Hochschule Neubrandenburg
University of Applied Sciences

Hochschule Neubrandenburg

Studiengang Geoinformatik

Integriertes Authoring von modularen Prozessdialogen

Bachelorarbeit

vorgelegt von: *Martin Scherpinski*

Zum Erlangen des akademischen Grades

„Bachelor of Engineering“ (B.Eng.)

Erstprüfer: Prof. Dr.-Ing. Andreas Wehrenpfennig

Zweitprüfer: Prof. Dr.-Ing. Bodo Urban

Bearbeitet vom 04.07.2011 bis 29.08.2011

URN: urn:nbn:de:gbv:519-thesis2011-0073-1

Diese Arbeit ist in Kooperation mit dem Fraunhofer-Institut für Graphische Datenverarbeitung in Rostock entstanden.

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Neubrandenburg, den

Unterschrift

Danksagung

Das Anfertigen einer Bachelorarbeit ist ein langwieriger Prozess, der ohne die fachliche und menschliche Unterstützung einer Vielzahl von Personen nicht möglich ist. An dieser Stelle soll all jenen gedankt werden, die ihren Anteil bei der Entstehung der vorliegenden Arbeit hatten.

An der Hochschule Neubrandenburg soll Prof. Dr.-Ing. Andreas Wehrenpfennig für die Betreuung des Themas und für die fachliche Beratung gedankt werden.

Als Zweitgutachter gilt der Dank Prof. Dr.-Ing. Bodo Urban vom Fraunhofer-Institut für graphische Datenverarbeitung in Rostock.

Ein weiteres Dankeschön geht an das IMAP-Projektteam des Fraunhofer-Instituts für graphische Datenverarbeitung in Rostock, das mir bei der Durchführung des der Arbeit zugrunde liegenden Praktikums mit Rat und Tag zur Seite stand:

Dipl.-Inf. Mario Aehnelt, Dipl.-Inf. (FH) Marleen Rosenkranz M.A., Dipl.-Inf. Andreas Müller, Dipl.-Päd. Petra Müsebeck LL.M.Crim., Dr.-Ing. Jörg Voskamp, Dr.-Ing. Karina Oertel.

Ein besonderer Dank gilt dabei Mario Aehnelt und Marleen Rosenkranz, die diese Arbeit während ihrer Entstehung fachlich betreut haben.

Abschließend soll allen Freunden und Familienmitgliedern für motivierende Worte, orthografische Korrekturen und die allgemeine Unterstützung gedankt werden.

Kurzfassung

In der Geschäftswelt sind es gerade die kleinen und mittelständischen Unternehmen, die gezwungen sind, ihre Produktion so effizient wie möglich zu gestalten, um ihre Chancen auf dem Markt zu bewahren. Häufig führt dies langfristig zu einer notwendigen Optimierung des Betriebes, bei der vermehrt die Geschäftsprozesse eines Unternehmens und deren Automatisierung in den Fokus rücken.

Diese Arbeit beschäftigt sich mit Möglichkeiten zur integrierten und einfachen Modellierung dieser Geschäftsprozesse mit dem Ziel der Automatisierung unter Berücksichtigung realer Unternehmensstrukturen, ohne dass explizites Fachwissen seitens der Modellierer nötigt ist. In diese automatisierten Prozesse sollen zudem Formulare und Eingabemasken integriert werden, damit eine Kommunikation und Interaktion der Nutzer mit dem ausführenden System möglich wird. Entsprechende Systeme existieren, sind in der Regel aber durch die Vernachlässigung realer Strukturen und einer zu komplizierten Bedienung gekennzeichnet.

Nach der Erläuterung von theoretischen Grundlagen zur Modellierung von Geschäftsprozessen und Anforderungen an ein mögliches Modellierungswerkzeug wird dazu der Prototyp eines intuitiven, praxistauglichen System entwickelt, mit dem eine einfache, nutzerfreundliche Modellierung und Ausführung von Geschäftsprozessen mit integrierten Prozessdialogen zur Ablaufsteuerung, Monitoring-Zwecken und Dateneingabe möglich wird.

ACM Classification Keywords

D.2.2 [SOFTWARE ENGINEERING]: Design Tools and Techniques --- *User interfaces;*
D.2.2 [SOFTWARE ENGINEERING]: Design Tools and Techniques --- *Flow charts;*
H.4.1 [INFORMATION SYSTEMS APPLICATIONS]: Office Automation --- *Workflow management;*
H.5.3 [INFORMATION INTERFACES AND PRESENTATION]: Group and Organization Interfaces --- *Organizational design;*
J.1 [ADMINISTRATIVE DATA PROCESSING]: *Business --- Process Management;*
K.4.3 [COMPUTERS AND SOCIETY]: Organizational Impacts --- *Computer-supported collaborative work*

Abstract

Small and medium sized companies are forced to keep their business as efficient as possible to have a chance at the global market. Often this leads to the necessity of optimizing the companies with focus on business processes.

This thesis wants to discover prospects of simple and integrated business process modeling that aims on automation and considers existing company infrastructures. While modeling these processes, people should only need basic knowledge about business process modeling. Also there should be forms for interaction and communication connected and integrated into the business processes. There are a couple of systems providing this, but they are complicated in usage and are not matching the company structures very well.

After providing some theoretical base knowledge and explaining some requirements for a modeling tool, a prototype of a practicable system will be developed. With that tool it will be possible to model business processes and user interfaces in a simple, user-friendly way and execute them for monitoring, data input and flow control.

Inhaltsverzeichnis

1 Einführung in das Thema

1.1 Motivation

In der heutigen Zeit fortschreitender Globalisierung ist es gerade für kleine und mittelständische Unternehmen des sekundären Wirtschaftssektors von essentieller Bedeutung, die eigenen Produktionskapazitäten so kostengünstig und zeitsparend wie möglich zu halten. Um dieses Ziel zu erreichen, rücken immer mehr die in den Unternehmen vorliegenden Geschäftsprozesse in den Vordergrund.

Eine Optimierung von diesen birgt großes Potential. Für die kleinen und mittelständischen Unternehmen geht das einher mit einer Automatisierung der vorliegenden Geschäftsprozesse, um die Arbeiten in den produzierenden Bereichen des Betriebes zu unterstützen und auf Dauer auch verbessern zu können. Eine solche Automatisierung setzt eine Modellierung der Geschäftsprozesse voraus. Für eine Optimierung ist es außerdem sinnvoll, einem Nutzer in Form von Prozessdialogen die Möglichkeit zu geben, mit dem ausführenden System zu kommunizieren.

Für die Modellierung der Geschäftsprozesse zu Automatisierungszwecken und für die Erstellung der entsprechenden Eingabemasken gibt es separate Lösungen. Diese haben in der Regel jedoch das Problem, dass sie einen sehr hohen Abstraktionsgrad aufweisen und dementsprechend nur mit Fachkenntnissen zu bedienen sind. Zudem bilden zwar die meisten verfügbaren Tools die gewünschten Prozesse ab, die Möglichkeiten von Nutzereingaben in Form zugehöriger Dialoge werden jedoch oft nicht berücksichtigt. Häufig müssen vorhandene Produkte an die Unternehmensinfrastrukturen angepasst werden, berücksichtigen dabei jedoch in den meisten Fällen die konkreten Bedingungen vor Ort nicht ausreichend. Oftmals sind Kompromisse notwendig, um mit gegebenen Werkzeugen die internen Abläufe möglichst realitätsnah abbilden zu können.

Doch gerade kleine und mittelständische Unternehmen haben meist nicht die finanzielle Möglichkeit, externe Dienstleister für die fachgerechte Modellierung unter Berücksichtigung aller Bedingungen einzusetzen. Dementsprechend werden praxistaugliche Werkzeuge für das Authoring, also die Modellierung dieser Prozesse und Prozessdialoge benötigt. In dem Forschungsprojekt IMAP finden Untersuchungen dazu statt.

Die vorliegende Arbeit beschäftigt sich mit der Fragestellung nach Möglichkeiten der integrierten Modellierung von Geschäftsprozessen und deren Verknüpfung mit Nutzereingaben für eine IT-gestützte, automatisierte Ausführung.

Dabei soll eine Basis zur Optimierung und Unterstützung der Arbeitsabläufe innerhalb von kleinen und mittelständischen Unternehmen des produzierenden Gewerbes unter Berücksichtigung der real vorhandenen Unternehmensinfrastrukturen entstehen.

Bei den Untersuchungen rückt vor allem die Integration und Verknüpfung der ausführbaren Geschäftsprozesse mit entsprechenden Eingabemasken und Formularen in den Vordergrund, sodass entsprechenden Prozessschritten zugehörige Prozessdialoge zugeordnet werden können, welche bei einer Ausführung an gegebener Stelle die Interaktion mit dem System ermöglichen.

Im Zuge der Betrachtung dieser Fragestellung soll ein integriertes Authoring-Werkzeug für die Modellierung von Geschäftsprozessen und zugehörigen Prozessdialogen entworfen und prototypisch umgesetzt werden, wobei der Fokus auf der erwähnten Integration liegt.

Dazu wird zunächst das für die Entwicklung relevante Wissen durch theoretische Betrachtungen zu dieser Fragestellung hergeleitet, das Projektes IMAP und der Kontext des Werkzeugs innerhalb des Projektes werden beschrieben. Anforderungen an das System werden erläutert und ausgewertet und auf Basis dessen wird eine entsprechende Anwendung konzeptioniert, entworfen und prototypisch umgesetzt.

Die hier im Verlauf dargestellten Ausführungen beruhen auf Erkenntnissen aus dem Forschungsprojekt IMAP.

1.2 Das Projekt Intelligenter, mitarbeiterzentrierter Montageprozess in KMU (IMAP)

Diese Arbeit beschäftigt sich mit der Optimierung von Geschäftsprozessen kleiner und mittelständischer Unternehmen durch deren Digitalisierung und Automatisierung. Untersuchungen zu diesem Thema werden unter anderem in dem Forschungsvorhaben IMAP durchgeführt. Hinter dem Akronym IMAP verbirgt sich ausgeschrieben *Intelligenter, mitarbeiterzentrierter Montageprozess in KMU.*

Bei dem Forschungsprojekt IMAP handelt es sich um ein Verbundprojekt von zwei Fraunhofer-Instituten und Anwendungspartnern aus der Wirtschaft, wobei dies im Speziellen kleine und mittelständische Unternehmen (KMU) sind. Dieses Projekt wird durch das Bundesministerium für Wirtschaft und Technologie gefördert.

Die Projektteilnehmer haben sich das Ziel gesetzt, die Arbeit in KMU in Zukunft effizienter, flexibler und auch innovativer zu gestalten. Beispielhaft soll dies an den Projektpartnern aus der maritimen Industrie erprobt werden.

Diese Zielvorstellung beruht auf dem Vorhaben, die Montageprozesse und auch die Montageplanung der KMU zu verbessern und optimiert zu gestalten. Innerhalb des Projektverlaufs werden verschiedene Konzepte und Methoden der Montageplanung und auch der Montageprozessorganisation entwickelt und anschließend evaluiert (vgl. [IMAP 01]).

Eine Besonderheit dieser Konzepte und Methoden ist einerseits die Integration in bestehende Arbeitsprozesse und Unternehmensinfrastrukturen, andererseits aber auch die Mitarbeiterzentrierung.

Die betriebsinternen (Montage-)Prozesse werden dabei nicht aus einem traditionellen Blickwinkel betrachtet, welcher zum Beispiel die Entwicklung eines Produktes über verschiedene Etappen wie Vertrieb, Einkauf, Konstruktion, Fertigung, Versand und Rechnungslegung umfasst, sondern aus der Perspektive der an der Erstellung beteiligten Personen (vgl. [HMS11] S.15). Ein Prozess umfasst dann beispielsweise den Arbeitsablauf eines Mitarbeiters, beginnend bei seiner Ankunft am Arbeitsplatz, über die Auswahl eines Arbeitsauftrages, dessen Erledigung und das Verlassen des Arbeitsplatzes.

Gerade mit dieser Sicht auf die Geschäftsprozesse des Betriebes besteht ein großes Optimierungspotential, da solche mitarbeiterzentrierten Sichten in der Regel nicht bzw. nur schlecht dokumentiert sind.

Wird die allgemeine Dokumentationssituation in den KMU betrachtet, sind die Aufgaben der einzelnen Mitarbeiter meist bekannt und auch erfasst. In der Regel sind die optimalen Abläufe, um ein bestimmtes Ziel zu erreichen, aber nur dem Ausführenden bekannt, welcher sich diese über Jahre angeeignet und für sich selbst optimiert hat. Diese Kompetenzen gehen jedoch verloren, sollte der entsprechende Mitarbeiter aus dem Betrieb ausscheiden. Um diesen Wissensverlust zu vermeiden, wird versucht, innerhalb des Projektes die Ressource Wissen explizit zu berücksichtigen.

Diese Auseinandersetzung mit dem in den Unternehmen vorliegenden Wissen führt neben der Wissensbewahrung zu einer Erarbeitung technischer Konzepte, um jedem Mitarbeiter an dem Ort und zu der Zeit, an denen bestimmte Informationen benötigt werden, diese auch bereitzustellen (vgl. [IMAP 02]). Das hier erwähnte Wissensmanagement stellt nur eines von mehreren Arbeitspaketen des Projektes IMAP dar.

Um die hier dargestellten, vielfältigen Ziele zu erreichen, soll als Ergebnis des Projektes ein sogenanntes integriertes Montageportal entstehen. Dabei handelt es sich um eine integrierte Softwarelösung, welche sich aus verschiedenen Modulen zusammensetzt, die für unterschiedliche Unternehmen zusammengestellt werden können (vgl. [IMAP 01]). Alle Konzepte, Methoden und Werkzeuge, die in den unterschiedlichen Arbeitspaketen des Projektes erarbeitet wurden, werden zu einem modularen System vereint (vgl. [IMAP 03]).

Die Entwicklung dieser Software obliegt dem Fraunhofer-Institut für graphische Datenverarbeitung in Rostock, mit dessen Kooperation diese Arbeit entstehen konnte.

Zwar wird das Montagportal für den beispielhaften Einsatz bei den Praxispartnern entwickelt, soll aber generisch angelegt sein, sodass eine Migration auf jedes beliebige KMU mit Montageprozessen einfach und schnell durchführbar ist. In dem Portal sollen alle relevanten, im Unternehmen vorliegenden Daten zusammengeführt und Werkzeuge bereitgestellt werden, mit deren Hilfe die Montage innerhalb der Unternehmen mit dem Fokus auf dem Mitarbeiter und arbeitsprozessintegriert geplant und auch durchgeführt werden kann (vgl. [IMAP 01]).

Hier lässt sich diese Arbeit einordnen. Das integrierte Montageportal soll der Planung und Durchführung von Arbeiten innerhalb der KMU dienen. Da hierbei die Montageprozesse in den Vordergrund rücken, ist es für die Montageplanung wichtig, dass diese Prozesse im Portal abgebildet werden können.

Integriertes Authoring von modularen Prozessdialogen bedeutet in diesem Kontext, ein Werkzeug bereitzustellen, um die Montageprozesse aus Sicht der Mitarbeiter modellieren und so im Montageportal zu Planungszwecken verfügbar machen zu können. Dabei sollen die realen Arbeitsschritte der Montage möglichst realitätsnah abgebildet werden.

Dieses Werkzeug, welches im Folgenden als IMAP-Editor bezeichnet werden soll, stellt dabei einen Teil des integrierten Montageportals dar und soll im Rahmen dieser Arbeit entwickelt werden.

Neben der Montageplanung, die durch die Modellierung der Prozesse ermöglicht wird, soll auch die eigentliche Montage mit Hilfe des Portals durchgeführt werden. Hierbei soll es als Assistenzsystem wirken, um beispielsweise an entsprechenden Stellen benötigte Informationen bereitzustellen. Um dies zu ermöglichen, werden Oberflächen benötigt, die begleitend zu einzelnen Prozessschritten der Montage Informationen darstellen, aber auch entgegennehmen können. Mit Hilfe solcher Formulare wird die Optimierung der einzelnen Prozesse unterstützt, da beispielsweise genau dargestellt wird, wann was zu tun ist, welches Material dafür benötigt wird und wie viel Zeit veranschlagt ist.

Diese Interaktionsmasken stellen die zweite Anforderung an die Funktionalität des IMAP-Editors dar. Sie sollen ebenfalls mit dem Editor modellierbar sein und dabei einzelnen Prozessschritten zugeordnet werden können.

Diese Arbeit verfolgt das Ziel, ein Werkzeug zu entwerfen und umzusetzen, mit dem die Integration von Formularen und Prozessdarstellungen möglich wird. Einen Ansatz, beides intuitiv, einfach und verknüpft modellieren zu können, existiert in einer solchen Form in der Praxis nicht. Es werden dazu verschiedene Konzepte der Integration vorgestellt und bewertet, um einen geeigneten Ansatz für den IMAP-Editor zu finden.

Der Schwerpunkt der Arbeit liegt auf der integrierten Modellierung von Prozessen und Formularen. Eine Ausführung der Modelle durch eine entsprechende Laufzeitumgebung stellt dabei einen Zusatz dar und soll dabei nur im Ansatz verfolgt werden.

Zunächst sollen jedoch einige theoretische Grundlagen zu der Modellierung von Prozessen erläutert werden.

2 Theoretische Grundlagen

Im vorherigen Kapitel wurde der Rahmen für diese Arbeit skizziert. Es wurde die Intention erläutert, Montageplanung und -durchführung in den KMU zu verbessern, um deren Stellung auf dem globalen Markt zu festigen. Erreicht werden soll dies unter anderem durch ein integriertes Montageportal, das verschiedene Werkzeuge bereitstellt, um Vorgänge innerhalb der Unternehmen darzustellen, zu automatisieren, Informationen zielgerichtet bereitzustellen und so Optimierungsmöglichkeiten schaffen zu können.

Als Teil dieses Portals und im Zuge dieser Arbeit soll ein Werkzeug entstehen, welches die Modellierung unternehmensinterner Prozesse intuitiv und einfach ermöglicht und dabei eine Verknüpfung mit Formularen schafft. Zusätzlich entsteht eine Laufzeitumgebung, mit der die Ergebnisse der Modellierung veranschaulicht werden sollen, jedoch liegt der Fokus der Betrachtungen auf dem Editor. Dennoch muss bei der Modellierung darauf geachtet werden, dass die abgebildeten Prozesse auch ausführbar sein sollten.

Auch vor IMAP hat es bezüglich des Prozessmanagements und der Automatisierung von Prozessen schon Untersuchungen gegeben. In diesem Kapitel werden einige dieser Konzepte, Begriffe und Standards erläutert.

Dazu wird zunächst auf Definitionen und Ansätze des Prozessmanagements eingegangen, die Theorie der Modellierung wird erläutert, das Workflowmanagement vorgestellt und es werden Standards zur Modellierung untersucht, die im Konzept des IMAP-Editors Verwendung finden könnten.

2.1 Grundlagen des Prozessmanagements

Die Notwendigkeit einer Optimierung der Abläufe innerhalb eines Betriebes ist permanent gegeben. In der heutigen Zeit der Globalisierung ist es essentiell, dass Kosten und Zeit eingespart werden, wo immer dies möglich ist. Produkte müssen möglichst schnell und möglichst günstig produziert werden, um im Wettbewerb bestehen zu können. Dabei wird zunehmend ein Fokus auf die im Betrieb etablierten Prozesse gelegt. Ohne entsprechende Vorbetrachtungen wird es jedoch keinem Verantwortlichen möglich sein, die Vorgänge in seinem Unternehmen zu verbessern.

Um eine Optimierung der Unternehmen zu ermöglichen, muss, zunächst der IST-Zustand in den Betrieben analysiert werden. Die dabei ermittelten Informationen werden ausgewertet und auf dieser Grundlage werden Modelle erstellt, die potentielle, zukünftige Vorgänge abbilden.

In [SpS03] werden Gründe zur Prozessoptimierung genannt. Dazu gehören unter anderem im internen Bereich die Einsparung von Kosten, die Verkürzung der Bearbeitungszeiten, Minimierung von Liegezeiten, aber auch extern eine größere Kundennähe oder eine verbesserte Produktqualität. Diese Gründe werden im Kontext dessen genannt, was durch eine Optimierung erreicht werden soll und sind somit ebenfalls auf den Zweck einer Geschäftsprozessverbesserung zurückführbar.

Beispiele zeigen, dass sich bereits mit relativ geringem Aufwand nennenswerte Ergebnisse erzielen lassen. Betrachtet man beispielsweise den Reifenhersteller Dunlop, so erkennt man die Effektivität in der Verbesserung der Arbeitsabläufe. Vor Ort werden Bestellungen per Fax angenommen.

Diese müssen per Hand in das vorliegende Verwaltungssystem eingegeben und durch einen Mitarbeiter kontrolliert werden. Das kostet vor allem Zeit. Durch die Automatisierung der Bestellannahme konnte bei gleichbleibender Kontrolle eine Zeitersparnis von 70 % erreicht werden. Die Faxe werden gescannt und digital erfasst. Der Mitarbeiter muss nun nicht mehr von Hand die Eingaben tätigen, sondern lediglich kontrollieren (vgl. [Gad08] S.42). Eine Optimierung von Geschäftsprozessen ist also ein anzustrebender Arbeitsbereich für viele Unternehmen. Im Folgenden soll ausgeführt werden, wie dies grundsätzlich möglich ist.

Bisher wurde vermehrt von Prozessen und Geschäftsprozessen und deren notwendiger Optimierung gesprochen. Um ein Verständnis für dieses Aufgabenfeld aufzubauen, sollen die genannten Begriffe definiert werden.

In [BeK03] wird von einem Prozess im Zusammenhang mit Unternehmen von einer abgeschlossenen Reihenfolge unterschiedlichster Aktivitäten gesprochen, welche zeitlich und logisch nacheinander durchlaufen werden, wobei ein betriebswirtschaftlich relevantes Objekt bearbeitet wird. Davon ausgehend wird der Geschäftsprozess als Spezialform des Prozesses benannt, der als sein Ziel die Erfüllung der Unternehmensziele sieht und dabei ein Geschäftsfeld beschreibt.

Nimmt man diese Definitionen als Grundlage, so kann ein Geschäftsprozess als eine Zusammensetzung verschiedenster Teilschritte gesehen werden, die nacheinander ausgeführt werden, um eine Aufgabe innerhalb des Unternehmens zu erfüllen. Diese Grunddefinition ähnelt sich in vielen Betrachtungen unterschiedlicher Autoren. Im Grunde nennen alle eine Abfolge von bestimmten Aufgaben und die Abarbeitung einer Folge von Teilschritten. Zum Teil werden die Ausführungen um die Hilfe durch IT-Technologie ergänzt (vgl. [Gad08] S.45).

Um eine solche Abfolge von Aktivitäten zu verbessern, gibt es unterschiedlichste Theorien. Ein Hauptziel besteht dabei in einer langfristigen und nachhaltigen Optimierung. Bisher wurde in deutschen Unternehmen eher versucht, über Kostensenkungen mögliche Probleme zu lösen.

Dazu zählen beispielsweise Outsourcing oder Personalabbau (vgl. [SeS08] S.4). Dies soll bei den KMU, für die IMAP Lösungen entwickelt, vermieden werden. Solche Maßnahmen haben in der Regel auch nur kurzzeitigen Erfolg und bekämpfen nur die Symptome, nicht aber die Ursachen von Problemen innerhalb der Unternehmen. Deshalb wird in der Regel versucht, eine Optimierung durch die Verbesserung der Abläufe als solche zu erreichen.

Die zwei gängigsten Theorien dazu heißen Business Reengineering und Geschäftsprozessoptimierung (GPO). Business Reengineering geht auf die Ausführungen von Hammer und Champy zurück.

Diese bezeichnen ihre Theorie als eine Radikalkur für ein Unternehmen. Sie zielen auf grundlegende Veränderungen und die komplette Umstrukturierung ab. Statt der Optimierung bestehender Prozesse sollen diese neu formuliert werden (vgl. [Gad08] S.32).

Bei der GPO muss die Benutzung des Begriffes klar abgegrenzt werden, da Geschäftsprozessoptimierung als grundlegendes Thema von IMAP und dieser Arbeit genannt waren. Im Kontext der folgenden Ausführungen ist jedoch die konkrete Theorie gemeint.

Diese zielt auf eine schrittweise Veränderung der Prozesse ab, bei der im kleinen Rahmen Neuerungen angebracht werden. Beide Theorien befassen sich mit der Restrukturierung der Prozesse (vgl. [Gad08] ebenda). Dem erläuterten Ansatz des IMAP-Projektes folgend sind die Arbeiten im Projekt der Geschäftsprozessoptimierung zuzuordnen. Ziel war die Integration in bestehende Strukturen und damit die Verbesserung vorhandener Abläufe. Diese sollen mit Hilfe von IT-Technologien automatisiert werden. Dabei ist die Verbindung der IT-Systeme mit den vorhandenen Prozessen entscheidend. Nach [SeS08] könnten dabei Problemen entstehen, wenn die Geschäftsprozesse vorher nicht klar definiert sind. Die IT könnte dabei am Ziel vorbei schießen.

Um die in den KMU vorliegenden Geschäftsprozesse mit Hilfe einer Automatisierung optimieren zu können, müssen diese zunächst erfasst und abgebildet werden. Bei dieser Modellierung sind jedoch einige Besonderheiten zu beachten.

2.2 Besonderheiten bei der Modellierung von Geschäftsprozessen

Im vorhergehenden Kapitel wurde der Begriff des Geschäftsprozesses geprägt. Diese Prozesse sollen im Zuge der Arbeiten an IMAP mit Hilfe einer Automatisierung optimiert und verbessert werden. Dabei stellt sich die Frage, wie ein Geschäftsprozess in eine automatisierte Form gebracht werden kann. Dazu muss dieser modelliert werden. Jedoch muss dabei beachtet werden, dass grundsätzlich zu unterscheiden ist, ob ein Geschäftsprozess automatisiert oder nicht automatisiert dargestellt wird. Um diese Abgrenzung adäquat vornehmen zu können, soll das Modell des integrierten Geschäftsprozess- und Workflowmanagements erläutert werden, welches in Abbildung 1 dargestellt ist.

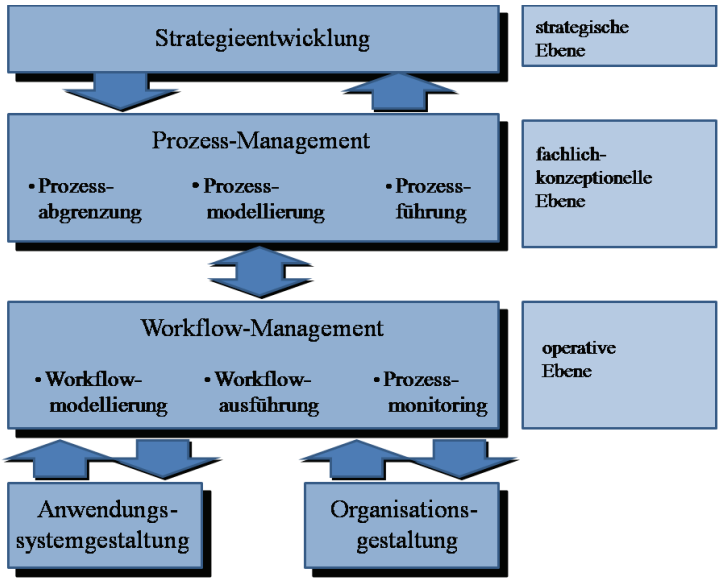


Abbildung 1: Integriertes Geschäftsprozess- und Workflowmanagement (aus [Gad08])

Das Modell des Integrierten Geschäftsprozess- und Workflowmanagements nach [Gad08] ist in 3 Ebenen gegliedert. Auf der strategischen Ebene steht die Unternehmensstrategie. Diese ist in den KMU maßgeblich für die internen Abläufe, welche an dieser Strategie ausgerichtet werden. Ist der Rahmen für die Ziele und Intentionen des Unternehmens klar, können sich die einzelnen Prozesse entwickeln, die Arbeiten innerhalb der KMU laufen nach bestimmten Mustern ab.

Damit diese Prozesse verbessert werden können, müssen sie im Bereich des Prozessmanagements erst einmal von einander abgegrenzt und anschließend modelliert werden. Dies geschieht auf der fachlich-konzeptionellen Ebene und führt zu einer nicht-automatisierten Darstellung der betriebsinternen Abläufe. Hier wäre eine Optimierung schon möglich, jedoch noch keine Automatisierung.

Die einzelnen, nicht-automatisierten Prozessmodelle werden in der Regel von Verantwortlichen in den Betrieben dokumentiert und dadurch auch modelliert.

Sollen die Prozesse automatisiert werden, so werden sie auf operativer Ebene innerhalb des Workflowmanagements mit Hilfe der Workflowmodellierung dargestellt und ausführbar gemacht. Dazu wird später noch Genaueres erläutert. Grundsätzlich leiten sich die Workflowmodelle aus den Prozessmodellen ab. Die Workflows und automatisierten Prozesse können mit Hilfe der IT-Technologie überwacht, gesteuert und auch optimiert werden. Da sich die Workflows aber aus den realen Prozessen ableiten, muss eine Änderung dieser auch wieder auf die höhere Ebene durchgereicht werden.

Diese Optimierung der Prozesse auf Workflowebene und damit verbundene Veränderung der eigentlichen Prozesse wird durch das Workflow-Life-Cycle Modell dargestellt, welches in Abb. 2 dargestellt ist.

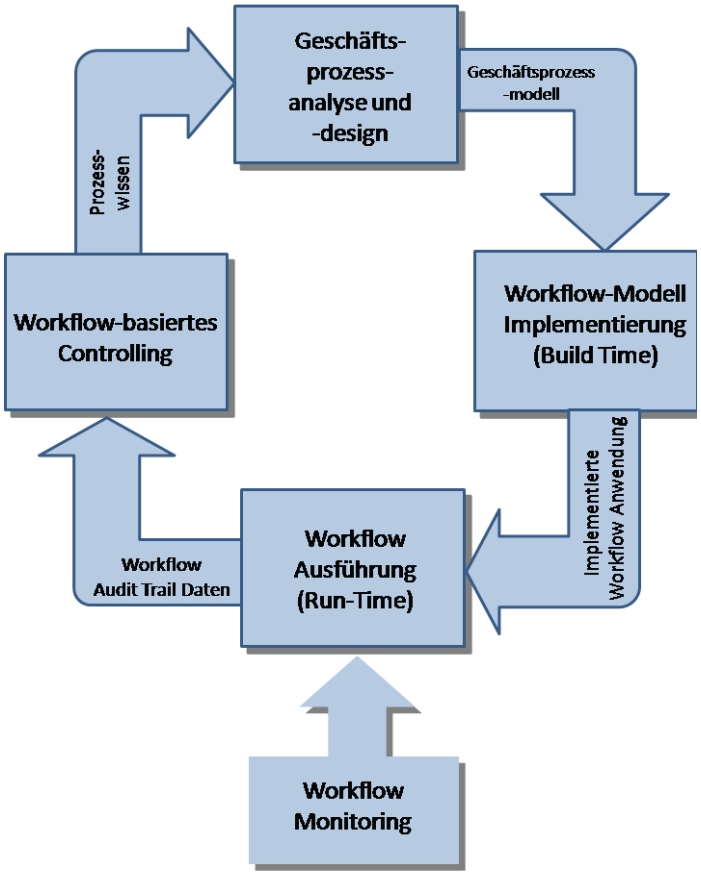


Abbildung 2: Workflow-Life-Cycle (aus [HzM03])

Hier werden zuerst einmal die Geschäftsprozesse des Unternehmens analysiert und in ein Geschäftsprozessmodell gegossen. Dieses wird in ein Workflowmodell überführt und kann daraufhin ausgeführt werden. Die ausgeführten Workflows werden überwacht und analysiert.

Die gewonnenen Erkenntnisse werden dazu genutzt, die eigentlichen Geschäftsprozesse zu verbessern. Diese werden dann erneut in einem Geschäftsprozessmodell verarbeitet und der Kreislauf beginnt von vorne.

Doch wie sieht so ein Geschäftsprozessmodell aus? Wie bei anderen Modellen auch, handelt es sich dabei um eine abstrakte Abbildung der Wirklichkeit. Geschäftsprozesse werden in einer Prozessdefinition spezifiziert. Diese kann in Teilprozesse zerlegt werden. Im Regelfall besteht eine Spezifikation aus einzelnen Aktivitäten. Dies geht mit den bereits erläuterten Definitionen eines Geschäftsprozesses konform. Diese Teilschritte sind entweder manueller oder automatisierter Natur (vgl. [Gad08] S.269).

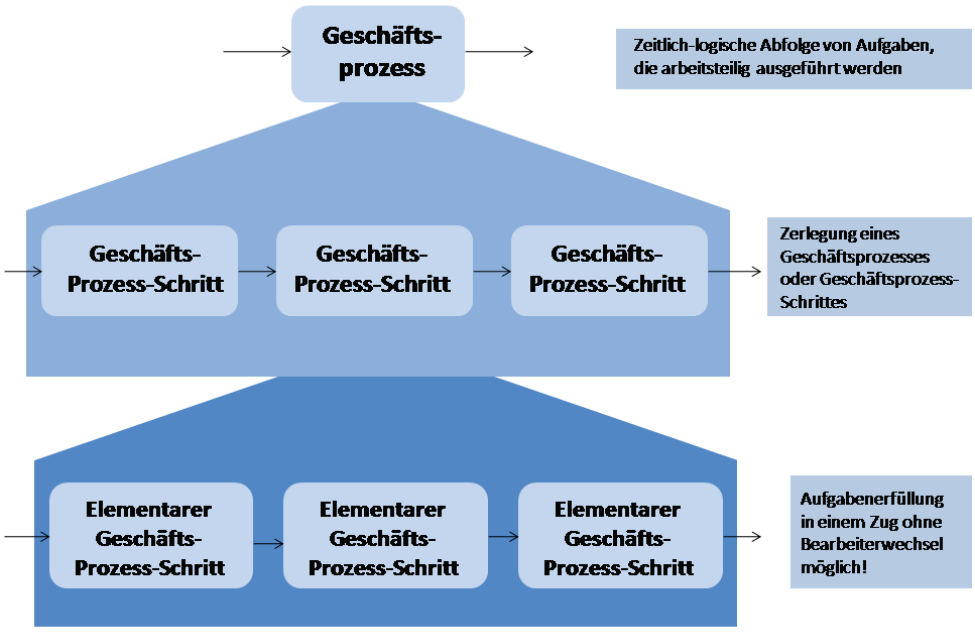


Abbildung 3: Zerlegung von Geschäftsprozessen (Prinzip) (aus [Gad08])

In Abb. 3 ist eine mögliche Zerlegung eines Geschäftsprozesses dargestellt, wie sie auch in der Modellierung berücksichtigt werden würde. Die Geschäftsprozessmodellierung als solche ist ein Unterstützungsinstrument des Prozessmanagements. Dabei werden die Bereiche der Konstruktion, Anwendung und Wartung von Modellen der Geschäftsabläufe eines Unternehmens abgedeckt. Die Geschäftsprozessmodelle sind spezielle Informationsmodelle, welche die Geschäftsprozesse repräsentieren. Sie beinhalten Aussagen über die Reihenfolge, Aufgaben, Daten und Akteure der Geschäftsprozesse (vgl. [WI Enz 01]).

Nach einer Modellierung von Prozessen bezüglich ihrer Aktivitäten und deren Reihenfolge soll der Schritt zur Prozessautomatisierung gemacht werden. Dabei ist zu beachten, dass Automatisierung nicht zwangsläufig bedeutet, dass keinerlei Interaktion zwischen Mensch und Maschine stattfindet (vgl. [JTB10] S.6).

Bei der Prozessautomatisierung laufen in der Regel die Prozessschritte nacheinander ab und der Nutzer erhält die Möglichkeit, an entsprechender Stelle Interaktionen zu tätigen. Im Mittelpunkt der Automatisierung steht eine Prozess-Engine. Diese arbeitet auf Grundlage eines technischen Prozessmodells und steuert den Ablauf der einzelnen Schritte, indem Beteiligte über ihre Aufgaben informiert, Eingaben verarbeitet und bestimmte IT-Systeme aufgerufen werden (vgl. [JTB10] S. 6f).

Die Grundlage hierbei bildet die Abbildung der realen Geschäftsprozesse auf ein technisches Modell. Diese Modelle sind die sogenannten Workflowmodelle. Im Folgenden sollen diese und auch deren Modellierung im Bereich des Workflowmanagements erläutert werden.

2.3 Grundlagen des Workflowmanagements

Neben dem Prozessmanagement und der Prozessmodellierung nicht-automatisierter Prozesse soll im Zuge der Automatisierung vor allem der Bereich des Workflowmanagement (WFM) betrachtet werden, der sich mit der Modellierung automatisierbarer Prozesse beschäftigt. Das WFM wird wie folgt definiert:

„Workflow-Management bezeichnet dagegen ein operatives Konzept zur Umsetzung der von der strategischen Unternehmensplanung vorgegebenen Geschäftsprozessziele. Zu diesem Zweck stellt das Workflow-Management Methoden und Werkzeuge zur computergestützten Analyse, Planung, Simulation, Steuerung und Überwachung von Arbeitsabläufen bereit.“ [Gad08] S.60.

Das WFM dient somit also der Ausführung und damit der Automatisierung von Prozessen.

Um den Schritt vom Prozess zum Workflow verstehen zu können, soll zunächst der Begriff des Workflows definiert werden. Dabei handelt es sich um einen Geschäftsprozess, welcher ganz bzw. zum Teil automatisiert ist. In diesem sind Spezifikationen enthalten, die eine automatisierte Ablaufsteuerung des Prozesses ermöglichen. Darunter fallen zeitliche, fachliche und auch ressourcenbezogenen Aspekte (vgl. [Gad08] S.53). Ein Workflow umfasst also Beschreibungen seiner Teilaufgaben, eine Festlegung bzgl. der Ausführung von diesen, die Beziehung unter den Aufgaben und die Ressourcen (vgl. [BeVo96] S.20).

Zur Modellierung und Ausführung dieser Workflows sind spezielle Softwaresysteme nötig, welche Workflowmanagementsysteme (WFMS) genannt werden.

Im Allgemeinen handelt es sich bei WFMS um Softwaresysteme, die der Modellierung, Ausführung und dem Monitoring, also der Überwachung von Workflows dienen.

Ein WFMS muss Workflows interpretieren können, die Bearbeitung bestimmter Arbeitsschritte durch das System oder den Nutzer veranlassen und Informationen bereitstellen, zum Teil auch durch externe Software (vgl. [Gad08] S.266f). Hier lässt sich eine Verbindung zum IMAP-Editor herstellen. Auch dieser dient, den Definitionen folgend, der Modellierung von Workflows, die Ausführung und das Monitoring werden durch das Montageportal übernommen, weshalb der IMAP-Editor als Teil des Montageportals und dieses als Ganzes ein Workflowmanagementsystem darstellen. Damit kann hier noch einmal der Kontext dieser Arbeit abgegrenzt werden.

Kernaufgabe eines solchen Workflowmanagementsystems ist die Unterstützung der Prozessabläufe durch die Koordination der verschiedenen Aktivitäten (vgl. [HzM03] S.385). WFMS enthalten detaillierte Informationen über geplante, aber auch in Ausführung befindliche Unternehmensabläufe. Dies soll den Informationsfluss stark verbessern (vgl. [Gad08] S.63).

