HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Rico Beier

Ergänzung einer Software-Suite für halbautomatische Dokumenterstellung um eine Anwendung zur Strukturanalyse und Textbausteinverwaltung

2010

Fakultät **Mathematik/Naturwissenschaften/Informatik**

BACHELORARBEIT

Ergänzung einer Software-Suite für halbautomatische Dokumenterstellung um eine Anwendung zur Strukturanalyse und Textbausteinverwaltung

Autor:
Rico Beier

Studiengang:
Informatik

Seminargruppe:
IF07w1-B

Erstprüfer:
Prof. Dr.-Ing. Wilfried Schubert

Zweitprüfer:
Dipl.-Inf. Ingo Gringer

Mittweida, 2010

Bibliografische Angaben

Beier, Rico: Ergänzung einer Software-Suite für halbautomatische Dokumenterstellung um eine Anwendung zur Strukturanalyse und Textbausteinverwaltung, 83 Seiten, 51 Abbildungen, Hochschule Mittweida (FH), Fakultät Mathematik/Naturwissenschaften/Informatik

Bachelorarbeit, 2010

Satz: LaTeX

Bzr Revision: 388

Referat

Verbindliche Vorgaben für Layouts und Wording sind in Unternehmen und Institutionen maßgebliche Anforderungen für die Gestaltung der Korrespondenz zu Kunden und Partnern. Um diese zu gewährleisten, werden vorlagenbasierte Dokumenterstellungssysteme eingesetzt. Diese Arbeit befasst sich mit der Erweiterung eines solchen Systems unter Einbeziehung des Entwicklungsprozesses, dem die Vorlagen unterliegen.

Die Techniken der Softwareentwicklung wurden über viele Jahre verfeinert und optimiert, so dass in diesem Gebiet eine Reihe etablierter Vorgehensmodelle verfügbar sind. Da diese jedoch nur auf die eigentliche Softwareentwicklung spezialisiert sind, profitieren verwandte Bereiche wie die Vorlagenentwicklung kaum davon. Diese Arbeit untersucht die Anwendbarkeit der Vorgehensmodelle der Softwareentwicklung auf den Bereich der Entwicklung von Dokumentvorlagen.

Zudem entsteht als Ergänzung einer bereits existierenden Software-Suite für halbautomatische Dokumenterstellung eine Anwendung, welche neben Textbausteinverwaltung und Strukturanalyse auch eine technische Unterstützung für den gefundenen Entwicklungsprozess für Vorlagen enthält.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

AIIM	Association for Information and Image Management
AIR	Adobe Integrated Runtime
API	Application Programming Interface
CI	Coded Information
CRUD	Create Read Update Delete
DIN	Deutsche Institut für Normung
DTP	DeskTop Publishing
EAR	Enterprise ARchive
ECM	Enterprise Content Management
EE	Enterprise Edition
EN	European Norm
ES	Enterprise Suite
FDD	Feature Driven Development
HTML	HyperText Markup Language
ICR	Intelligent Character Recognition
ISO	International Organization for Standardization
IT	InformationsTechnik
NCI	Non Coded Information
OCR	Optical Character Recognition
OMS	Output Management System
PDF	Portable Document Format
QM	Quality Management
RIA	Rich Internet Application
TEd	Textbaustein EDitor
URL	Unified Resource Locator
WYSIWYG	What You See Is What You Get
XDP	Xml Data Package
XML	eXtensible Markup Language
XSL-FO	eXtensible Stylesheet Language - Formatting Objects
XSLT	eXtensible Stylesheet Language Transformation
XT	eXtreme Tailoring

V. Danksagung

> „Es ist nicht gut, daß der Mensch alleine sei, und besonders nicht, daß er alleine arbeite; vielmehr bedarf er der Teilnahme und Anregung, wenn etwas gelingen soll.“
>
> Johann Wolfgang von Goethe

So möchte ich mich bei all denen bedanken, die mich im Laufe meines Studiums - bewusst oder unbewusst - unterstützt haben, egal auf welche Weise. Ohne eure Teilnahme und Anregung wäre es nicht gelungen.

Für die Zeit, in der diese Bachelorarbeit entstand, gilt mein Dank besonders Herrn Prof. Schubert und Herrn Ingo Gringer, die mir als Betreuer stets mit offenem Ohr und kritischem Blick zur Seite standen.

