprozess-Monitoring bei der Implementierung war im zeitlichen Rahmen der Arbeit nicht mehr möglich, kann jedoch relativ einfach über den Java Message Service und einen entsprechenden Adapter zum GPM-System realisiert werden. Die Events, die diese Nachrichten abschicken, können direkt im Prozessmodell definiert werden.
Darüber hinaus sind die standardmäßigen Monitoring-Funktionen von Camunda auch bereits in der jetzigen Implementierung über das Camunda-Cockpit nutzbar.

Vermeidung von Redundanz bei Prozessmodellen

Da im Rahmen dieser Implementierung nur ein Prozess beispielhaft umgesetzt wurde, kann dieser Punkt nicht ausreichend bewertet werden.

Transaktionssicherheit innerhalb von Prozessen

Die jetzige Lösung baut auf dem bereits in Camunda integrierten Transaktionskonzept auf. Sollen eigene Transaktionsmanager verwendet werden, so können diese einfach nachträglich integriert werden.

Testabdeckung

Für den modellierten Prozess wurde ein lauffähiger Unit-Test geschrieben. Tiefergehende Integrationstests sind nicht mit in die Implementierung eingeflossen.

6.2 Nicht-funktionale Anforderungen

Möglichkeit zur Integration in bestehende Infrastruktur

Wie in Abschnitt 4.3 gezeigt, sind die verschiedenen Integrationsmöglichkeiten der Camunda-Workflow-Engine grundsätzlich flexibel. Für die Implementierung wurde der geteilte, container-verwaltete Ansatz für eine Workflow-Engine mit einem Wildfly-Application-Server gewählt. Dies scheint prinzipiell auch in der Praxis machbar, da die betriebliche Banken-Anwendung in einem ähnlichen Kontext betrieben wird. Genauere Aussagen über eine tatsächliche Integrierbarkeit können aber aufgrund der in Abschnitt 5.3 beschriebenen Probleme nicht getroffen werden.

Erhöhung der Effizienz

Auch über die Erhöhung der Effizienz lassen sich nur Mutmaßungen anstellen, zumal keine Quantifizierung der Effizienz des alten Systems vorliegt, die man zu einem Vergleich heranziehen könnte. Hierzu kann nur allgemein gesagt werden, dass die Erhöhung der Transparenz und Nachvollziehbarkeit der durchgeführten Geschäftsprozesse auch zu einem besseren unternehmensweiten Verständnis für diese Prozesse führt, was insgesamt die Tendenz schafft, diese Prozesse zu verbessern.

Geringere Lizenzkosten

Geringere Lizenzkosten wären mit der neuen Implementierung durch die Verwendung von Open-Source-Software in der Tat verbunden.

7 Fazit und Ausblick

Die im Rahmen dieser Arbeit vorgestellte Implementierung hat zwar die zuvor gestellten Anforderungen zum großen Teil erfüllt, allerdings kann sie noch keinen großen Beitrag zur Bewertung der Sinnhaftigkeit des Einsatzes einer Worfklow-Engine bei dem betrachteten Finanzinstitut leisten. Dazu waren zum einen die Voraussetzungen aus betrieblicher Sicht nicht gegeben, Live-Systeme konnten nicht einfach zum Testen verwendet werden und die Ablösung wichtiger Systeme kann überhaupt nur langfristig gedacht werden. Zum anderen stand bei dieser Arbeit ohnehin der explorative Charakter des Vorhabens im Vordergrund. Zukünftige Aufgaben für eine gründlichere Machbarkeitsstudie oder gar ein Pilotprojekt sind müssen es daher sein, das entwickelte System noch besser in die IT-Infrastruktur des Unternehmens zu integrieren und somit zu besseren Erkenntnissen hinsichtlich der abteilungsübergreifenden Akzeptanz von technischem Geschäftsprozessmanagement mithilfe von Workflow-Engines.

Anhang A Implementierung der Klasse `ProcessApplication`

```
@SpringBootApplication(scanBasePackages = {"bpm", "cmis", "graphql"})
@EnableProcessApplication
@EnableConfigurationProperties
public class ProcessApplication extends SpringBootServletInitializer {

    private static final Logger LOGGER =
        Logger.getLogger(ProcessApplication.class.getName());

    @Autowired
    protected CamundaBpmProperties camundaBpmProperties;

    @Autowired
    private CmisRepositoryScheduler cmisRepositoryScheduler;

    @Autowired
    private CmisRepository cmisRepository;

    @Override
    protected SpringApplicationBuilder
        configure(SpringApplicationBuilder application) {
        return application.sources(ProcessApplication.class);
    }

    public static void main(String... args) {
        SpringApplication.run(ProcessApplication.class, args);
    }

    @EventListener
    public void processPreUndeploy(PreUndeployEvent event) {
        LOGGER.info("do anything...");
    }

    @EventListener
    private void processPostDeploy(PostDeployEvent event) {
        LOGGER.info("Started process engine with name " +
            event.getProcessEngine().getName());
        LOGGER.info("Process engine properties: " +
            camundaBpmProperties.toString());
        // Activate scheduler to wait for start events
        cmisRepositoryScheduler.checkForNewCmisDocuments();
    }
}
```

Anhang B Frontend-Screenshot

Literaturverzeichnis

Allweyer, T. (2014). *BPMS: Einführung in Business Process Management-Systeme*. BoD–Books on Demand.

Bischoff, B. & van Dinther, C. (2016). Workflow management systems-an analysis of current open source products. In *DEC* (S. 147–160).

Dumas, M., La Rosa, M., Mendling, J. & Reijers, H. A. (2013). *Fundamentals of Business Process Management*. doi:10.1007/978-3-642-33143-5

Freund, J. & Rücker, B. (2016). *Praxishandbuch BPMN: mit Einführung in CMMN und DMN*. Carl Hanser Verlag GmbH Co KG.

Komus, A. (2011). *BPM best practice: Wie führende Unternehmen ihre Geschäftsprozesse managen*. Springer.

Lotz, D. (2015). *Geschäftsprozessautomatisierung der Sterilgutaufbereitung mit der BPMN 2.0 am Beispiel der Open Source Camunda BPM-Plattform* (Diss.).

Moormann, J. (2018). Enzyklopädie der Wirtschaftsinformatik, Stichwort: Kernbanksystem. Zugriff 12. Mai 2018 unter http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/informationssysteme/Sektorspezifische-Anwendungssysteme/Finanzsektor--Anwendungssysteme-im/kernbanksystem

Schaffry, A. (2009). Hohe Erwartungen an Geschäftsprozess-Optimierung. Zugriff 2. Mai 2018 unter https://www.cio.de/a/hohe-erwartungen-an-geschaeftsprozess-optimierung,871871

Süß, M. (2012). *Einführung eines Workflow Management Systems* (Diss., Hochschule für Telekommunikation Leipzig).

Thiede, C. (2010). CMIS - neuer Standard für die ECM-Industrie. Zugriff 12. Mai 2018 unter https://www.heise.de/developer/artikel/CMIS-neuer-Standard-fuer-die-ECM-Industrie-1082119.html

Van der Aalst, W. M. P. (2013). Business Process Management: A Comprehensive Survey. *ISRN Software Engineering*, *2013*(1), 1–37. doi:10.1155/2013/507984

Zahn, C. (2013). *Integration einer BPM-Engine in eine komplexe Geschäftsanwendung* (Diss.).