Doch welchen Nutzen bergen das Konzept der Workflows und die Nutzung von WFMS? Nach [HzM03] gibt es eine Reihe von direkt messbaren, aber auch nebenher zu registrierenden Effekten, die von Vorteil sind.

Hauptnutzen ist hierbei die klar herauszustellende Zeiteinsparung. Transport- und Liegezeiten werden durch einen besseren Informationsfluss auf ein Minimum reduziert. Ein Mitarbeiter bekommt direkt und konkret die von ihm benötigten Informationen elektronisch breitgestellt und spart so die Suchzeit in Papierunterlagen. Dies führt zu einer Verkürzung der Durchlaufzeiten. Auch Leerphasen, in denen einige Arbeiter „aufgabenlos“ dastehen, können minimiert werden, da eine automatische Benachrichtigung über neue Tätigkeiten erfolgt. Die eigentlichen Abläufe können ebenfalls effizienter bearbeitet werden.

Mit Prozessinstanzen ist ein Priorisieren möglich und auch die Wahrscheinlichkeit des Untergehens „unwichtiger“ Prozesse kann durch die elektronische Bearbeitung vermindert werden. Diese Vorteile sind direkt messbar. Ein indirekter, positiver Effekt ist die Transparenz der Prozesse. Mit Hilfe der Workflows und deren Monitoring kann jederzeit der Zustand und Bearbeiter eines Prozesses ermittelt werden.

Die Kommunikation zwischen Kunde und Prozessbearbeiter kann so verbessert werden, was zu einer höheren Produktqualität führt. Die Auslastung unterliegt dem Monitoring ebenfalls, sodass Kapazitäten besser verteilt werden können. Durch die Rollensicht auf die Prozesse kann zudem effektiv der Ausfall von Mitarbeitern durch den Einsatz von Stellvertretern überbrückt werden (vgl. [HzM03] S.391f).

Doch wie wird aus einem Geschäftsprozess ein Workflow?

Um Prozessbeschreibungen ausführbar zu machen, müssen diese innerhalb der WFMS in Workflowmodellen beschrieben werden. Diesen Bereich deckt die Workflowmodellierung ab. Nicht jeder Geschäftsprozess ist allerdings für die Modellierung und Automatisierung mit Hilfe von Workflows geeignet. Um einen wirklichen Nutzen daraus ziehen zu können, sollten Prozesse modelliert werden, die nach einem festen Muster ablaufen, sich häufig wiederholen und dabei aus vielen Einzelschritten aufgebaut sind (vgl. [SeS08] S.30).

Die Workflowmodellierung ist die Überführung eines betriebswirtschaftlichen Geschäftsprozessmodells in ein ausführbares Workflowmodell (vgl. [Gad08] S.77). Bei der Ausführung wird zwischen manuellen Tätigkeiten außerhalb des WFMS und denen, die durch das WFMS kontrolliert werden, unterschieden. Dies geschieht durch die Instanziierung der Prozessdefinitionen bzw. Workflowmodellen (vgl. [Gad08] S. 269).

Ein wichtiger Aspekt der Workflowmodellierung ist, dass im Regelfall spezielle Workflowmodellier für sie zuständig sind. Oftmals wird diese Aufgabe auch Softwareentwicklern zuteil. Hierfür sind dann meist spezielle Kenntnisse der Modellierungssoftware nötig (vgl. [Gad08] S.77). Gerade im Zuge des Projektes IMAP ist es jedoch wichtig, dass die Modellierung sehr einfach möglich und nicht an solche Fachkenntnisse gebunden ist. Dafür sind einfache Werkzeuge nötig.

Neben dieser reinen Nutzbarkeit gibt es auch eine Reihe von Auswahlkriterien für Modellierungswerkzeuge, die sich auf die reine Technologie beziehen.

Nach [Gad08] sind dies unter anderem der Bedarf an IT-Ressourcen (Speicher, Hardware), die Nutzerinteraktionen, Import und Exportmöglichkeiten unterschiedlichster Formate, rollenbasierte Rechte für die User und die Möglichkeit der Einbindung externer Dokumente zu Informationszwecken.

Der Auswahl eines geeigneten Werkzeuges kommt also eine große Bedeutung zu. Dementsprechend wird später noch genau darauf eingegangen werden.

Neben der Wahl der Modellierungswerkzeuge kommt der Wahl der Beschreibungssprache und Darstellungsform der Workflows und Prozessmodellen ebenfalls eine entscheidende Rolle zu. Es gibt eine Vielzahl von Methoden und Sprachen, die skriptbasiert oder grafisch sein können (vgl. [Gad08] S.81). Im folgenden Unterkapitel sollen einige Standards zur Modellierung vorgestellt werden.

2.4 Standards für die Modellierung von Geschäftsabläufen

Für die Modellierung von Workflows und Prozessbeschreibungen gibt es Standards, die sich in der Praxis etabliert haben. Damit eine Spezifikation markttauglich wird, muss sich diese bestimmten Anforderungen stellen. Das folgende Unterkapitel soll einige dieser Besonderheiten aufzeigen.

Zusätzlich werden drei der meistverwendeten Modellierungsdarstellungen kurz erläutert und ihre potentielle Tauglichkeit für die Nutzung im integrierten Editor abgewogen. Dabei soll erneut zwischen Standards zur automatisierten und nicht-automatisierten Darstellung unterschieden werden.

2.4.1 Standards und Konventionen

Bei der Modellierung von Unternehmensprozessen gibt es eine große Spanne zwischen intuitiver Modellierung und hochfunktionalen Darstellungen. Natürlich-sprachliche Beschreibungen der Prozesse sind intuitiv und leicht verständlich, haben aber viele Freiheitsgrade und können relativ ungenau sein. Formale Sprachen dagegen sind sehr exakt, aber teilweise nur Experten verständlich (vgl. [WI Enz 02]). Häufig handelt es sich bei den Beschreibungen der Workflows um grafische Diagrammsprachen. Ein WFMS benötigt daher eine grafische Modellierungskomponente. Es gibt zwar einige Modellierungssprachen, die sich durchgesetzt haben, andere sind konforme Standards. Dennoch gibt es keinen einheitlichen Standard, der allgemein akzeptiert ist und als „DIE“ Methode für die Workflowmodellierung gilt (vgl. [HzM03] S.388).

Dies liegt daran, dass an die Modelle je nach Einsatzzweck unterschiedlichste Anforderungen gestellt werden. Werden Workflowmodelle für Simulation oder das Workflowmanagement eingesetzt, müssen diese sehr detailliert und formal exakt sein. Der Fokus liegt hierbei auf einer exakten Spezifikation und nicht auf der grafischen Darstellung.

Andere Einsatzzwecke wie die Organisationsgestaltung oder die prozessorientierte Reorganisation erfordern eher eine anschauliche Darstellung. Nur geschulte Mitarbeiter verwenden hier komplexere Modelle (vgl. [DRS03] S.76f).

Wurde eine Modellierungsspezifikation gewählt, so sollte diese innerhalb eines Unternehmens einheitlich sein. Dies wird mit Modellierungskonventionen erreicht (vgl. [DRS03] S.78). Fasst man alle Modellierungskonventionen zusammen, so erhält man einen Modellierungsstandard (vgl. [DRS03] S. 100). Dies ist notwendig, da bei der Modellierung verschiedenste Möglichkeiten gegeben sind, um das gleiche Ziel zu erreichen.

In der Geschäftswelt ist es nicht auszuschließen, dass es zu Personalwechseln innerhalb eines Betriebes kommen kann. Damit im Falle eines solchen Mitarbeiterwechsels der Sinn und die Intention eines Modells nicht verloren gehen und dieses auch bei einem neuen Modellierer oder neuen Nutzern weiterhin sinngemäß verwendet werden kann, sollte es bestimmten Regelungen unterliegen. Hier wäre beispielsweise eine Dokumentation sinnvoll. Es gilt für viele Bereiche eines KMU, dass eine sinnvolle Dokumentation Komplikationen bei einem Personalwechsel vermindert.

Da nun bestimmte Anforderungen an eine Modellierungssprache dargestellt wurden, sollen im Folgenden mehrere Varianten aufgezeigt werden, die in der Praxis häufig zum Einsatz kommen. Bei den Modellierungstechniken handelt es sich um die EPK, die BPMN und die Flowcharts.

Hier muss erneut zwischen Ausführung und nicht-Ausführung unterschieden werden. Während die EPK und die Flowcharts eher dazu dienen, Prozesse darzustellen, zielt die BPMN auf die Darstellung und Ausführung ab.

2.4.2 Die Ereignisgesteuerte Prozesskette

Die Ereignisgesteuerte Prozesskette (EPK) zählt zu den im deutschen Sprachraum am meisten genutzten Notationen zur Darstellung von Prozessen. Sie dient der anschaulichen Modellierung von Kontrollflüssen und soll auch für Nutzer ohne Modellierungswissen geeignet sein. Die EPK ist Bestandteil der ARIS-Architektur (Architektur Integrierter Informationssysteme) und basiert auf Petri-Netzen (vgl. [WI Enz 02] und [DRS03] S. 67).

Prozesse werden durch 3 Basiselemente dargestellt (vgl. [DRS03] S. 67ff). Dabei handelt es sich um Funktionen (Tätigkeiten), Ereignisse und Konnektoren. Ereignisse sind dabei Zustandsänderungen (Create, Delete, Update). Sie können Funktionen auslösen oder Zustände dokumentieren. Konnektoren dienen als Verknüpfungsoperatoren und führen Abläufe zusammen oder spalten diese.

Bei der Modellierung von EPK's gelten grundsätzliche Regeln. So werden immer nur unterschiedliche Knoten verbunden, also Ereignisse, die Funktionen anstoßen, und Funktionen, die ein Ereignis als Ergebnis haben. Prozessketten beginnen immer mit Ereignissen.

Zu den Grundelementen können noch weitere Informationsobjekte hinzugefügt werden und es gibt unterschiedliche Formen der EPK (zum Beispiel die erweiterte EPK: eEPK). Abb. 4 zeigt die Grundelemente und eine Beispiel.

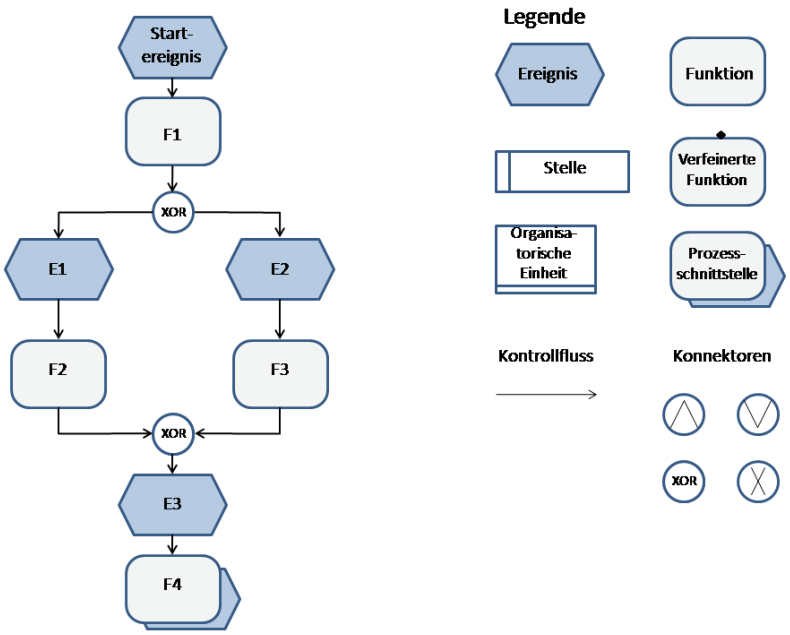


Abbildung 4: Ereignisgesteuerte Prozesskette: Objekttypen und Beispiel (aus [DRS03])

Nach [JTB10] haben EPK's allerdings den Nachteil, dass sie für die Prozessautomatisierung eher ungeeignet sind.

2.4.3 Flowcharts

Bei den Flowcharts handelt es sich um eine Notationsform, welche sehr häufig unter dem Namen Programmablaufplan (PAP) bekannt ist. Sie stammt aus dem Jahr 1966 und ist zweckmäßig eigentlich dafür vorgesehen, Algorithmen innerhalb von Programmen anschaulich darzustellen. Allerdings wird diese Notationsform häufig auch für andere Einsatzgebiete genutzt, da sich hiermit gut allgemeine Abläufe darstellen lassen. Programmablaufpläne sind normiert und in der DIN 66001 zusammengefasst und hinterlegt (vgl. [HMS11] S.26f).

Auch bei dieser Notationsform für die Darstellung von aufeinanderfolgenden Aktivitäten gibt es eine bestimmte Symbolik.

Die meistverwendeten dieser Symbole sind der Vorgang, das Unterprogramm, die Verzweigung, die Grenzstelle, Ein- und Ausgabeelemente sowie die Ablauflinie, die optional durch einen Pfeil erweitert werden kann. Diese grafischen Elemente sind in Abbildung 5 dargestellt.

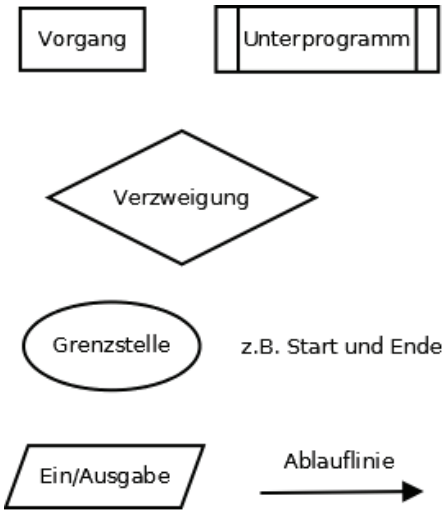


Abbildung 5: Elemente der Flowchart-Darstellung

Ähnlich wie die EPK, dienen die Flowcharts eher der rein grafischen Darstellung und sind für eine Ausführung nicht vorgesehen. In den Partnerunternehmen des Projektes kommt es jedoch vermehrt zum Einsatz dieser Notation zur Modellierung der vorhandenen Geschäftsprozesse. In den QM-Handbüchern sind in der Regel die Abläufe mit PAP's notiert. Dieser Modellierungsstandard kann dementsprechend bei den Projektpartnern als bekannt vorausgesetzt werden.

2.4.4 Business Process Model and Notation

Im Gegensatz zu der nicht für die Automatisierung geeigneten EPK und den Flowcharts stellt es sich bei der BPMN anders dar. Die Business Process Model and Notation in der Fassung 2.0 ist die Definition einer standardisierten Sprache zur Erstellung grafischer Modelle von Geschäftsprozessen. Sie hat dabei die Intention, für alle Beteiligten leicht verständlich zu sein (vgl. [MG 01]).

Die BPMN verfolgte von Anfang der Entwicklung an das Ziel, eine für die Automatisierung von Prozessen geeignete, grafische Notation zu beschreiben (vgl. [JTB10] S.8).

Sie verknüpft also eine einfache, gut zu verstehende grafische Beschreibung mit der Möglichkeit, die erstellten Modelle unkompliziert ausführen zu können, ohne sie in eine kompliziertere Beschreibungssprache überführen zu müssen. Die BPMN ist standardisiert. Die Version 2.0 ist seit Anfang 2011 öffentlich und damit ein momentan aufkommender Standard, der in vielen Anwendungen zum BPM Verwendung findet.

Ähnlich wie die EPK besteht die BPMN aus einigen Grundelementen, die allerdings um viele spezifische Eigenschaften erweitert werden können.

Die Grundelemente sind Aktivitäten, Gateways für Entscheidungen, Events, Sequenzflüsse zum Verbinden, Datenobjekte, Nachrichten, Gruppierungen und Anmerkungen. Hinzu kommen sogenannte Swimlanes, welche für am Prozess beteiligte Rollen stehen (vgl. [JTB10] S.21).

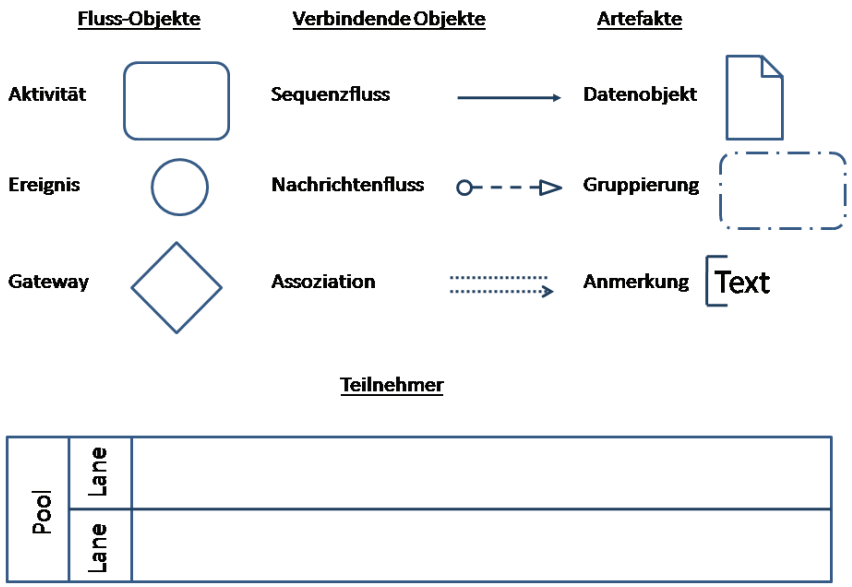


Abbildung 6: Die Kernelemente der BPMN (aus [JTB10])

Abbildung 6 zeigt die grundlegenden Elemente der BPMN. Diese können allerdings noch vielfältig erweitert und spezifiziert werden. So können Task (Aktivitäten) als UserTask spezifiziert werden, welche durch eine Interaktion mit dem Nutzer beendet werden.

Die fertig modellierten Prozesse können als XML-Dateien gespeichert werden und sind damit leicht ex/importierbar. Innerhalb der Spezifikation ist vorgeschrieben, wie welches Element in XML darzustellen ist (vgl. [JTB10] S.199).

Neben dieser syntaktischen Spezifikation beschreibt die BPMN 2.0 auch noch die Ausführungssemantik, also wie die einzelnen Elemente bei der Ausführung durch eine beliebige, BPMN 2.0 – konforme Prozess-Engine reagieren (vgl. [JTB10] S.200).

Im Endeffekt hat die BMPN zwei Funktionen: neben der Darstellungsmöglichkeit für andere XML-basierte Ausführungssprachen wie WSBPEL, die von Prozess-Engines gelesen werden können, sollen die Modelle auch direkt ausführbar sein (vgl. [MG 01]).

Nach dieser umfassenden Darstellung theoretischer Grundlagen zu Management, Modellierung und Standards von sowohl ausführbaren, als auch nicht-ausführbaren Prozessen und Prozessbeschreibungen sollen im Folgenden die Rahmenbedingungen für den IMAP-Editor skizziert werden. Dazu wird das Montageportal näher erläutert und es werden Anforderungen an ein Modellierungswerkzeug gestellt, welche sich aus den Arbeitsbereichen des IMAP-Projekts ergeben.

3 Ausgangssituation für den IMAP-Editor

Das folgende Kapitel dient der Erläuterung der grundsätzlichen Ausgangssituation des IMAP-Editors. Dabei wird dessen Position innerhalb des Projektes deutlich gemacht und es werden Anforderungen an den Editor erläutert, welche anhand der Arbeiten innerhalb der Teilbereiche von IMAP erarbeitet wurden.

3.1 Das Montageportal

Als eines der Ergebnisse des zuvor beschriebenen Forschungsvorhabens IMAP soll mit dem sogenannten integrierten Montageportal eine Softwarelösung entstehen, die sowohl alle unternehmensweiten Daten vereint und vernetzt, als auch Anwendungen bereitstellt, um eine effiziente Montageplanung und –steuerung innerhalb der KMU durchführen zu können und damit die Grundlage für eine Geschäftsprozessoptimierung innerhalb dieser Unternehmen zu legen.

Bisher wurde dazu ein Grundverständnis für die Inhalte des Projektes aufgebaut und es wurden theoretische Grundlagen vermittelt.

Das Hauptthema dieser Arbeit befasst sich mit der Erstellung eines Modellierungswerkzeuges zur integrierten Modellierung von Prozessbeschreibungen und Formularen. Es stellt dabei einen Teil des Montageportals dar. Die Integration der Formulare in die Prozessbeschreibungen zur Interaktion des Nutzers mit dem System während der Prozessausführung ist in der Praxis kaum berücksichtigt. Deshalb zielt der IMAP-Editor darauf ab, Möglichkeiten aufzuzeigen, Prozessbeschreibungen und Nutzerinteraktionen miteinander zu verknüpfen und diese praktisch umzusetzen.

Um eine Verbindung zwischen dem Projekt als Ausgangspunkt und dem im weiteren Verlauf hergeleiteten Konzept des IMAP-Editors herstellen zu können, ist es notwendig, das integrierte Montageportal (MP) näher zu erläutern. Dabei wird zunächst auf die genauen Aufgaben des Portals eingegangen und spezifische Anforderungen werden aufgezeigt. Nach einer genauen Betrachtung der einzelnen Komponenten und deren Funktionalität und Verknüpfung wird auf die Position des IMAP-Editors innerhalb des Portals und des unterliegenden Systems eingegangen.

3.1.1 Konzeption des Montageportals

Das integrierte Montageportal stellt ein modulares System dar, welches alle Daten eines Unternehmens zusammenfasst und auf dieser Grundlage Methoden und auch Anwendungen bereitstellt, um die Montage in den KMU mitarbeiterzentriert und auch arbeitsprozessintegriert planen und durchführen zu können (vgl. [HMS11] S.3).

Die Mitarbeiterzentrierung (vgl. Kapitel 1.2) wird durch die Betrachtung der Arbeitsprozesse aus Sicht des Mitarbeiters erreicht, die Arbeitsprozessintegration hat ihre Ursache in der Verwendung realer Daten der Unternehmen und schon vorhandener Infrastrukturen.

Weitere wichtige Aspekte des Montageportals stellen die sogenannten Incentives dar. Dabei handelt es sich um Motivationsanreize, die das entstandene Portal und die Arbeit mit diesem in den KMU etablieren und eine Nutzung der Software zur Verbesserung motivieren sollen. Den Arbeitern muss dabei bewusst werden, dass das MP eine wirkliche Verbesserung ihrer Arbeitssituation hervorbringen kann.

Hinzu kommt der schon erwähnte Wissenstransfer, sodass bestimmte Kompetenzen nicht verloren gehen und Informationen zielgerichtet Verwendung finden können.

Bei dem Portal soll es sich um ein unternehmensweit zugreifbares Framework handeln (vgl. [HMS11] S.4). Entscheidend ist eine Nutzbarkeit für jeden Mitarbeiter zu jeder Zeit auf unterschiedlichen Medien (vgl. [IMAP 01]). Jedoch soll sich nicht nur der Zugriff auf das Portal über das gesamte Unternehmen erstrecken. Auch soll eine unternehmensweite Vernetzung unterschiedlicher Systeme stattfinden.

Alle Konzepte und Technologien, die in den anderen Arbeitsbereichen des Projektes IMAP entwickelt werden, fließen in das Portal ein und werden mit den schon im Unternehmen vorhandenen Systemen (Hard- und Software) integriert. Um dies zu erreichen müssen einheitliche Schnittstellen geschaffen werden.

Neben der Verknüpfung von etablierten mit neu entwickelten Systemen bekommt der Nutzer die Möglichkeit, spezifische Informationen angezeigt zu bekommen. Diese sind auf seine Rolle bzw. seine Aufgabe abgestimmt. Dabei werden verschiedenste Datenquellen zugänglich gemacht. Da das Portal auf einem gemeinsamen Datenmodell basiert, das versucht, reale Strukturen nachzubilden, ist nicht nur eine Wiedergabe dieser Daten entscheidend, sondern auch die Möglichkeit, Eingaben von Nutzerseite entgegen nehmen zu können und diese in das Datenmodell einfügen zu können (vgl. [HMS11] S.4).

Das Portal erfüllt damit zwei große Aufgabenkomplexe: die Informationspräsentation und die Informationserfassung, beides auf dem Datenmodell aufbauend (vgl. [HMS11] S.13). Hinzu kommen Dienste zur Kommunikation und zur Wissensverwaltung. Da die Präsentation der Informationen an den Nutzer gebunden ist, muss zu jeder Zeit bekannt sein, wo sich der Nutzer befindet. Zudem muss das System wissen, welches Ein-/Ausgabemedium der Nutzer zur Verfügung hat (Terminal-PC, Mobiles Endgerät).

Neben einer Auskunft über die aktuellen Aufgabengebiete muss auch die Form bekannt sein, in der die entsprechenden Informationen dargestellt werden (vgl. [HMS11] S.13).

Für die Erfassung einfacher Daten durch den Nutzer in das Datenmodell oder über spezifische Schnittstellen in die lose verlinkten, anderen vorhandenen Softwaresysteme muss im Portalsystem bekannt sein, welche Arten von Informationen erfasst werden sollen, und wo diese abgelegt werden (vgl. [HMS11] S.14).

Ein komplexes Softwareprojekt, wie es das integrierte Montageportal darstellt, bringt eine Reihe spezifischer Anforderungen mit sich. Diese sind im Folgenden aufgelistet, wobei zunächst die Funktionalität erfasst werden soll:[1]

- Das MP soll als logistischer Leitstand für alle am Prozess beteiligten Personen dienen.

- Innerhalb des MP sollen die Prozesse möglichst realitätsnah erfasst, modelliert und auf dieser Grundlage optimiert werden.

- Das Portalsystem soll eine konsistente Zeiterfassung unterstützen, um Produktionsprozesse effizienter zu gestalten. Dies umfasst die Registrierung der tatsächlichen Arbeitszeiten an Geräten und die Dauer der Arbeitsschritte.

- Die Daten der Zeiterfassung sollen als direkte Planungsgrundlage für die Arbeitsvorbereitung dienen, um Aufwände besser abschätzen zu können, das Zeitmanagement zu verbessern und Durchlaufzeiten zu verringern.

- Das Montageportal soll den Materialfluss transparent halten, sodass ein Vorarbeiter zum Beispiel immer weiß, wie viel von welchem Material wo verarbeitet wird und wo es zu Engpässen kommen könnte. Eine Aktualisierung soll dabei zeitgenau stattfinden.

- Veränderungen während der Prozesse sollen sauber erfasst und Informationen direkt weitergeleitet werden. Der Mitarbeiter erhält die Möglichkeit, direkt Probleme zu melden.

- Erfasste Daten sollen möglichst lückenlos im Portal abgelegt werden können, alle Änderungen oder Probleme werden dokumentiert. Dabei soll unnötiges Datenmaterial vermieden werden und ein Übergang von analogen Dokumenten zu digitalen Darstellungen erfolgen.

- Soziale Aspekte sollen berücksichtigt werden. Der Mitarbeiter muss bereit sein, mit dem Portalsystem zu arbeiten.

[1] Vgl. dazu [IMAP 03], [HMS11] S.5f

Neben diesen Anforderungen an die Funktionalität des Portals gibt es auch noch einige, welche die Architektur betreffen[2]:

- Für das Montageportal ist ein generischer Charakter entscheidend, um es in diversen KMU einsetzen zu können.
- Eine Anpassung des Montageportals auf die Bedürfnisse einzelner KMU soll ohne Programmierkenntnisse durchführbar sein.
- Das Portal erfordert eine hochflexible und skalierbare Architektur, da Prozesse, die IT-Infrastruktur und Mitarbeiter kontinuierlichen Veränderungen unterliegen.
- Innerhalb eines Unternehmens werden offene Schnittstellen und Datenformate benötigt, um externe Anwendungen in das Portal integrieren zu können.
- Konfigurationswerkzeuge sollen intuitiv und einfach zu bedienen sein. Hierzu zählt auch der IMAP-Editor.
- Das Montageportal vereint unterschiedliche Anwendungen. Dies umfasst unter anderem Modellierungswerkzeuge, semantische Suchdienste und Kommunikationsmöglichkeiten. Hinzu kommt eine Laufzeitumgebung zum Ausführen erstellter Prozessbeschreibungen.

Die unterschiedlichen Arbeiten der einzelnen Arbeitspakete und die damit verbundenen Analysen der Partnerunternehmen dauern zum Zeitpunkt dieser Arbeit noch an. Gleiches gilt für die Umsetzung des Montageportals. Das Projekt läuft zu diesem Zeitpunkt noch mindestens ein Jahr, sodass es sich bei dem im Folgenden erläuterten Prototypen des Portals noch nicht um die endgültige Lösung handelt. Durch die iterative Arbeit an der Lösung werden nach und nach mehr Anforderungen realisiert, weshalb noch nicht alle zuvor erläuterten Funktionalitäten umgesetzt sind.

Die folgenden Ausführungen beschreiben die erste prototypische Umsetzung einiger Grundfunktionen des Montageportals und dessen Komponenten zum Zeitpunkt Juni 2011.

Abbildung 7 zeigt die einzelnen Teile des Systems. Diese werden folgend näher erläutert.

[2] Vgl. dazu [HMS11] S. 7, 13 und 14

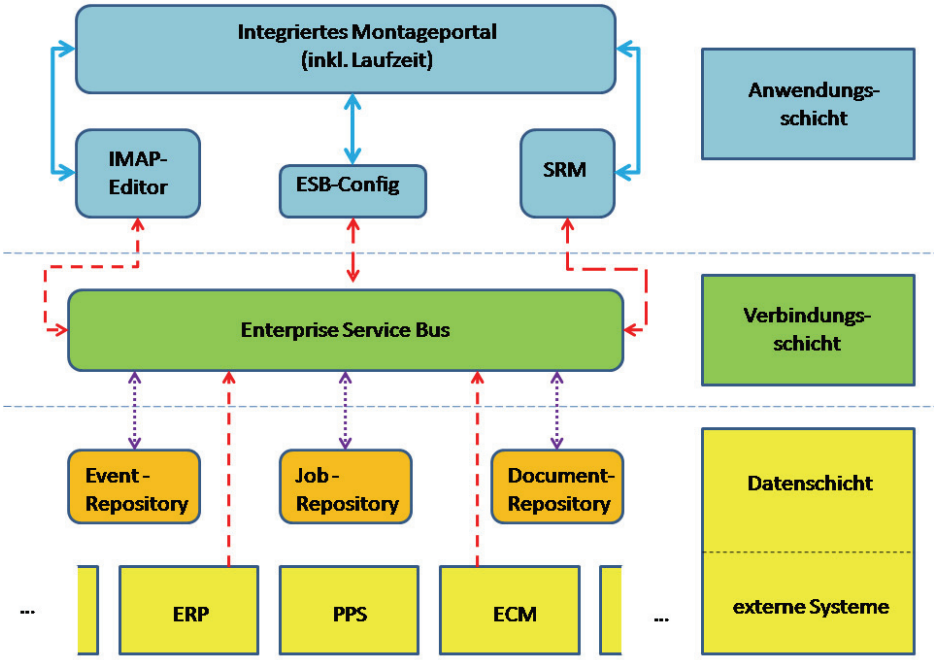


Abbildung 7: Komponenten des Montageportals

Die Komponenten des integrierten Montageportals lassen sich in drei Schichten einordnen. Die Anwendungsschicht bilden die Teile des Systems, die eine direkte Interaktion mit dem User ermöglichen. Dazu gehört das eigentliche Portal, welches die Laufzeitumgebung zur Ausführung der automatisierten Prozesse verkörpert. Hinzu kommen weitere Komponenten, wie das ESB-Config-Werkzeug und der Semantic Reference Manager, die später noch erläutert werden.

Zu dieser Anwendungsschicht gehört zudem der IMAP-Editor. All diese Komponenten sind im Portal vereint und integriert. Besonders die Verbindung zwischen dem Editor zur Modellierung der Prozesse und Formulare und dem Portal als ausführende Instanz ist hier hervorzuheben. Der Editor dient der Modellierung und schafft dementsprechend erst die Grundlage für die Nutzung und Optimierung der anderen Komponenten innerhalb der KMU, da durch ihn die realen Strukturen der Unternehmen für eine automatisierte Nutzung aufbereitet werden.

Die Verbindungsschicht bildet der Enterprise Service Bus (ESB). Dabei handelt es sich um die Schnittstelle zwischen allen Komponenten[3].

Das schon erwähnte, gemeinsame Datenmodell des Portals wird hier mit den Unternehmensdaten aus externen Quellen in Verbindung gesetzt und auch ein Zugriff interner Komponenten auf Teile des Datenmodells wird über diese Schnittstelle realisiert. Im Projekt IMAP kommt die ESB-Implementierung MULE zum Einsatz.

[3] Bei einem ESB im Allgemeinen handelt es sich um eine Integrationsinfrastruktur, die dazu dient, verschiedenste Softwaresysteme miteinander in Verbindung zu bringen (vgl. [ESB 01]).

In dieser werden zum Beispiel die Repositorien, auf die gleich noch eingegangen wird, als Beans eingebunden und es wird festgelegt, auf welchem Port bzw. auf welcher Adresse http-Calls (REST) eingehen, worauf die interne Kommunikation basiert[4].

Genutzt wird hierfür die JAX-RS, die Java API for RESTful Web Services und deren Referenzimplementierung Jersey[5]. Diese ermöglichen es, Java-Klassen als Ressourcen über einen einzigartigen URI anzusprechen. Je nach Art und Methode dieses Calls, können innerhalb der Ressource verschiedene Reaktionen definiert und unterschiedliche Rückgabewerte, wie XML-Strukturen, ermöglicht werden (vgl. [Rest 01]).

Für die Kommunikation innerhalb des Montageportals wird diese Technik verwendet, um beispielsweise Datenobjekte als XML-Struktur auszutauschen. Durch Mule wird die Grundadresse definiert, auf der die Ressourcen die Calls entgegennehmen. Werden diese wie ein WebService angesprochen, werden im Java-Code der Ressourcen Datenobjekte des gemeinschaftlichen Datenmodells in XML serialisiert und als Response verfügbar gemacht.

Neben der Anwendungsschicht und der Verbindungsschicht existiert zudem die Datenschicht.

Diese umfasst die erwähnten Repositorien und die externen Softwaresysteme der vorhandenen Unternehmensinfrastruktur innerhalb der KMU. Innerhalb der Systemgrafik (vgl. Abbildung 7) werden sie durch die gelben Rechtecke dargestellt.

ERP steht für Enterprise Ressource Planning. Dabei handelt es sich um Planungssoftware. Bekannteste Lösung ist hierbei SAP, in den Partnerunternehmen sind dies vor allem Microsoft Navision und Sharepoint. PPS sind Produktionsplanung- und Steuerungssysteme, ECM sind Enterprise Content Management Systeme. Dies ist die externe Software, die durch das Portal mit den eigenentwickelten Komponenten integriert werden muss.

In IMAP wird versucht, die Integration mit Hilfe der mit ESB-Config bezeichneten Komponente zu lösen. Dabei handelt es sich um ein Tool, dass es ermöglichen soll, die Daten aus den externen Systemen einfach und intuitiv in das Datenmodell zu übernehmen. Dieses Tool befindet sich noch in der Entwicklung. Für alle prototypischen Umsetzungen, die bisher entwickelt wurden, kamen noch keine realen Daten der Firmen zum Einsatz. Es wurde mit Demo-Daten gearbeitet, um die grundsätzliche Machbarkeit der Konzeption der einzelnen Komponenten aufzuzeigen.

Weitere Module des Systems sind die Repositorien. Hier seien nur beispielhaft das Document Repository und das Event Repository genannt. Diese Komponenten gehören ebenfalls zum internen Datenmodell und stellen bestimmte Informationen strukturiert dar. Beispielsweise verbirgt sich hinter dem Event Repository eine Hibernate-Anbindung zu einer MySQL-Datenbank, in die zum Beispiel bei einer Störung ein neuer Eintrag gemacht wird.

[4] REST oder Representational State Transfer ist ein Architekturstil, der im Grunde leichtgewichtige WebServices ermöglicht, also der Datenübertragung per http-Protokoll dient.

[5] Vgl. http://jersey.java.net/ [15.08.2011]

Das Document Repository ist ein Pool von Dokumenten, auf den der SRM dann zugreifen kann. Hinter diesem Semantic Reference Manager (SRM) verbirgt sich eine Komponente, die semantisch und kontextbezogen aus einem Dokumentenfeld die passendsten Informationen herausfiltert, Volltextsuchen durchführt oder Dokumente indiziert. Es werden hier Ähnlichkeitsbetrachtungen durchgeführt und Listen entsprechender Dokumente erstellt.

Nach diesem Überblick über das gesamte Portal stellt das folgende Unterkapitel den IMAP-Editor im Kontext des Montageportals dar.

3.1.2 Der IMAP-Editor im Kontext des Montageportals

Sämtliche bisherigen Ausführungen bezüglich des Projektes IMAP und des Montageportals dienten dazu, ein Grundverständnis aufzubauen, in welchem Kontext das Thema der Arbeit zu sehen ist. Wie aus den Beschreibungen hervorgeht, dient das Montageportal dem Zweck, eine Möglichkeit zu schaffen, Geschäftsprozesse zu optimieren. Dabei soll die entstehende Lösung möglichst generisch und damit auf eine Vielzahl von KMU übertragbar sein.

Da die Geschäftsprozesse dieser Unternehmen automatisiert werden sollen und eine möglichst intuitive Bedienbarkeit des Systems unabhängig von externen IT-Dienstleistern gefordert ist, werden, wie erwähnt, Werkzeuge benötigt, um das System nach den Wünschen der KMU zu konfigurieren. Im Detail heißt dies nichts anderes, als die im Unternehmen vorhandenen Prozesse in das IMAP-System übertragen zu können.

Bei den entsprechenden Konfigurationswerkzeugen handelt es sich zum einen um einen Prozesseditor zur Darstellung der Abläufe und ein Formulareditor zur Generierung der Eingabemasken für die Benutzer.

Auf das Thema zurückgeführt ist es das Ziel der Arbeit, eben solche Konfigurations- bzw. Modellierungswerkzeuge zu entwickeln. Dabei soll der IMAP-Editor als Teil des Montageportals entstehen.

Ausgehend von der geforderten Automatisierung sollen also maschinenlesbare Prozess- und Formularbeschreibungen entstehen. Ein entscheidender Aspekt liegt hierbei auf der Integration von Prozessen und Formularen. Es ist gefordert, dass zu bestimmten Prozessschritten Nutzerinteraktionen zugeordnet werden. Sei es ein einfaches Log-In vor dem Arbeiten oder die Meldung fehlender Materialien beim Vorarbeiter.

Diese konkrete Zuordnung von Interaktionen zu bestimmten Prozessschritten, also die Integration von Prozess und Formular ist eine der Hauptanforderungen an das Montageportal und damit auch an den IMAP-Editor als Teil dieses Portals. Diese konkrete Funktionalität ist in der Praxis bisher kaum und nur kompliziert umsetzbar verfügbar. Eine möglichst intuitiv und leicht zu bedienende Variante dieser Integration soll der IMAP-Editor ermöglichen.

Auf dem Markt gibt es eine Vielzahl von einzelnen Prozesseditoren[6] und auch eine Vielzahl von Formulareditoren[7].