1 Einleitung

In einer Zeit zunehmender Digitalisierung spielen Dokumente weiterhin eine wichtige Rolle als Auslöser und Gegenstand von Geschäftsprozessen vieler Unternehmen. Sie unterliegen dabei zumeist strengen Designvorgaben der unternehmensweit eingeführten Corporate Identity. Es ist daher notwendig, die Technologien der Erzeugung solcher Dokumente weiterzuentwickeln und zu optimieren.

Diese Arbeit behandelt eine solche Weiterentwicklung am Beispiel einer konkreten praktischen Aufgabenstellung aus dem Projektumfeld der msg.

1.1 msg systems ag

Die msg systems ag ist ein Systemhaus mit Hauptsitz in Ismaning bei München. Neben ihrem Hauptsitz hat die msg weitere Geschäftsstellen und Landesgesellschaften in Deutschland, Österreich, der Schweiz, Rumänien, Spanien, Indien, Singapur und den USA. Abbildung 1.1 „msg systems ag, Geschäftsstelle Chemnitz“ zeigt das Bürogebäude der Geschäftsstelle Chemnitz. Das 1980 gegründete Unternehmen beschäftigt derzeit über 3000 Mitarbeiter.

Abbildung 1.1: msg systems ag, Geschäftsstelle Chemnitz

Ihren Tätigkeitsbereich beschreibt die msg wie folgt:

> .consulting .solutions .partnership – diese drei Begriffe bringen unser zentrales Anliegen auf den Punkt: die ganzheitliche Unterstützung unserer Kunden, von der Beratung bis zum lauffähigen System. Unsere Kompetenz in den Bereichen Technologie und Betriebswirtschaft verbinden wir mit detailliertem Branchenwissen, um in enger partnerschaftlicher Zusammenarbeit mit unseren Kunden die für sie optimale Systemlösung zu finden.

> Der Fokus unserer Arbeit liegt auf der Entwicklung von komplexen, individuell gestalteten Anwendungssystemen und Standardsoftware für die Branchen Automotive, Finanzdienstleistungen, Gesundheitswesen und Versicherungen. Als Systemintegrator sorgen wir für das reibungslose Zusammenspiel aller Systemkomponenten. [msg10]

Die Geschäftsstelle Chemnitz arbeitet sowohl in der Produktentwicklung für den Bereich Versicherungen als auch im Projektbetrieb für Versicherungen und öffentliche Auftraggeber. Schwerpunkt ist zum einen das Java EE-Competence Center im Geschäftsbereich Versicherungen, zum anderen die Arbeit als Silver Solution Partner von Adobe an Lösungen im Enterprisebereich mit der *LiveCycle*-Technologie. In der msg wird dazu ein breites Spektrum verschiedener Technologien eingesetzt.

1.2 Ziele der Arbeit

In einem bestehenden Kundenprojekt wird zur Erstellung von Geschäftsdokumenten eine Lösung, basierend auf der *Adobe LiveCycle Enterprise Suite*, eingesetzt. Damit lassen sich automatisch auf generischen oder spezifischen Vorlagen basierende Dokumente und interaktive Formulare im PDF-Format generieren, die mit Daten aus einem vorgelagerten System befüllt werden.

Sachbearbeiter können in den Dokument-Generierungsprozess eingreifen, indem sie vor dem Aussenden der Dokumente Textmanipulationen direkt oder über eine spezielle Sachbearbeiteroberfläche im PDF-Dokument vornehmen. Um dabei die Sachbearbeiter bestmöglich bei ihrer Arbeit zu unterstützen und zugleich die Anforderungen aus dem Bereich „Unternehmenssprache/rechtliche Formulierungen“ (Legal Wording/Corporate Wording) zu gewährleisten, werden vorgefertigte und parametrisierte Textbausteine eingesetzt.

Im Verlauf der Entwicklung der Vorlagen müssen auch die Textbausteine im System erfasst und gewartet werden. Diese nicht im Produkt *Adobe LiveCycle* vorhandene Funktionalität soll nun durch eine zu entwickelnde Anwendung *Textbausteineditor* ergänzt werden, die sich in den Arbeitsablauf der Vorlagenentwicklung gut integrieren soll.