Kombinierte Varianten, die sehr einfach auf die Bedürfnisse der KMU angepasst werden können, und dabei auch noch intuitiv zu bedienen sind, gibt es aber mehr oder weniger gar nicht. Da es aber konkrete Anforderungen an die in IMAP benötigten Modellierungswerkzeuge gibt, ist es nicht leicht, das „beste“ System für das Projekt zu finden. Im Vorfeld dieser Arbeit wurden eine Vielzahl von Systemen getestet und evaluiert[8], ohne dabei das „Non-Plus-Ultra“ finden zu können.

Darum soll diese Arbeit die Frage nach einem Modellierungswerkzeug beantworten, dass zum einen in das Portal und Konzept des Projektes passt und intuitiv, einfach und ohne Programmierkenntnisse dazu geeignet ist, die entsprechenden Abläufe der Unternehmen digital darzustellen. Zudem sollen auf sinnvolle Art und Weise Prozessbeschreibungen so mit Formulareingaben verknüpft werden, dass an der richtigen Stelle die richtigen Eingaben getätigt werden können. Erreicht werden soll dies durch den IMAP-Editor.

Mit der Positionierung des IMAP-Editors als Teil des Portals wurde der Gesamtrahmen für die Ziele dieser Arbeit abgesteckt. Die Notwendigkeit der Integration von Prozessbeschreibungen und Formularen wurde erläutert. In den folgenden Kapiteln wird das Modellierungswerkzeug zur Erstellung der benötigten Beschreibungen Schritt für Schritt entworfen und umgesetzt. Einige grundlegende Anforderungen an den IMAP-Editor wurden bereits dargestellt. Da aber neben der essentiellen Integration von Prozessen und Formularen und der intuitiven Bedienbarkeit bei der Modellierung noch andere spezielle Anforderungen an diese Anwendung gestellt werden, beschäftigt sich das nächste Kapitel mit diesen.

[6] Beispielsweise:
Signavio (http://www.signavio.com/de.html [15.08.2011]),
Enhydra JaWE (http://www.together.at/prod/workflow/twe [15.08.2011])
Bonapart (http://www.btc-ag.com/bonapart [15.08.2011])

[7] Beispielsweise:
Orbeon (http://www.orbeon.com/ [15.08.2011])
Cirali (http://www.fjd.de/ced-main.htm [15.08.2011])

[8] Beispielsweise:
Oryx (http://code.google.com/p/oryx-editor/ [15.08.2011])
Bonita Open Studio (http://www.bonitasoft.org/ [15.08.2011])

3.2 Anforderungen an den IMAP-Editor

Bei IMAP handelt es sich um ein Forschungsvorhaben, dass neue Technologien und Konzepte erarbeitet, weshalb bestehende Definitionen, wie sie in Kapitel 2 erarbeitet wurden, eher dem allgemeinen Verständnis dienen sollten. Dementsprechend bringt ein solches Projekt ganz eigene, spezielle Anforderungen mit sich, welche sich aus den Zielen und Vorhaben ergeben. Dieses Kapitel soll einen Überblick über solche Anforderungen geben, auf deren Grundlage der IMAP-Editor entwickelt werden kann.

Dabei werden zunächst die Zielgruppe und die Einsatzgebiete bzw. das IMAP-Szenario erläutert. Nach diesen Rahmenbedingungen werden konkrete Anforderungen an Prozess- und Formulareditor gestellt. Von essentieller Bedeutung ist dabei die Integration der Formulare und Interaktionsmasken in den Ablauf der Workflows. Diese Möglichkeit der Verknüpfung ist nur in wenigen existierenden Tools vorhanden und in vielen Prozessspezifikation vom Funktionsumfang nicht einmal vorgesehen. Deshalb ist die Integration auch einer der Hauptvorteile des IMAP-Editors und damit auch wichtiger Teil dieser Arbeit.

Zunächst soll jedoch der Rahmen für die Anforderungen abgesteckt werden.

3.2.1 Die Zielgruppe des IMAP-Editors

Bevor die Entwicklung des IMAP-Editors beginnen kann, muss im Vorfeld erst einmal klar sein, wer die Zielgruppe dieses Vorhabens ist. Den Anforderungen des Projektes und des Montageportals folgend, muss der entstehende Editor in der Arbeitsumgebung der KMU und in bestehenden Infrastrukturen einsetzbar sein.

Hier jedoch davon auszugehen, dass „kleine und mittelständische Unternehmen“ als Zielgruppendefinition ausreichend ist, wäre jedoch zu unpräzise. Deshalb ist eine Einschränkung der möglichen Zielgruppe nötig.

Die Analysen der Ist-Zustände innerhalb der Unternehmen konnten mehrere Nutzergruppen identifizieren. Für den IMAP-Editor relevant ist dabei die Gruppe des Qualitätsmanagements (QM). Andere Nutzergruppen existieren, kommen jedoch nicht mit der Nutzung des IMAP-Editors in Berührung, weshalb sie an dieser Stelle nicht berücksichtigt werden sollen.

Bei den QM-Verantwortlichen innerhalb der KMU muss von folgender Ausgangssituation ausgegangen werden: Sie besitzen fachliche Kompetenz, was die internen Prozesse angeht. Diese sind meist in Papierform und im Stile von Programmablaufplänen modelliert. Zum Einsatz kommen dabei Modellierungswerkzeuge wie Microsoft Visio, mit denen die Modellierung allerdings zum Teil sehr lange dauert. Viele relevante Daten werden in Microsoft Excel dargestellt. Zu Planungszwecken kommen Systeme wie Navision oder Sharepoint zum Einsatz. Es kann also davon ausgegangen werden, dass ein Grundverständnis des Umgangs mit einem PC vorhanden ist.

Was allerdings in der Regel nicht vorhanden ist, sind Programmierkenntnisse. Das heißt zum Beispiel, eine Abbildung der grafisch erstellten, bekannten Prozesse auf eine XML-basierte Ausführungssprache wie WS- BPEL[9] zur Automatisierung der Geschäftsprozesse wird ohne weiteres nicht möglich sein. Gefordert ist deshalb ein intuitives System, dass der QM-Abteilung ohne viel Mehraufwand ermöglicht, die Prozesse digital darzustellen und sie ausführbar zu machen, ohne dass programmiert werden muss. Auch eine Anpassung des Systems an interne Bedürfnisse (Templates, Corporate Design) soll intuitiv von statten gehen können. Im Allgemeinen hat das Qualitätsmanagement des Unternehmens die Aufgabe, die betriebsinternen Geschäftsprozesse in Workflowmodellen darzustellen. Durch das QM werden die Interaktionsformulare erstellt und mit den Workflows verknüpft.

Nach dem Abstecken der Zielgruppe und deren Aufgaben folgen die konkreten Anforderungen an den Prozesseditor und den Formulareditor als Teil des IMAP-Editors.

3.2.2 Anforderungen an das integrierte Modellierungswerkzeug

In diesem Abschnitt sollen die Anforderungen an die Komponenten des IMAP-Editors dargestellt werden. Zu Beginn soll der Prozesseditor zur Modellierung der Workflows untersucht werden.

Hierbei werden zunächst die funktionalen Anforderungen betrachtet:

- Der Prozess- oder Workfloweditor soll der Modellierung von Prozessdefinitionen dienen.
- Die Prozessdefinitionen sollten folgende Grundelemente enthalten , wobei der Fokus auf der Mitarbeiterzentrierung und der Rollensicht eines Prozesses liegen soll (vgl. [HMS11] S. 15):
    - Teilschritte
    - die Reihenfolge dieser Teilschritte
    - die zu den Teilschritten gehörenden Interaktionsformulare

  Um die oben genannten Grundfunktionen modellieren zu können, sollen folgende Modellierungselemente verwendet werden (vgl. [HMS11] S.17):

- Prozessschritte (einzelne Aktivitäten)
- zu einem Prozessschritt gehöriges Formular

[9] Vgl. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html [15.08.2011]

- mögliche Prozessvariablen zur Weiterverwendung bestimmter Eingaben (direkte Verwendung auf der Arbeitsfläche ist optional)
- Sequenzflüsse, Verbindungselemente
- Entscheidungen und Bedingungen
- Teil- oder Unterprozesse (eigenständige Prozesse, die separat modelliert werden können)
- Verzweigungen zum parallelen Ablauf (optional)
- Warte-Elemente (Reaktion auf Ereignisse, optional)
- klar spezifizierte Darstellungen von Start und Ende (optional, Start = Aktivität ohne Eingangsverbindung, Ende = Aktivität ohne Ausgang)

Neben diesen umzusetzenden Grundelementen gibt es noch einige weitere Anforderungen, auf denen eine hohe Priorität liegt (vgl. [HMS11] S.18):

- Es soll der WYSIWYG-Grundsatz Anwendung finden (What you see is what you get, Darstellung zum Modellierungszeitpunkt entspricht Darstellung zur Ausführungszeit).
- Der Prozesseditor soll intuitive und (nahezu) selbsterklärende Prozessbeschreibungen modellieren.
- Der Editor soll einen eher geringen Funktionsumfang aufweisen, um die Übersichtlichkeit und damit die einfache Bedienung zu fördern.
- Die Menge an unterschiedlichen Symbolen und Icons soll klein gehalten werden und diese solle klar unterscheidbar sein.
- Der Aufbau der Prozessbeschreibungen soll modular sein. Das heißt, dass ein Prozess nicht nur aus Teilschritten sondern auch aus Teilprozessen aufgebaut werden kann, wobei diese gesondert zu kennzeichnen sind.
- Auch Copy und Paste Funktionalitäten sollen im Sinne der Einfachheit und Usability unterstützt werden. Prozesse könnten durch das Kopieren schon vorhandener Teilschritte viel schneller erstellt werden.

Neben diesen Anforderungen an die Modellierung der Prozessbeschreibungen, sollen diese zudem ausführbar sein. Dazu müssen sie, unabhängig der gewählten Notationsform, in ein ausführbares Format gebracht werden. Das Format ist in der Regel XML-basiert, sodass unterschiedliche Prozess-Engines dieses lesen können (vgl. [HMS11] S.18).

Neben den Forderungen, die an den Prozesseditor gestellt werden, soll die gewählte Notationsform intuitiv und leicht erlernbar oder bereits im Unternehmen etabliert sein.

Wird sich bei der Umsetzung des Prozesseditors für die Verwendung eines bestehenden Systems entschieden, soll es günstige Lizenzbedingungen aufweisen. Hier wird der Open-Source-Bereich bezüglich der Anpassungsmöglichkeit bevorzugt betrachtet.

Grundsätzlich ist jedoch eine Abwägung der Entwicklungszeit vorzunehmen. Ist diese in einem angemessenen Rahmen, so soll für den Prozess-Editor eine bereits bestehende Lösung Verwendung finden, welche an die genannten Anforderungen angepasst wird. Ist dies nicht in einer adäquaten Zeitspanne möglich, soll der Prozesseditor durch eine Eigenentwicklung entstehen.

An den Formulareditor werden ebenfalls funktionale Anforderungen und Bedingungen bezüglich der Modellierung gestellt (vgl. [HMS11] S.33, 34, 5):

- Zur Darstellung der Formulare soll auf bekannte Steuerelemente grafischer Oberflächen zurückgegriffen werden, wie sie aus der alltäglichen Benutzung eines Windows-Systems oder dem Internet bekannt sind. Dazu gehören zum Beispiel Checkboxen und Textfelder.

- Zusätzlich sollen spezielle, selbstentworfene Elemente wie Tabellen mit teileditierbaren Feldern oder Dokument-Elemente erstellt werden können, welche es ermöglichen, durch gängige Interaktionsmetaphern externe Programme beispielsweise zur Betrachtung von PDF-Dateien zu starten.

- Die verwendeten Elemente sollen spezifische Eigenschaften zur Konfiguration erhalten. Dazu zählt zum Beispiel, ob ein Element lediglich angezeigt wird oder ob es interaktiv zu bearbeiten oder gar der Eingabe von Daten in das System dient.

- Formularinhalte sollen dynamisch erzeugt werden können, beispielsweise durch vorher gesetzte Werte (Prozessvariablen).

- Der Formulareditor soll ebenfalls eine WYSIWYG-Oberfläche besitzen. Formulare sollen während der Erstellung genauso aussehen, wie die Formulare, welche durch die Prozess-Engine geladen und dargestellt werden.

Die bisher genannten Anforderungen an den Formulareditor entstanden durch die Analyse der Bedingungen innerhalb der Partnerunternehmen. Anhand der Anforderungen wurde ein nicht funktionaler Prototyp modelliert. Dieser stellt eine mögliche Visualisierung bestimmter Prozesse und Arbeitsabläufe dar, welche durch beispielhafte Formulardarstellungen realisiert wird.

Bei dem Prototypen handelt es sich um eine Folge von HTML-Seiten, die mit Hilfe von simulierten Daten mögliche Funktionen der Laufzeitkomponenten darstellen. Das mögliche Aussehen der zu erstellenden Eingabemasken und die Interaktionen innerhalb dieser Formulare werden dabei gezeigt. Abbildungen 8 und 9 zeigen beispielhaft einige Eingabemasken.

Abbildung 8: Fertigungsauftragsdetails – Formularprototyp

Abbildung 9: Störungsgründe- Formularprototyp

Dieser Prototyp ermöglicht jedoch nur eine Visualisierung der Formulare und nicht das Integrieren realer Funktionen oder Daten. Zudem wurde er mit Hilfe von Web-Entwicklungstools erstellt, welche über die Kompetenzen und Programmierfähigkeiten eines QM-Verantwortlichen hinaus gehen. Der Prototyp dient lediglich dazu, den Projektpartnern und den KMU vor Ort das mögliche System näher zu bringen.

Allerdings kann auf Grundlage dieses Prototypen eine iterative Entwicklung des Formulareditors betrieben werden, da hier schon einmal dargestellt wird, was der Formulareditor letztendlich umsetzen sollte.

Schritt für Schritt soll dabei versucht werden, die hier beispielhaft dargestellten Funktionalitäten so umzusetzen, dass sie von einem Qualitätsmanager modelliert und mit realen Daten und Funktionen ausgestattet werden können.

Aufgrund der Nutzung des Prototypen zur iterativen Entwicklung ergeben sich für den Formulareditor einige weitere Anforderungen, welche auf der Grundlage dieser Darstellungen basieren:

- Die Formulare sollen in der Lage sein, mit den anderen Komponenten des Systems, wie zum Beispiel der Datenbasis, kommunizieren zu können. Das entspricht den eingangs erwähnten Grundsätzen der Informationspräsentation und –erfassung und ergibt sich aus der Darstellung der im Prototypen verwendeten Demodaten, welche durch reale Daten ersetzt werden sollen.

- Innerhalb des Formulareditors sollen Templates verwendet werden können, um den Corporate Design-Gedanken des Unternehmens unterstützen zu können. Der Prototyp beruht auf den Prozessen eines Unternehmens, dessen Logo innerhalb der Formulare dargestellt wird. Dies soll auch für andere KMU möglich sein.

- Auch die Formulare sollen modular aufgebaut sein (vgl. [HMS11] S.35):

    - Mehrere Elemente sollen zu definierten Blöcken zusammengeschlossen werden können (zum Beispiel mehrere Textfelder als Adresse).

    - Bestimmte, immer wiederkehrende Elemente sollen mit speziellen Eigenschaften vordefiniert werden können.

    - Copy und Paste soll ebenfalls unterstützt werden.

- Grundsätzlich sollte der Formulareditor, ebenso wie der Prozesseditor plattformunabhängig sein.

- Zwar ist der mobile Einsatz des Editors nicht geplant, aber die zur Laufzeit generierten Formulare sollten auch mobil funktionieren und die gleiche Optik wie auf einem Desktop-PC haben. Durch den Prototypen werden einige Interaktionsmetaphern dargestellt, bei denen geprüft werden muss, inwiefern diese beispielsweise durch Fingergesten eines Tablet-PC's umgesetzt werden können.

Dies sind die grundsätzlichen Anforderungen, die der Formulareditor erfüllen sollte. Bei der Auswahl eines Werkzeuges müssen diese Rahmenbedingungen erfüllt sein. Gleiches gilt für den Workfloweditor und damit auch für den gesamten IMAP-Editor.

In diesem Kapitel wurden diverse Anforderungen an das Montageportal, und die Editoren aufgelistet.

Bevor im Folgenden auf Grundlage dieser Bedingungen der IMAP-Editor entworfen werden soll, werden zunächst das Montageportal und der Editor in die theoretischen Grundlagen eingeordnet, um so die Rahmenbedingungen zu komplettieren.

3.3 Einordnung von MP und IMAP-Editor in die theoretischen Grundlagen

In den bisherigen Ausführungen wurde, neben Anforderungen an die verschiedenen Komponenten, eine Vielzahl theoretischer Grundlagen vermittelt, welche sich auf die Bereiche beziehen, in denen das Projekt IMAP an der Optimierung der Prozesse von KMU forscht und dafür Lösungen entwickelt. Auf dieser Grundlage können Vergleiche zwischen IMAP und den theoretischen Ausführungen gezogen werden, sodass die Rahmenbedingungen für die Entwicklung des IMAP-Editors komplettiert werden können.

Aufgrund der Ausführungen zu den theoretischen Grundlagen ist aus dem Prozessmanagement, für welches das Montageportal von IMAP ausgelegt war, ein Workflowmanagement geworden und aus der Prozessmodellierung eine Workflowmodellierung.

Betrachtet man dazu das Konzept des Integrierten Geschäftsprozess- und Workflowmanagement in [Gad08][10] und stellt diese den Betrachtungen des IMAP-Projektes gegenüber, so ist das integrierte Werkzeug zur Erstellung modularer Prozessdialoge nicht im Bereich der Prozessmodellierung anzusiedeln, sondern im Bereich des Workflowmanagements. Die Bereitstellung von Methoden und Werkzeuge zur Planung, Steuerung und Überwachung, die Modellierung und Ausführung sowie das Monitoring von automatisierten Prozessbeschreibungen sind ebenfalls Anforderungen, welche das Workflowmanagement umsetzt und welche gleichzeitig vom Portal gefordert werden. Damit kann das Montageportal in Zusammenhang mit dem IMAP-Editor als Workflowmanagementsystem klassifiziert werden.

Die Automatisierung der unternehmensinternen Prozessabläufe sollte im Zuge dieser Erkenntnis durch die Modellierung von Workflows vorgenommen werden. Die in [HzM03] aufgelisteten Vorteile von Workflows zur Prozessautomatisierung decken sich mit den Vorteilen, die das Montageportal laut der Konzeption in IMAP mit sich bringen soll. Dazu zählen die Transparenz der Prozesse oder die Verminderung der Durchlaufzeiten eines Produktes. Dies belegt noch einmal, dass das es sich bei dem Montageportal um ein WFMS handelt. Der IMAP-Editor als Teil des Montageportals übernimmt dabei die Rolle der Planung und Modellierung der Workflows, während die Laufzeit des Montageportals die Bereiche Steuerung und Monitoring abdeckt.

[10] Vgl. Abbildung 1

Wird zudem noch die Modellierung der Eingabemasken hinzugefügt, ergibt sich ein Gesamtbild für den IMAP-Editor. Die Bedeutung der Interaktionsformulare wurde an anderer Stelle bereits benannt, soll hier aber noch einmal aufgeführt werden.

Die Nutzung integrierter Formulare innerhalb der Workflows kann direkt zur Optimierung der Prozesse innerhalb der KMU beitragen. Durch sie wird es ermöglicht, begleitend zu einzelnen Aktivitäten, Informationen darzustellen, aber auch entgegenzunehmen.

Neben detaillierten Arbeitsanweisungen ermöglicht dies eine transparente Prozessgestaltung, effektives Monitoring und allgemein die Nutzung des Montageportals als Assistenzsystem für die Arbeiten.

Um die Optimierung adäquat umsetzen zu können, ist die Integration der Formulare in die Workflows von essentieller Bedeutung und muss vor allem bei der Modellierung, aber auch bei der Ausführung berücksichtigt werden. Neben der Möglichkeit, ein Formular direkt zu einem Prozessschritt erstellen zu können, muss dieses auch bei Erreichen der entsprechenden Aktivität aufgerufen werden.

Grundsätzlich werden bei der Modellierung sehr detaillierte, ausführbare Modelle benötigt, die aber dennoch grafisch anschaulich sind und leicht und intuitiv erstellt werden können. Im IMAP-Szenario soll die Modellierung von Mitarbeitern ohne spezifische Fachkenntnisse übernommen werden. Daher ist bei der Auswahl bestimmter Modellierungswerkzeuge Vorsicht geboten.

Diese sind häufig nicht ohne Modellierungskenntnisse einsetzbar (vgl. [Gad08] S.122). Gerade hier liegt die Herausforderung von IMAP im Kontext der Workflowmodellierung.

Neben den einfachen Modellierungswerkzeugen ist es auch essentiell, dass die gewählte Beschreibungssprache für die Workflows intuitiv darstellbar ist. Zudem muss sie trotzdem so formal sein, dass eine Ausführung und damit Automatisierung durch eine Prozess-Engine eines WFMS möglich ist. Außerdem muss, wie erwähnt, berücksichtigt werden, dass ein entsprechendes Format auch mit Formulareingaben verknüpft werden kann. Ist dies von Haus aus nicht gegeben, müssen Konzepte entwickelt werden, um diese Funktionalität bereitstellen zu können. Solche Konzepte werden in Kapitel 4.2 erläutert.

Nach der bereits erfolgten Darstellung einzelner Standards stellt sich die Frage, welche Variante und Notation in IMAP Verwendung finden sollte. Die EPK ist einfach, muss aber zur Automatisierung erst kompliziert in ein anderes Format gebracht werden.

BPMN dagegen ist einfach auszuführen und auch einfach zu benutzen. Da sie stark in der Zukunft Verwendung finden wird, gibt es eine Vielzahl von Modellierungstools. Großes Problem bei der BPMN ist jedoch, dass auch hier ein gewisses Verständnis für die Sprache vorhanden sein muss, damit gültige Modelle erstellt werden können.

Dies stellt im Hinblick auf das IMAP- Szenario ein mögliches Problem dar. Einfach Modelle sind schnell erstellt mit relativ wenig Einarbeitungszeit.

Allerdings bietet diese Notation eine sehr große Menge an Spezialisierungen der Grundelemente. Prozessdiagramme können dadurch sehr schnell sehr unübersichtlich werden. Der Sprachumfang ist für einen unerfahrenen Nutzer vielleicht zu groß.

Die erwähnten Flowcharts sind bei den Mitarbeitern, für die der IMAP-Editor entwickelt wird, bekannt, jedoch in der Regel nicht für eine Ausführung geeignet. Bei der Wahl einer Modellierungssprache muss aber nicht nur die reine Funktionalität berücksichtigt werden, sondern auch die Akzeptanz durch die Nutzer. Diese sind mitunter nicht bereit, eine Notation, die so „kompliziert“ wie die BPMN ist zu lernen, wenn es ihrer Ansicht nach einfachere Varianten gibt.

Unter den besonderen Voraussetzungen und Anforderungen von IMAP wäre es denkbar, ein eigenes, sehr einfaches Datenformat und eine dazugehörige Notation zu entwickeln. Sie hätte einen sehr geringen Umfang und würde damit die Prozesse der Projektpartner bestmöglich darstellen, ohne dass große Kenntnisse bestimmter Sprachen notwendig wären. Hierbei müsste allerdings neben dem Format eine eigene Prozess-Engine entwickelt werden. Zudem wäre das Format nicht konform zu anderen Standards. Die Problemstellung könnte jedoch durch einen Export in andere Sprachen gelöst werden.

Es bleiben also noch einige Fragen offen, bevor die eigentliche Applikation zur Erstellung der Arbeitsabläufe mit integrierten Formularen erstellt werden kann, wobei inzwischen eine Klassifizierung als Teil eines Workflowmanagementsystems möglich ist. Das Folgende Kapitel versucht, einige dieser Fragen zu beantworten, indem einige Beschreibungen zur Konzeption vorgenommen werden. Dabei werden verschiedenen Ansätze für ein mögliches System vorgestellt und auf Grundlage der Anforderungen eine Entscheidung für einen dieser Ansätze getroffen, welcher dann umgesetzt wird.

4 Konzept des IMAP-Editors

Nach der Auseinandersetzung mit theoretischen Grundlagen zum Workflowmanagement und der Workflowmodellierung hat sich ergeben, dass es sich bei dem IMAP-Montageportal und dem IMAP-Editor um Komponenten eines Workflowmanagementsystems handelt. Diese Erkenntnis und auch die Zielstellungen des Montageportals, welche sich mit gängigen Definitionen decken, bringen eine Reihe von allgemeinen Anforderungen mit sich. Dazu gehören unter anderem die Bereitstellung von Informationen, die Modellierung von Prozessbeschreibungen und die Ausführung von diesen mit Hilfe einer Laufzeitkomponente. Zusammen mit den detaillierten Ausführungen bezüglich der spezifischen Anforderungen an die Komponenten des IMAP-Editors, dem Prozesseditor und dem Formulareditor, wurde eine theoretische Basis geschaffen, auf deren Grundlage der IMAP-Editor entworfen werden kann. Das folgende Kapitel befasst sich mit der grundlegenden Konzeption dieser Anwendung. Dabei sollen mögliche Ansätze für den Editor gegeneinander abgewogen werden.

Neben grundlegenden Konzepten bezüglich der Integration von Workflows mit Formularbeschreibungen, welche essentiell für den IMAP-Editor ist, werden zwei Ansätze vorgestellt, nach denen die Anwendung entwickelt werden könnte. Sie basieren jeweils auf unterschiedlichen Technologien und verkörpern auch unterschiedliche Grundgedanken und Konzepte. Abschließend werden die Ansätze gegeneinander abgewogen, sodass eine Technologieentscheidung gefällt werden kann, auf deren Grundlage dann der IMAP-Editor entworfen wird.

4.1 Allgemeines zur Konzeption

Im Zuge der Analysen des Projektes IMAP und bei der Konzeption des Montageportals wurden verschiedenste Technologien und auch bereits existierende Softwarelösungen für die Umsetzung des Montageportals und der Modellierungswerkzeuge auf Grundlage der bereits erläuterten Anforderungen verglichen.

Im Bezug auf den Prozesseditor haben die Evaluationen innerhalb des Projektes gezeigt, dass auf kommerzieller Ebene kein Tool den geforderten Bedingungen entspricht (vgl. [HMS11] S.32). Entweder fehlen bestimmte essentielle Funktionen, oder der Funktionsumfang ist überdimensioniert im Vergleich zu dem, was nötig ist. Auf Open Source-Ebene gibt es einige Tools, die es ermöglichen würden, sie entsprechend anzupassen. Auf die Verwendung einer bereits vorhandenen Softwarelösung wird im Verlauf noch eingegangen.

Auch bei Formulareditoren wurde nach einem passenden Werkzeug gesucht (vgl. [HMS11] S.45). Innerhalb der Partnerunternehmen sind viele Microsoft Produkte im Einsatz.

Hier könnte auch für die Formulare aus der reichhaltigen Produktpalette gewählt werden (InfoPath), allerdings wären diese Systeme proprietär und nicht anzupassen. Im Open Source-Bereich gibt es einige Lösungen, doch ist auch hier eine Auswahl schwierig.

Da eine Vernetzung mit den Workflows entscheidend ist, sollte eine Entscheidung in Abhängigkeit von der Wahl des Workfloweditors getroffen werden (vgl. ebenda).

Auch hier gilt die Maßgabe, den Aufwand der Anpassung gegenüber der Eigenentwicklung abzuschätzen, wobei vermutlich letzteres zum Tragen kommen wird, da die Studien innerhalb des Projektes auch hier keine bevorzugte Lösung ergaben.

Somit kommt der Wahl des Workfloweditors eine entscheidende Rolle zu. Den vorherigen Ausführungen zu möglichen Standards folgend, ist damit verbunden jedoch eine weitere wichtige Entscheidung zu treffen. Sie umfasst das Format der Workflowbeschreibungen.

Bei der Wahl der Notationsform für die Prozessbeschreibungen ist es wichtig, dass berücksichtig wird, dass sich der IMAP-Editor in den KMU etablieren muss. Dort herrschen feste Abläufe vor, die QM-Abteilung hat ihre eigenen Vorstellungen, was die Modellierung von Prozessen angeht. Die Akzeptanz und auch die zeitlichen Möglichkeiten, eine neue Notationsform einzuführen, sind in der Regel in den KMU nicht vorhanden. Dementsprechend ist es entscheidend, wie leicht erlernbar und wie verständlich die gewählte Notation letztendlich ist (vgl. [HMS11] S.18).

Um die Überführung in eine ausführbare Variante sollte sich das Qualitätsmanagement dabei keine Gedanken machen, hier zählt letztendlich die Bedienbarkeit. Dabei sollte ebenfalls überprüft werden, ob nicht auf schon bekannte Notationen zurückgegriffen werden kann. Bei den Projektpartnern wäre dies eine Flowchart-Darstellung (Programmablaufplan), EPK (vgl. [HMS11] S.32). Die Flowcharts nehmen jedoch den größten Anteil ein.

Bei der Wahl einer Beschreibungsform könnte es aber zu Problemstellungen kommen, was das Überführen des Montageportals in projektunabhängige KMU angeht. Diese nutzen nicht zwangsläufig die Möglichkeiten, die in den Partnerunternehmen etabliert sind. Hier stellt sich die Frage, was am Ende sinnvoller ist, ein eigenes Format entwickeln, welches alle Anforderungen umsetzt, einfach und intuitiv ist, oder auf einen etablierten Standard zurückgreifen, der dann aber vielleicht nicht bei allen Firmen auf Akzeptanz stößt. Ein eigenes Format wäre natürlich proprietär und würde keinem etablierten Standard entsprechen.

Dafür hätte es den Vorteil, dass man unabhängig von den Softwareprodukten anderer Hersteller und möglicher neuer Releases wäre, bei denen mögliche Schnittstellen verändert werden könnten. Die Kompetenzen würden in der eigenen Hand liegen. Trotzdem könnte es schwierig sein, die Unternehmen von einem nicht standardisierten Format zu überzeugen. Es müsste dabei sehr einfach sein, sodass der „Konkurrenz“ gegenüber viele Vorteile aufzuzeigen wären.

An dieser Stelle soll erneut betont werde, dass neben den schon genannten Fragestellungen bezüglich des Ausgabeformates der Prozessbeschreibungen/Workflows ebenfalls berücksichtigt werden muss, inwiefern Prozesse und Formulare in Einklang gebracht werden können. Doch gerade diese Integration von Formularen und Workflows soll das Element sein, was dem IMAP-Editor einen Vorteil gegenüber anderen Workfloweditoren verschafft.

Durch die Formulare werden durch die Informationsbereitstellung und –erfassung die Grundlagen für eine verbesserte Arbeit innerhalb der KMU geschaffen, da so beispielsweise Materialflüsse besser überwacht werden können und durch die digitale Darstellung aller relevanten Informationen wie Arbeitsanweisungen und unterstützende Dokumente auf einen Blick sofort verfügbar sind. Gleiches würde in analoger Form, wie bisher in den meisten Fällen stattfindend, einen großen Zeitaufwand bei der Suche bedeuten. Es kann so festgestellt werden, wo potentielle Schwachstellen der Arbeit liegen, um sie konkret verbessern zu können. Die Formulareingaben sind dementsprechend essentiell.

Hierbei ist jedoch zu bedenken, dass nicht jede Notationsform standardmäßig die Möglichkeit mit sich bringt, eine solche Verbindung zu modellieren. Für dieses Fall müsste ein Ausgleich geschaffen werden. Da diese Fragestellung von entscheidender Bedeutung ist, werden im nächsten Kapitel mögliche Konzepte zur Integration von Workflow und Nutzereingabe vorgestellt.

4.2 Konzepte zur Integration von Workflows und Formularen

Bei den Anforderungen an den IMAP-Editor wurden Prozesseditor und Formulareditor getrennt voneinander betrachtet. Das Thema dieser Arbeit lautet allerdings „Integriertes Authoring von modularen Prozessdialogen“, zielt also grundsätzlich darauf ab, Prozesse mit Formularen in Verbindung zu bringen.

Auf dem Markt gibt es einige Tools[11], die eine direkte Verknüpfung von Workflows mit Formularen ermöglichen. Sie sich jedoch meist dadurch gekennzeichnet, dass entweder die Formulare umständlich programmiert werden müssen, was gegen das IMAP-Konzept spricht, oder für die Zielgruppe zu kompliziert zu bedienen sind, da hier nicht vorhandene Fachkenntnisse benötigt werden. Da durch genannte Anforderungen relativ strikte Vorgaben herrschen, was der Formulareditor umsetzen sollte, stoßen die Möglichkeiten aber oft an Grenzen, die auch durch eine Erweiterung sehr schwer zu umgehen sind.

Während der Suche nach einer geeigneten Lösung innerhalb des Projektes scheiterten diverse Softwareprodukte, weil eine individuelle Anpassung durch die Komplexität des vorhandenen Codes den Zeitrahmen von IMAP gesprengt hätte. Hier ist Bonita Open Solution (BOS) zu nennen. BOS bietet eine sehr gute Integration, bei der mehrere Formulare zu einem Prozessschritt zugeordnet werden können (vgl. [HMS11] S.63).

[11] Beispielsweise Bonita oder ORYX

Jedoch bietet beispielsweise der Formulareditor nicht alle geforderten Eigenschaften und müsste dementsprechend angepasst werden. BOS bringt aber das Problem mit sich, dass es keine Entwicklerdokumentation oder kommentierten Quellcode gibt und die Projektstruktur zu unübersichtlich ist, als das einzelne Abläufe nachvollzogen werden könnten (vgl. [HMS11] S. 72).

Für das angestrebte Modellierungswerkzeug müssen also Konzepte entwickelt werden, wie Prozess und Formular möglichst einfach und intuitiv verbunden werden können, wobei das Ganze mit dem oder den Editoren simpel zu modellieren ist, da hierfür bisher keine geeigneten Lösungen gefunden werden konnten. Solche Konzepte sollen im Folgenden erläutert und bewertet werden.

Bei der Erarbeitung von Konzepten zur Verknüpfung von Formularen und Prozessbeschreibungen sollte grundsätzlich berücksichtig werden, ob die Modellierung mit einem Werkzeug stattfindet, dass beide Funktionalitäten vereint oder ob zwei Editoren Verwendung finden.

Bei der Modellierung mit zwei Editoren sind mehrere Ansätze zur Integration denkbar:

- **Ansatz A (verknüpfte Editoren):**
    - Separate Modellierung des Workflows
    - kontextabhängige Modellierung der Formulare an entsprechender Stelle
    - Modellierung der Formulare zur Laufzeit des Workfloweditors
    - Starten des Formulareditors aus dem Workfloweditor heraus
  - **Ansatz A1 (verknüpfte Editoren, verlinkte Formulare):**
    - externes Ablegen der Formulare
    - Verlinkung des entsprechenden Formulars innerhalb des Workflows
  - **Ansatz A2 (verknüpfte Editoren, ein Format):**
    - Formulare werden nicht verlinkt, sondern bilden Teil des Workflows in einem Format
  - **Ansatz A3 (verknüpfte Editoren, Verbindung durch Dateien):**
    - Hinterlegen der Verbindungsinformationen in Konfigurationsdateien

- **Ansatz B (unabhängige Modellierung):**
    - Modellierung des Workflows mit entsprechendem Editor
    - Modellierung des Formulars mit entsprechendem Editor
    - Nachträgliches Zusammenführen beider Formate
    - **Ansatz B1 (unabhängige Modellierung, verlinkte Formulare):**
        - Verlinkung des Formulars in dem Workflow
    - **Ansatz B2 (unabhängige Modellierung, ein Format):**
        - Verbindung beider Outputs zu einer neuen Datei/neuem Format
    - **Ansatz B3 (unabhängige Modellierung, Verbindung durch Dateien):**
        - Verbindung beider Formate erst während der Prozessausführung anhand von Konfigurationsdateien

- **Ansatz C (Modellierung durch Analyse):**
    - Modellierung des Workflows
    - Analyse des Workflows
    - Modellierung der Formulare kontextabhängig zu Prozessschritten
    - Modellierung der Formulare NICHT zur Laufzeit des Workfloweditors
    - Start des Formulareditors unabhängig des Workfloweditors
    - **Ansatz C1 (Modellierung durch Analyse, verlinkte Formulare):**
        - Verlinkung des entstandenen Formulars im Workflow
    - **Ansatz C2 (Modellierung durch Analyse, ein Format):**
        - Verknüpfung von beiden Formate zu einem Format
    - **Ansatz C3 (Modellierung durch Analyse, Verbindung durch Dateien):**
        - Verbindung der Formate durch Zuordnung in externen Konfigurationsdateien

Dies sind unterschiedliche Integrationsansätze, die bei der Verwendung von zwei unabhängigen Editoren denkbar wären. In Kurzform wären dies also zum einen die Modellierung des Formulars zur Laufzeit aus dem Workfloweditor heraus (A), die nachträgliche Verbindung von Workflows und Formularen, die unabhängig modelliert wurden (B) und die Modellierung der Formulare auf Grundlage des Workflows, jedoch erst nach deren Modellierung von diesem (C). Für die Integration selbst werden 3 Fälle unterschieden: die Verlinkung der Formulare in den Workflows (1), die Verbindung beider zu einem einzigen Format (2) und die Verbindung beider erst zum Zeitpunkt der Prozessausführung auf Grundlage von in Konfigurationsdateien gespeicherten Verbindungsinformationen (3).

Neben diesen Ansätzen, die bei der Verwendung von zwei unabhängigen Modellierungswerkzeugen denkbar wären, besteht noch die Möglichkeit, die Modellierung mit nur einem Werkzeug zu konzipieren. Dabei würde diese Applikation sowohl die Möglichkeit der Modellierung von Workflows als auch Formularen als Funktion bereitstellen.

Dies würde Ansatz A ähneln, nur dass hierzu kein externes Tool gestartet werden müsste, sondern einfach die entsprechende Funktion aktiviert werden könnte:

- **Ansatz D (ein Editor)**
    - Modellierung des Workflows
    - kontextabhängige Modellierung der Formulare an entsprechender Stelle
    - Formulareditor als Teil des Workflows
  - **Ansatz D1 (ein Editor, verlinkte Formulare):**
    - Verlinkung innerhalb des Workflows
  - **Ansatz D2 (ein Editor, ein Format):**
    - Eigenes Datenformat, Formular Teil des Workflows

Da hier eine direkte Verbindung zwischen Formulareditor und Workfloweditor besteht, sollte ein Ansatz D3 (dem obigen Schema folgend) als eher unwahrscheinlich betrachtet werden, da die Verbindung anhand von externer Dateien eher dazu gedacht sein sollte, zwei völlig unvereinbare Formate zu verknüpfen, was bei einer bestehenden Integration beider Editoren als wenig wahrscheinlich erachtet wird.

Abbildung 10 stellt die unterschiedlichen Konzepte noch einmal übersichtlich dar. Link steht dabei für Integrationsform 1 (verlinkte Formulare), Part für 2 (ein Format) und Conf für 3 (Verbindung durch Dateien).

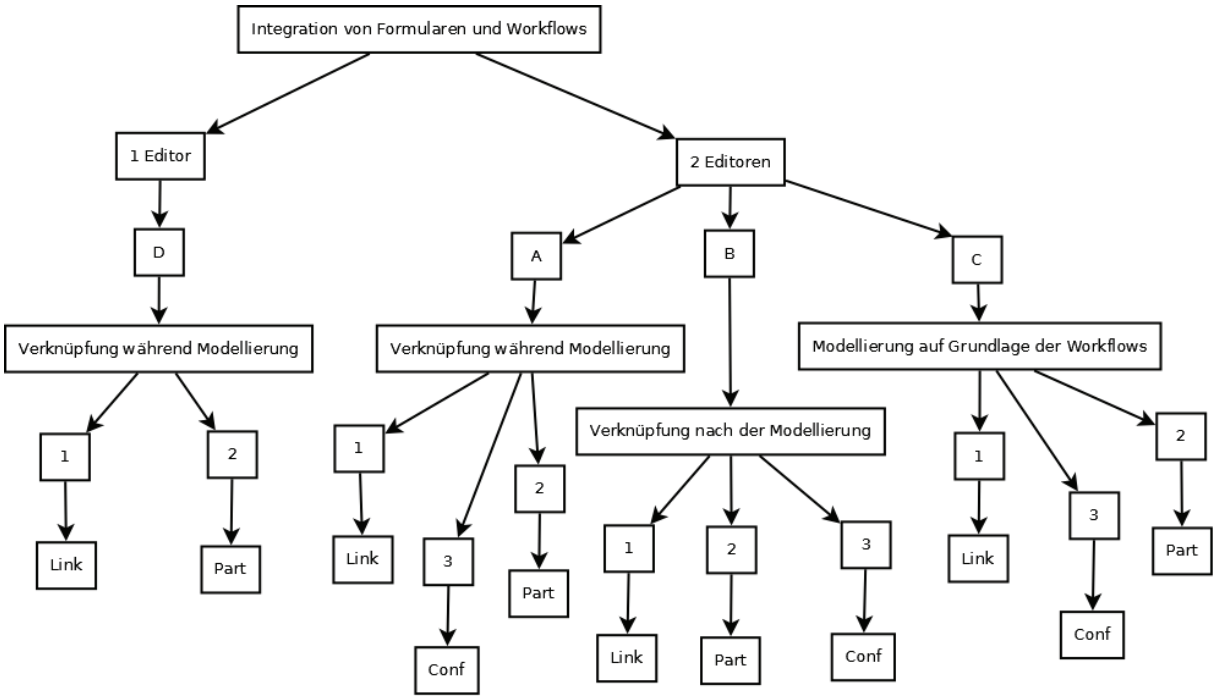


Abbildung 10: Übersicht der Integrationskonzepte

Nach dieser Auflistung möglicher Integrationskonzepte sollen diese gegeneinander abgewogen und bewertet werden. Zur Bewertung werden dabei verschiedene Kategorien herangezogen. Eine ausführliche Übersicht bezüglich der Ansätze und deren Bewertung befindet sich im Anhang. Im Folgenden sollen kurz die KO-Kriterien und möglichen Auswahlen für den IMAP-Editor dargestellt werden.

Bei dieser Beurteilung werden zunächst die Modellierungsansätze A – D betrachtet. Diese werden nach den Kategorien Implementierungsaufwand, Nutzungsaufwand für den User und möglichen, bereits vorhandenen Systemen verglichen. Dabei besteht ein Zusammenhang zwischen schon existierenden Lösungen und dem eigenen Aufwand bei der Implementierung.

Ansatz A (verknüpfte Editoren) zeichnet sich dadurch aus, dass hier die Nutzung für den Modellierer sehr übersichtlich ist. Zudem sollte hierbei, je nach Komplexität des verwendeten Systems, der Implementierungsaufwand im Rahmen bleiben.

Dies hängt jedoch größtenteils von dem verwendeten Workfloweditor ab und wie hierbei der Formulareditor eingebunden werden könnte. Grundsätzlich könnten hierbei viele Workflow- und auch Formulareditoren verwendet werden, die allerdings an die Bedürfnisse und Anforderungen von IMAP angepasst werden müssten. Insgesamt ist dieser Ansatz als „Gut“ zu betrachten.

Ansatz B (unabhängige Modellierung) ermöglicht zwar die Verwendung jedes Workflow- und Formulareditors, der auf IMAP angepasst wurde, es ist jedoch nötig, für die Verknüpfung ein Extra-Tool zu entwickeln, dass das Mapping übernimmt.

Da es hierbei zu Problemen mit Prozessvariablen kommen könnte, müssen dafür ausgereifte Konzepte entwickelt werden. Auch für den Nutzer stellt ein weiteres Tool, bei dem umständlich die Formulare mit den Workflows verknüpft werden, einen Mehraufwand da. Dieser Ansatz ist dadurch eher als „nicht gut“ einzuordnen.

Bei Ansatz C (Modellierung durch Analyse) handelt es sich im Grunde um die Vorstufe zu A (verknüpfte Editoren). C (Modellierung durch Analyse) stellt eine gute Alternative dar, falls es zu kompliziert sein sollte, den Formulareditor in den Workfloweditor zu integrieren. Hier wird es ermöglicht, die Formulare kontextabhängig und auch unter Berücksichtigung möglicher Variablen zu entwickeln. Allerdings müsste bei diesem Ansatz das Analysetool zusätzlich entwickelt werden. Hierfür gibt es aber diverse Möglichkeiten. Basiert das Workflowformat beispielsweise auf XML könnte für die Analyse XSLT[12] verwendet werden. Für den Nutzer entsteht hierbei nur ein geringer Mehraufwand, da das Analysetool so entwickelt werden kann, dass es kaum Interaktionen erfordert. Der Ansatz ist aufwändig, ist jedoch besser zu bewerten als B (unabhängige Modellierung).

Ansatz D (ein Editor) stellt sich in sofern als gut dar, dass mit einem kombinierten Werkzeug ein sehr kompaktes System vorhanden wäre. BOS stellt beispielsweise ein solches System dar. Da hier jedoch nach einer Analyse eine Anpassung als zu aufwändig eingestuft wurde, würde ein kombinierter Editor vermutlich auf eine Eigenentwicklung hinauslaufen, da keine anderen, ähnlichen Systeme bekannt sind.

Dies bringt zwar eine Menge Aufwand in der Entwicklung mit sich, jedoch besteht bei diesem System ein geringer Nutzeraufwand, da hier alle Funktionen kompakt verfügbar sind. Bei vergleichbar großen Aufwänden bei der Anpassung bestehender Systeme sollte Ansatz D (ein Editor) als Alternative betrachtet werden.

Bezüglich der eigentlichen Integration wurden drei verschiedene Ansätze unterschieden. Diese sollen ebenfalls nach dem Implementierungsaufwand und nach vorhandenen Systemen geprüft werden.

Ansatz 1, also das Verlinken, stellt eine gute Variante dar, sofern passende Systeme dafür gefunden werden. Ist das nicht der Fall, sind viele Veränderungen notwendig, da die Verlinkung keinem Standard entspricht. Es würde ein entsprechendes Format, aber auch eine Engine betreffen.

Da mit Activiti[13] jedoch ein System vorhanden ist, dass eine Engine bereitstellt, die eine Verlinkung erlaubt, sollte es für diesen Ansatz verwendet werden.

[12] Vgl. http://www.w3.org/TR/xslt20/ [15.08.2011]
[13] Vgl. http://www.activiti.org [15.08.2011]

Ansatz 2 (ein Format) dagegen läuft auf eine Eigenentwicklung hinaus. Gerade da es keine Formate gibt, die das direkte Einbinden der Formulare als Teil der Workflows unterstützen, wurde diese Arbeit relevant. Grundsätzlich müsste hierbei ein Datenformat erschaffen werden, dass die Formulare auf genau diese Art abbildet, ebenso wie eine eigenständige Engine. Hier besteht also ein sehr hoher Aufwand. Dennoch ist die eigentliche Integration hier am besten gelöst, weshalb dieser Ansatz durchaus in Betracht gezogen werden sollte.

Der 3. Ansatz, also die Verknüpfung durch Ablegen der nötigen Informationen in Konfigurationsdateien, bietet eine eher schlechte Integration. Hierbei wird das Verbinden relativ kompliziert gelöst, Engines gibt es für diesen Ansatz nicht. Bezüglich der Vereinigung ist hier bereits zum Modellierungszeitpunkt eine Testumgebung notwendig. Dadurch lassen sich Aufwände bei dieser Integration nicht abschätzen. Da es sich eher um eine „Notlösung“ handelt, sollte dieser Ansatz vermieden werden.

Nach dieser Beurteilung der unterschiedlichen Ansätze für die Modellierung und Verknüpfung sollen im Folgenden die Kombinationen von diesen im Ganzen betrachtet werden. Dabei fallen die Urteile der einzelnen Teile des Ansatzes ebenso ins Gewicht wie die grundsätzliche Umsetzbarkeit durch bestehende Systeme.

Aufgestellt wurden 11 unterschiedliche Ansätze. Durch die schon geschilderten Schwierigkeiten sollte bei einer Wahl des am besten für die Nutzung im IMAP-Editor geeigneten Integrationskonzepts auf die Verwendung der Ansätze B (unabhängige Modellierung) und 3 (Verbindung durch Dateien) verzichtet werden, da hierbei bessere Varianten zur Verfügung stehen.

Die Ansätze A2 (verknüpfte Editoren, ein Format) und C2 (Modellierung durch Analyse, ein Format), also die Verwendung von unterschiedlichen Workflow- und Formulareditoren bei einem kombinierten Format, dass vermutlich eine Eigenentwicklung darstellen würde, scheiden ebenfalls aus, da es sich als sehr schwierig darstellen würde, ein eigenes Format in einen vorgegebenen Editor einzubauen. Wären die Editoren selbst ebenfalls Eigenentwicklungen, so wäre es im Grunde sinnvoller, hier gleich einen kombinierten Editor zu erstellen.

Bei D1 (Ein Editor, verlinkte Formulare), also der Verwendung eines kombinierten Editors mit einem im Workflow verlinkten Formular, besteht das Problem, dass es bei vorhandenen Systemen schwierig sein könnte, die komplette Funktionalität des Formulareditors in den Workfloweditor einzubauen. BOS arbeitet zwar nach diesem Prinzip (wobei nicht genau nachvollzogen werden kann, wie genau die Formulare integriert sind), ist aber zu aufwändig in der Anpassung. Wird ein kombinierter Editor eigenentwickelt, so kann gleich auf die optimalere Integration (D2) zurückgegriffen werden.

Es bleiben also die Ansätze A1 (Ein Editor, verlinkte Formulare), C1 (Modellierung durch Analyse, verlinkte Formulare) und D2 (ein Editor, ein Format). Bezüglich der Ansätze mit Kennnummer 1 wurde bereits erwähnt, dass mit dem System Activiti die Möglichkeit gegeben ist, Formulare in den Workflow zu verlinken. Genaueres dazu wird im folgenden Kapitel dargestellt. Activiti bietet zudem einen eigenen Workfloweditor.

Den Ansätzen folgend, würde dementsprechend Ansatz A1 (Ein Editor, verlinkte Formulare) die für den IMAP-Editor optimalste Variante darstellen. Hierbei hängt alles davon ab, inwiefern der vorhandene Workfloweditor angepasst werden könnte.

Wäre das Einbinden des Formulareditors durch den vorhandenen Code zu aufwändig, wäre mit Variante C1 (Modellierung durch Analyse, verlinkte Formulare) als Vorstufe eine ebenfalls gute Integration möglich, die jederzeit mit etwas Aufwand zu A1 (Ein Editor, verlinkte Formulare) ausgebaut werden könnte. Hauptvorteil liegt, wie erwähnt, in der Unterstützung durch die vorhandene Engine.

Allerdings müsste hierbei ein entsprechender Formulareditor gefunden oder eigenentwickelt werden, der dann für diesen Ansatz verwendet werden kann.

Sollte jedoch der Workfloweditor zu kompliziert für eine Erweiterung sein, wäre es denkbar, auf den Ansatz D2 (ein Editor, ein Format) zurückzugreifen. Dies wäre die komplette Eigenentwicklung eines kombinierten Editors mit einem eigenen Format, das die Formulare als Teil des Workflows darstellt. Hinzu käme eine entsprechende Engine zur Ausführung. Dies würde zwar einen hohen Implementierungsaufwand bedeuten, jedoch wäre die Integration optimal und statt kompliziert die Editoren an bestimmte Bedürfnisse anpassen zu müssen bestünde hierbei von vorneherein die Möglichkeit, den Editor entsprechend aller Vorgaben zu entwickeln. Jedoch müssten die Aufwände gegenüber den anderen beiden Ansätzen verglichen werden.

Diesen Betrachtungen folgend kommen für den IMAP-Editor also die Integrationsansätze eines eigenen Formates, das Formulare und Workflows als eins darstellt, und die Verlinkung der Formulare in die Workflows bei Verwendung zweier Editoren in Frage.

Ersteres würde einer Eigenentwicklung gleich kommen, während zweites die Nutzung eines vorhandenen Systems bedeuten würde. Im Folgenden sollen diese beiden Entwicklungsansätze im Detail erläutert und miteinander verglichen werden, sodass im Anschluss eine Entscheidung für einen dieser Ansätze gefällt werden kann.

4.3 Ansätze für die Umsetzung des IMAP-Editors

Wie erwähnt, sollen im Folgenden zwei Ansätze für die Umsetzung des IMAP-Editors erläutert werden. Bei dem einen handelt es sich um die Verwendung einer Open-Source-basierten Software, die benutzt und angepasst werden soll und zur Integration die Formulare in den Workflows verlinkt.

Der zweite Ansatz basiert auf der kompletten Eigenentwicklung von Prozess- und Formulareditor auf Grundlage der Anforderungen, sodass diese dann die gewünschten Funktionalitäten in einem Rahmen abdecken, die bei der Anpassung einer bestehenden Software schwer zu erreichen wäre. Integriert wird hierbei durch ein eigenes Datenformat, das die Formulare als Teil des Workflows beschreibt.

Bei der folgenden, kurzen Beschreibung beider Ansätze werden diese nach den Kriterien der verwendeten Technologie, der Modellierung, verwendeter Daten und Datenformate, der Integration und möglicher Probleme betrachtet.

4.3.1 Nutzung eines Open-Source-Produktes

Zunächst die Betrachtung der Open-Source-Variante. Hierbei soll Activiti genutzt werden. Dabei handelt es sich um eine leichtgewichtige Workflow- und Business Process Management-Plattform, die eine Modellierung und auch Ausführung von Geschäftsprozessen auf Grundlage des BPMN 2.0 Standards ermöglicht, wobei eine eigene Prozess-Engine verwendet wird (vgl. [ACT 01]).

Diese Engine bildet die Hauptkomponente des Projektes. Sie ermöglicht die Ausführung der modellierten Prozesse anhand der Ausführungssemantik von BPMN 2.0. Die Engine basiert auf Java und bringt zudem eine eigene Java-API mit sich (vgl. [ACT 02]).

Der große Vorteil dieser Softwarelösung ist es, dass Formulare an die Workflows angebunden und diese durch die Engine ausführbar gemacht werden können. Die Formulare können an entsprechender Stelle der Prozessausführung kontextabhängig zu einem Task aufgerufen und durch das Übergeben von Daten an die Engine ausgefüllt und komplettiert werden.

Für die Modellierung stellt Activiti einen Workfloweditor zum Erstellen von BPMN 2.0 Prozessmodellen zur Verfügung. Was Activiti jedoch fehlt, ist ein Formulareditor. Es können zwar Formulare an bestimmte Task angeheftet werden, allerdings gibt es keine mitgelieferte Möglichkeit, diese intuitiv und graphisch zu modellieren.

An anderer Stelle dieser Arbeit wurde bereits erwähnt, dass im Bereich der Formulareditoren keine passenden Lösungen gefunden werden konnten (vgl. Kapitel 4.1). Deshalb würde bei diesem Ansatz eine Eigenentwicklung eines Formulareditors mit der Verbindung des durch Activiti gegebenen Workfloweditors, dem Activiti Modeller, zum Tragen kommen.

Der Activiti Modeller ist eine an die Bedürfnisse von Activiti angepasste Version des Signavio Prozess-Editors (vgl. [ACT 02]). Allerdings wird die Zuordnung zwischen Task und Formular nicht durch den Workfloweditor bereitgestellt, obwohl die Activiti-Engine diese Zuordnung unterstützt. Sie müsste also auf andere Art erfolgen, was noch erläutert werden soll. Der Activiti Modeller ist eine webbasierte Anwendung und wird in einem Browser ausgeführt. Abbildung 11 zeigt die Oberfläche dieses Editors.

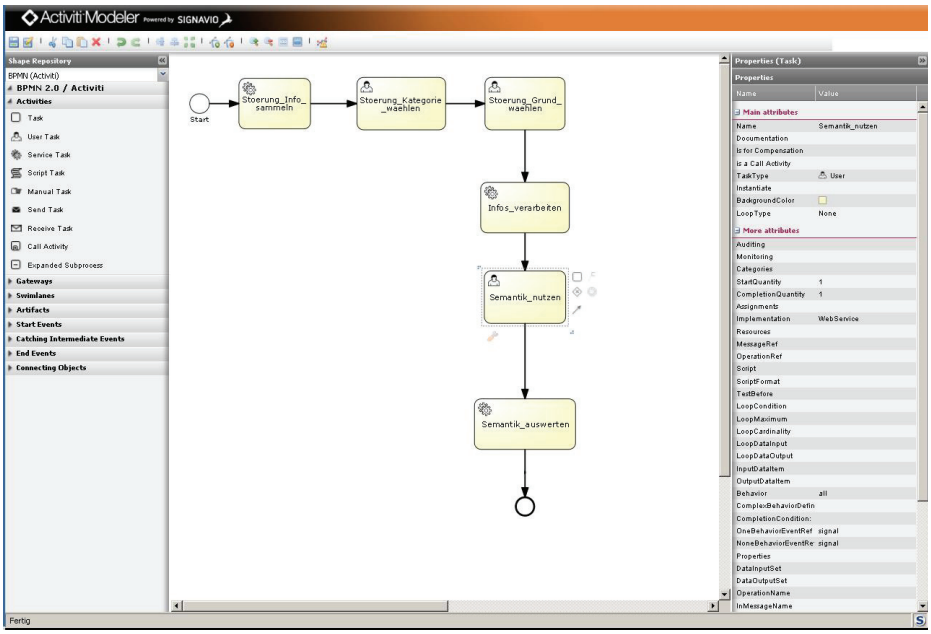


Abbildung 11: Oberfläche des Activiti Modeller

Der Formulareditor soll bei diesem Ansatz als Desktopanwendung konzipiert werden, da für den Betrieb der Engine und für die, durch den Integrationsansatz notwendigen Anpassungen ebenfalls eine Desktopanwendung sinnvoll erscheint, die mit dem Editor kombiniert werden könnte. Zudem wäre im Vergleich eine Webentwicklung viel aufwändiger und zeitintensiver. Bei der Entwicklung soll für den Formulareditor das Oberflächenframework Swing von Java zum Einsatz kommen.

Bezüglich der Dateiformate, die durch beide Editoren erzeugt werden sollen, ist festzuhalten, dass die Modellierung und Ausführung bei Activiti auf dem BPMN 2.0 Standard basiert. Die Formularanbindung, die eigentlich nicht durch die BPMN vorgesehen ist, jedoch durch Activiti ermöglicht wird, kann durch die Einbindung eines speziellen Activiti-Namensraums in die XML-Darstellung des BPMN-Modells möglich gemacht werden.

Da der im Zuge der Arbeiten am Projekt entstandene, nicht funktionale Prototyp mögliche Interaktionen und Interaktionsfolgen durch eine HTML-Repräsentation abdeckt, wäre eine Ausgabe der Formulare in XHTML denkbar. Zur Kommunikation dieser webbasierten Formulare mit der Java-API der Engine wäre der Einsatz von Jersey möglich.

Für die Integration muss berücksichtigt werden, dass die Files, die durch den Modeller erzeugt und in die Engine überführt werden sollen, zusätzlich den erwähnten eigenen Namespace benötigen. Dieser wird durch den Modeller, der standardkonformes BPMN produziert, nicht bereitgestellt, sodass ein Eingriff in das erzeugte File nötig ist.

Dadurch wird es jedoch möglich, für die Tasks bestimmte Keys zu hinterlegen. Nutzt man dies aus, so werden Formulare mit einem solchen Key im Workflow integriert. Zur Laufzeit kann der aktuelle Formkey ausgelesen werden, hinter dem sich dann zum Beispiel der Pfad einer durch den Formulareditor erstellten XHTML-Datei befinden kann.

Dadurch, dass das BPMN-File des Modellers bearbeitet werden muss, bevor eine Ausführung möglich wird, wäre hier der Integrationsansatz C1 denkbar, da Analyse und Veränderung kombiniert werden könnten. Für die Nutzung würde es bedeuten, dass die Formulare auf Grundlage der Aktivitäten des BPMN-Modells entwickelt werden. Nach der Erstellung des Modells würde der Workflow in einem Analyse-Tool geöffnet werden und zu den entsprechenden Aktivitäten des Workflows würde der Formulareditor gestartet. Das Formular wird erstellt und es wird im Workflow durch das Analyse-Tool ein entsprechender Verweis hinterlegt, da das durch den Modeller nicht vorgenommen wird. Neben dem Einfügen der Verlinkung könnte durch dieses Tool ebenfalls der benötigte Namespace eingefügt werden.

Dieser Ansatz klingt erst einmal vielversprechend, bringt jedoch auch einige Probleme mit sich. Grundsätzlich ist die Verlinkung der Formulare sehr vorteilhaft, jedoch ist es gerade der Modeller, der bei diesem Ansatz ein Problem darstellt. Er sollte beispielsweise erst einmal so belassen werden, wie er ist, da eine Anpassung zeitnah vom Aufwand her nicht möglich wäre. Dies liegt an den umfangreichen und nicht gut dokumentierten Code-Strukturen und der unübersichtlichen Entwicklung in JavaScript.

Trotz der speziellen Anpassung des Signavio-Modellers auf Activiti fehlt im Modeller die Möglichkeit, die Zuordnung zwischen Prozess und Formular zu modellieren. Das ist zwar BPMN-konform, aber inkonsistent, wird diese Funktion doch durch die Engine bereitgestellt. Obwohl es sinnvoll erscheinen würde, ist ein durch den Modeller erstelltes BPMN-File nicht direkt in der Engine ausführbar. Es ist also grundsätzlich ein Eingriff nötig, damit die erstellten Files durch die Activiti-Engine verarbeitet werden können. Eine entsprechende Funktion wäre nur unter sehr hohem Aufwand in den Modeller zu integrieren.

Zudem ist der Funktionsumfang des Workfloweditors sehr hoch. Es bestehen viele Möglichkeiten, die einen nicht fachkundigen Nutzer überfordern könnten. Zudem muss für eine Notation wie BPMN, die zwar standardisiert, jedoch nicht zwangsläufig in den KMU bekannt ist, erst einmal eine gewisse Akzeptanz geschaffen werden. Der viel zu umfangreiche Modeller ist dabei nicht hilfreich.

Hinzu kommen kleinere Probleme, wie die konsequente Umsetzung des WYSIWYG-Grundsatzes, da hier auf Grund vielfältiger Technologien zwischen den einzelnen Komponenten die Elemente mehrfach gerendert werden müssten.

Entscheidender Punkt bei diesem Ansatz sind jedoch die hohen Aufwände verbunden mit dem Activiti-Modeller. Wie schon mehrfach dargestellt, sollte in einem solchen Fall geprüft werden, ob nicht eine Eigenentwicklung mit weniger Aufwand, dafür aber mit einer besseren Umsetzung aller Anforderung machbar wäre. Entsprechend beschäftigt sich der im Folgenden dargestellte Ansatz mit dieser Eigenentwicklung.

4.3.2 Eigenentwicklung des gesamten IMAP-Editors

Dieses Unterkapitel beschäftigt sich mit der Eigenentwicklung des IMAP-Editors, einer eigenen Engine und eines eigenen Datenformates. Durch eine Eigenentwicklung ist grundsätzlich eine Umsetzung genau der Funktionen möglich, die wirklich gebraucht werden. Zudem würde eine einheitliche technologische Basis beider Funktionalitäten des IMAP-Editors geschaffen werden. Es sollte hierbei darauf geachtet werden, dass die gesamte Anwendung sehr simpel gehalten, aber dennoch sehr flexibel gestaltet wird, sodass Erweiterungen einfach vorgenommen werden können.

Es soll dabei generell ein webbasiertes System entstehen. Da es jedoch einen erheblichen Mehraufwand mit sich bringt, alles von Grund auf neu zu erstellen, muss dafür eine Technologie gewählt werden, mit der dies schnell und einfach möglich ist.

Dies soll durch die Nutzung des Entwicklungsframeworks Google Web Toolkit (GWT) erreicht werden. Dabei handelt es sich um einen Baukasten zur Entwicklung von komplexen Web-Applikationen. Diese sollen hochperformant sein, wobei die Entwicklung als solche sehr produktiv und auch ohne fundierte Kenntnisse über Ajax, XMLHttpRequest oder JavaScript möglich ist (vgl. [GWT 01]).

Die Verwendung von GWT hat einen entscheidenden Vorteil, einen Java-to-JavaScript Compiler. Die gesamte Entwicklung der Webanwendung wird dabei in Java vollzogen. Sie ähnelt dabei der Oberflächengestaltung von Java Swing. Der entsprechende Java Code wird anschließend in JavaScript umgewandelt, sodass eine browserunabhängige Webanwendung entstehen kann, ohne das auch nur eine Zeile JS geschrieben wurde. Das GWT – SDK stellt dabei eine Vielzahl von Java-Widgets bereit, mit denen die Anwendung gebaut werden kann (vgl. [GWT 01]).

Auch die Nutzung der GWT-Erweiterung SmartGWT wäre denkbar. Sie bringt neben einer Vielzahl grafisch sehr ansprechender Oberflächenelementen, welche optisch das „normale“ GWT übertreffen, ein sehr gut dokumentiertes Showcase mit, dass nicht nur zeigt, was mit dieser Erweiterung möglich ist, sondern gleichzeitig noch die dazugehörigen Code-Fragmente zur Verfügung stellt. Das macht eine Einarbeitung sehr einfach, sodass mit Hilfe von SmartGWT innerhalb von sehr kurzer Zeit gute Ergebnisse zu erzielen sind. Die reichhaltige Bibliothek an Oberflächenelementen stammt bei SmartGWT aus dem SmartClient, einem Framework für die Webentwicklung mit JavaScript (vgl. [Smart 01]).

Bei dieser Eigenentwicklung des IMAP-Editors soll der Formulareditor ein Bestandteil des Workfloweditors sein. Grundsätzlich sollten die Editoren sehr einfach gehalten werden. Die Modellierung soll mit nur sehr wenigen, dafür aber aussagekräftige Elementen geschehen. Die Verwendung von einfachen Funktionen, die vorkonfiguriert zur Modellierung der Prozesse bereitgestellt werden, ist angedacht. Diese Funktionen müssen im Workfloweditor nur auf die Arbeitsfläche gezogen und miteinander verbunden werden, sodass der Ablauf klar wird.

Neben dieser einfachen Möglichkeit zum Anlegen sehr intuitiver Workflows soll auch der Formulareditor sehr simpel angelegt sein. Durch eine Interaktion mit einem angelegten Task im Workfloweditor wird der Formulareditor gestartet.

Er ist dabei Teil des gesamten Editors und somit nicht eigenständig ausführbar. Die Tasks des Workflows sollen Schnittstellen zu realen Funktionen im System darstellen, welche gewisse Input- und auch Outputparameter enthalten. Sie werden dem Formulareditor übergeben und aus ihnen werden ebenfalls vorkonfigurierte Elemente erstellt, welche nur verteilt werden müssen.

Die Eigenentwicklung soll ebenfalls eine eigene Darstellungsform für die Workflows mit sich bringen. Sie basiert auf einem durchgehenden Datenmodell, welches dabei auf ein Minimum reduziert wird, sodass es nur Grundelemente wie Task und Links gibt. Die Formulare sollen dabei ein Teil der Task sein und selber als Container für eine Reihe von Formularelementen wirken.

Dieses Datenformat entspricht zwar keinem Standard, soll jedoch helfen, eine sehr intuitive Erstellung der Prozessbeschreibungen zu ermöglichen. Zusätzlich soll dem Modellierer das Neuerlernen einer Notationsform grundsätzlich erspart werden, da er nicht viel „falsch“ machen kann. Das Format der Formulare als solches ist auch nicht extra definiert, sondern gehört zum IMAP-Workflow-Format.

Auf Grundlage dieses Datenformates soll auch eine eigene Engine entwickelt werden. Da das Datenformat sehr einfach und anfangs linear zu halten ist, muss diese Engine bei der Ablaufsteuerung nicht viel leisten. Anhand der Out- und Inputs eines Tasks wird der Ablauf ermittelt. Bei einer Erweiterung auf beispielsweise Entscheidungen und Verzweigungen muss dies in der Engine entsprechend berücksichtigt werden. Da die Formulare direkter Teil des Workflows seien sollen, kann aus diesen direkt bei Erreichen eines Task eine Oberfläche für die Eingabe generiert werden. Diese Eingaben werden ebenfalls durch die Engine verarbeitet.

Durch eine Eigenentwicklung ist es generell möglich, die geforderte Integration optimal umzusetzen. Hierbei wird der Ansatz D2 verfolgt. Wie schon angedeutet, sollen die Formulare ein direkter Teil der Aktivitäten des Workflows sein. Im Gegensatz zu anderen Ansätzen ist das Formular also nicht verlinkt, sondern gehört direkt zu den Workflows. Dies stellt eine optimale Verknüpfung dar.

Doch auch dieser Ansatz bringt einige Probleme mit sich. Grundsätzlich basiert er auf keinem Standard, sondern orientiert sich nur an solchen. Die Notationsform entspricht keiner, in den KMU vorhandenen Darstellungsform, sodass es Akzeptanzprobleme geben könnte. Daneben bedeutet die komplette Eigenentwicklung einen sehr hohen Aufwand. Dieser mag anfangs überschaubar sein, jedoch könnte, bei einer schrittweisen Erweiterung, ein Komplexitätslevel erreicht werden, der nicht absehbar war. Grundsätzlich müssen die Aufwände verschiedener Ansätze gegeneinander abgewogen werden, um herauszufinden, ob eine Eigenentwicklung gegenüber einer Anpassung sinnvoller wäre.

Dafür bietet dieser Ansatz jedoch den Vorteil, dass die Integration bestmöglich erfolgt und ein System entwickelt wird, dass allen nötigen Anforderungen entsprechen kann, gerade was die Benutzung angeht.

Nach dieser Betrachtung beider möglicher Ansätze für den IMAP-Editor soll im Folgenden eine Gegenüberstellung stattfinden, sodass letztendlich eine Entscheidung für eine der Varianten getroffen werden kann.

4.4 Technologische Entscheidung auf Grundlage der Anforderungen

Nach den vorherigen Betrachtungen der Anforderungen, theoretischen Grundlagen und nach der Vorstellung zweier Ansätze bezüglich einer möglichen Umsetzung soll nun eine Entscheidung getroffen werden, nach welcher Vorstellung und auf welcher technologischer Grundlage der IMAP-Editor entworfen und umgesetzt werden soll.

Ein wichtiger Aspekt bei dieser Entscheidung ist die Umsetzung der Anforderungen, welche an den IMAP-Editor gestellt werden. Diese wurden im Kapitel 3 ausgiebig dargestellt. Einige dieser Bedingungen sind dabei mit einer höheren Priorität zu versehen. Andere wurden zwar genannt, eine Nicht-Umsetzung wäre jedoch nicht so gravierend. Der zu entwickelnde IMAP-Editor soll eine Grundfunktionalität bereitstellen, jedoch ist die komplette Umsetzung aller Anforderungen nicht zwingend, da es sich um einen Prototypen handelt. Im Anhang befindet sich eine ausführliche Darstellung der Anforderungen und welche durch welchen Ansatz umgesetzt werden. Grundsätzlich soll im Folgenden auf die wichtigsten eingegangen werden.

Beide Formulareditoren stellen dabei eine Eigenentwicklung dar. Da die geforderte Grundfunktion bei der Entwicklung dieser umsetzbar ist, sollte der Formulareditor-Teil des IMAP-Editors gewissen Ansprüchen genügen, jedoch sollte der Fokus bei der Entscheidung für einen Ansatz auf dem Prozesseditor liegen. Hierbei stehen die Konzepte einer Eigenentwicklung denen der Anpassung eines bestehenden Systems gegenüber.

In den Vorbetrachtungen des Projektes haben sich für die Auswahl eines Prozesseditors einige Kriterien ergeben, was eine Mindestanforderung angeht (vgl. [HMS11] S.32f). Dies sind die Erstellung von Prozessbeschreibungen, die auch ausführbar gemacht werden können, eine intuitive Modellierung bei geringem Funktionsumfang und die Integration, also die Verknüpfungsmöglichkeit der Workflows mit den Formularen.

Die Prozessbeschreibungen können sowohl mit dem Activiti-Modeller, als auch mit einer GWT-Eigenentwicklung umgesetzt werden.

Aufgrund der Modellierung in BPMN sollten die Workflowspezifikationen von Activiti grundsätzlich ausführbar sein, jedoch wurde die Problematik der Überführung des reinen BPMN in eine ausführbare Version für die Engine schon erläutert.

Dafür ist ein Extra-Tool notwendig, was die Nutzung komplizierter machen könnte, als sie notwendigerweise sein müsste.

Bei der Eigenentwicklung ist eine direkte Ausführung der erstellten Prozesse möglich. Sie kommt durch die eigene Engine zustande, welche das eigene Format direkt interpretieren und die Formularelemente entsprechend darstellen kann.

Ein weiterer wichtiger Aspekt ist die Möglichkeit der Verknüpfung der Formulare mit den Workflows. Bei der Eigenentwicklung stellt dies kein Problem dar und ist ohne weiteres möglich.

Die Formulare sind Teil des Workflows und der Formulareditor kann direkt und taskabhängig gestartet werden, sodass die Formulare im direkten Kontext des Workflows stehen. Bei der Benutzung von Activiti sieht dies ganz anders aus. Der Modeller bietet keine Möglichkeit, Formulareingaben zu hinterlegen. Das ist vermutlich darin begründet, dass dies durch die BPMN eigentlich nicht unterstützt wird. Grundsätzlich könnte das Problem auch durch das Extra-Tool gelöst werden, das trägt dann aber auch wieder zur Komplexität bei.

Die Komplexität führt direkt zur nächsten Anforderung, der einfachen Modellierung bei geringem Funktionsumfang. Hier muss ein großes Minus für das Activiti - System verzeichnet werden. Für die Modellierung sind grundsätzliche BPMN-Kenntnisse notwendig. Zudem ist der Modeller sehr umfangreich. Neben unzähligen Spezialisierungen der Grundelemente der BPMN, ist es vor allem die Property-Table zur Modifizierung der Elemente, die sehr erdrückend wirkt. Die unzähligen verschiedenen Eigenschaften wirken verunsichernd, wenn nicht genügend Kenntnisse der BPMN vorliegen. Die kleinste Änderung kann bewirken, dass die Prozessdefinition nicht mehr ausführbar ist. Für die Modellierer in den KMU, denen die Arbeit erleichtert werden soll, könnte dies eher eine Verkomplizierung bedeuten.

Grundsätzlich könnte die Funktionalität des Modellers umgeschrieben werden, das würde aber viel Zeit in Anspruch nehmen. Zwar muss bei einer Eigenentwicklung sämtliche Funktionalität neu entwickelt werden, durch die gegebenen Möglichkeiten der GWT-Programmierung (Java-to-JavaScript, Widgets) sollte das aber viel schneller machbar sein. Die intuitive Modellierung und der geringe Funktionsumfang kann bei einer Eigenentwicklung durch eine gute Konzeption gewährleistet werden.

Bei der Nutzung von Activiti sollte hervorgehoben werden, dass dieses System auf BPMN und damit einem aufkommenden Standard basiert. Die Engine bietet eine Vielzahl von Möglichkeiten. Auf den ersten Blick sieht das Activiti-System also sehr gut aus. Bei näherer Betrachtung ändert sich dieses Bild allerdings. Der Workfloweditor ist zu kompliziert, das erstellte BPMN muss nachbearbeitet werden. Die eigentliche Integration wird nur durch ein Extra-Tool möglich.

Um einen sinnvollen und funktionierenden Arbeitsablauf zu ermöglichen, müssen mehrere Anwendungen benutzt werden und es besteht viel Potential für mögliche Fehler und Verwirrungen seitens des Nutzers. Zusätzlich bestehen viele kleine Probleme.

Activiti als Projekt ist noch sehr jung, sodass mit vielen neuen Releases zu rechnen ist, die mögliche Schnittstellen verändern könnten.

Zudem ist bei diesem Ansatz die Migration zum mobilen Einsatz noch nicht berücksichtigt. Zwischen Formulareditor und Laufzeitkomponente kommt es zu einem doppelten Rendering und damit zu Inkonsistenzen in der Optik. Außerdem besteht keine einheitliche technologische Basis. Neben Java kommen JavaScript und HTML zum Einsatz. Dies könnte nötige Wartungen in den KMU erschweren.

Zwar bringt eine Eigenentwicklung auch einige Nachteile mit sich. Zum Beispiel basiert das Workflowformat auf keinem Standard und ist damit proprietär. Zudem muss bei der Entwicklung viel Eigenleistung eingebracht werden, was zum Beispiel das Entwickeln einer eigenen Engine angeht. Jedoch kann dies durch die Möglichkeiten zur Entwicklung, die GWT beinhaltet, bei einer guten Konzeption in relativ kurzer Zeit umgesetzt werden. Dem gegenüber stehen allerdings viele Vorteile.

Neben einer einheitlichen technologischen Basis ist hier vor allem die Unabhängigkeit externer Entwicklungen zu nennen. Bei paralleler, guter Dokumentation muss lediglich gesichert werden, dass teaminternes Wissen über die Quellcodes weitergegeben wird. Mit der nötigen Zeit ist mit einer Eigenentwicklung alles möglich, was die Anforderungen vorgeben.

Je nach Konzept wird die gesamte Nutzung sehr einfach und intuitiv. Die angestrebte Integration genügt allen Anforderungen. Zudem bietet GWT optisch sehr ansprechende Möglichkeiten zur Darstellung und zudem einen problemlosen mobilen Einsatz ohne viel Extra-Entwicklungszeit. Die Applikation kann sogar browserunabhängig verfügbar gemacht werden. Die Problematik des nicht-standardisierten Formates könnte durch eine im Nachhinein eingebaute Export-Funktion in andere Formate ebenfalls gelöst werden.

Es stehen also eine Reihe nicht unerheblicher Nachteile der Nutzung von Activiti einer ganzen Reihe von Vorteilen von einer Eigenentwicklung gegenüber. Activiti ist mit sehr hohen Aufwänden in der Anpassung des Editors und damit einem hohen Risiko verbunden. Wie schon erwähnt, sind die Aufwände bei der Eigenentwicklung zwar auch noch nicht abschätzbar, auf kurze Sicht stellt sie aber ein geringeres Risiko dar, da in kurzer Zeit ansprechende Ergebnisse geschaffen werden können. Activiti dagegen wartet mit eher schlechten Dokumentationen, vielen Ansatzpunkten für eine nötige Verbesserung und vielen internen Abhängigkeiten auf. Wenn das System gut entworfen ist, kann mit GWT dagegen sehr viel erreicht werden, was durch die Anforderungen von IMAP vorgegeben wird.

Dementsprechend sollte der IMAP-Editor auf dieser Technologie beruhen. Für eine optisch ansprechendere Entwicklung soll dabei die GWT-Erweiterung SmartGWT genutzt werden.