Da die Entwicklung von Vorlagen und Textbausteinen einem festgelegten Ablauf unterliegt, sollen deren einzelne Phasen durch die Anwendung ebenfalls unterstützt werden. Die Erarbeitung eines geeigneten Vorgehensmodells soll das zweite wesentliche Ziel der Arbeit darstellen. Dazu werden vorhandene Vorgehensmodelle der Softwareentwicklung zur Betrachtung herangezogen.

1.3 Abgrenzung

Um den Rahmen der Arbeit nicht zu sprengen, sind einige Abgrenzungen zu nennen:

- In den Ausführungen zur Dokumenterzeugung werden Enterprise Content Management und Output Management nur soweit angerissen, wie es notwendig ist, um den Kontext der Dokumenterzeugung verständlich darzustellen.
- Bei der Betrachtung der Vorgehensmodelle der Softwareentwicklung wurde nur eine kleine Auswahl aus den vorhandenen Modellen gewählt, die für die Vorlagenentwicklung relevant sein könnten. Die Beurteilung der Modelle bezieht sich lediglich auf deren Anwendbarkeit auf die Vorlagenentwicklung.
- In der technischen Konzeption wird nur auf eine Auswahl wichtiger Funktionen eingegangen. Die Umsetzung einer Versionisierung im Sinne einer Sourcecodeverwaltung, welche über Dateien und Metainformationen hinweg eine Historie führt, ist nicht Gegenstand der Arbeit.

1.4 Aufbau der Arbeit

Nach dieser kurzen Einleitung wird in Kapitel 2 „Dokumenterzeugung“ ab Seite 5 die Erzeugung von Dokumenten näher beleuchtet. Nach einer vorbereitenden Definition des Begriffs Dokument wird sich dem Thema schrittweise genähert. Beginn ist die Einordnung der Dokumenterzeugung in den großen Themenbereich des Enterprise Content Management sowie in den Workflow des Unternehmens. Daran anschließend wird der engere Rahmen - das Output Management - vorgestellt. Eine Übersicht der Dokumenterzeugungstechnologien, die besonderen Wert auf die vorlagenbasierte Dokumenterzeugung legt, bildet die Überleitung auf das bestehende und zu erweiternde Softwareprodukt *DokGen*.

Die dabei aufgedeckten Schwachstellen werden in einem neuen Produkt bereinigt, welches im Rahmen dieser Arbeit entwickelt wird. Die wichtigsten fachlichen Anforderungen an diese Entwicklung werden in Kapitel 3 „Fachliche Spezifikation der Anforderungen des TEd“ ab Seite 23 kurz vorgestellt.

Bei einer dieser Anforderungen reicht eine kurze Vorstellung nicht aus. Das Vorgehensmodell muss tiefergehend analysiert und festgelegt werden. In Kapitel 4 „Analyse und fachliche Spezifikation des Entwicklungsprozesses“ ab Seite 27 wird daher durch Analogie zur Softwareentwicklung und unter Einbeziehung von Erkenntnissen aus bestehenden Vorgehensmodellen der Softwareentwicklung ein Vorgehensmodell für die Entwicklung von Vorlagen entworfen und mit aktuellen Mitteln spezifiziert. Die Überlegung wird durch eine Betrachtung der fachlichen Umsetzung dieses Modells abgeschlossen.

In Kapitel 5 „Technische Konzeption“ ab Seite 45 werden technisch relevante Überlegungen zur Entwicklung des Programms angestellt. Hier finden sich auch die fachlichen Anforderungen wieder, die nun in Hinsicht auf ihre Realisierung detaillierter betrachtet werden. Etwas von der eigentlichen Entwicklung abgetrennt werden in Kapitel 6 „Entwicklungsumgebung und Softwareverteilung“ ab Seite 63 noch zwei weitere Punkte kurz vorgestellt.

Schließlich findet die Arbeit ihren Abschluss in Kapitel 7 „Zusammenfassung und Ausblick“ ab Seite 67. Hier werden die Ergebnisse noch einmal prägnant zusammengefasst und kritisch bewertet.

2 Dokumenterzeugung

Das Erzeugen von Dokumenten ist ein wichtiger Prozess im Geschäftsumfeld. Er ist einer der Bestandteile des Output Management, welches im großen Umfeld des Enterprise Content Management angesiedelt ist. Vor der Beschäftigung mit dem eigentlichen Thema der Dokumenterzeugung ist es vorteilhaft, sich den Begriff des Dokuments etwas genauer anzusehen.