Nachdem die Entscheidung getroffen ist, soll im Folgenden der IMAP-Editor basierend auf SmartGWT als Technologie in einer Eigenentwicklung entworfen und prototypisch umgesetzt werden.

5 Entwurf des Systems

Nachdem entschieden wurde, den IMAP-Editor auf Grundlage des zuvor vorgestellten Ansatzes einer Eigenentwicklung und damit auf der Basis von SmartGWT zu entwickeln, soll das System an dieser Stelle entworfen werden. Dazu wird zunächst das Feinkonzept des Editors erläutert, da bei dem vorgestellten Ansatz nur sehr oberflächlich die genaue Funktionalität des IMAP-Editors beschrieben wurde und eher die Vorteile von GWT in den Fokus rückten. Da die Entwicklung mit diesem Framework einige Besonderheiten mit sich bringt, welche nach der Entscheidung für diesen Ansatz an Bedeutung gewinnen, sollen diese nach dem Konzept erläutert werden. Nach einer Betrachtung der relevanten Daten für den IMAP-Editor, welche eine ausführliche Darstellung der Integration anhand der eigenentwickelten Notations- und Beschreibungsform enthält, wird die Architektur des Systems beschrieben. Abschließend wird die geplante Oberflächengestaltung erläutert.

5.1 Fein-Konzept des Editors

Dieses Unterkapitel soll das Konzept des IMAP-Editors noch verfeinern. Bisher ist ein Grobkonzept des Ansatzes bekannt. Der Nutzer startet den Workfloweditor und bekommt die Möglichkeit, sehr einfach eine Abfolge von Aktivitäten zu gestalten, die miteinander verbunden sind. Die dafür bereitgestellten Komponenten sind vorgegeben. Zu den Aktivitäten wird der Formulareditor gestartet, wo ebenfalls vorkonfigurierte Elemente zur Verfügung stehen, um die Formulare zu gestalten. Der Nutzer speichert und das Modellierte wird in der Laufzeit verfügbar. Dies ist das Grobkonzept. Abbildung 12 zeigt dazu ein Anwendungsfalldiagramm. Hierbei wird zwischen Modellierer und Endnutzer unterschieden. Der Modellierer kann sowohl Workflows als auch Formulare modellieren, wobei die Formulare ein Teil der Workflows sind. Darauf aufbauend (hier durch <<extends>> dargestellt, da auf die Ergebnisse der Modellierung zurückgegriffen wird) kann die Laufzeit durch den Nutzer verwendet werden.

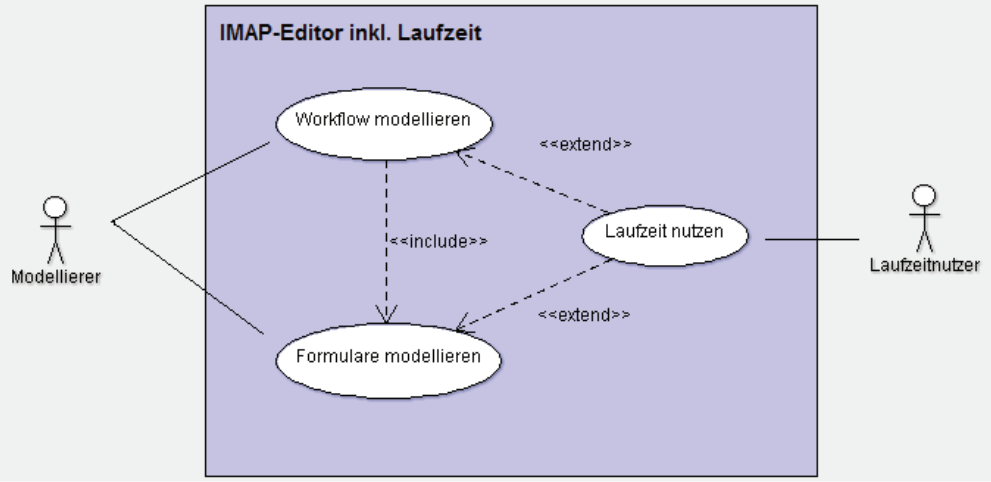


Abbildung 12: Anwendungsfälle des IMAP-Editors

Das Feinkonzept soll ein wenig tiefer ins Detail gehen und dabei die geplante Aktionsfolge näher beschreiben. Das Sequenzdiagramm in Abbildung 13 zeigt diese Abfolge. Im Folgenden soll der Ablauf näher beschrieben werden, sodass die einzelnen Schritte des Diagramms näher erläutert werden.

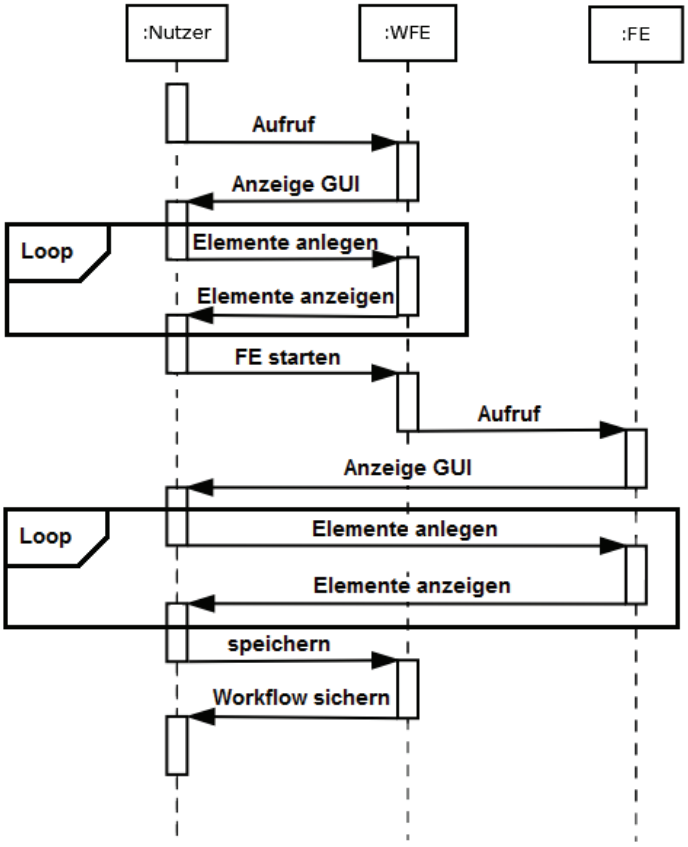


Abbildung 13: Sequenz der Aktionen bei der Modellierung

Der IMAP-Editor ist als Webanwendung konzipiert. Dementsprechend erfolgt der Start über den Aufruf der entsprechenden URL durch den Nutzer. Vor dem eigentlichen Start erhält er die Möglichkeit der Dateiverwaltung, das heißt, er wählt aus, ob ein schon vorhandener Workflow zur Bearbeitung geladen oder ein neuer angelegt wird.

Der Speicherort ist dabei nicht relevant, sämtliche Files werden zentral auf demselben Anwendungsserver hinterlegt, auf dem sich auch die Anwendung befindet. Dies hat den Vorteil, dass der IMAP-Editor von jedem Medium erreichbar ist, das in Verbindung zum Applikationsserver steht und einen Browser besitzt. Der Modellierer ist dementsprechend nicht an einen einzigen Rechner gebunden, sondern nutzt das Montageportal als „virtuellen Arbeitsplatz“.

Wird bei der Auswahl das Laden gewählt, so sollen anhand der hinterlegten Daten des Dateiformates und mit Hilfe der wiederverwendbaren Anzeigeelemente der Workflow und die Formulare wieder so hergestellt werden, wie sie gespeichert wurden.

Bei einem Neuanlegen wird eine leere Arbeitsoberfläche bereitgestellt. Um auf dieser einen Ablauf von Aktivitäten zu modellieren, bekommt der Nutzer eine Auswahl von Elementen zur Verfügung gestellt.

Hinter diesen verbergen sich die Aufgaben, die während der Ausführung dieses Workflows abgearbeitet werden sollen und bilden vorkonfigurierte Schnittstellen zu realen Funktionen und anderen Systemen innerhalb des Montageportals. In diesen Schnittstellen wird durch Eingabe- und Ausgabeparameter definiert, was notwendig ist, um diese Aufgabe von einer anderen Komponente des Portals ausführen zu lassen.

Um eine Aktivität zum Workflow hinzuzufügen, wird diese per Drag & Drop auf die Arbeitsfläche gezogen. Durch SmartGWT ist diese Funktionalität bei vielen Elementen schon vorgegeben. Die Aufgabe wird entsprechend angelegt und kann daraufhin mit anderen Aufgaben verbunden werden. Um die Modellierung so einfach wie möglich zu halten, steht für die Verbindung kein einzelnes Element in der Auswahlliste der möglichen Elemente zur Verfügung, sondern die Verknüpfung ist Teil der einzelnen Aktivitäten. Für eine Verbindung wird das, in den Task angezeigte, pfeilförmige Icon auf die nächste Aufgabe gezogen. Dabei soll die Richtung berücksichtigt werden, um eine Ablaufreihenfolge garantieren zu können.

Da der Prototyp des Editors vorerst sehr einfach konzeptioniert werden soll, werden in der ersten Iteration Verzweigungen und Entscheidungen noch nicht berücksichtigt. Diese folgen jedoch bei Weiterentwicklung und erhalten dort eine hohe Priorität.

Auf die geschilderte Art können die Arbeitsabläufe als Abfolge einzelner Aufgaben abgebildet und modelliert werden. Zu diesen Aufgaben gehört jeweils ein Formular, welches laut dem gewählten Integrationsansatz und dem eigenen Datenformat, in dem die Modellierungen hinterlegt werden, ein Teil dieser Tasks sind. Dabei ist vorgesehen, dass zu jedem Task genau ein Formular gehört. Die Formulare setzen sich zum Großteil aus Elementen zusammen, die sich aus den Eingabe-Parametern der Funktionen ergeben, zu denen die Aktivitäten im Workflow Schnittstellen bilden. Dementsprechend ergibt sich diese eins-zu-eins-Verbindung. Jedes Formular enthält folglich mindestens einen Button zum Beenden der Aufgabe. Dadurch kann gewährleistet werden, dass der Task abgearbeitet wird. Auch die Grundlage für eine genaue Zeitnahme wird gelegt.

Außerdem wird ermöglicht, optionale, zusätzliche Aufgaben, wie das Melden einer Störung, welches im HTML-Prototypen auf fast jedem Formular modelliert ist, in jeder Eingabemaske und somit bei jeder Aufgabe zur Verfügung zu stellen.

Dem gängigen Arbeitsablauf bei der Modellierung folgend, sollten nach dem Anlegen aller Aufgaben die einzelnen, jeder Aufgabe zugeordneten Formulare modelliert werden. Grundsätzlich ist der Zeitpunkt der Formularerstellung jedoch egal. Es sollte lediglich beim Speichern des Workflows darauf geachtet werden, dass alle Formulare modelliert sind, sodass das Ergebnis ausführbar ist und keine Fehler produziert.

Der Formulareditor wird durch einen Doppelklick gestartet, wobei der Workfloweditor beim Aufruf des Formulareditors als Parameter die Funktion übergibt, die bearbeitet wird. Das ist damit begründet, dass die meisten Formularelemente aus den Eingabeparametern der Funktionen generiert werden.

Da durch diese Vorgabe alle benötigten Elemente für das Ausführen der Funktion abgedeckt werden, ist die Verteilung dieser Elemente auf dem Formular zwingend, da ohne sie eine fehlerfreie Ausführung des Formulars zur Laufzeit der Engine nicht möglich ist. Hinzu zu diesen Eingabeparametern, die verteilt werden müssen, kommen optionale Elemente.

Hierbei wird es sich zu Beginn um ein Label, also um eine Beschriftung handeln, um einfachen Text im Formular anzeigen zu können. Damit kann dann beispielsweise ein Titel oder die Beschreibung der Aufgabe des Formulars angezeigt werden. Zudem besitzt jede Funktion, wie erwähnt, den Button für ihre Ausführung zur Laufzeit. Er ergibt sich jedoch nicht aus den Eingabeparametern, muss aber dennoch verteilt werden.

Neben dem ersten Anlegen der Formulare, können sie auch im Nachhinein verändert werden. Dazu werden die bereits erstellen Elemente beim Start des Formulareditors geladen und bereits auf der Arbeitsfläche verteilt, wo diese verschoben oder bearbeitet werden können.

Nach der Modellierung aller nötigen Formularelemente zu den entsprechenden Aktivitäten kann der Workflow gespeichert werden. Bei diesem Vorgang sollte zukünftig eine Prüfung erfolgen, inwiefern alle nötigen Elemente verteilt wurden, sodass Fehler vermieden werden können. Das ist für den ersten Prototypen jedoch noch nicht vorgesehen.

Abbildung 14 zeigt die Modellierung in einem Aktivitätsdiagramm. Dabei sind wesentliche Schritte abgebildet. Unter anderem wird berücksichtigt, dass es egal ist, zu welchem Zeitpunkt der Modellierung die Formulare zu einem Task erstellt werden, sofern am Ende alle nötigen Eingaben getätigt wurden.

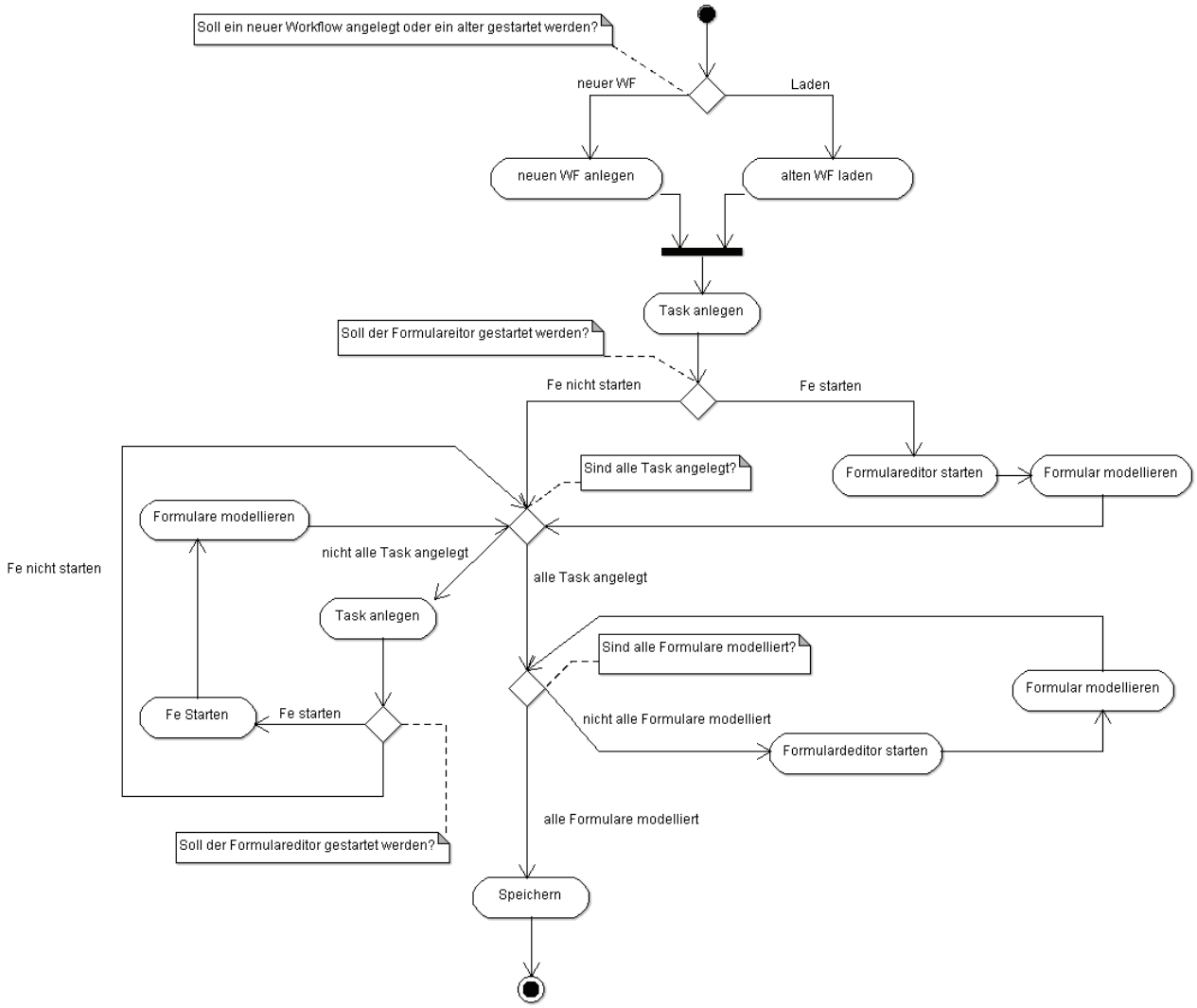


Abbildung 14: Aktivitätsdiagramm der Modellierung

Die so erstellten Workflows sind in der ebenfalls selbst entworfenen Laufzeit ausführbar. Diese soll jedoch nur am Rande dieser Arbeit betrachtet werden, der Fokus der Arbeit auf der Modellierung liegt.

Anhand des gespeicherten Workflows wird in dieser Engine die Abfolge der unterschiedlichen Aufgaben gesteuert. Da jeder Task mit Out- und Input-Link ausgestattet ist, wird die Reihenfolge ermittelt. An entsprechender Stelle wird das Formular des Tasks geladen. Die Funktionen, die durch die Formulare abgebildet sind, werden durch den entsprechenden Button ausgeführt. Dabei werden die eingetragenen Werte aus den Formularfeldern ausgelesen und die Funktion wird im ESB aufgerufen. Die Aktivität wird als erfüllt markiert und die Ablaufsteuerung ermittelt den nächsten Task. Dies ist das Feinkonzept des IMAP-Editors, wie es mit Hilfe von GWT umgesetzt werden soll. Im Folgenden sollen einige Besonderheiten erläutert werden, die bei der Benutzung von GWT für die Entwicklung einer Webapplikation berücksichtigt werden müssen.

5.2 Besonderheiten von GWT

Bei dem Entwurf des IMAP-Editors mit Hilfe des Entwicklungsframeworks SmartGWT sind einige Besonderheiten zu beachten, welche die Nutzung von GWT mit sich bringt. Sie sollen an dieser Stelle erläutert werden, damit die Applikation im Detail entworfen werden kann[14].

Wie bereits erläutert, besteht der große Vorteil von GWT gegenüber anderen Webentwicklungsframeworks darin, dass die Oberflächen in Java geschrieben werden, um dann in JavaScript kompiliert zu werden. Dabei muss berücksichtigt werden, dass nicht jeder in Java geschriebene Code kompatibel zu JavaScript ist. Dementsprechend findet bei GWT eine Trennung statt.

Auf der Clientseite eines Projektes wird der Code verwaltet, der am Ende in JavaScript transformiert wird. Dabei werden alle Oberflächen und deren Logik, wie beispielsweise Event-Handler, programmiert.

Sollen im Quellcode jedoch externe Bibliotheken verwendet werden, so kann dies zu Problemen bei der Übertragung zu JavaScript führen. Nicht alles kann und soll überführt werden. Die Benutzung von externen Codes wird auf ein internes Servlet ausgelagert. Entsprechende Funktionen können dann per Remote Procedure Calls angesprochen werden.

Die fertige Applikation wird nach der Entwicklung und Kompilierung in einem „war“-Archiv abgelegt, um es auf einen Tomcat oder ähnlichen Applikationsserver übertragen zu können. In diesem befinden sich dann die kompilierten JavaScript-Ressourcen und die .class-Dateien der Servlets.

So viel zu den Grundlagen der Entwicklung einer GWT-Applikation. Das nächste Unterkapitel beschäftigt sich etwas intensiver mit dem Entwurf. Dabei werden die Daten näher betrachtet, die im IMAP-Editor verwendet werden. Zudem wird erläutert, wo sie herkommen. Der ESB rückt bei der Betrachtung in den Vordergrund.

5.3 Daten

Dieses Kapitel beschäftigt sich mit den im IMAP-Editor verwendeten Daten. Dabei handelt es sich vor allem um die Daten, welche durch die Anbindung an den ESB im Editor verfügbar sind. Da es angedacht ist, innerhalb der Editoren viele vorkonfigurierte Elemente bereitzustellen, um dem Modellierer seine Arbeit noch leichter zu machen, muss geklärt werden, was sich hinter diesen Informationen verbirgt und woher diese stammen.

Schon zu Beginn der Ausführungen über das Montageportal war von einem einheitlichen Datenmodell der Komponenten die Rede. Es soll hier näher spezifiziert werden. Zudem wird erläutert, wie die Daten aus diesem Modell in den Editor kommen und was das für die Applikation zu bedeuten hat.

[14] Die generelle Benutzung von GWT zur Entwicklung einer Webanwendung ist sehr ausführlich auf den offiziellen Projektseiten dargestellt: http://code.google.com/intl/de-DE/webtoolkit/gettingstarted.html [15.08.2011]

Wie schon angemerkt, sollen durch den ESB „Funktionen“ vorgegeben werden. Dabei handelt es sich um Schnittstellen anderer Dienste des Montageportals. Innerhalb dieser Schnittstellen wird definiert, wo sich diese Dienste befinden und wie diese angesprochen werden können. In den meisten Fällen verbirgt sich hinter einer solchen Schnittstelle ein WebService, dem bestimmte Informationen beim Ansprechen übergeben werden, woraufhin eine spezifische Reaktion ausgelöst wird. Beispielsweise wird ein Nutzer nach Angabe von Name und Passwort angemeldet. Auf welche Art und Weise die benötigten Input-Parameter an die entsprechende Adresse geschickt werden, also per GET oder POST, in welcher Reihenfolge und mit welchem Bezeichner, wird durch die Laufzeitkomponente geregelt.

In der Schnittstellendefinition werden die Eingabeparameter bereitgestellt und im System zu Eingabeelementen für den Formulareditor verarbeitet. Solche Schnittstellen werden durch das erwähnte, einheitliche Datenmodell abgebildet. Ein Klassendiagramm dieses Modells ist in Abbildung 15 zu sehen.

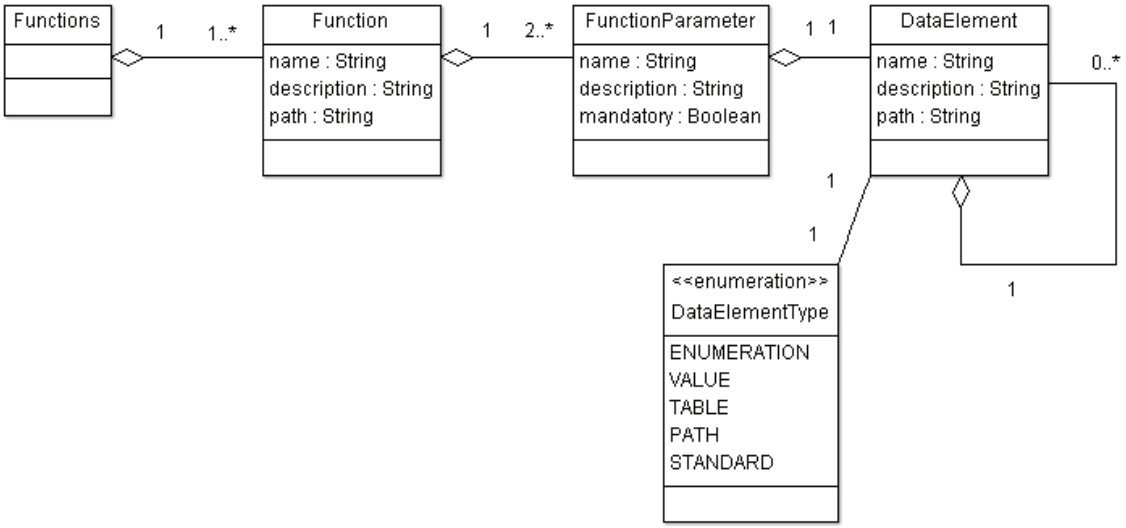


Abbildung 15: Klassendiagramm des gemeinschaftlichen Datenmodells

Grundsätzlich gliedert sich dieses Datenmodell wie folgt: An oberster Stelle steht Function. Dieses ist ein Container für die im ESB vorgehaltene und für die Modellierung verwendete Schnittstelle. Das Element Functions, welches der Function vorgelagert ist, dient der verbesserten Darstellung, da eine Serialisierung ohne dieses zusätzliche Element nicht möglich wäre. Auf die Serialisierung wird im Folgenden noch eingegangen.

Die Function ist durch mehrere Attribute genau beschrieben. Dazu zählen Name, Beschreibungstext, aber auch ein Pfad.

Hinter diesem verbirgt sich eine REST-Ressource, an die die spezifischen Eingaben eines ausgefüllten Formulars bei Betätigung des Ausführen-Button gesendet werden sollen, um diesen Dienst zu starten. Beispielsweise wird hier der Log-In veranlasst.

Was an diese Ressourcen gesendet werden soll, wird durch die Abbildung des WebService auf das Datenmodell ebenfalls dargestellt. Eine solche Funktion besitzt, wie im Modell gezeigt wird, Eingabe- und Ausgabeparameter, wobei für die Modellierung die Eingabeparameter von besonderem Interesse sind. Diese werden durch die Klasse FunctionParameter abgebildet.

Hinter diesen verbergen sich einzelne Eingabeelemente. Attribute dieser Klasse sind neben dem Namen ebenfalls eine Beschreibung, die zur Beschriftung der Elemente in den Editoren verwendet werden kann. Hinzu kommt ein Boolean-Wert bezüglich der Notwendigkeit dieses Elements, da nicht zwingend jeder Eingabewert nötig ist. Optionale Werte werden hier geduldet. Ein weiteres Attribut stellt hier das sogenannte DataElement dar. Dies ist das eigentliche Eingabeelement, anhand dessen das grafische Element letztendlich konfiguriert wird. Hier existieren noch einmal extra Name, Beschreibung, Pfad und Typ. Das hat im Grunde aber nur Identitätsgründe. Es wird hier beispielsweise unterschieden, ob es sich bei einem Eingabeparameter um ein Textfeld oder einen Button handelt. Diese Zuordnung geschieht durch die Aufzählung DataElementType.

Das DataElement enthält eine Besonderheit, die zusätzliche Eigenschaft eines Containers weiterer Datenelemente. Ein DataElement kann sich also selbst enthalten. Dies ist damit begründet, dass ein Eingabeelement, welches die Auswahl mehrerer Unterelemente ermöglicht, ebenfalls abgebildet werden soll. Ein Beispiel wird bei der Beschreibung des Datenformates gegeben.

Damit ist die Beschreibung des gemeinschaftliches Datenmodells, auf das alle Teile des Montageportals zugreifen, abgeschlossen. Es ist noch komplexer, für den IMAP-Editor werden aber nur die angesprochenen Teile des Modells verwendet, weshalb die anderen Teilbereiche nicht erläutert werden sollen.

Von den Datenobjekten des Modells werden Instanzen erstellt und durch den ESB für den IMAP-Editor bereitgestellt. Die entsprechenden Informationen müssen zum Zeitpunkt dieser Arbeit noch von Hand in das System eingepflegt werden und basieren noch nicht auf realen Unternehmensdaten, was sich jedoch ändern soll. Das sich in Entwicklung befindliche Konfigurationswerkzeug, welches in der Systemübersicht aus Kapitel 3.1.1 als ESB-Config dargestellt wird, soll es ermöglichen, ebenfalls einfach und intuitiv, Unternehmensdaten aus ERP und anderen Systemen in das IMAP-Portal zu überführen und so das Datenmodell zu pflegen.

Entscheidend ist die Fragestellung, wie die Repräsentationen der Datenobjekte, die im System vorliegen, in den IMAP-Editor übernommen werden können. Hierbei kommt die Serialisierung von Objekten zur Datenübertragung zum Einsatz.

Dabei werden Datenobjekte als Instanzen in einer bestimmten Form dargestellt, um diese für den Austausch von Daten oder zur Persistenz, also dem Speichern des Zustands eines Objektes, verfügbar zu machen.

Gerade der Persistenz-Aspekt ist auch ein Grund dafür, warum für das Abspeichern der Workflows ebenfalls eine Form der Objektserialisierung zum Bewahren des Zustandes eines Objektes verwendet wird. Dazu aber später mehr.

Objekte können auf verschiedene Arten und in verschiedenen Formen serialisiert werden. So ist eine Übertragung in einen sequentiellen, binären Bytestrom möglich, wie er von Haus aus durch Java unterstützt wird. Dazu muss eine Klasse das Interface Serializable implementieren und eine ID besitzen. Daraufhin können sie mit einem ObjectInputStream binär in eine Datei geschrieben werden. Es existieren jedoch noch andere Formen der Serialisierung.

Sehr häufig für den Austausch von Daten verwendet wird die Abbildung von Objekten auf eine XML-Struktur. Diese Form wird auch zum Austausch der Datenobjekte bei IMAP genutzt. Die im System vorliegenden Objekte des gemeinsamen Datenmodells werden durch eine externe Bibliothek in einen XML-Baum überführt. Um dies zu ermöglichen, muss das gemeinschaftliche Datenmodell in XML serialisierbar sein. Das wird erreicht, in dem die Klassen als XmlRootElement annotiert und die Variablen und Attribute der Klassen als protected deklariert werden.

Die Serialisierung als solche findet dann mit Hilfe von JAXB statt, der Java Architecture for XML Binding[15]. Hierfür wurde eine Helferklasse erstellt, welche ebenfalls für alle Komponenten des Montageportals verfügbar ist. Dieser RESTserializer genannte Helfer beinhaltet Methoden, um aus XML-Strukturen Objekte und umgekehrt zu erzeugen.

Um diese XML-Darstellungen der Funktionen für die verschiedenen Komponenten verfügbar zu machen, wird erneut Jersey und damit REST eingesetzt. Für die Verbreitung des XML kommt eine Ressource zum Einsatz. Wie schon erwähnt, kann diese so konfiguriert werden, dass nur bestimmte Rückgabewerte erlaubt sind. Wird die entsprechende Ressource angesprochen, werden die Objekte in XML serialisiert und diese Struktur wird als Response für den Call zurückgeschrieben. Damit ist die XML-Repräsentation dann verfügbar, der Rückgabewert muss nur noch verarbeitet werden. Abbildung 16 zeigt einen Zugriff mit entsprechendem Rückgabewert auf so eine Ressource. Hier dargestellt ist das Ermitteln aller möglichen Funktionen innerhalb des Workfloweditors unter Berücksichtigung der Tatsache, dass das gemeinsame Datenmodell nicht JavaScript-geeignet ist, was jedoch später noch genau erläutert wird.

[15] Vgl. http://jaxb.java.net/ [15.08.2011]

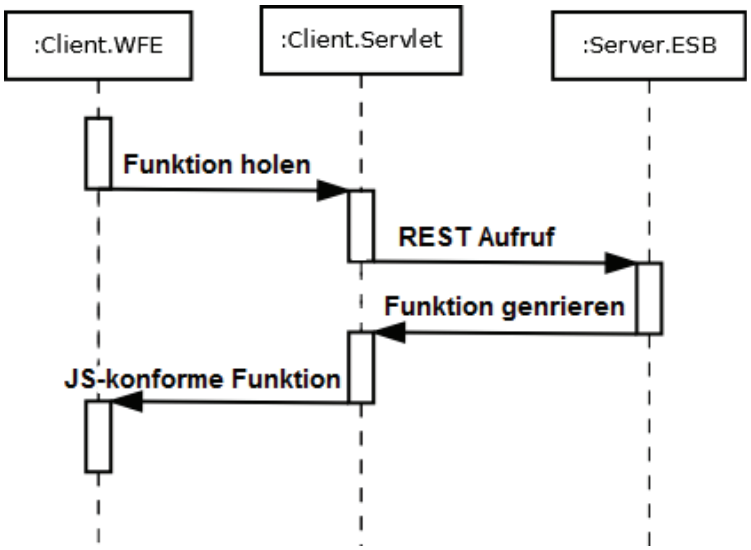


Abbildung 16: Sequenzdiagramm der Datenanbindung

Im IMAP-Editor sollen also durch Calls auf bestimmte URLs die im System vorliegenden Dienste verfügbar gemacht werden. Dazu soll ebenfalls der RESTserializer verwendet werden. Der für alle Komponenten des Portals verfügbar gemachte Helfer gehört jedoch zu den externen Bibliotheken, welche nicht im GWT-Client verwendet werden können. Dementsprechend muss die Funktionalität des Datenholens auf ein Servlet ausgelagert werden. Auf diesem sollen dann aus der XML-Struktur wieder entsprechende Datenobjekte erzeugt werden. Allerdings kommt es auch hierbei zu weiteren Problemen.

Zwar soll das Datenmodell für alle Teile des Montageportals verfügbar sein, jedoch kann es nicht einfach so in einer GWT-Applikation verwendet werden, da eine Umwandlung in JavaScript nicht möglich ist. Das liegt unter anderem an den Annotationen, die notwendig sind, um die Objekte überhaupt in XML serialisieren zu können. Gelöst werden soll dieses Problem dadurch, dass eine GWT-konforme Abbildung des gemeinschaftlichen Datenmodells erstellt werden soll. Diese wäre dann JavaScript-konform, würde also nur aus Grunddatentypen (Int, String, Bool) bestehen beziehungsweise aus Zusammensetzung von diesen. Beim Datenholen sollten dann auf Serverseite Objekte aus den XML-Strukturen erstellt und anschließend auf das JavaScript- und GWT-konforme Datenmodell im Client abgebildet werden. Damit soll dann im Editor gearbeitet werden, so dass trotz allem die Daten aus dem ESB authentisch vorhanden sind.

Neben dem gemeinschaftlichen Datenmodell und der Datenanbindung an den IMAP-Editor gehört auch das für den Editor zu schaffende, in gewissem Sinne proprietäre Dateiformat und damit die eigentlich Integration von Workflow- und Formularbeschreibungen in den Bereich der Daten.

In den Ausführungen zu den theoretischen Grundlagen wurde zwischen Beschreibungsformaten zur rein grafischen Notation und solchen zur automatisierten Ausführung gedachten, jedoch grafisch komplizierten Formaten unterschieden. Das IMAP-Format soll dabei die Vorteile verschiedener Variationen vereinen.

Die Grundlage dieses Formates beruht auf einem konsistenten, serialisierbaren Datenmodell. Hierbei liegt der Fokus auf der technischen Seite und damit auf der Ausführbarkeit. Da das Format aber, wie auch schon mehrfach dargelegt wurde, sehr einfach aufgebaut sein soll, ist es ohne weiteres möglich, auf das Datenmodell eine einfache, verständliche Notation aufzusetzen.

Dabei wäre hier sogar ein rein grafischer Austausch der Elemente im Sinne des Corporate-Design-Gedanken möglich. Da der Prototyp des IMAP-Editors jedoch erst einmal für ein KMU entsteht, soll dem Modell zunächst aber eine vorgegebene Darstellungsform aufgesetzt werden. Optische Aspekte finden sich im Kapitel der Oberflächengestaltung wieder.

Zunächst soll eine ausführliche Beschreibung des Dateiformates stattfinden. Das zugehörige Klassendiagramm ist in Abbildung 17 dargestellt.

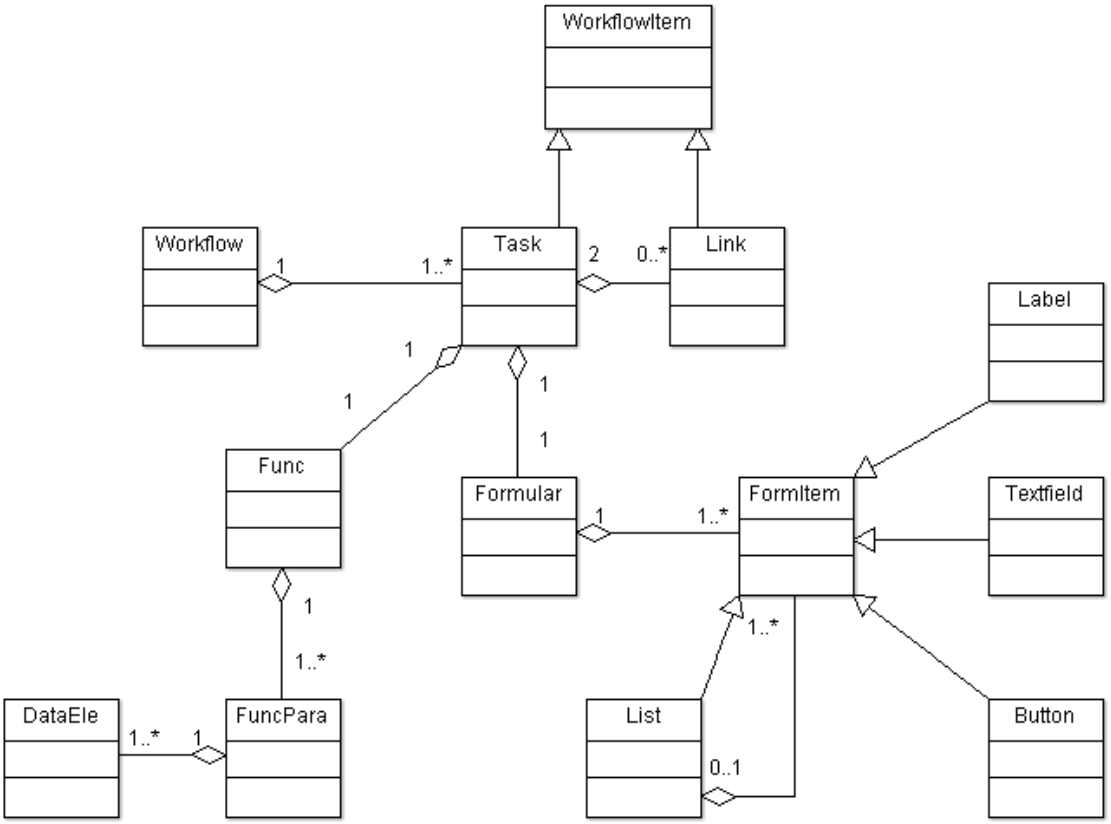


Abbildung 17: Klassendiagramm des IMAP-Datenformates

Ausgangspunkt des Datenmodells ist der Workflow. Dieser besteht aus mindestens einem Task. Eine Begrenzung nach oben gibt es hier nicht. Jeder Task besitzt ein Formular. Somit ist das Formular ein direkter Bestandteil des Dateiformates. Die Integration zwischen Formular und Workflow findet dementsprechend an dieser Stelle statt. Neben dem Formular hat ein Task noch weitere essentielle Bestandteile. Zu jedem Task gehört genau eine Funktion. Diese wird durch die Klasse Func abgebildet.

Dabei handelt es sich um die JavaScript –konforme Abbildung einer Funktion aus dem gemeinschaftlichen Datenmodell aller Komponenten des Montageportals. Die Klassen FuncPara und DataEle entsprechen den weiteren Teilen dieser Repräsentation. Dabei wird allerdings der Output-Parameter nicht berücksichtig. FuncPara steht dementsprechend nur für die Input-Parameter, von denen eine Func einen oder mehrere haben kann.

Genauso verhält sich DataEle, von denen ein FuncPara ebenfalls mehrere besitzen kann. Dies dient der Abbildung bestimmter Sachverhalte, die an anderer Stelle noch dargestellt werden sollen.