2.1 Der Begriff Dokument

Eine sehr kurze Definition orientiert sich an der DIN EN ISO 9000. Dort wird ein Dokument als eine Information mit ihrem Trägermedium beschrieben. [iso00] Etwas aussagekräftiger ist hingegen die folgende Definition aus dem Bereich der Dokumentationstechnik:

> Ein Dokument ist eine auf einem Trägermedium festgelegte Information, die bei entsprechender Bedeutung als Gesamtheit durch ein Dokumentkennzeichen eindeutig identifizierbar und dem zugelassenen Zugriff zugänglich ist.[1]

Als Dokumentkennzeichen können hier beispielsweise Aktenzeichen oder Dateinamen fungieren. Dies klingt zwar nach reinen Schriften, jedoch beschränkt sich die Form eines Dokuments nicht nur auf die Verwendung von Text. Auch Bild, Ton und Video können Teil eines Dokuments sein. In diesem Fall spricht man gemeinhin von multimedialen Dokumenten. [hmc10] Zum Dokument gehören weiterhin auch alle ergänzenden Angaben, wie beispielsweise Metadaten, die zum Verständnis der Primärinformation notwendig sind. [Bun05]

Die Dokumenterzeugung, wie sie hier behandelt wird, meint in aller Regel Dokumente, die sich aus Text und Grafiken zusammensetzen. Liegen diese in Papierform vor, so werden sie im weiteren Verlauf dieser Arbeit *analoge Dokumente* genannt. Bei *digitalen*, oder auch *elektronischen Dokumenten* muss eine weitere Einteilung vorgenommen werden. Das Ergebnis einer digitalen, bildlichen Erfassung eines analogen Dokuments, dessen Inhalt erst durch Verfahren wie Optical Character Recognition (OCR) und Intelligent Character Recognition (ICR) aus den Bildpunkten zurückgewonnen werden muss, wird Non Coded Information (NCI) - Dokument genannt. Andernfalls spricht man von einem Coded Information (CI) - Dokument.[2]

[1] [GK08], S. 494
[2] [Rig09], S. 61

2.2 Einordnung der Dokumenterzeugung

Die Betrachtung der Dokumenterzeugung sollte auch den umliegenden Kontext mit einbeziehen. Das Erzeugen von Dokumenten fällt hierbei in den Kontext des Enterprise Content Management.

2.2.1 Enterprise Content Management

Vielfach wird der Begriff als „Content Management“ auf „Enterprise“-Ebene verstanden. Dabei ist die primäre Assoziation zu Content Management die Verwaltung von Websites. Das Verständnis von Enterprise Content Management als die Verwaltung von großen Websites ist jedoch falsch. Vielmehr geht es darum, den „Enterprise Content“ - also alle Unternehmensinformationen - zu „managen“. [jdk10]

Enterprise Content umfasst dabei im Wesentlichen drei Bereiche. Das sind zum einen alle unstrukturierten Informationen des Unternehmens, wie sie im Ein- und Ausgang beispielsweise in Geschäftsbriefen vorkommen. Zum anderen gehören die strukturierten Informationen dazu, die die Geschäftsprozesse und -objekte beschreiben. Als dritte Komponente kommt noch das Wissen über die im Unternehmen ablaufenden Prozesse hinzu. [pir06]

Enterprise Content Management ist als ein Sammelbegriff für sämtliche Produkte, Techniken und Prozesse anzusehen, die verwendet werden, um strukturierte und unstrukturierte Informationen zu erfassen, bearbeiten, verwalten, publizieren und archivieren. Dabei kann der Begriff Enterprise Content Management als Wortschöpfung der Non-Profit-Organisation Association for Information and Image Management (AIIM) gelten[3]. Sie ist eine etablierte Community, die Wissen bereitstellt, sowie Forschung betreibt, um Unternehmen zu helfen, ihre Informationen zu organisieren, steuern und optimieren[4].

Das Enterprise Content Management lässt sich, wie in Abbildung 2.1 „Enterprise Content Management, Komponenten, nach [aii10b]“ auf der folgenden Seite dargestellt, in folgende fünf Komponenten einteilen, welche einen groben Lebenszyklus der Dokumente skizzieren:

Erfassung (Capture) Die *Capture*-Komponente beinhaltet die Funktionalität zur Erfassung, Aufbereitung und Verarbeitung von analogen wie auch elektronischen Dokumenten. Dies reicht vom einfachen Einscannen eines Briefes bis hin zu einer komplexen Aufbereitung und Klassifikation der enthaltenen Informationen.

[3] [Rig09], S. 4
[4] frei übersetzt nach [aii10a]

Abbildung 2.1: Enterprise Content Management, Komponenten, nach [aii10b]

Verwaltung (Manage) Die *Manage*-Komponente dient zur Verwaltung, Bearbeitung und Nutzung der Informationen. Sie verteilt sich unter anderem auf folgende Anwendungsfelder: Dokumentenmanagement, Collaboration, Web Content Management und Workflow Management.

Ausgabe (Deliver) Die Ausgabe wird auch als Output Management bezeichnet. Neben dem althergebrachten Druck von Dokumenten kann hier auch die Ausgabe digitaler Dokumente in Form von PDF, die Präsentation von Inhalten im Web, wie auch die Erzeugung von eBooks eingeordnet werden.

Archivierung (Preserve) Aufgrund rechtlicher Vorschriften und auch im eigenen Interesse des Unternehmens gibt es zahlreiche Informationen, die langfristig aufbewahrt werden müssen. Handelsbriefe unterliegen beispielsweise einer Aufbewahrungspflicht von sechs Jahren[5]. Das Archiv zur stabilen und unveränderbaren Aufbewahrung dieser Informationen über lange Zeiträume wird von der Komponente *Preserve* bereitgestellt.

Speicherung (Store) Nicht alle Informationen sind archivierungswürdig bzw. archivierungspflichtig. Dazu zählen unter anderem Daten, die von laufenden Prozessen verwendet werden, aber auch unvollständig bearbeitete Dokumente aus der Komponente *Capture*. Die temporäre Speicherung dieser Informationen wird von der Komponente *Store* übernommen. Sie stellt damit eher eine Ablage als ein Archiv dar.

2.2.2 Einordnung im Enterprise Content Management

ECM deckt einen weiten Raum ab. Die Dokumenterzeugung ist ein Teilbereich aus dem Output Management, welcher sich mit der Erzeugung von Dokumenten in digitaler Form (PDF) und gegebenenfalls deren Ausdruck in Papierform befasst. Die Dokumenterzeu-

[5] [HGB09], §257

gung ist also als Teil des Output Management im Gesamtkontext des ECM in der Komponente *Deliver* anzusiedeln. Das Output Management wird in Abschnitt 2.3 „Output Management“ auf der nächsten Seite genauer erläutert.

Trotz dieser klaren Einordnung soll erwähnt werden, dass auch die *Capture*-Komponente durch die Art der Dokumenterzeugung profitieren kann. Der Aufwand, Dokumente zu erfassen, steigt mit deren Unstrukturiertheit. Nach dem Einscannen von Dokumenten in Papierform liegen diese erst als Non Coded Information (NCI) vor. Sie müssen vor einer weiteren maschinellen Verwendung in Coded Information (CI) überführt werden. Dies ist mit Hilfe von OCR-Software möglich. Eine manuelle Korrektur ist dabei nicht unüblich. Der OCR nachgestellt erfolgt eine strukturelle Zuordnung der Daten aufgrund ihres Inhalts und ihrer Position auf dem eingescannten Dokument (ICR).

Hier spielt die Erzeugung digitaler Dokumente ihre Stärke aus. Es entstehen direkt CI-Dokumente, welche ohne den Zwischenschritt der Texterkennung im System weiterverarbeitet werden können. Informationen, also Daten und ihre strukturelle Zuordnung, können so wesentlich einfacher aus den Dokumenten extrahiert werden. Wichtige Informationen können dem Dokument zusätzlich in Form von Metadaten beigefügt werden, was die Zuordnung eines fachlichen Kontextes wesentlich einfacher macht.

2.2.3 Einordnung in den Workflow des Unternehmens

Neben dem eben beschriebenen Lebenszyklus eines Dokuments ist dessen Rolle im Workflow eines Unternehmens wichtig. Hierbei unterscheidet man in Kerngeschäftsprozesse und Unterstützungsprozesse.