Neben dem Task gibt es noch ein weiteres Element, welches vom Workflowitem erbt. Dies ist der Link. Dahinter verbirgt sich ein Verbindungselement für die einzelnen Task. Ein Link hat dabei die Attribute Start-Task und End-Task, hinter denen sich der Ausganspunkt und das Ziel einer Verbindung verbergen. Ein Task als solcher muss dabei nicht zwangsläufig einen Link haben, kann aber mehrere besitzen.

Das zu einem Task gehörende Formular ist ein Container für FormItems. Hinter diesen verbergen sich die eigentlichen Formularelemente, welche von dieser Klasse erben. Ein Formular besitzt dabei mindestens ein FormItem, den Button zum Bestätigen der Funktion.

Grundsätzlich kann das Datenmodell um beliebig viele FormItems erweitert werden, je nachdem, was benötigt wird. Für den Prototypen reichen hier zunächst vier Items aus. Dabei sind Button, Label und Textfeld selbsterklärend. Beim Textfeld gibt es zudem die Besonderheit, dass zwischen Passwort und normaler Eingabe unterschieden wird.

Hinzu kommt das Element List. Es dient dazu, eine Auswahl abzubilden. Hierfür ist es notwendig, dass ein Element mehrere Unterelemente beinhaltet, die einzelnen Auswahloptionen. Dies wird im gemeinschaftlichen Datenmodell durch die Datenelemente in den Datenelementen abgebildet. Hat eine Funktion mit einem Textfeld einen Input-Parameter Eingabe, so hat dieser Parameter beispielhaft ein Datenelement Textfeld. Bei einer List ist dies anders. Ein Input-Parameter, welcher der Auswahl dient, besitzt ebenfalls ein Datenelement (bis hier hin analog zum Textfeld), jedoch beinhaltet dieser weitere Datenelemente, nämlich die eigentlichen Einträge. Das wird durch das FormItem List abgebildet, welches mehrere andere FormItems enthalten kann.

Auf das Datenmodell des Workflowformates werden dann entsprechend grafische Elemente aufgesetzt.

Zur Laufzeit der Editoren werden während der Modellierung von diesem Datenmodell Objekte instanziiert, welche den, auf der Arbeitsfläche dargestellten, Workflow repräsentieren. Diese Abfolge von Aktivitäten soll nach der Modellierung auch gespeichert werden können, um sie für die Ausführung in der Laufzeitumgebung nutzbar zu machen. Dieses Abspeichern soll ebenfalls durch eine Objektserialisierung durchgeführt werden. Dabei ist beim Anlegen des Datenmodells auch wieder entscheidend, dass es serialisierbar ist. Für das Abspeichern soll an dieser Stelle nicht eine XML-Repräsentation gewählt werden, sondern eine Überführung in eine binäre Darstellung, welche per Byte-Strom in ein File geschrieben wird. Dieses erhält die Endung .wf (**W**ork**F**low).

Beim Speicherort der Dateien soll, wie bereits erwähnt, die Idee eines Repositories Verwendung finden. Dahinter verbirgt sich ein Ort, an dem alle Workflowmodelle abgelegt werden. Bedingt dadurch, dass das Montageportal und vor allem auch die Laufzeitumgebung unternehmensweit einsetzbar sein sollten, ist es sinnvoll, dieses Repository nicht auf einem lokalen Rechner zu positionieren. Dieser müsste jedes Mal vor der Ausführung ein Update durchführen und prüfen, ob sich der Bestand der Modelle geändert hat. Sinnvoller wäre es hierbei, die Workflows dort zu speichern, wo auch die Webanwendung als solche liegt, nämlich auf einem Web-Server. Dieser Gedanke der „IMAP-Cloud“ würde dann veranlassen, dass der Endnutzer nur noch einen Browser benötigt, um die Laufzeit zu benutzen, da diese auf die auf demselben Server gelegenen Dateien zugreift. Zudem müsste sich der Modellierer keine Gedanken über den Speicherort seiner Workflows machen.

Nach dieser Beschreibung des IMAP-Workflow-Formats soll nun die Beschreibung der eigentlichen Architektur des IMAP-Editors folgen.

5.4 Architektur

Das folgende Kapitel beschäftigt sich mit der Architektur des zu entwerfenden Systems des IMAP-Editors. Eine Architekturbeschreibung ist in der Regel geprägt durch die Darstellung der einzelnen Komponenten eines Systems, deren Schnittstellen und Beziehungen untereinander. Diese Bereiche sollen auch hier abgedeckt werden, wobei zunächst der Editor als Gesamtsystem betrachtet werden soll, bevor auf die einzelnen Komponenten näher eingegangen wird.

Bei den einzelnen Komponenten des IMAP-Editors handelt es sich um den Workfloweditor und den Formulareditor. Etwas abseits davon ist die Laufzeitumgebung beziehungsweise Engine zu sehen. Sie wird durch das erläuterte, eigene Datenformat mit dem Editor verbunden. Abbildung 18 zeigt diese einzelnen Komponenten.

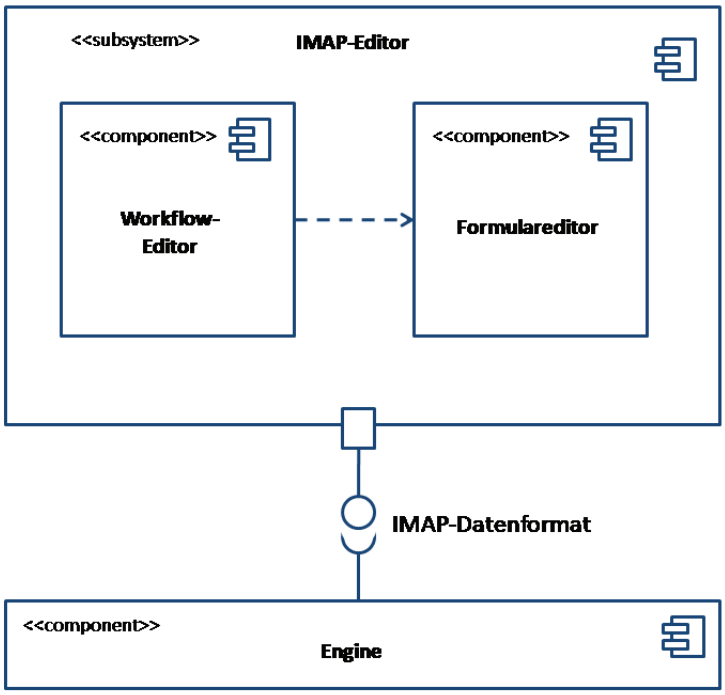


Abbildung 18: Komponenten des IMAP-Editors

Zwischen den Komponenten des Gesamtsystems des IMAP-Editors herrschen bestimmte Beziehungen und Abhängigkeiten. Zentraler Ausgangspunkt ist der Workfloweditor. Dieser ruft den Formulareditor mit den Daten des Workflows auf. Das hat zur Folge, dass der Formulareditor vom Workfloweditor abhängig ist. Ein Start ohne den Workfloweditor ist nicht vorgesehen. Außerdem arbeitet der Formulareditor direkt auf den Daten des Workfloweditors. Das erspart ein nachträgliches Einknüpfen des Formulars in den Workflow und dient damit der Integration. Allerdings wird damit auch die Abhängigkeit der einen Komponente von der anderen geschaffen.

Abseits dieser Komponenten ist die Laufzeitumgebung angeordnet. Sie ist vom Editor im Grunde unabhängig, muss jedoch mit dem Output des Editors gestartet werden. Das stellt eine indirekte Abhängigkeit dar.

Im Grunde wäre allerdings ein Ausführen der Laufzeitumgebung ohne eine zugrunde liegende Workflowbeschreibung sinnlos, da dies die Aufgabe der Laufzeitumgebung verfehlt.

Neben der grundsätzlichen Betrachtung der Komponenten sollen hier kurz generelle Informationsflüsse zwischen den Komponenten des Editors, dem Nutzer und dem ESB verdeutlicht werden. Das dient vor allem der Schnittstellendefinition der einzelnen Elemente des Systems.

Der IMAP-Editor und auch die Laufzeitumgebung sollen sich im Gesamtsystem auf einem Apache Tomcat[16], also einem Anwendungsserver befinden, sodass sie von jeglichem PC oder alternativem Endgerät, das einen Browser und eine Verbindung zu diesem Server besitzt, genutzt werden können.

[16] Vgl. http://tomcat.apache.org/ [15.08.2011]

Um die Anwendungen auf diesem Server ausführen zu können, muss der Nutzer in einem Browser einen Aufruf an die entsprechende URL der Anwendung per HTTP absetzen. Der Server liefert dementsprechend den Zugriff auf die Applikationen zurück, indem die Oberflächenkomponenten, die mit GWT erstellt wurden in Form von JavaScript in eine entsprechende HTML-Seite eingebunden werden.

Die Kommunikation zwischen Editor und Engine mit dem ESB (MULE) erfolgt dagegen über REST, um die vorliegenden Daten in die Applikationen zu bekommen beziehungsweise die durch den Workflow bereitgestellten Dienste anzusprechen. Diese Kommunikation ist in Abbildung 19 dargestellt.

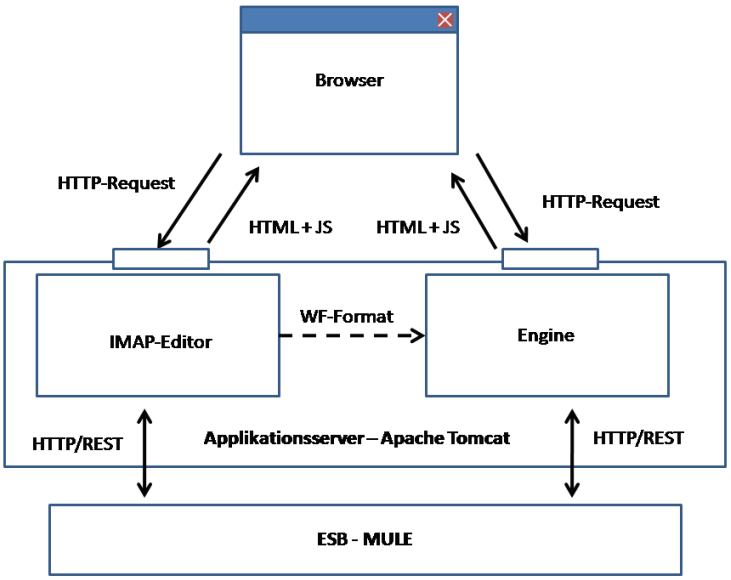


Abbildung 19: Kommunikation im Montageportal

Nach der Betrachtung der Gesamtarchitektur mit den einzelnen Komponenten, Schnittstellen und Beziehungen soll im Folgenden auf die Teilbereiche eingegangen werden.

5.4.1 Details des Workfloweditors

Dieses Unterkapitel soll den Aufbau und die Struktur der Komponenten des Workfloweditors darstellen. Bei dem Workfloweditor handelt es sich um die Hauptkomponente des IMAP-Editors, da der Formulareditor ohne diesen nicht ausführbar wäre.

Bei den Komponenten des Workfloweditors handelt es sich um das FunctionsPanel, den WorkflowView und das zugehörige Datenmodell, aus dem das eigene Datenformat abgeleitet wird. Abbildung 20 zeigt diese Komponenten im Überblick.

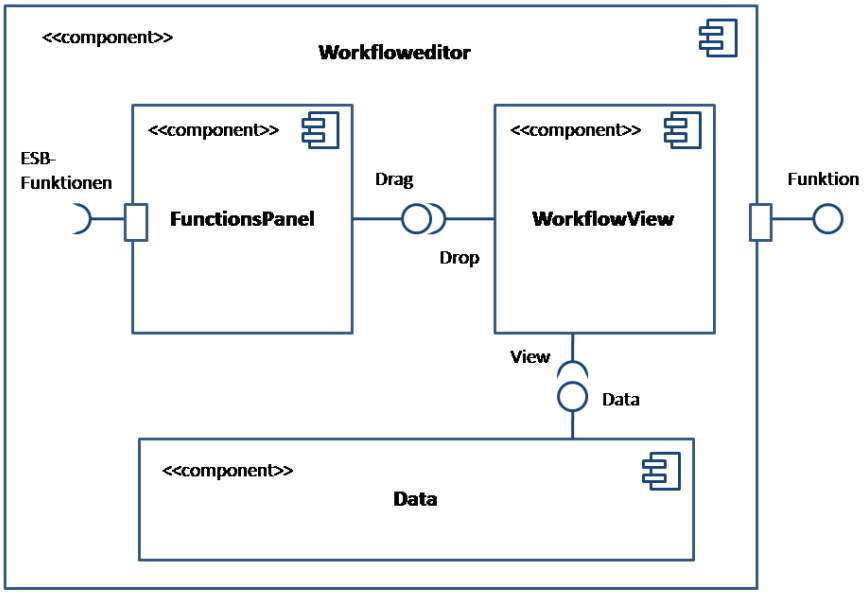


Abbildung 20: Komponenten des Workfloweditors

Das FunctionsPanel dient zum einen als Auswahlliste für die möglichen Funktionen, die zu Taskfolgen zusammengestellt werden sollen, zum anderen aber auch dazu, um eben diese Funktionen aus dem ESB zu erfragen. Dies geschieht durch einen RPC auf das entsprechende Servlet, auf dem die XML-Strukturen deserialisiert und auf das JavaScript-konforme Modell abgebildet werden (vgl. Abb. 16).

Beim WorkflowView handelt es sich um die grafische Repräsentation des Workflowobjektes, mit dem der Workfloweditor instanziiert wird. Dies ist darauf zurückzuführen, dass der IMAP-Editor nach dem Model-View-Controller-Ansatz aufgebaut werden soll, wie es bei grafischen Oberflächen üblich ist. Dabei werden die eigentlichen Daten strikt von den grafischen Repräsentationen, den Views, getrennt. Dementsprechend verbergen sich hinter der dritten Komponente die eigentlichen Daten des Workflows, die mit Hilfe der Views auf dem WorkflowView, der als Arbeitsfläche fungiert, zusammengebaut werden.

Wird dabei beispielsweise ein Task durch Drop auf die Arbeitsfläche angelegt, so wird sichtbar ein Element dargestellt. Gleichzeitig wird im Modell aber auch ein entsprechendes Objekt instanziiert. Werden auf grafischer Ebene Änderungen durchgeführt, so werden diese in das Modell durchgereicht. Ein TaskView, die grafische Repräsentation einer Aufgabe, besitzt außerdem noch die Besonderheit, dass bei einem Doppelklick auf diesen der Formulareditor gestartet werden soll.

Ihm werden diverse Parameter übergeben, die an entsprechender Stelle erläutert werden sollen. Außerdem soll im TaskView die Funktionalität des Verbindens geregelt werden, in dem ein kleines Element, hinter dem sich der Link befindet, von einem TaskView auf den nächsten gezogen werden soll.

Als nächstes folgt die Beschreibung des Formulareditors.

5.4.2 Details des Formulareditors

An dieser Stelle soll der Formulareditor nach seinem Aufbau beschrieben werden.

Der Aufbau des Formulareditors ist sehr ähnlich dem des Workfloweditors. Auch hier wird zwischen Arbeitsfläche und Auswahlliste unterschieden. Da auch hier MVC berücksichtigt werden soll, kommt das Datenmodell hinzu. Abbildung 21 stellt dies dar.

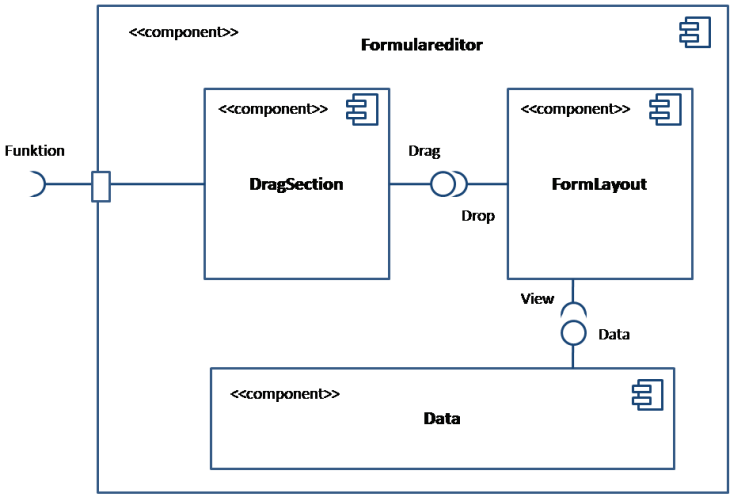


Abbildung 21: Komponenten des Formulareditors

Bei der Trennung zwischen Daten und Darstellung wird beim Formulareditor allerdings zwischen Anzeigeelementen für die Auswahlliste und denen für das Formular unterschieden. Dies dient der Übersichtlichkeit in der Auswahlliste und hat optische Gründe. Die View-Elemente für das Formular werden zusätzlich in der Laufzeitumgebung wiederverwendet, um ein doppeltes Rendering zu vermeiden.

Wie schon erläutert, ist der Formulareditor abhängig vom Workfloweditor. Das liegt unter anderem da ran, dass dieser direkt auf den Workflowdaten des Workfloweditors arbeitet. Zum anderem rührt es aber auch da her, dass beim Start des Formulareditors einige Parameter übergeben werden. Das sind Funktion, Formular und Task. Aus der Funktion kommen grundsätzliche Infos, zum Beispiel die Input-Parameter, welche für die zu verteilenden Elemente entscheidend sind. Anhand des Formulars wird geprüft, ob schon einmal ein Formular für den entsprechenden Task erstellt wurde. Ist dies der Fall, wird mit diesem gearbeitet, ansonsten ist dieser Wert null, weshalb ein neues Formular angelegt wird. Dazu dient dann der übergebene Task, sodass das neue Formular direkt in den Task, zu dessen Eigenschaften ein Formular gehört, eingefügt werden kann.

Wurde das Formular aus den vorgegebenen und optionalen Elementen modelliert, kann es geschlossen werden. Durch das Arbeiten auf den Daten des Workflows werden alle Änderungen direkt übernommen.

Nach der Darstellung der Komponenten von beiden Teilen des IMAP-Editors soll im Folgenden die Laufzeitumgebung kurz beschrieben werden.

5.4.3 Details der Laufzeitumgebung

Die genaue Struktur und die Besonderheiten der Laufzeitumgebung sollen im Folgenden dargestellt werden.

Wie auch beim Editor sind die Funktionalitäten der Laufzeit zwischen Workflows und Formularen getrennt. Die Workflow-Komponente dient der Ablaufsteuerung, also dem Ermitteln des nächsten Tasks, dem Beenden von diesen und zusätzlich dem Aufruf der Formularkomponente an geeigneter Stelle. Das dient der Darstellung der Formulare und beinhaltet auch die Logik zum Ermitteln der Eingaben und das Weiterleiten von diesen an entsprechende Stellen zu den zugehörigen WebServices. Abbildung 22 zeigt die Komponenten.

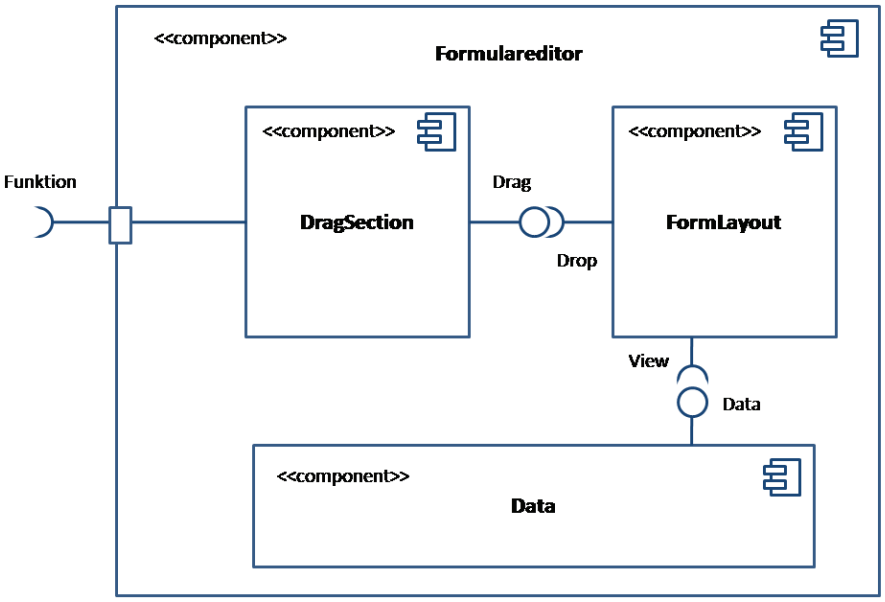


Abbildung 22: Komponenten der Laufzeitumgebung

Im Anhang befindet sich eine grafische Darstellung der genauen Funktionsweise der Engine. Nach dieser Ausführung zur Architektur und dem Aufbau der Applikation soll folgend die Oberflächengestaltung erläutert werden.

5.5 Oberflächengestaltung

Grundsätzlich ist zu der Oberflächengestaltung des IMAP-Editors zu sagen, dass versucht werden soll, diese sehr intuitiv zu halten. Die verwendeten Elemente sollen dabei ansprechend wirken, den Nutzer allerdings auch nicht überfordern. Darum sollen auf der Oberfläche nur die nötigsten Funktionen bereitstehen. Es soll versucht werden, die Bandbreite der von SmartGWT ermöglichten Darstellungsformen adäquat zu nutzen, sodass eine optisch ansprechende Oberfläche entstehen kann, ohne dass diese zu sehr auf unnötige Effekte setzt.

Für den IMAP-Editor ist die Nutzung von Tabs vorgesehen. Das bedeutet, dass beim Start des Editors zuerst das Hauptfenster dargestellt werden soll, welches dem Workflow entspricht.

Wird der Formulareditor gestartet, so wird für das entsprechende Formular ein Tab hinzugefügt. Zwischen diesen kann gewechselt werden, sie können aber auch weggeklickt werden, ohne dass zuvor getätigte Modellierungen verloren gehen.

Damit der gesamte Workflow gespeichert werden kann, ist auf Höhe der Tabstruktur ein Drop-Down-Menü vorgesehen, das grundlegende Funktionen wie Laden und Speichern enthält.

Wie bei der Architektur schon angedeutet, sind sowohl Formular- als auch Workfloweditor zweigeteilt. Auf der linken Seite sollen sich in einklappbaren Sektionen die verschiedenen, per Drag & Drop auf die Arbeitsfläche zu ziehenden Elemente befinden.

Innerhalb dieser Elemente wird beim Formulareditor zwischen optionalen Elementen unterschieden und solchen, die zwingend verteilt werden müssen, damit der Workflow ohne Fehler ausführbar wird. Die notwendigen Elemente sollen nach deren Anlegen aus der Auswahlliste verschwinden.

Grundsätzlich sollen alle angelegten Elemente auf der Arbeitsfläche verschiebbar sein. Beim Workfloweditor ist hierfür ein Raster mit möglichen Punkten vorgesehen, um die Task zu positionieren. Innerhalb des Formulareditors ist jedoch nur eine lineare Verschiebung in der Vertikalen möglich. Das vermeidet eine unnötige Verzerrung innerhalb der Formulare zur Laufzeit, sollte sich die Bildschirmbreite auf unterschiedlichen Endgeräten ändern.

Farblich orientieren sich die Formulare und damit der Formulareditor an der Farbgebung des HTML-Prototypen. Der Workfloweditor erhält einen anderen Farbton, damit eine Abgrenzung intuitiv möglich ist, um ein „Durcheinanderkommen“ innerhalb der Tab-Strukturen zu vermeiden.

Was im Gegensatz zu anderen Editoren nicht verwendet werden soll, ist eine Property Table zur Konfiguration spezifischer Eigenschaften. Da viele Elemente schon durch die Anbindung an den ESB vorgegeben sind, ist sie nicht zwingend notwendig. Um bestimmte Eigenschaften ändern zu können, wie den Text einer Beschriftung, sollen intuitive Metaphern bereitgestellt werden, beispielsweise ein Doppelklick oder ein per Rechtsklick zu öffnendes Kontextmenü. Ein Entwurf der Oberfläche, zusammengestellt mit den Möglichkeiten von SmartGWT, ist in Abbildung 23 zu sehen.

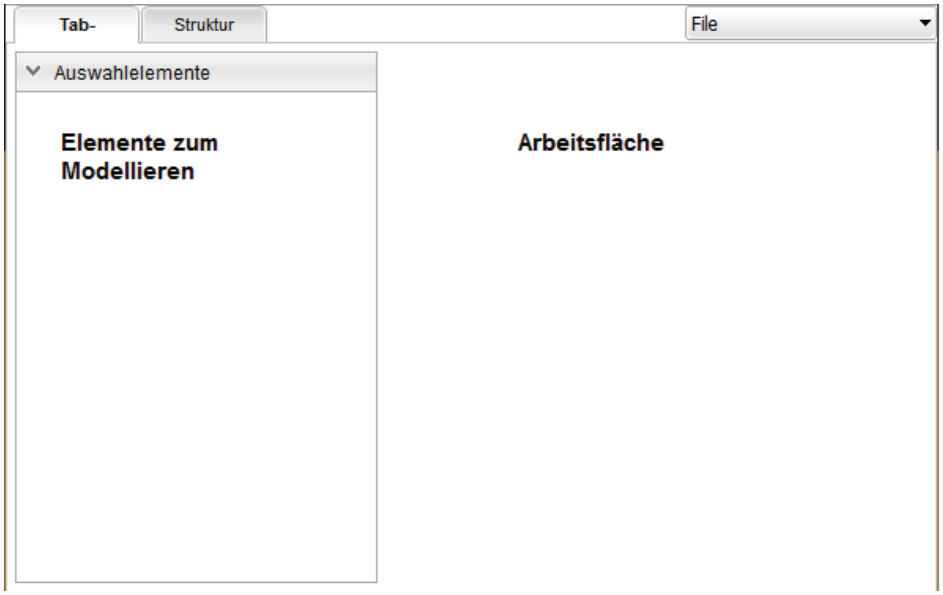


Abbildung 23: Oberflächenentwurf IMAP-Editor

Nachdem das System des IMAP-Editors und der Laufzeitumgebung entworfen wurde, konnte eine prototypische Umsetzung dieser Applikation erfolgen. Das nachfolgende Kapitel erläutert die Umsetzung und damit den entstandenen Editor.

6 Prototypische Umsetzung des IMAP-Editors

Nachdem in den vorherigen Kapiteln Grundlegendes zum IMAP-Editor, dessen Anforderungen und zu theoretischen Grundlagen dargestellt, das System selbst konzeptioniert und anschließend entworfen wurde, soll sich dieses Kapitel mit dem konkreten IMAP-Editor befassen, welcher prototypisch umgesetzt wurde.

6.1 Allgemeines zum entstandenen System

Nach der Entwicklung des Systems von Grund auf, wurde mit dem erarbeiteten Wissen der IMAP-Editors erstellt. Es handelt sich dabei jedoch um einen Prototypen. Dementsprechend sind nicht komplett alle in Kapitel 3.2 dargestellten Anforderungen an das System konsequent und bis ins letzte Detail umgesetzt worden. Vielmehr soll der entstandene IMAP-Editor dazu dienen, aufzuzeigen, wie ein intuitiver Editor zur Erstellung modularer Prozessbeschreibungen nach den zuvor geschilderten Prinzipien aussehen könnte. Zudem soll er als Arbeitsgrundlage für die weiteren Arbeiten am Projekt IMAP dienen. Demzufolge wird die Applikation iterativ um weitere Funktionalitäten erweitert.

6.1.1 Umgesetzte Funktionalität des IMAP-Editors

In dem entstandenen Prototypen wurden die zuvor erläuterten Komponenten umgesetzt. Dabei handelt es sich um den Workfloweditor und den Formulareditor. Zusätzlich ist eine Laufzeitkomponente entstanden, um das Ergebnis der Modellierung, auf welcher der Fokus lag, deutlich zu machen.

Grundsätzlich ermöglicht der IMAP-Editor die Erstellung von Workflows in dem erläuterten, eigenentwickelten Datenformat. Die modellierten Aktivitäten des Workflows können direkt mit Nutzereingaben in Verbindung gebracht werden. Dafür benötigte Formulare können innerhalb des Editors passend zu den entsprechenden Aufgaben erstellt werden.

Für die Modellierung stehen 4 verschiedene Funktionen zur Verfügung, die noch genauer erläutert werden sollen. Die Daten für eine Abbildung dieser Funktionen in einem Workflow werden als Schnittstellen durch den ESB bereitgestellt. Hinter den Schnittstellen verbergen sich real existierende Dienste und WebServices, die durch die Aktivitäten dargestellt werden sollen. Der Editor als Teil des Montageportals ist dementsprechend in der Lage, die benötigten, realen Unternehmensdaten für Modellierung aus den entsprechenden Komponenten zu ermitteln.

Da es sich bei den modellierbaren Funktionen um reale Vorgänge innerhalb des Systems handelt, werden zur Modellierung der zugehörigen Formulare die entsprechend nötigen Elemente anhand der Eingabe-Parameter der Funktionen bereitgestellt.

Der Modellierer benötigt dabei keine Kenntnisse über die anderen Komponenten des Systems.

Neben den aus den Parametern generierten, notwendigen Elementen zur Formulargestaltung wird zudem ein veränderbares Textelement als optionales Modellierungselement zur Verfügung gestellt.

Die so entstehenden Formulare und die zugehörigen Workflows können nach der Modellierung durch die Laufzeitkomponente ausgeführt werden. Dabei werden die Aufgaben anhand ihrer Abfolge abgearbeitet. Komplettiert werden diese durch das Ausfüllen der zu der Aufgabe gehörenden Formulare. Dabei werden die WebServices und Dienste mit den entsprechenden Daten innerhalb des ESB angesprochen.

Im Folgenden sollen die einzelnen Funktionen kurz beschrieben werden.

6.1.2 Modellierbare Aktivitäten in den Workflows

Wie erwähnt, wurden im Prototypen des IMAP-Editors vier verschiedene Funktionen umgesetzt. Dazu gehören unter anderem das An- und Abmelden. Hinter den Eingaben verbirgt sich allerdings keine ausgefeilte Nutzerverwaltung oder gar ein LDAP-Server, sondern ein einfacher WebService zur Demonstration von Textfeldern.

Daneben stehen noch zwei weitere Funktionen bereit. Bei der einen handelt es sich um eine Funktion zur Störungsmeldung. Sie stellt als Input-Parameter zwei Auswahlen bereit, die Störungskategorie und den Störungsgrund mit diversen Möglichkeiten, welche durch eine Auswahl an Buttons dargestellt werden.

Die vierte Funktion dient der Veranschaulichung möglicher Kommunikationswege. Hierbei handelt es sich um eine SMS-Funktion. Sie wird durch einen externen Provider bereitgestellt und ermöglicht das Versenden einer Textnachricht an eine Telefonnummer. Beide werden in Textfeldern erfasst. Diese Funktion läuft nicht über den ESB, als Pfad dient hier die URL des Providers, welche als Parameter Text und Nummer angehängt bekommt. Für die Eingabe des Textes wurden Platzhalter geschaffen. Erfolgt das Versenden nach der Störungsmeldung, so können Grund und Kategorie, automatisch mit übertragen werden.

Das ist der erreichte Funktionsumfang des IMAP-Editors. Bevor er umgesetzt werden konnte, wurde eine Reihe von Anforderungen an den Editor gestellt.

6.1.3 Vergleich des entstandenen Systems mit den Anforderungen

An dieser Stelle sollen die Anforderungen an den IMAP-Editor mit dem erreichten Stand verglichen werden. Die Anforderungen sind jedoch sehr umfangreich. Eine ausführliche Auflistung aller Anforderungen im Vergleich zum entstandenen Editor befindet sich im Anhang. Hier sollen die wichtigsten erläutert werden.

Wird die Auflistung innerhalb des Anhangs betrachtet, so kann festgestellt werden, dass schon eine Vielzahl der Anforderungen an das System umgesetzt wurde. Dennoch sind einige der Anforderungen noch nicht berücksichtig worden.

Bei vielen von diesen handelt es sich jedoch um solche, die innerhalb der nächsten Iteration der Entwicklung den Weg in den IMAP-Editor finden sollen. Näheres zum Ausblick auf die künftige Entwicklung des Editors ist im Kapitel 7 aufgelistet. Um aufzuzeigen, dass der Prototyp des IMAP-Editors dennoch die notwendige Grundfunktionalität erfüllt, sollen an dieser Stelle die Anforderungen betrachtet werden, die auch bei der Auswahl eines Ansatzes zur Umsetzung in Betracht gezogen wurden. Das sind die Erstellung ausführbarer Prozessbeschreibungen, eine intuitive Modellierung bei geringem Funktionsumfang und die Verknüpfung von Prozessen mit Formularen für Nutzereingaben.

Zuerst soll letzteres betrachtet werden. Das Thema der Integration von Prozessbeschreibungen und Formularen zieht sich durch diese gesamte Arbeit und ist als eines der Hauptanliegen des IMAP-Editors zu sehen. Es wurde adäquat umgesetzt, in dem im eigens entwickelten Datenformat die Formulare ein essentieller Bestandteil einer Aktivität innerhalb des Workflows sind. Sie sind dementsprechend direkt in die Prozessbeschreibungen integriert, was das Ziel der Arbeit dargestellt hat. Formulare können mit dem IMAP-Editor auch nur in direktem Zusammenhang mit einem Task erstellt werden.

Wie aus dem vorherigen Abschnitt hervorgeht, ist mit dem IMAP-Editor die Erstellung von Prozessbeschreibungen möglich. Sie sind zudem ausführbar durch die Komponente der Laufzeitumgebung. Diese Anforderung ist dementsprechend auch abgedeckt.

Bezüglich der einfachen, intuitiven Modellierung bei geringem Funktionsumfang ist eine Aussage schon schwieriger. Beim Funktionsumfang der Editoren wurde versucht, diesen so gering wie möglich zu halten. Im Grunde kann bei der Modellierung nicht viel falsch gemacht werden. Nach einer kurzen Einweisung sollte das Modellieren sehr leicht machbar sein.

Jedoch wurde diese Annahme mit dem Wissen erfahrener PC-Nutzer gemacht. Den Analysen der Vorbetrachtungen folgend, bestehen bei vielen der potentiellen Nutzer aber nur geringe Computerkenntnisse. Ob die Modellierung dementsprechend wirklich intuitiv und einfach ist, müssen Evaluationen innerhalb der KMU zeigen.

Grundsätzlich sind jedoch die wichtigsten Anforderungen umgesetzt worden. Jene, die noch nicht in das System eingebaut wurden, sollen in der nächsten Iteration entwickelt werden.

Die potentielle Einfachheit einer Modellierung mit dem IMAP-Editor soll im Folgenden gezeigt werden. Die im System hinterlegten Interaktionsmetaphern sollen anhand eines typischen Benutzungsvorgangs des Editors beschrieben werden.

6.1.1 Interaktionsmetaphern zur Nutzung des IMAP-Editors

Das folgende Kapitel soll die Benutzung des entstandenen Systems aufzeigen und dabei mögliche Interaktionen erläutern.

- Nach dem Start des Systems ist die Auswahl zwischen einem Neuanlegen oder Laden möglich. Ein Klick auf einen der angebotenen Namen entspricht dem Laden. Wird ein neuer Name eingetragen, so wird dieser Workflow neu angelegt.

- Mit „OK“ wird die Auswahl bestätigt und die Arbeitsfläche des Editors wird geladen. Da die Laufzeitumgebung bisher noch nicht mit einer solchen Auswahl ausgestattet ist, sollte der Name des Workflows mit „imap“ angegeben werden.

- Nach erfolgreichem Laden können die Funktionen aus der Leiste links per Drag & Drop auf der Arbeitsfläche verteilt werden.

- Die Aktivitäten sind bei gedrückter Maustaste auf der Fläche anhand eines Rasters verschiebbar.

- Innerhalb der dargestellten Task sind mehrere Interaktionen möglich:

    - Ein Rechtsklick öffnet ein Kontextmenü, in welchem der Task entfernt werden kann.

    - Hinter dem kleinen Icon in Pfeilform verbirgt sich die Möglichkeit, die Aktivitäten untereinander zu verbinden. Wird das Icon ebenfalls per Drag & Drop bewegt, so ist die Mausposition durch ein Linkelement mit dem Ausgangstask verbunden. Findet der Drop auf einem anderen Task statt, so werden diese miteinander verbunden.

Mit diesen erläuterten Interaktionen lassen sich die Workflows modellieren. Neben dieser Modellierung der Aufgabenabfolge ist das Erstellen der zugehörigen Formulare möglich:

- Soll der Formulareditor zu einem Task gestartet werden, genügt ein Doppelklick auf diesen.

- Der Formulareditor bietet ebenfalls eine Liste mit Elementen zum Drag & Drop an.

- Anders als beim Workfloweditor ist eine Bewegung der angelegten Elemente nur an der vertikalen Achse möglich, die Drop-Position wird entsprechend angezeigt.

- Mit den angelegten Formularelementen ist größtenteils keine Interaktion möglich. Die Buttons können zwar geklickt werde, auch ein Schreiben in die Textfelder ist möglich, jedoch findet keine Aktion statt.

- Etwas anders verhält es sich beim Label, also der Beschriftung. Bei einem Doppelklick auf dieses öffnet sich ein Textfeld, in das der Text der Beschriftung eingetragen werden kann.
    - Durch Klick auf das grüne Icon, welches hinter dem Textfeld bereitsteht, wird die Auswahl bestätigt und übernommen.
    - Hinter dem blauen Icon verbirgt sich eine Vorschlagfunktion, welche den Beschreibungstext der durch dieses Formular abgearbeiteten Funktion in das Textfeld einträgt.
- Wurden mit den erläuterten Aktionen der Workflow und alle zugehörigen Formulare modelliert, kann die Arbeit mit der, in der rechten oberen Ecke befindlichen, Menüstruktur gespeichert werden.

Innerhalb der Laufzeitumgebung ist eine nähere Beschreibung der einzelnen Aktionen nicht notwendig. In die Textfelder sind entsprechende Angaben einzutragen, die Buttons reagieren auf einen einfachen Klick. Bei den Störungslisten wird der gewählte Wert durch optische Kennzeichnung deutlich gemacht. Im Allgemeinen sind die Formulare erst dann zu beenden, wenn sie vollständig sind, das heißt, ein Weiterschalten ist erst dann möglich, wenn alle Angaben getätigt wurden.

Da innerhalb des Editors mit den anderen Komponenten des Montageportals kommuniziert und interagiert wird, bestehen für die Nutzung des IMAP-Editors bestimmte Voraussetzungen und Abhängigkeiten. Diese sollen im Folgenden dargestellt werden.

6.1.2 Nutzungsvoraussetzungen und Abhängigkeiten

Der IMAP-Editor war durch die Bereitstellung von Daten und als Teil des Montageportals nie als Stand-Alone-Anwendung konzipiert. Dementsprechend ergeben sich beim Prototypen gewisse Abhängigkeiten.

Der Editor stellt eine Webanwendung dar. Eine Entwicklung dieser Applikation mit dem Framework GWT führt dazu, dass der fertige Editor in einem war-Archiv verpackt ist. Damit der Editor ausführbar wird, muss das Archiv auf einem geeigneten Applikationsserver deployed werden. Im Zuge der Entwicklungen des Projektes wurde dazu ein Apache Tomcat gewählt. Neben dem Editor liegen auf diesem die anderen Komponenten, wie beispielsweise der Semantic Reference Manager. Außerdem werden auf einem Port dieses Servers die Aufrufe der Jersey-Ressourcen entgegen genommen.