Dokumente spielen in den Kerngeschäftsprozessen eine bedeutende Rolle. Die Verträge, die im Policierungsprozess entstehen, bilden beispielsweise die rechtsverbindliche Grundlage des Versicherungsverhältnisses, ohne die von beiden Seiten keine Ansprüche möglich wären. Aber auch die Rolle als Anstoß und Ergebnis eines Workflows ist nicht vernachlässigbar. So ist die Schadensmeldung der Auslöser einer Schadenregulierung, welche wiederum ein Antwortschreiben als Ergebnis liefert. Eine weitere Kategorie von Dokumenten, die in den Kernprozessen angesiedelt werden, sind die, die als Vertriebsmaßnahmen und Werbung im Bereich Kundenakquise verwendet werden.

In den unterstützenden Geschäftsprozessen sind die Dokumente genau so wenig wegzudenken wie in den Kernprozessen. Die Bereitstellung von Informationen im Internetportal, wie auch die betriebsinterne Verteilung von Informationen über das Intranet nutzen Dokumente. Deren Erstellung unterliegt den gleichen Rahmenbedingungen wie die Erstellung der Dokumente aus den Kernprozessen.

Da jedoch die Kernprozesse den Großteil der betrieblichen Wertschöpfung ausmachen, kommt den Dokumenten aus den Kerngeschäftsprozessen eine höhere Wichtung zu als denen in den unterstützenden Prozessen.

2.3 Output Management

Bei dem Begriff Output Management handelt es sich um den Teilbereich des Enterprise Content Management, der in Abbildung 2.1 „Enterprise Content Management, Komponenten, nach [aii10b]“ auf Seite 7 durch den Ausgang „Deliver“ dargestellt wurde. Eine feste Definition des Begriffs gibt es nicht. Deshalb sollte man sich diesem Thema über seinen Aufgabenbereich nähern.

2.3.1 Aufgaben und Definition

Einfach ausgedrückt ist die Aufgabe des Output Management, das richtige Dokument im richtigen Format zur richtigen Zeit zu den jeweils günstigsten Kosten zum Empfänger zu bringen[6].

Formaler ausgedrückt, umfasst das Output Management die Wertschöpfungskette der Geschäftskommunikation von der Anlieferung der Daten aus den betrieblichen Informationssystemen über die Erstellung und Generierung von digitalen und analogen Dokumenten bis hin zum Massenversand[7]. Dies wird vereinfacht in Abbildung 2.2 „Output Management, Überblick“ gezeigt. Die dafür verwendeten Systeme können daher auch als wichtige Kommunikationsinstrumente des Unternehmens verstanden werden[8].

Aus der Definition ergeben sich drei wichtige Komponenten: Dokumenterzeugung, Ausgabe im Zielformat und Versand.

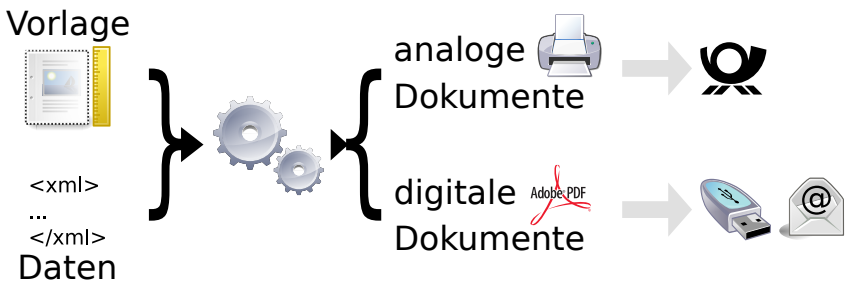

Abbildung 2.2: Output Management, Überblick

[6] [BSB09], S. 3
[7] [LSL08], S. 3
[8] [LSL08], S. 8

Potential zur Optimierung gibt es an vielen Stellen dieses Weges. Dabei sind die Möglichkeiten bei Druckkosteneinsparung und Portooptimierung offensichtlicher und häufiger ausgeschöpft als die, die sich im Themenfeld der Dokumenterzeugung ergeben.

2.3.2 Dokumenterzeugung

Wie bereits erwähnt, steht am Anfang dieses Weges immer die Erzeugung eines Dokuments. Hierunter zählt die Produktion von personalisierten Geschäftsdokumenten sowie deren Verwaltung und Weiterverarbeitung. Dabei wird ein breites Spektrum abgedeckt, welches von Serienbriefen in Massenauflage bis zum Erstellen von individuellen Schreiben mit zum Teil juristisch abgesicherten Vertragstexten reicht. In jedem Fall werden aber die Designvorgaben des jeweiligen Unternehmens (Corporate Identity) umzusetzen sein.