Welcher Port das ist, wird unter anderem durch Mule geregelt. Dabei handelt es sich um eine ESB-Implementierung, welche bei IMAP verwendet wird. Hier befinden sich die WebServices, aber auch die Jersey-Ressourcen zur Bereitstellung der Daten.

Wird innerhalb des Systems ein Call zum Ausführen einer Funktion an den Applikationsserver abgesetzt, so kommt er in Mule an. Das Ergebnis der Interaktion wird dabei in der Konsole von Mule dargestellt.

Neben dem Tomcat und Mule wird zudem ein MySQL-Server benötigt, auf dem das IMAP-spezifische Datenbankschema verfügbar ist. Das ist notwendig, damit beispielsweise eine gemeldete Störung im System hinterlegt werden kann.

Sind diese drei Komponenten richtig konfiguriert und gestartet, so sollte auch die Ausführung des IMAP-Editors möglich sein. Die vorgenommene Beschreibung der Umsetzung wurde sehr theoretisch gehalten. Das folgende Kapitel dokumentiert den umgesetzten Stand des IMAP-Editors anhand einiger Screenshots anschaulich.

6.2 Dokumentation des entstandenen Systems

Bisher wurde der erreichte Stand des Prototypen des IMAP-Editors in theoretischer Form betrachtet. In diesem Kapitel soll anschaulich aufgezeigt werden, wie die Oberflächengestaltung des Editors umgesetzt wurde. Dies soll anhand ausgewählter Screenshots geschehen. Im Anhang befindet sich eine ausführliche Bildserie zur Benutzung des IMAP-Editors und der Laufzeit. An dieser Stelle sollen einige Auszüge zur Dokumentation beitragen.

In Abbildung 24 ist die Oberfläche des Workfloweditors dargestellt. Links befindet sich die Auswahlliste der verfügbaren Funktionen. Auf der Arbeitsfläche sind hier einige angelegte Aktivitäten und deren Verbindung zu sehen.

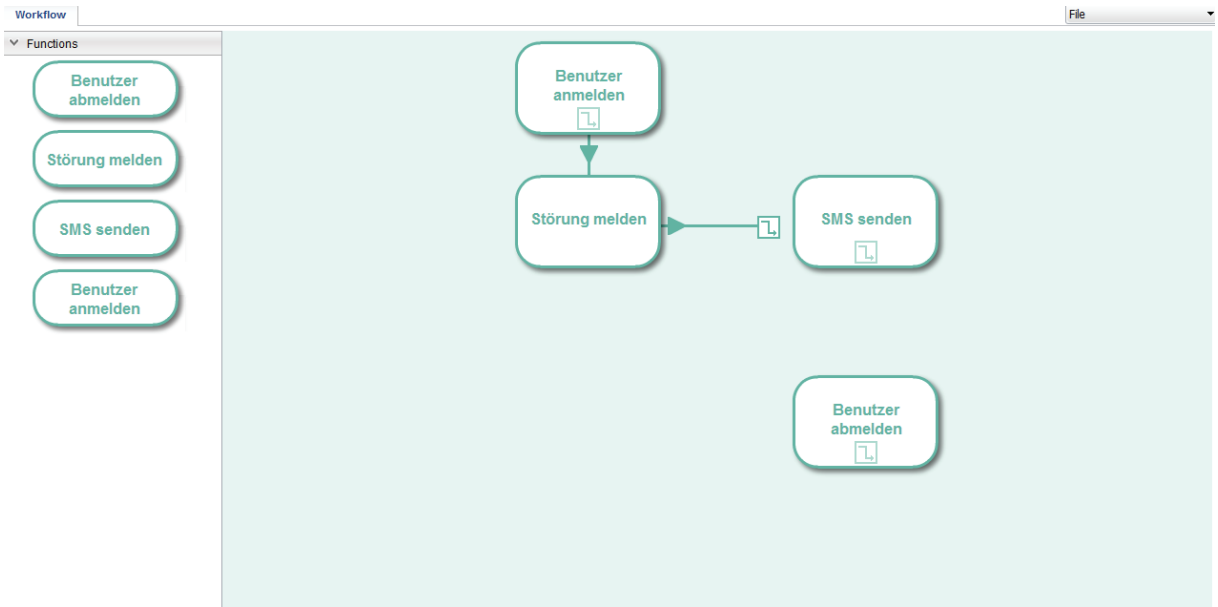


Abbildung 24: Oberfläche des Workfloweditors

Durch einen Doppelklick wird, wie erwähnt, der Formulareditor zu den einzelnen Task gestartet. Dieser ist in Abbildung 25 dargestellt.

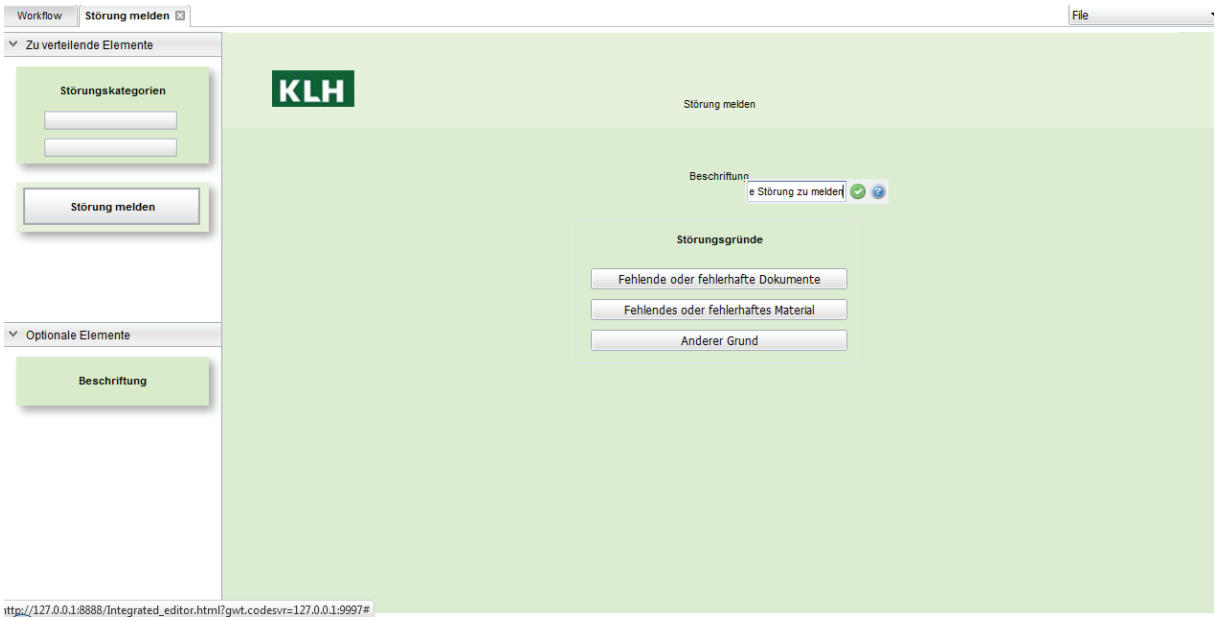


Abbildung 25: Oberfläche des Formulareditors

Auch hier befinden sich auf der linken Seite die verfügbaren Elemente, getrennt nach Notwendigkeit und freier Verwendung. Sie können nach dem Anlegen innerhalb des Formulars frei verschoben werden. Zudem ist hier die Bearbeitung des Labeltextes dargestellt.

Abbildung 26 zeigt einen Workflow während der Ausführung zur Laufzeit in der Laufzeitumgebung. Hierbei wird die Nutzung der SMS-Funktion demonstriert. Dabei wird mit Platzhaltern gearbeitet, welche durch die Werte innerhalb der Engine ersetzt werden.

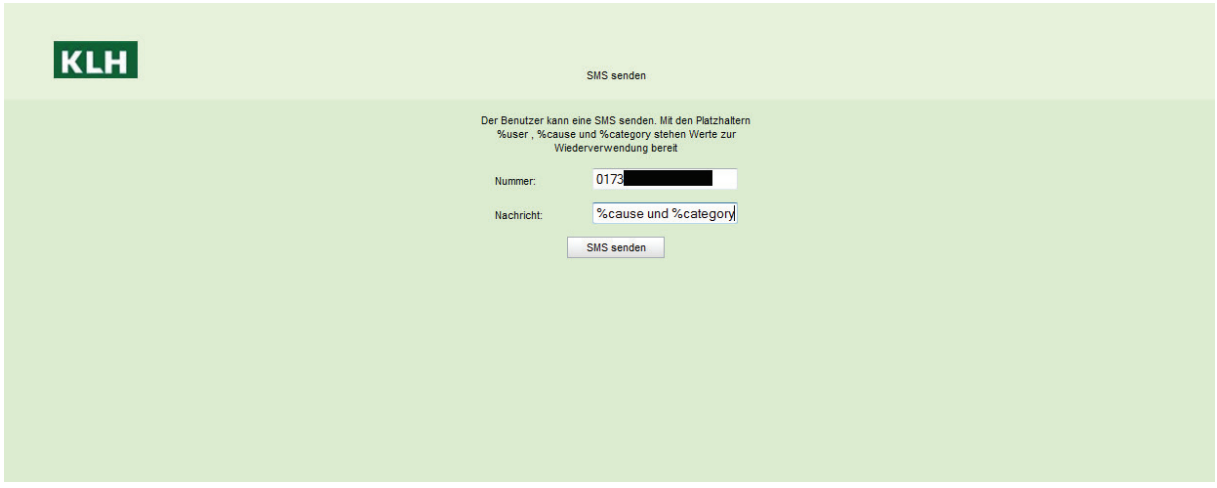


Abbildung 26: Oberfläche Laufzeitumgebung, SMS-Funktion

Das Ergebnis dieser Funktion wird in Abbildung 27 dargestellt. Dabei handelt es sich um die SMS, welche durch die Funktion versendet wird.

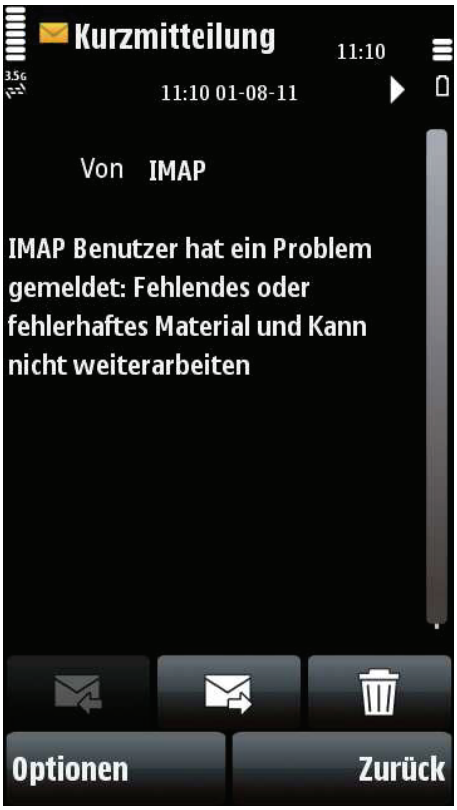


Abbildung 27: Kurzmitteilung, versendet durch SMS-Funktion

Nachdem in diesem Kapitel die Umsetzung der zuvor hergeleiteten Ansätze und Konzepte dargelegt wurde, soll im Folgenden eine Zusammenfassung dieser Arbeit erfolgen. Dabei wird ein Ausblick gegeben, wie die Entwicklung des IMAP-Editors im Projektkontext voranschreiten könnte.

7 Zusammenfassung und Ausblick

Nachdem in den vorhergehenden Kapiteln eingehend alle Grundlagen, Anforderungen und Konzepte bezüglich des IMAP-Editors entwickelt worden sind, wurde er auf der genannten Basis in Form eines Prototypen umgesetzt. Die entstandene Anwendung erhebt in keiner Weise den Anspruch, ein komplett fertiges System darzustellen. Lediglich dient es zur Demonstration der Umsetzbarkeit der gestellten Forschungsfrage auf Grundlage der Arbeiten im Projekt IMAP. Zudem soll der entstandene Editor als Basis für das weitere Vorgehen innerhalb des Projektes genutzt werden.

Das folgende Kapitel dient einem Ausblick auf kommende Arbeiten am Werkzeug für das integrierte Authoring von modularen Prozessdialogen und der Zusammenfassung der Ergebnisse der vorliegenden Arbeit. Da das System bisher nur in einem prototypischen Zustand vorliegt, ergeben sich einige weitere Aufgaben, damit der IMAP-Editor allen Anforderungen des Projektes entspricht.

Im Folgenden wird erläutert, welche Schritte konkret für das weitere Vorgehen innerhalb des Entwicklungsteams des Fraunhofer-IGD geplant sind. Dies soll als kurzfristiger Blick nach vorne dienen. Etwas weiter voraus blicken soll dagegen das Unterkapitel, welches die Möglichkeiten und Grenzen des Editors auf lange Sicht aufzeigt.

7.1 Ausblick

In diesem Kapitel soll ein kleiner Blick in die Zukunft geworfen werden. Da es sich beim IMAP-Editor lediglich um einen Prototypen handelt und dieser, wie im vorangegangenen Kapitel dargestellt wurde, noch nicht alle Anforderungen umsetzen konnte, stellt sich die Frage, wie es mit dem Editor weiter gehen soll. An dieser Stelle soll versucht werden, diese Frage zu beantworten.

Dazu soll zwischen dem kurzfristigen Blick auf die konkret geplanten Arbeiten zur Weiterentwicklung innerhalb des Projektteams und einem längerfristigen Ausblick auf Möglichkeiten und Grenzen des IMAP-Editors unterschieden werden.

7.1.1 Vorgehen innerhalb des Projektes

Da der IMAP-Editor in seiner jetzigen Form die Grundlage für weitere Arbeiten innerhalb des Entwicklungsteams des IMAP-Projektes darstellen soll, bestehen konkrete Vorstellungen, was als nächstes implementiert werden sollte.

Bisher ist nur die Modellierung sehr einfacher Workflows in linearer Abfolge möglich. Dies soll sich in der nächsten Iteration ändern. Das Stichwort hierfür lautet „komplexe Workflows“. Hinter dieser Komplexität verbergen sich mehrere Dinge.

Zum einen beinhaltet das die Möglichkeit, innerhalb der Workflows Verzweigungen und Entscheidungen modellieren zu können. Damit wird die lineare Abfolge gebrochen. Das ist notwendig, da nicht alle Arbeitsläufe innerhalb der KMU strikt linear verlaufen. Zudem erlaubt eine modellierte Entscheidung weitaus mehr Möglichkeiten. Aktivitäten können beispielsweise an bestimmte Bedingungen geknüpft werden.

Da hierdurch allerdings die Komplexität steigt, muss bei der Umsetzung darauf geachtet werden, dass eine Modellierung immer noch einfach und intuitiv bleibt. Diese neuen Möglichkeiten innerhalb der Abfolge von Aktivitäten müssen zudem nicht nur im Editor als solchem modellierbar sein, sondern ebenfalls in der Laufzeitumgebung berücksichtigt werden. Deren Konzept muss in sofern erweitert werden, als dass nicht nur anhand eines In- und Outputs der nächste Task ermittelt wird. Darüberhinaus muss hierbei vielleicht über den Einsatz von separaten Deklarationen für Start und Ende nachgedacht werden.

Ein weiteres Element dieser steigenden Komplexität ist die Modellierung unterschiedlicher Rollen. Hier soll es möglich werden, nicht nur kleine Teile eines Workflows für einen Mitarbeiter zu erstellen, sondern größere Sachverhalte abzudecken, die sich über mehrere Teilnehmer erstrecken.

Hinzu kommt zudem die Umsetzung „komplizierterer“ Elemente, als sie bisher dargestellt wurden. Beispielsweise sieht das Szenario die Anzeige von Stücklisten und Ähnlichem in tabellenartigen Strukturen vor. Solche Strukturen müssen sowohl optisch, als auch im Datenmodell geschaffen werden. Zusätzlich ist hier eine Datenanbindung bereits zum Modellierungszeitpunkt notwendig, sodass Überlegungen notwendig sind, wie bestimmte, dafür zulässige Elemente mit den Daten aus dem ESB verbunden werden können. Das gilt vor allem auch für das dynamische Rendern der Formulare, da sich Stücklisten auch zur Laufzeit ändern können. Bisher war diese Dynamik nur zum Modellierungszeitpunkt gegeben.

All diese Überlegungen beinhaltet der Terminus komplexe Workflows, die zukünftig durch die Bearbeiter umgesetzt werden sollten. Neben diesen Aktivitäten sind noch weitere Schritte geplant, die im Editor verankert werden sollen. Dazu zählt beispielsweise die Modellierung mehrerer Funktionen auf einem Formular. Insbesondere ist hier beispielsweise ein Knopf für den Logout vorgesehen.

Außerdem soll das Konzept der Nutzung von Prozessvariablen verbessert werden. Bisher sind diese nur unter bestimmten Voraussetzungen innerhalb der Laufzeit verwendbar. Es soll jedoch auch während der Modellierung geprüft werden, welche Variablen durch die Reihenfolge der Tasks potentiell vorhanden wären. Dementsprechend muss hier bereits eine Ablaufsteuerung zum Modellierungszeitpunkt zu Previewzwecken stattfinden.

Ein weiteres Arbeitsfeld ist das gemeinschaftliche Datenmodell aller Komponenten. Bisher wurde nur mit Demo-Daten gearbeitet.

Eine große Problematik besteht darin, reale Unternehmensdaten aus gängigen ERP-Systemen in das Datenmodell zu überführen, da aufgrund von Versionskonflikten und unzähligen, weit verbreiteten Tools keine einheitliche Ausgangsituation vorliegt. Dementsprechend muss das Datenmodell sehr generisch angelegt sein, was bisher nicht der Fall war.

Für die Pflege des Datenmodells befindet sich das Tool ESB-Config in der Entwicklung. Sollte sich auf Grundlage der Suche nach einer generischen Lösung der Datenanbindung dass Datenmodell ändern, muss der Editor dementsprechend an die Veränderungen angepasst werden.

Damit sind die Ziele und der konkrete Rahmen für die geplanten Arbeiten am Editor aufgezeigt. Die folgenden Unterkapitel sollen einem weiteren Blick über diesen Plan hinaus ermöglichen und dabei generelle Möglichkeiten und Grenzen des Produktes IMAP-Editor aufzeigen.

7.1.2 Grenzen und Möglichkeiten des Systems

Nach dem im vorherigen Kapitel ein konkreter Ausblick auf die geplanten Arbeiten am IMAP-Editor für die nahe Zukunft gegeben wurde, soll im folgenden Unterkapitel ein etwas weiterer Blick getätigt werden. Dabei soll geprüft werden, wo grundlegende Möglichkeiten und auch Grenzen des Systems liegen könnten.

Grundsätzlich ist dazu festzuhalten, dass mit einem gewissen Zeitaufwand sehr vieles möglich wäre. Dabei ist jedoch zu beachten, dass hierbei, genau wie es schon an anderen Stellen erläutert wurde, immer der Aufwand gegenüber dem Nutzen abzuwägen ist. Das gilt vor allem für einen angebrachten Zeitrahmen. Ist die Zeit jedoch gegeben, so ist bei der Implementierung neuer Funktionalitäten vieles möglich.

Da das System iterativ entwickelt wird, kann Stück für Stück geprüft werden, inwiefern bestimmte geplante Schritte in der vorgegebenen Zeit realisierbar sind. Der Editor kann so nach und nach wachsen. Eine Anforderung nach der anderen kann umgesetzt werden.

Dabei darf jedoch nicht aus den Augen verloren werden, dass neue Schritte und Funktionen gut durchdachte Konzepte voraussetzen. Sind diese stimmig entwickelt und zeitlich umsetzbar, so sind dem Funktionsumfang des Editors kaum Grenzen gesetzt.

Das Ziel sollte dabei sein, weg von einem Prototypen, der sich an bestimmten Szenarien innerhalb des Gesamtkonzeptes orientiert, hin zu einem realen System, welches mit Hilfe von Echtdaten der KMU nicht nur für die Darstellung und Ausführung von Workflows verantwortlich ist, sondern effektiv zur Optimierung der Prozesse innerhalb der KMU sorgt. Das umfasst dann Aspekte, welche innerhalb der Untersuchungen zum Montageportal erläutert wurden, also genaueste Zeitnahmen und Kontrollen der Materialflüsse. Um diese darstellen zu können, wird dann wieder der IMAP-Editor verwendet.

Weitere denkbare Aspekte der Möglichkeiten des Systems sind relativ profane Dinge, die bisher nicht berücksichtigt wurden. Dazu könnten beispielsweise Exportfunktionen in standardisierte Notationsformen gehören. Die Modellierung erfolgt im IMAP-Format, auf Knopfdruck wird daraus dann BPMN generiert. Auch der Corporate Design- Gedanke könnte in diese Kategorie fallen. Hier würden nicht nur die Prozesse eines Unternehmens mit Hilfe des IMAP-Editors dargestellt werden, auch der Editor als solcher würde komplett an die Wunsch-Optik der KMU anpassbar sein.

Dies sind nur einige der potentiellen Ideen, wie der IMAP-Editor in Zukunft entwickelt werden könnte. Dennoch gibt es einige Grenzen und Hindernisse, die berücksichtigt werden müssen, welche im Folgenden dargestellt werden sollen.

Grundsätzlich sollte bei allen Plänen der zeitliche Aspekt und der benötigte Aufwand abgeschätzt werden. Sind diese in einem entsprechenden Rahmen, ist es möglich, bestimmte Funktionen zu implementieren. Doch gerade die Zeit stellt hier ein Problem dar. Das Projekt IMAP läuft noch etwa ein Jahr. Das ist für eine Entwicklung nicht sehr lange. Es muss dementsprechend überlegt werden, welche Funktionalitäten höher zu priorisieren sind, sodass die wichtigsten auf Kosten einiger eher unwichtiger Aspekte den Weg in das System finden.

Bei allen Arbeiten ist zu berücksichtigen, dass die Einfachheit in der Benutzung die höchste Priorität beansprucht. Momentan sind das Datenformat und die Modellierung sehr einfach und intuitiv gehalten. Die Komplexität der Applikation und auch der Notationsform ist im Vergleich zu beispielsweise BPMN sehr gering.

Wenn jedoch neue Funktionalitäten hinzu kommen und der IMAP-Editor immer weiter entwickelt wird, ist es bisher nicht absehbar, ob es möglich bleiben wird, die Einfachheit und geringe Komplexität beibehalten zu können. Ob bei Projektende das Datenformat immer noch weitestgehend einfacher ist, als beispielsweise andere Notationsformen, kann zu diesem Zeitpunkt nicht garantiert werden. Um alle Funktionen umzusetzen, die durch die Anforderungen vorgegeben werden, kann es sein, dass die Einfachheit hinter der reinen Funktionalität zurückstecken muss, was eigentlich vermieden werden sollte.

Ein weiterer wichtiger Aspekt bei der Betrachtung von Grenzen und Hindernissen ist die Datenanbindung. Wie schon erwähnt, stellt die Überführung der Unternehmensdaten in den ESB eines der elementaren Probleme des IMAP-Portals dar. Auch für die Datenbereitstellung innerhalb des IMAP-Editors müssen schlüssige Konzepte entwickelt werden, sodass die durch den ESB bereitgestellten ERP-Daten zur Modellierung und Laufzeit dynamisch verwendet werden können.

Was bisher gar nicht berücksichtigt wurde, ist die mobile Nutzung. Beispielsweise stehen dabei nicht immer die Verbindungen zu den Server-Komponenten bereit. Auch hier sind schlüssige Konzepte nötig, deren Entwicklung erneut einiges an zeitlichem Aufwand mit sich bringt. Diese Zeit könnte dann wieder an anderer Stelle fehlen.

Von elementarem Interesse ist jedoch ein ganz anderes Themengebiet, welches am ehesten die Grenze des IMAP-Editors darstellt. Dabei handelt es sich um die Akzeptanz der erstellten Produkte innerhalb der KMU. Alle Konzepte, Umsetzungen und Ideen mögen für ein Entwicklungsteam schlüssig und gut klingen. Wenn die dabei entstehenden Produkte allerdings, bei den eigentlichen „Kunden“ auf Ablehnung stoßen, nutzt auch das beste System nichts. Dementsprechend ist es von essentieller Bedeutung, dass an entsprechender Stelle die Ergebnisse der Arbeiten des Projektes vor Ort und im Praxistest bewertet werden, sodass gewährleistet ist, dass die entstehenden Produkte auch Verwendung finden. Das beste System nutzt nichts, wenn niemand damit arbeiten möchte.

Eine weitere Grenze stellt die Übertragbarkeit des IMAP-Konzeptes auf andere KMU als die Projektpartner dar. Das entstehende System soll möglichst generisch angelegt sein, sodass es auch für projektfremde KMU einsetzbar ist.

Da das Portal jedoch auf Grundlage der Strukturen innerhalb der Partnerunternehmen entwickelt wird, kann nicht mit Sicherheit gesagt werden, ob das Portal auch auf andere Unternehmen übertragbar ist.

Grundsätzlich soll vermieden werden, dass es sich bei dem Montageportal um eine bessere Individualsoftware handelt. Dementsprechend werden durch diese Tatsache die Möglichkeiten der Entwicklung weiter eingegrenzt. Letztendlich müssen alle Entwicklungen so veranlagt sein, dass sie potentiell übertragbar bleiben.

Nachdem ausgehend von der fertigen, prototypischen Umsetzung des IMAP-Editors, welche im Zuge dieser Arbeit entwickelt wurde, ein kleiner Ausblick in die Zukunft gegeben wurde, soll im Folgenden die vorliegende Arbeit zusammengefasst und ein Fazit gezogen werden.

7.2 Zusammenfassung

Um im Zuge des Projektes IMAP einen Teil zur Optimierung der Geschäftsprozesse von kleinen und mittelständischen Unternehmen des produzierenden Gewerbes beizutragen, soll mit Hilfe automatisierter Prozesse und in Verbindung mit Nutzerinteraktionen zur Laufzeit dieser Abläufe eine Basis für die Verbesserung der unternehmensinternen Abläufe innerhalb der KMU geschaffen werden. Dazu werden praxistaugliche Werkzeuge für die Modellierung der Prozesse und Prozessdialoge benötigt, die einfach und intuitiv zu bedienen sind.

Diese Arbeit hat sich mit der Frage nach solchen Werkzeugen beschäftigt. Es sollte nach Möglichkeiten für eine integrierte Modellierung ohne Fachkenntnisse gesucht werden, um automatisierte Geschäftsprozesse und zugehörige Interaktionen während der Ausführung abbilden zu können. Dabei sollte vor allem die Integration der Eingaben mit den eigentlichen Prozessdarstellungen berücksichtigt werden, ebenso wie die Anbindung an real existierende Infrastrukturen der Unternehmen. Auf Grundlage dieser Untersuchungen sollte ein prototypisches Werkzeug für die Modellierung entworfen und umgesetzt werden.

Ermöglicht werden sollte die Beantwortung der Forschungsfrage durch die Betrachtung unterschiedlichster Anforderungen und theoretischer Grundlagen. Hinzu kamen die Konzeption und der Entwurf eines solchen Systems.

Entstanden ist ein Werkzeug, der IMAP-Editor, welcher es ermöglicht, Geschäftsprozesse zu modellieren. Die Prozesse basieren auf einer eigenen, sehr einfachen Notations- und Darstellungsform. Innerhalb dieser sind direkt Formulareingaben verankert, die, kontextbezogen zu den einzelnen Aufgaben zugeordnet, in demselben Werkzeug modelliert werden können. Die für die Modellierung notwendigen Elemente werden dabei detailgetreu durch andere Komponenten bereitgestellt, welche im Zuge von IMAP entwickelt wurden und ebenfalls einen Teil des IMAP-Montageportals darstellen, genau, wie es der Editor tut.

Diese Daten werden im Zuge der weiteren Entwicklungen direkt auf den real vorhandenen Planungsdaten der KMU basieren, sodass die realen Unternehmensinfrastrukturen berücksichtigt werden.

Die Modellierung ist mit vorkonfigurierten, auf realen Funktionen basierenden Elementen so angelegt, dass nicht viel falsch gemacht werden kann, was eine einfache und intuitive Darstellung ermöglichen soll.

Zudem sind, wie erwähnt, die Nutzereingaben konkret mit gewissen Aufgaben verknüpft. Neben der Modellierung wurde zudem eine Laufzeitumgebung geschaffen, die es ermöglicht, die dargestellten Prozessdefinitionen auszuführen, wobei an entsprechender Stelle die erstellten Formulare aufgerufen und bearbeitet werden können. Dadurch wird eine Kommunikation mit den anderen Komponenten des Systems zur Ausführungszeit möglich.

Die Grundlage für eine Optimierung der Geschäftsprozesse ist also gegeben. Da ein Prototyp entstanden ist, kann das System in folgenden Weiterentwicklungen so erweitert werden, dass daraus konkrete Nutzen innerhalb der KMU gezogen werden können.

Während der Betrachtungen innerhalb dieser Arbeit und bei der Entwicklung des Prototypen lies sich absehen, dass im Grunde sehr viel von dem möglich ist, was in den Arbeitsbereichen des Projektes IMAP für die Optimierung der Abläufe innerhalb der KMU konzeptioniert wurde. Die Umsetzung des Prototypen hat allerdings auch gezeigt, dass dabei die Zeit einen wesentlichen Faktor darstellt. Bei einer Weiterentwicklung der entstandenen Konzepte muss dementsprechend darauf geachtet werden, dass eine Umsetzung in angemessenen Zeitrahmen möglich ist, um letztendlich einen effektiven Nutzen aus dem System für die Optimierung ziehen zu können. Auch sollte darauf geachtet werden, dass die konkreten Konzepte nicht nur für ein einziges Partnerunternehmen stimmig sind, sondern sich auch auf andere Unternehmen der gleichen Sparte übertragen lassen.

Grundsätzlich hat diese Arbeit jedoch gezeigt, dass die Möglichkeiten zur integrierten Darstellung der Geschäftsprozesse mit verknüpften Formulareingaben auf Grundlage realer Unternehmensstrukturen, welche das Untersuchungsgebiet dieser Ausführungen dargestellt haben, vorhanden sind und es möglich ist, sie in konkreten Anwendungen sichtbar zu machen. Dies hat die Entwicklung des IMAP-Editors auf konzeptioneller und prototypischer Ebene deutlich gemacht.

Quellenverzeichnis

Literaturquellen

[BeK03]

Becker, Jörg /Kahn, Dieter:
Der Prozess im Fokus in: Becker, Jörg/ Kugeler, Martin/ Rosemann, Michael (Hrsg.): Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung, 4. Auflage, Springer-Verlag, Berlin, Heidelberg, New York, 2003.

[BeVo96]

Becker, Jörg/Vossen, Gottfried: Geschäftsprozeßmodellierung und Workflow-Management: Eine Einführung in: Becker, Jörg/Vossen, Gottfried (Hrsg.): Geschäftsprozeßmodellierung und Workflowmanagement. Modelle, Methoden, Werkzeuge, 1. Auflage, International Thomson Publishing, Albany, Bonn, 1996.

[DRS03]

Rosemann, Michael/Schwegmann, Ansgar/Delfmann, Patrick:
Vorbereitung der Prozessmodellierung in: Becker, Jörg/ Kugeler, Martin/ Rosemann, Michael (Hrsg.): Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung, 4. Auflage, Springer-Verlag, Berlin, Heidelberg, New York, 2003.

[Gad08]

Gadatsch, Andreas: Grundkurs Geschäftsprozessmanagement. Methoden und Werkzeuge für die IT-Praxis: Eine Einführung für Studenten und Praktiker, 5. Auflage, Friedr. Vieweg & Sohn Verlag, Wiesbaden, 2008.

[HMS11]

Herzig, Claudia/Musielak, Marleen/Schmidt, Katharina:
AP420-E1: INTEGRATIONSKONZEPT/ AP420-E3: DESIGNSTUDIE MONTAGEPORTAL. Ergebnisse im Projekt IMAP aus dem AP 400 Integriertes Montageportal, Fraunhofer-Institut für Graphische Datenverarbeitung, interner Bericht, Rostock, 2011.

[HzM03]

Zur Mühlen, Michael/ Hansmann, Holger:
Workflowmanagement in: Becker, Jörg/ Kugeler, Martin/ Rosemann, Michael (Hrsg.): Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung,
4. Auflage, Springer-Verlag, Berlin, Heidelberg, New York, 2003.

[JTB10]

Freund, Jakob/Henninger, Thomas/Rücker, Bernd:
Praxishandbuch BPMN, 1. Auflage, Carl Hanser Verlag, München, Wien, 2010.

[SeS08]

Schmelzer, Herman J./Sesselmann, Wolfgang: Geschäftsprozessmanagement in der Praxis. Kunden zufrieden stellen – Produktivität steigern – Wert erhöhen, 6. Auflage,
Carl Hanser Verlag, München, 2008.

[SpS03]

Schnetgöke, Norbert/ Speck, Mario:
Sollmodellierung und Prozessoptimierung in: Becker, Jörg/ Kugeler, Martin/ Rosemann, Michael (Hrsg.): Prozessmanagement. Ein Leitfaden zur prozessorientierten Organisationsgestaltung, 4. Auflage, Springer-Verlag, Berlin, Heidelberg, New York, 2003.

Internetquellen

Letzter Zugriff in eckigen Klammern.

[ACT 01]

Webseite der BPM-Plattform Activiti,
URL: http://activiti.org/index.html
[15.08.2011]

[ACT 02]

Webseite der BPM-Plattform Activiti. Bestandteile der Software,
URL: http://activiti.org/components.html
[15.08.2011]

[ESB 01]

Demolsky, Markus:
Enterprise Service Bus. Die Konzepte (April 2008),
URL: http://it-republik.de/jaxenter/artikel/Enterprise-Service-Bus-1662.html
[15.08.2011]

[GWT 01]

Entwicklungsseite zum Google Web Toolkit,
URL: http://code.google.com/intl/de-DE/webtoolkit/overview.html
[15.08.2011]

[IMAP 01]

Aehnelt, Mario:
Webpräsenz Projekt IMAP (14.12.2010), URL: http://projekt-imap.de/projekt/
[15.08.2011]

[IMAP 02]

Aehnelt, Mario:
Webpräsenz Projekt IMAP. Knowledge Engineering (14.12.2010), URL: http://projekt-imap.de/projekt/knowledge-engineering/
[15.08.2011]

[IMAP 03]

Aehnelt, Mario:
Webpräsenz Projekt IMAP. Integriertes Montageportal (14.12.2010),
URL: http://projekt-imap.de/projekt/integriertes-montageportal/
[15.08.2011]

[MG 01]

Götz, Manuel:
BPMN 2.0 Tutorial. Kompakte Einführung in die BPMN 2.0,
URL:
http://www.itransparent.de/sites/default/files/BPMN_2_0_Tutorial_Business_Process_Modeling_Notation_Deutsch.pdf
[15.08.2011]

[Rest 01]

Ullenboom, Christian:
Jersey-Tutorial: REST, JAX-RS, RESTful Web-Services mit Java (27.07.2010), URL: http://www.tutego.de/blog/javainsel/2010/07/jersey-tutorial-rest-jax-rs-restful-web-services-mit-java/
[15.08.2011]

[Smart 01]

Entwicklungsseite zu SmartGWT,
URL: http://code.google.com/p/smartgwt/
[15.08.2011]

[WI Enz 01]

Becker, Jörg:
Geschäftsprozessmodellierung (23.09.2008),
URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/Systementwicklung/Hauptaktivitaten-der-Systementwicklung/Problemanalyse-/Geschaftsprozessmodellierung
[15.08..2011]

[WI Enz 02]

Vom Brocke, Jan:
Prozessmanagement (26.09.2008),
URL: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/Informationsmanagement/Informationsmanagement--Aufgaben-des/Prozessmanagement/index.html
[15.08.2011]

Abbildungsverzeichnis

Anhang

Anhang I – Vergleich Modellierungsansätze für Integration

Ansatz	Voraussetzung	Implementierungsbedarf	Aufwand Implementierung	Nutzung	Aufwand Nutzung	Mögl. Systeme	Urteil
A	WFE nach Anforderungen (externer) FE nach Anforderungen Verbindung (FE aus WFE starten)	Anpassung WFE Anpassung FE Externer Aufruf Integration FE in WFE Kontextabhängigkeit der Formulare	Abhängig von gewählten Editoren Integration und Aufruf je nach Ausgangslage Aufwand mittel	WFE starten FE starten Modellieren Speichern	Gering für Nutzer Grundfunktionen ohne Mehraufwand	Fast jeder WFE Systeme wie BOS bevorzugt, die Integration schon haben	Nutzung gut Impl. Mittel Verwendung bestehender Systeme möglich gut
B	WFE nach Anforderungen FE nach Anforderung Mapping-Tool o.Ä.	Anpassung WFE Anpassung FE Extra-Tool muss entwickelt werden Benötigt gutes Konzept, um z.B. P-Var zu berücksichtigen	Anpassung Neuentwicklung Tool inkl. Konzept Sehr hoher Entwicklungsaufwand	WFE Modellieren FE Modellieren Extra Tool benutzen Manuelles Verknüpfen (konzeptabhängig)	Hoch durch Tool Unnötig kompliziert (z.B. P-Var setzen)	jeglicher WFE und FE Tool komplette Eigenentwicklung	Hohe Aufwände Ansatz nicht gut
C	WFE nach Anforderungen FE nach Anforderung Analysetool Vorstufe zu A, falls Integration in WFE nicht möglich	Anpassung WFE Anpassung FE -> kontextabhängiger Aufruf Analysetool entwickeln	Sehr aufwändig, aber leichter als B P-Var besser berücksichtigt Analyse je Format, Ansätze verfügbar	WFE starten Analysetool starten FE starten Modellieren speichern	1 Tool mehr starten, aber keine Gedanken an Verknüpfung nötig Mittlerer Nutzungsaufwand	Alle WFE Vermutlich FE Eigenentwicklung oder starke Anpassung Ana-Tool eigenes	Mittlere bis hohe Aufwände, aber guter Kompromiss, falls A nicht sofort umsetzbar Vorstufe
D	Ein Werkzeug mit beiden Funktionen	Ein Tool anpassen Oder Eigenentwicklung	Nur einmal in Code einarbeiten Gering Bei Eigenentwicklung hoch	Ein Werkzeug starten Modellieren Speichern	Sehr gering Sowohl bei Übernehmen als auch eigenem Wenige Grundfunktionen	BOS, aber zu Aufwändig in Anpassung Eigenes, da so kompakt	Guter Ansatz weil kompakt Geringer Nutzungsaufwand vermutlich hoher Entwicklungsaufwand bei eigenem

Anhang II – Vergleich Integrationsansätze

Ansatz	Beschreibung	Voraus-setzung	Implemen-tierungsaufwand	Systeme	Bemerkung	Urteil
1	Verlinkung der Formulare, Einklinken in den Workflow 2 verknüpfte Formate	WF-Format muss einbinden unter-stützen (z.B. durch XML) Engine muss ausführen können	Format muss verändert oder neu erfunden werden Engine muss neuentwickelt oder verändert werden Wenn System vorhanden, dann kaum	Activiti unterstützt Einbindung von Formularen und bringt Engine mit Bei Eigen-entwicklung alles möglich	Von keinem Standard unterstützt, dennoch machbar nicht mehr konform Export schwer	Wenn ent-sprechendes System nutzbar, sehr gut, ansonsten viele Ver-änderungen Note 2
2	Formulare stellen einen direkten Teil des Workflows dar 1 Format	Komplett neues Format Engine wird benötigt	Neues Format muss erstellt werden Komplette Engine neu zu entwickeln	Keine Systeme vorhanden (deshalb Arbeit überhaupt) Eigen-entwicklung	Optimale Integration, da Formulare bestmöglich angebunden	Kein System vorhanden, dadurch viel Arbeit, aber optimale Integration Note 2
3	2 Formate, Verknüpfungs-Informationen in Konfigurations-Dateien hinterlegt	Betrifft nur die Engine, nicht die Editoren Engine muss ent-sprechend bearbeitet werden oder extra Tool direkt vor Ausführung	Braucht gute Konzeption für die Verknüpfung Benötigt Bekanntheit möglicher Variablen zur Modellierungszeit -> Testumgebung vll. nötig Vermutlich eigene Engine Aufwand nicht absehbar	Keine Systeme vorhanden (deshalb Arbeit überhaupt) Eigen-entwicklung	Könnte Probleme mit P-Var machen Unnötiger Aufwand Mehr Notlösung Wenig elegant Viele Abhängigkeiten	Aufwand nicht abschätzbar, Integration nicht optimal Note 3

Anhang III – Vergleich kombinierte Integrationsansätze

Ansatz	Aufw. Impl. Modell	Aufw. Impl. Integr.	Aufw. Nutz.	System Modell	System Integr.	Urteil Modell	Urteil Integr.	Bemerkung	Gesamt-urteil
A1	Mittel	Hoch	Gering	An-passung mögl.	Activiti	Gut	gut	Activiti macht Nutzung einfach, Aufwand bei FE in WFE	Am besten
A2	Mittel	Sehr hoch	Gering	An-passung mögl.	eigenes	Gut	Opti-mal	Eigenes Format in bestehendes System = schlecht, komplette Eigenentw. kann auch gleich alles in einem	Geht besser
A3	Mittel	Am höch-sten	Gering	Anpassun g mögl.	kein	Gut	mittel	Sollte wegen Int. nicht	Vermeiden
B1	Sehr hoch	Hoch	Hoch	Anp. Plus extra Tool	Activiti	Schlecht	Gut	Zu hoher Aufwand bei Mod	zu kompliziert
B2	Sehr hoch	Sehr hoch	Hoch	Anp. Plus extra Tool	Eigenes	Schlecht	Opti-mal	2 Edit. Bei einem Format nicht gut	Zu kompliziert
B3	Sehr hoch	Am höch-sten	Hoch	Anp. Plus extra Tool	kein	schlecht	Mittel	Sollte wegen Int. Und Mod nicht	Schlecht
C1	Hoch	Hoch	Mittel	Anp. Plus extra Tool	Activiti	mittelgut	Gut	Gut mit Activiti, als Vorstufe zu A1 denkbar	Gute alternative zu A1
C2	Hoch	Sehr hoch	Mittel	Anp. Plus extra Tool	Eigenes	mittelgut	Opti-mal	Eigenes Format und bestehende Systeme	Geht besser
C3	Hoch	Am höch-sten	Mittel	Anp. Plus extra Tool	kein	mittelgut	Mittel	Sollte wegen Int. nicht	vermeiden
D1	Gering oder hoch	Hoch	Sehr gering	BOS, eigenes	Activiti	Gut	Gut	Wenn komb. Editor nicht von Haus aus, dann sehr schwierig (z.B. FE komplett in Activiti bauen)	Geht besser
D2	Gering oder hoch	Sehr hoch	Sehr gering	BOS, eigenes	Eigenes	gut	Opti-mal	Komplette Eigenentwicklung, bringt alle Vorteile, aber auch viel Arbeit	In Betracht ziehen

Anhang IV – Anforderungen Prozesseditor und Umsetzung durch jeweiligen Ansatz

Nach [HMS11]

	Activiti		SmartGWT	
Anforderung	**erfüllt**	**Bemerkung**	**erfüllt**	**Bemerkung**
Grundbedingungen:				
Modellierung von Prozessdefinitionen	Ja		Ja	
Teilschritte, Reihenfolge, zugehörige Interaktionsformulare	Ja, Ja, Nein	Formulare auf Grundlage der WF, aber nicht im WFE	Ja, Ja, Ja.	
Prozess aus dem Blickwinkel des Mitarbeiters	z.T.	Je nach Modellierung in BPMN	Ja	Funktionen vorgegeben, deshalb möglich
Modellierbare Elemente:				
Aktivitäten	Ja		Ja	
Formular	Nein		Ja	
Prozessvariablen	Nein		Ja	Platzhalter
Sequenzflüsse	Ja		Ja	
Entscheidungen	Ja		Nein	
Teil- oder Unterprozesse	Ja		Nein	Iterativ
Verzweigungen (o)	Ja		Nein	Iterativ
Warte-Elemente (o)	Ja		Nein	
Start und Ende (o)	Ja		Nein	Anfangs nicht notwendig
Allgemeines:				
WYSIWYG	Ja		Ja	
nahezu selbsterklärendes Format	z.T.	BPMN-Kenntnisse, Komplexität	Ja	
Modellierung sehr intuitiv	Nein	BPMN-Kenntnisse, Modeller sehr kompliziert	Ja	
Geringer Funktionsumfang	nein	Modeller sehr umfangreich	Ja	
kleine Menge an unterschiedlichen Symbolen und Icons	nein	Sehr komplex	Ja	
Aufbau der Prozessbeschreibungen modular	ja		Nein	Keine Teilprozesse
Copy und Paste	Nein		Nein	
ausführbar	z.T.	Wenn richtig modelliert und aufgearbeitet	Ja	
Notationsform intuitiv und leicht erlernbar	z.T.	BPMN komplex, aber auch einfach modellierbar	ja	
Anpassung	Ja		Nein	
Eigenentwicklung	z.T.	Laufzeit mit Parser, FE	Ja	
günstige Lizenzbedingungen	ja	Open Source	Ja	Keine

Anhang V – Anforderungen Formulareditor und Umsetzung durch jeweiligen Ansatz

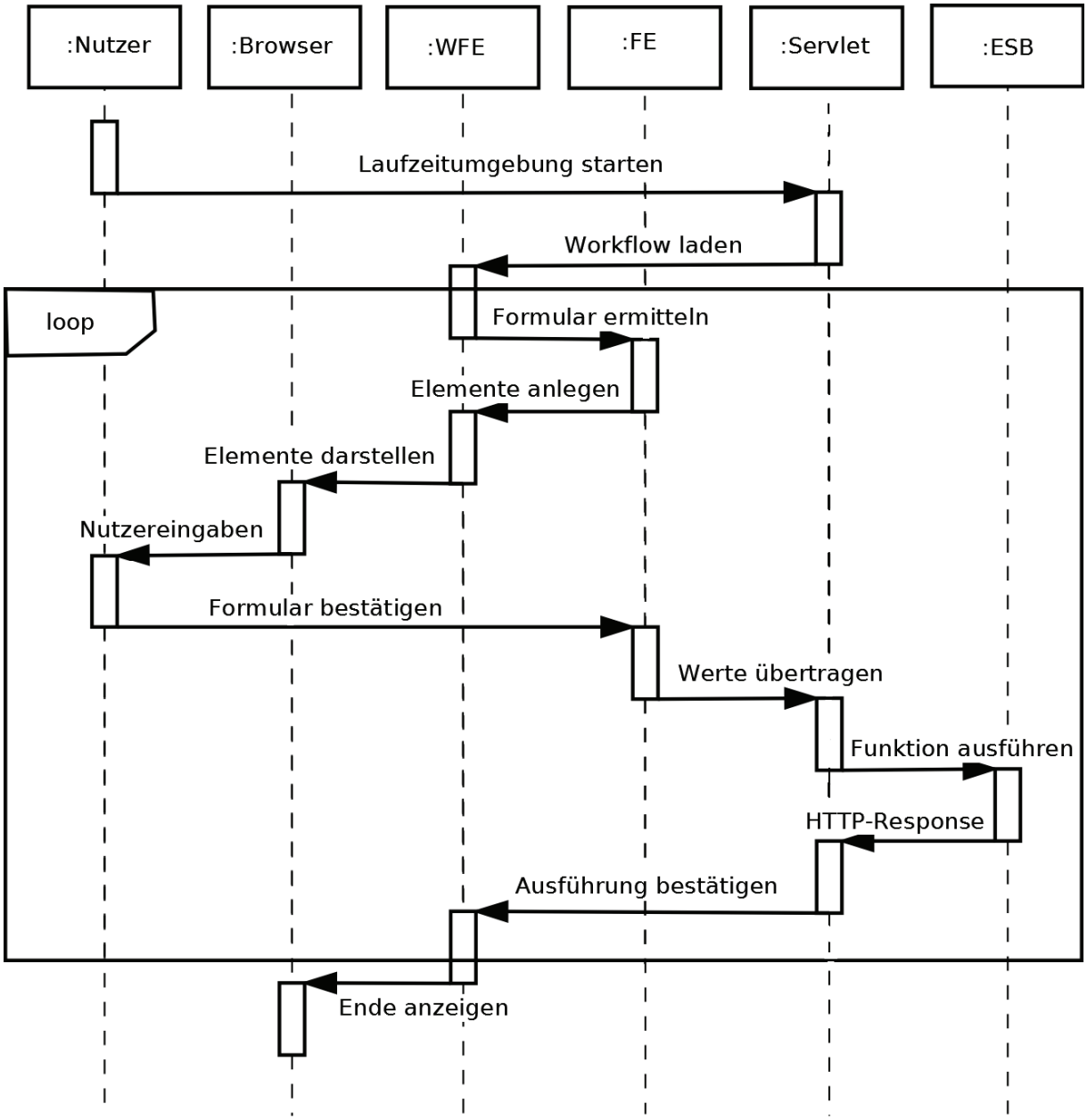
Nach [HMS11]

	Activiti		SmartGWT	
Anforderung	**erfüllt**	**Bemerkung**	**erfüllt**	**Bemerkung**
Erstellung benutzerfreundlicher Formulare	Ja		Ja	
kontextabhängig	Ja	Auf Grundlage der Task	Ja	
Entsprechende Task bekannt	Ja		Ja	
Prozessvariablen bekannt	z.T.	Könnte übergeben werden	z.T.	Durch Description der Funktion
bekannte Steuerelemente	Ja		Ja	
selbstentworfene Elemente	Ja		Ja	
dynamische Erzeugung der Formularinhalte	z.T.	Elemente dynamisch, aber nur während Modellierung	Ja	z.B. Username
spezifische Eigenschaften	Ja	Property Pane	Nein	Nicht notwendig
WYSIWYG	z.T.	„Doppeltes Rendering“	Ja	
Datenbasis, Kommunikation	Ja	Zur Laufzeit	Ja	
Templates	z.T.	vorgesehen	Nein	
Formulare modular	z.T.		Ja	Verschiebbare Elemente
definierten Blöcken	Nein		Nein	
speziellen Eigenschaften vordefiniert	z.T.	Daten aus ESB	z.T.	Daten aus ESB
Copy und Paste	Nein		Nein	
plattformunabhängig	Ja		ja	Und Browserunabhängig
Laufzeit mobil	nein		ja	

Anhang VI – Vergleich Anforderungen mit entstandenem System

Anforderung	Umgesetzt	Bemerkung
Prozesseditor:		
Modellierung von Prozessdefinitionen	Ja	
Teilschritte, Reihenfolge, zugehörige Interaktionsformulare	Ja Ja Ja	
Prozess aus dem Blickwinkel des Mitarbeiters	Ja	
Elemente:		
Aktivitäten	Ja	
Formular	Ja	
Prozessvariablen	z.T.	Innerhalb der Laufzeit, Konzept zu verbessern
Sequenzflüsse	Ja	
Entscheidungen	Nein	Nächste Iteration
Teil- oder Unterprozesse	Nein	
Verzweigungen (o)	Nein	Nächste Iteration
Warte-Elemente (o)	Nein	
Start und Ende (o)	Nein	Noch nicht benötigt
Allgemeines:		
WYSIWYG	Ja	
nahezu selbsterklärendes Format	Ja	
Modellierung sehr intuitiv	z.T.	Evaluation muss dies beweisen
Geringer Funktionsumfang	Ja	
kleine Menge an versch. Symbolen und Icons	Ja	
Aufbau der Prozessbeschreibungen modular	Ja	
Copy und Paste	Nein	Durch Vorgabe der Elem. kaum benötigt
ausführbar	Ja	
Notationsform intuitiv und leicht erlernbar	Ja	
Formulareditor:		
Erstellung benutzerfreundlicher Formulare	Ja	
kontextabhängig	Ja	
entsprechende Task bekannt	Ja	
Prozessvariablen bekannt	Nein	Nächste Iteration
bekannte Steuerelemente	Ja	
selbstentworfene Elemente	Ja	
dynamische Erzeugung der Formularinhalte	z.T.	Während der Modellierung
spezifische Eigenschaften	Nein	Nicht benötigt, da viel Vorgegeben
WYSIWYG	Ja	
Datenbasis, Kommunikation	Ja	
Templates	z.T.	Header in Bildform
Formulare modular	Ja	
definierten Blöcken	z.T.	Vorgaben durch Funktionen
speziellen Eigenschaften vordefiniert	Ja	
Copy und Paste	Nein	Durch Vorgabe der Elem. kaum benötigt
plattformunabhängig	Ja	
Laufzeit mobil	ja	

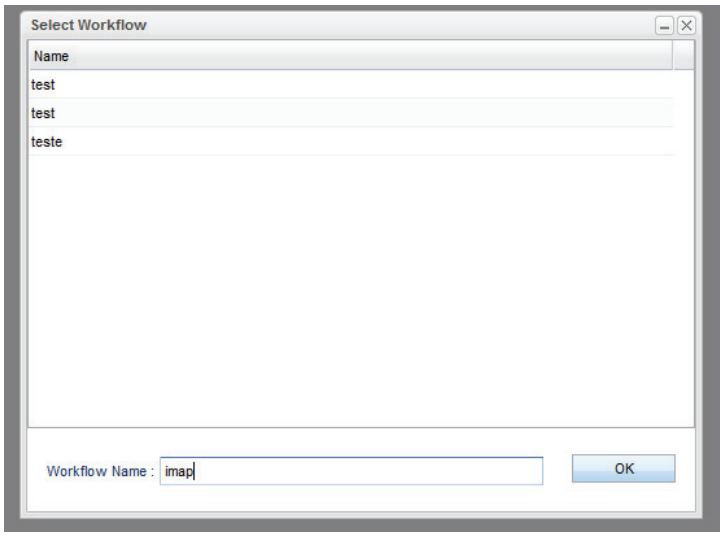
Anhang VII – Sequenzdiagramm der Funktionsweise der Laufzeitumgebung



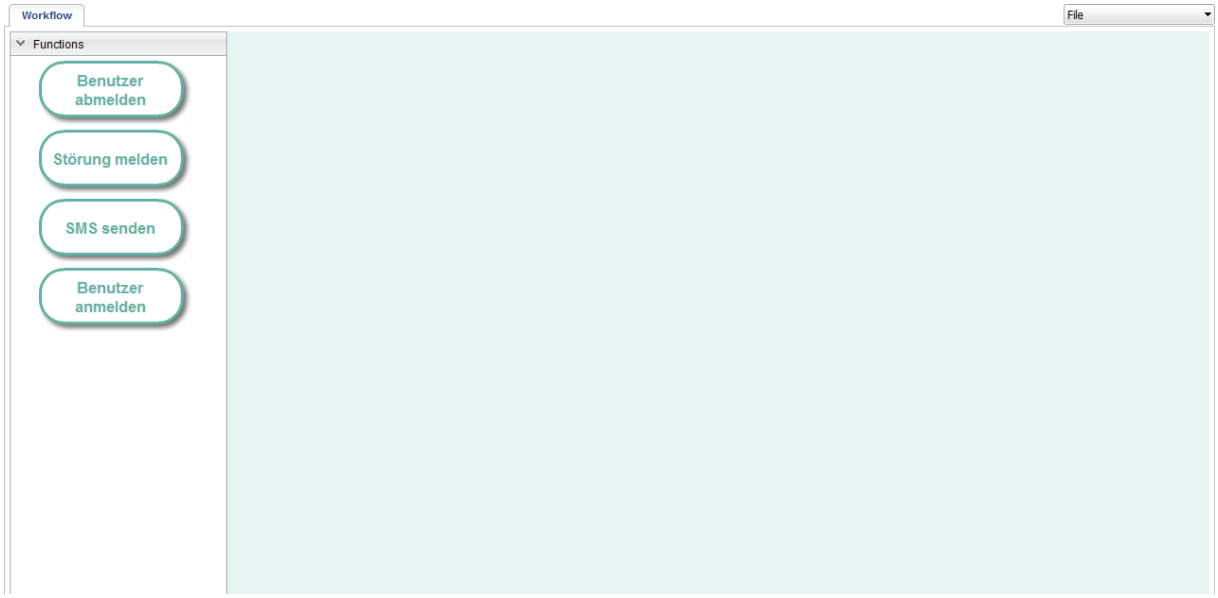
Anhang VIII – Bildserie zur Nutzung des IMAP-Editors

Die im Folgenden dargestellte Bildserie dient der Dokumentation der Nutzung des IMAP-Editors sowie der Laufzeit. Einige ausgewählte Darstellungen wurden bereits im Kapitel 6 abgebildet. Hier soll nun eine ausführliche Visualisierung des entstandenen Systems erfolgen.

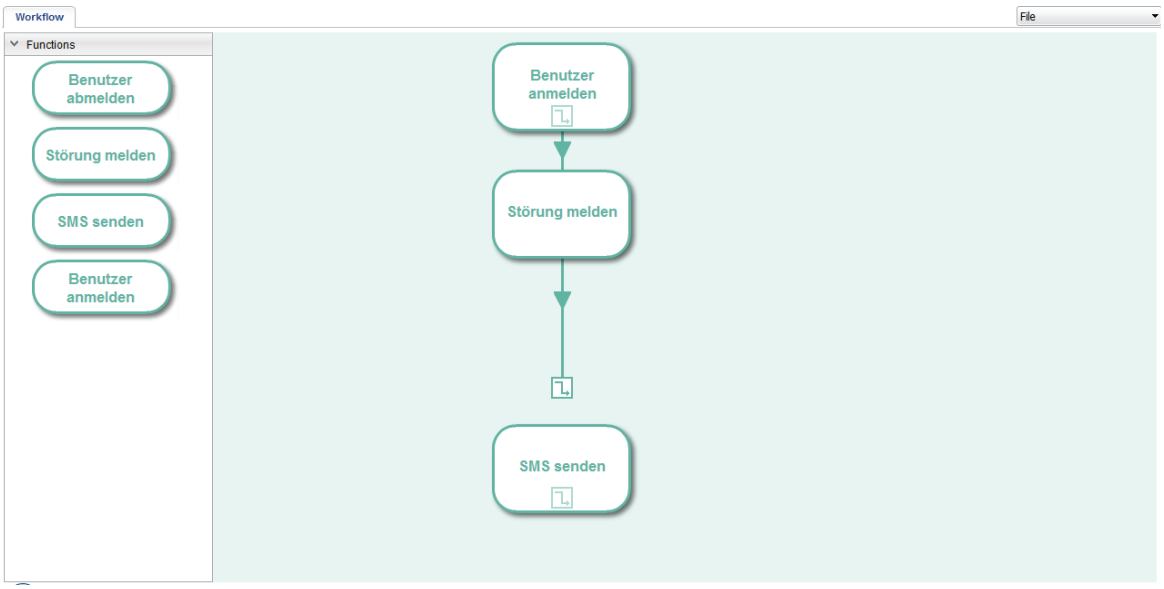
Beim Start des Systems bekommt der Nutzer zunächst die Möglichkeit, einen Workflow auszuwählen, der im Anschluss bearbeitet werden soll. Wird ein noch nicht angebotener Name eingegeben, so wird der entsprechende Workflow neu angelegt.



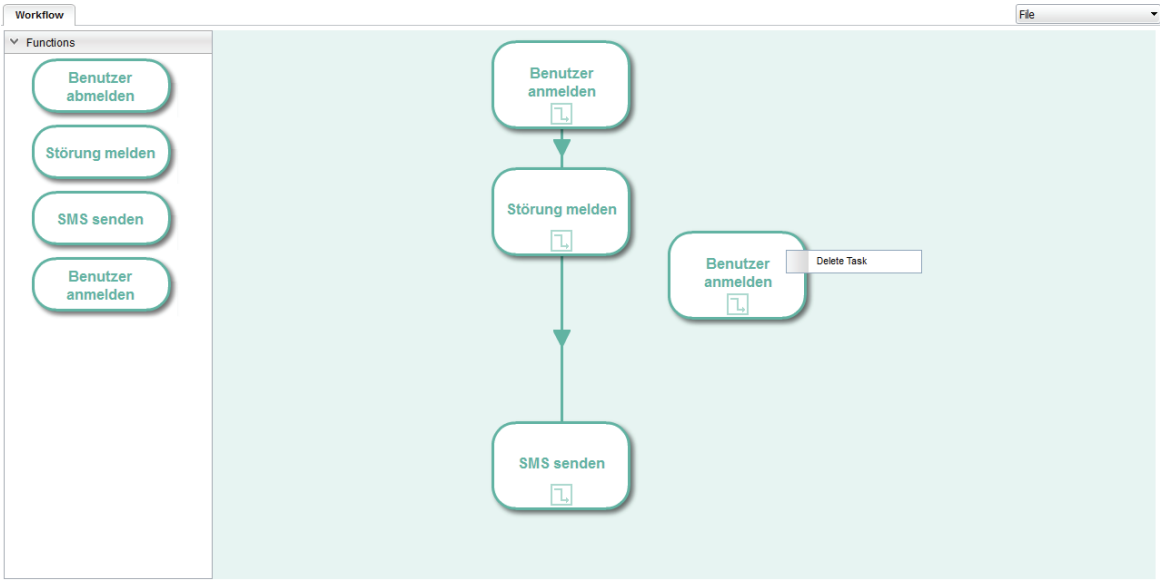
Nach der entsprechenden Auswahl erscheint die Oberfläche des IMAP-Editors. Sie zeigt die Sicht zum Modelliereren der Workflows an. Auf der linken Seite befinden sich die Aktivitäten, welche zur Modellierung bereitstellen. Hinter diesen verbergen sich die Repräsentationen der Schnittstellen zu den entsprechenden Diensten innerhalb des Systems, die durch den ESB für den IMAP-Editor bereitgestellt werden. Die gefärbte Fläche auf der rechten Seite stellt die eigentliche Arbeitsfläche zur Modellierung dar.



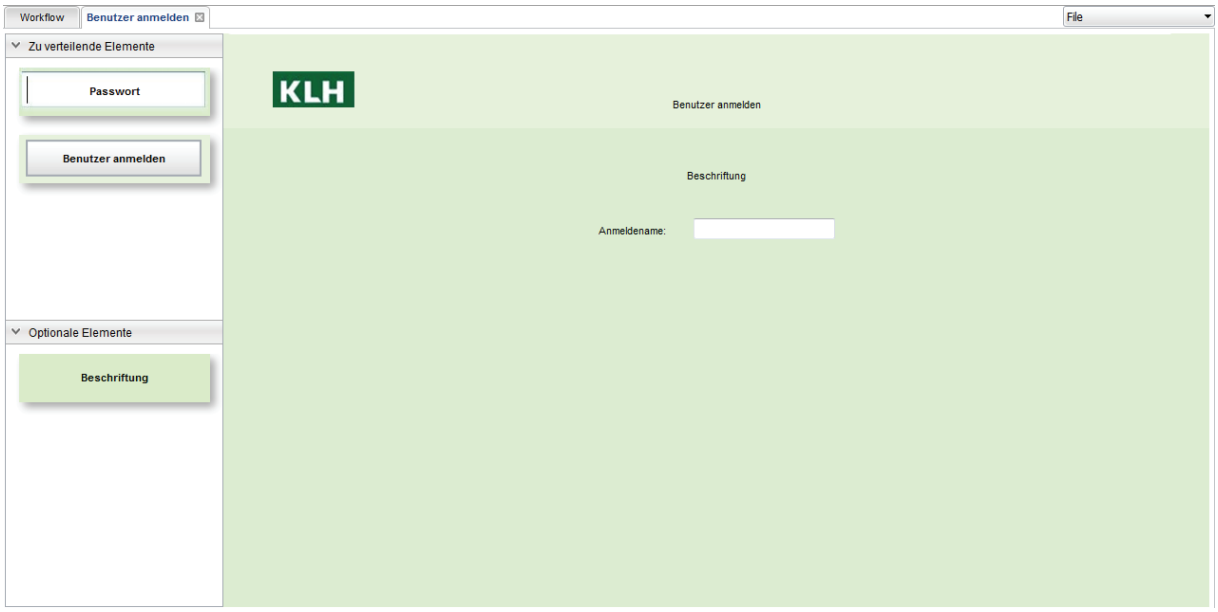
Zur Modellierung einer Abfolge von Aktivitäten werden diese per Drag & Drop auf die Arbeitsfläche gezogen. Anschließend können sie verbunden werden, indem das kleine, viereckige Pfeil-Icon innerhalb der Aktivitäten auf die nächste gezogen wird.



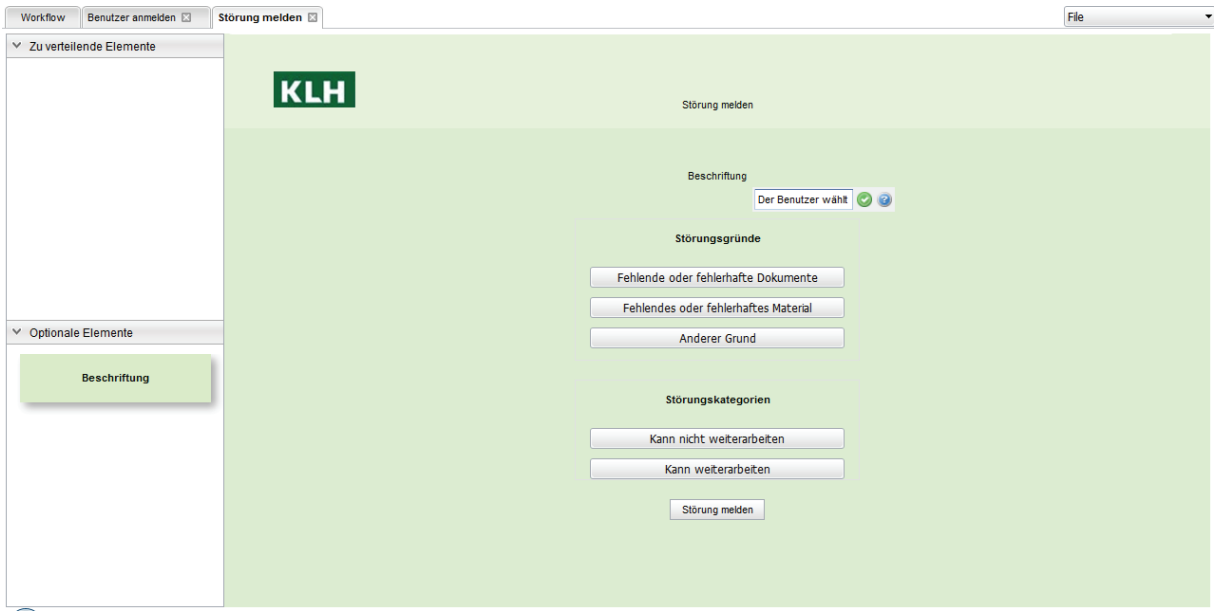
Wurde versehentlich eine Aktivität falsch angelegt, so kann sie auch wieder entfernt werden. Dazu wird zu einem entsprechenden Task mit einem Rechtsklick ein Kontextmenü geöffnet, in welchem die Aufgabe entfernt werden kann.



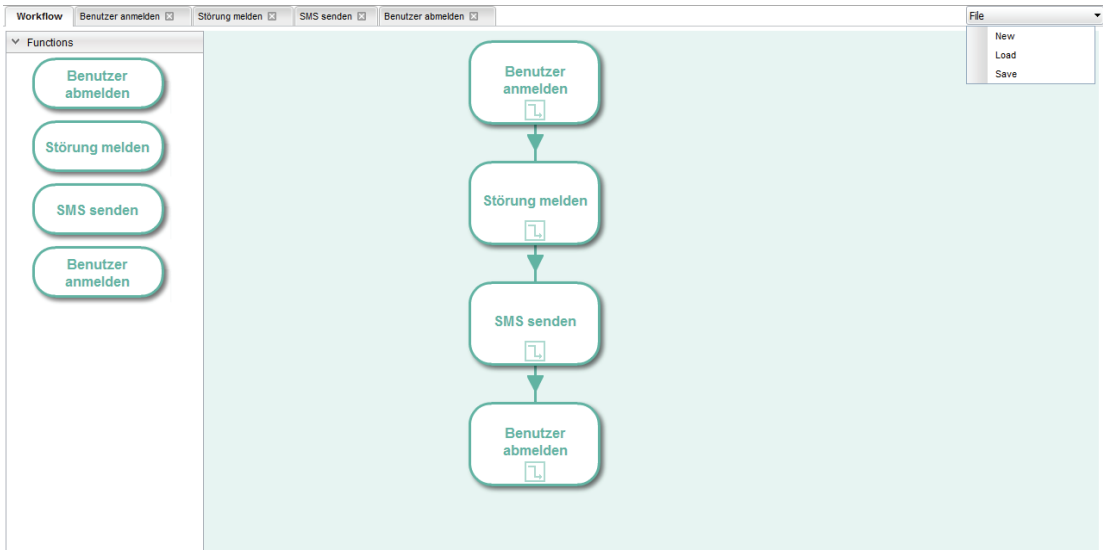
Während der Modellierung oder auch nach deren Vollendung kann zu einem entsprechenden Task der zugehörige Formulareditor gestartet werden. Für diesen wird ein neuer Tab angelegt. Zudem ist er abhängig vom jeweiligen Task, es kann also nur zu einem bestimmten Task das jeweilige Formular erstellt werden. Auch hier befinden sich links die möglichen Elemente, wobei diese unterschieden werden nach notwendigen und optionalen Elementen. Sie werden ebenfalls per Drag & Drop der Arbeitsfläche hinzugefügt und können hier in ihrer vertikalen Anordnung verschoben werden.



Neben der Anordnung und der möglichen, vertikalen Verschiebung kann beispielsweise die Beschriftung verändert werden. Um einen neuen Text einzutragen genügt ein Doppelklick auf das angelegte Element. Das sich öffnende Textfeld kann gefüllt werden. Mit dem grünen Icon wird die Eingabe bestätigt. Der blaue Knopf dient einem Vorschlag, in dem hier die Beschreibung der Funktion, zu der das Formular modelliert wird, als Beschriftung übertragen wird.



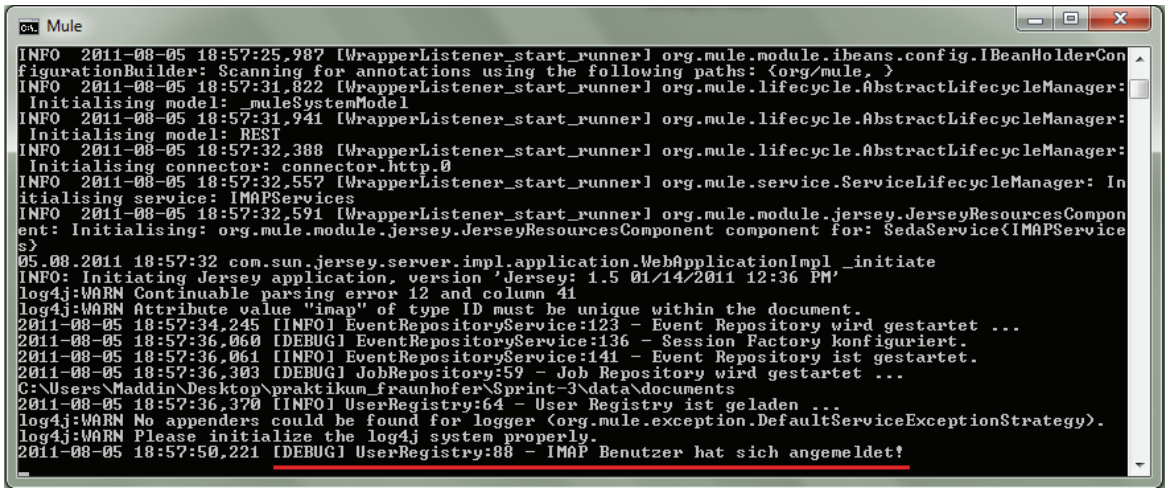
Sind zu allen Aktivitäten, die modelliert wurden die entsprechenden Formulare ebenfalls erstellt, so kann der Workflow gespeichert werden. Das geschieht über das Drop-Down Menü rechts oben.



Ist der Workflow gespeichert, so kann er in der Laufzeitumgebung ausgeführt werden. Die folgenden Abbildungen zeigen die 4 Funktionen, welche mit dem IMAP-Editor modelliert werden können, während ihrer Ausführung zur Laufzeit. Zusätzlich sollen hier die Ergebnisse der Funktionsausführung dokumentiert werden.

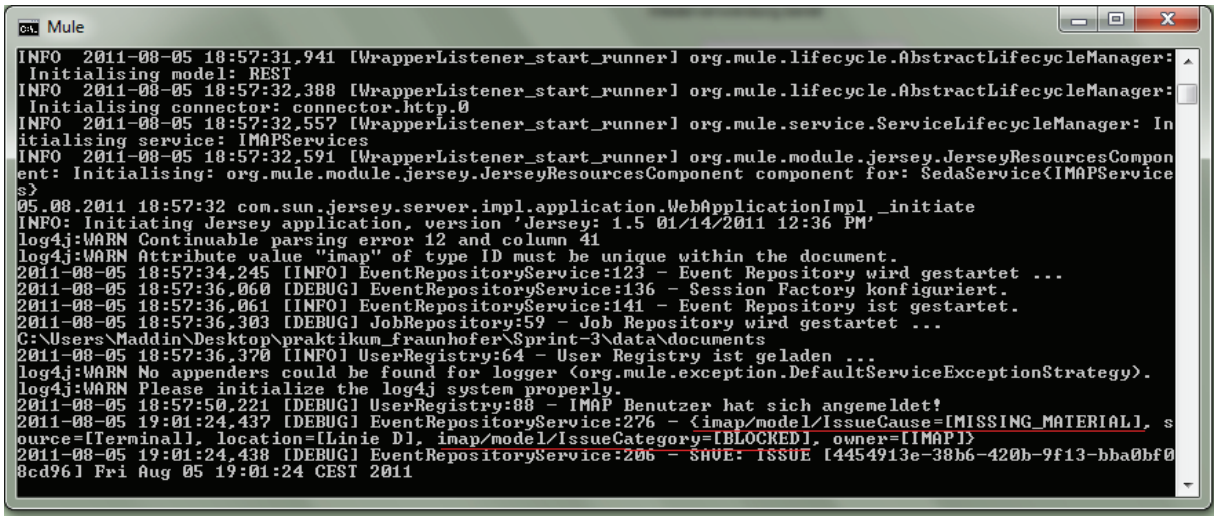
I Funktion „Benutzer anmelden“

Bei Ausführung dieser Funktion wird ein WebService angesprochen, der den entsprechenden User im ESB (Mule) registriert. Das ist hier dargestellt:

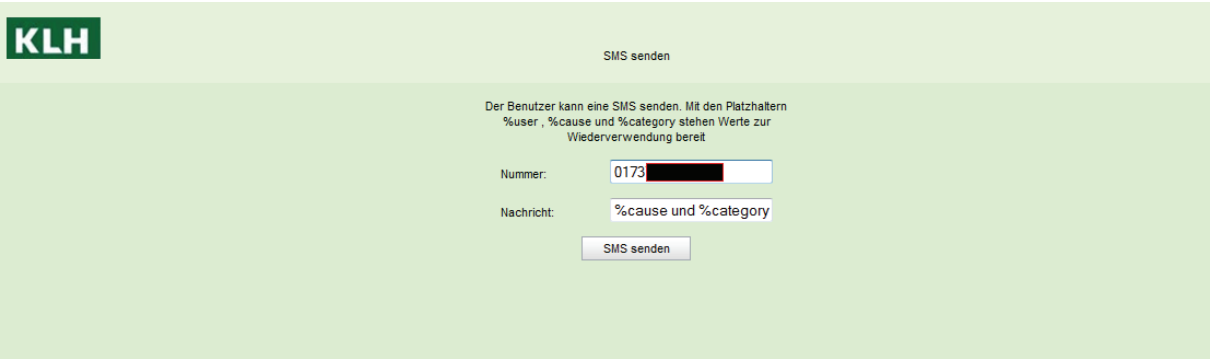


II Funktion „Störung Melden“

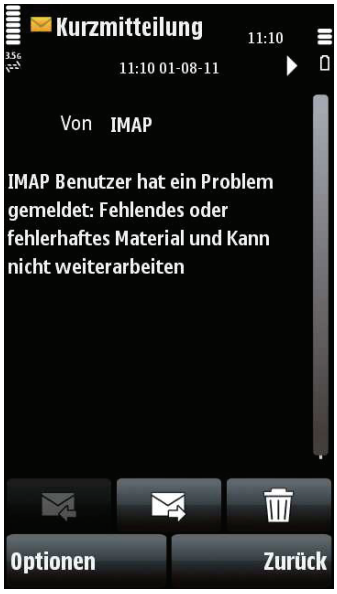
Bei dieser Funktion wird ein Eintrag in die IMAP-Datenbank gemacht. Dabei wird ein Event vom Typ Issue gemeldet. Die ausgewählten Grund und Kategorie werden ebenfalls übernommen.



III Funktion „SMS senden“

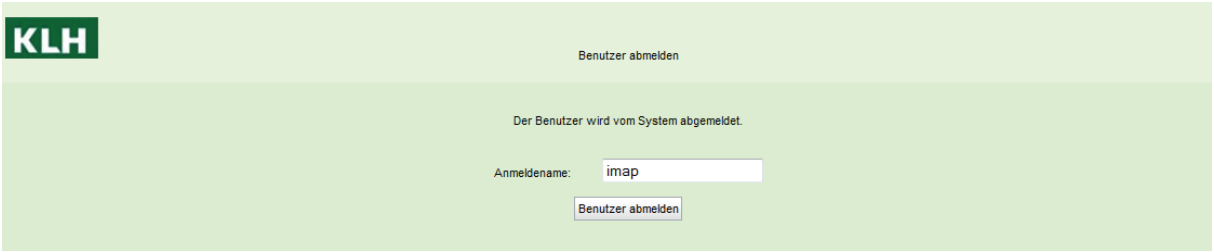


Diese Funktion lässt neben dem reinen Versenden einer Kurzmitteilung über einen externen Anbieter (http://smsout.de/) zusätzlich die Verwendung von Platzhaltern zu, um in das System eingetragene Werte, wie beispielsweise die zuvor gemeldeten Störungsgründe, in der Nachricht zu verwenden. Eine dieser Nachrichten sieht dann beispielsweise wie folgt aus:



IV Funktion „Benutzer abmelden“

Diese Funktion ist das Gegenstück zur Anmeldung.



Auch hier wird ein WebService angesprochen.