Je nach Bedarf und Stand der Technik können die Geschäftsdokumente manuell, halbautomatisch mit individueller Nachbearbeitung oder vollautomatisch produziert werden[9]. Um den Aufwand gering zu halten, ist die Anbindung an vorhandenen Bestandssysteme wichtig. Daten werden so direkt übernommen. Dabei liegt die Herausforderung im Angleich der Datenstrukturen von Bestandssystem und Dokumenterzeugungssystem.

2.3.3 Druck und Versand

Ob digital oder analog, Dokumente werden weiterversendet. Geschieht dies nicht durch automatisierten E-Mail-Versand, sondern auf dem Postweg, so müssen die Dokumente gedruckt und versandfertig gemacht werden. Hier gibt es unterschiedliche Ansätze:

dezentral Der dezentrale Druck ist ein Ansatz, der nur in Unternehmen denkbar ist, die ein geringes Druckaufkommen haben, so dass sich die Einrichtung eines zentralen Druckservers nicht lohnt. Kuvertierung und Freimachung von Briefen muss hier von Hand erledigt werden.

zentral In Hinsicht auf die Druckkosten bietet der zentrale Druck wesentlich höhere Optimierungsmöglichkeiten. Gerade im Versicherungsumfeld, in dem ein hohes Druckaufkommen existiert, bietet sich die Anbindung einer Druckstraße an. Neben dem eigentlichen Druck kann hier auch das Falten, Kuvertieren und die Freimachung der Briefe automatisiert werden.

Druck und Versand können auch durch externe Anbieter übernommen werden. Diese bieten umfassende Versandlösungen an. Ihnen werden elektronische Dokumente geliefert, die über den Postweg versendet werden sollen. Hierauf liegt jedoch nicht der Fokus dieser Arbeit.

[9] [BSB09], S. 10

2.4 Technologien, um Dokumente zu erzeugen

Es existiert eine Vielzahl von Technologien, die dazu verwendet werden können, Dokumente zu erzeugen. Eine Auswahl davon wird hier vorgestellt.

2.4.1 Textverarbeitungssoftware

Diese Lösungen sind sehr weit verbreitet. Für einen Individualbrief, der keinen weiteren Prozessbezug aufweist und folglich auch kaum relevante Daten aus dem Bestandssystem nutzt, ist es die ideale Lösung. Die direkte Arbeit am Dokument im WYSIWYG-Editor macht Textverarbeitungslösungen wie *Microsoft Word* oder *Open Office Writer* attraktiv für die Dokumenterzeugung im nichtgeschäftlichen Umfeld. Die Nutzung von Schriftstilen wird ermöglicht, aber nicht erzwungen. So können kleinere Dokumente ohne den zusätzlichen Aufwand für die Definition von Formatvorlagen erstellt werden, wohingegen bei größeren Dokumenten die Vorteile der zentral gepflegten Stile genutzt werden können.

Dem prozessbezogenen Anspruch der geschäftlichen Dokumenterzeugung sind diese Standardtextverarbeitungsprogramme jedoch nicht gewachsen. Wenngleich in diesen Lösungen auch Vorlagen zum Einsatz kommen können, gestaltet sich die Durchsetzung eines firmenweiten Corporate Design der erstellten Dokumente ebenso, wie auch die Verwendung von Geschäftsdaten aus dem Bestandssystem zu aufwändig.

2.4.2 Desktop Publishing

DTP legt seinen Fokus auf die Erzeugung hochwertiger Dokumente für den Druck. Punktgenaue Ausrichtung von Text und Bild ermöglichen die Umsetzung komplexer Layouts. Dies ist eher geeignet für die Entwicklung einmaliger, grafisch sehr anspruchsvoller Dokumente wie Broschüren, Plakate und Kalender.

Geschäftsbriefe zu erzeugen, die einem Corporate Design unterliegen sollen, ist mit DTP-Software wie *Adobe InDesign* zu aufwändig für den alltäglichen Einsatz. Zudem sind DTP-Systeme nicht darauf ausgelegt, über leistungsfähige Schnittstellen prozessbezogene Daten aus Umsystemen abzugreifen oder von diesen Systemen angesteuert zu werden.

2.4.3 Textsatz

Im Gegensatz zu den beiden vorigen Technologien hat der Autor eines Dokuments bei Textsatzsystemen wie *LaTeX* nur indirekten Einfluss auf das entstehende Layout. Zwar kann das Layout anhand verschiedener Kriterien prinzipiell festgelegt werden, jedoch

geschieht die genaue Verteilung von Text und Bild im entstehenden Dokument dann allein anhand dieser Festlegungen. Abhängig von der Betrachtungsweise kann das Vorteil wie Nachteil sein.

Scheinbar ist die Tatsache nachteilig, dass hier kein WYSIWYG-Editor existiert. Die Dokumente werden in einem speziellen Format erstellt, welches Rohtexte und Skriptelemente enthält. Sie werden anschließend in ein Zielformat kompiliert. Das ist nur auf den ersten Blick ein Nachteil. Tatsächlich konzentriert sich der Ersteller eines Dokuments nun stärker auf den Inhalt.

Für die manuelle Generierung von Dokumenten sind Systeme wie LaTeX durch das spezielle Format, welches einer vorigen Einarbeitung bedarf, eher ungeeignet. Nutzt man LaTeX hingegen im Hintergrund eines Systems als Template, sodass der Benutzer damit nicht vordergründig umgehen muss, ist der Einsatz durchaus denkbar.

2.4.4 XSL-Formatting Objects

XSL-FO ist eine Sprache zur Formatierung von XML-Daten für die Ausgabe auf Bildschirm, Papier und anderen Medien[10]. Ein XML-Dokument wird mithilfe einer XSL-Transformation in einen Baum aus Seitenbeschreibungselementen umgeformt. Dieser kann dann in das jeweilige Zielformat gerendert werden.

Ein wichtiger Punkt bei diesem Ansatz ist die Trennung von Layout und Inhalt. XML bietet hierbei eine gute Anbindungsmöglichkeit an bestehende Softwaresysteme, mit der man die zur Individualisierung eines Schreibens notwendigen Inhalte beisteuern kann. Das Gesamtlayout wird dabei in Form einer XSLT gehalten.

XSLT ist zwar noch leistungsfähiger als *LaTeX*, im Gegenzug aber auch wesentlich komplizierter. Sein Einsatz beschränkt sich daher auf vollautomatisch generierte Dokumente, die während oder nach der Erzeugung nicht mehr von einem Sachbearbeiter verändert werden müssen.

2.4.5 Report-Generatoren

Reportgeneratoren wie *Crystal Reports* und *Jasper Reports* sind dafür konzipiert, Daten aus einer Datenbank oder auch aus anderen Datenquellen in Form von Reports für den Menschen zugänglich zu machen und verständlich, zumeist auch zusammengefasst, darzustellen.

[10] Frei übersetzt nach [w3s10]

Dabei werden grundsätzlich zwei Prinzipien unterschieden: Die einfach zu bedienenden, Query-basierten Reportgeneratoren nutzen als Datenquellen ausschließlich vorgefertigte Datensichten. Mit Kenntnis des Datenbankmodells können in Tabellen-basierten Reportgeneratoren die Datensichten hingegen individuell definiert werden. [rep10]

Zwar könnte man einen Reportgenerator dazu nutzen Geschäftsbriefe zu erzeugen, jedoch wäre dies im Sinne der Zielsetzung von Reportgeneratoren eine unzweckmäßige Nutzung. Zu dem ist es aufwändiger, mit Mitteln der Reporterzeugung Briefe zu gestalten.

2.4.6 Vorlagensysteme

Die letzte Gruppe ist die der vorlagenbasierten Dokumenterzeugungssysteme. Ähnlich wie bei XSL-FO werden hier Layout und Inhalt getrennt gehalten.

Auch wenn es bei den anderen Systemen möglich ist, Vorlagen zu erstellen und zu verwenden, sind die Funktionen der darauf spezialisierten Vorlagensysteme wesentlich ausgereifter. So lassen sich bereits die Vorlagen selbst in einem WYSIWYG-Modus beispielhaft betrachten.

Im folgenden Abschnitt werden diese Systeme näher vorgestellt.

2.5 Vorlagenbasierte Dokumenterzeugungssysteme

In diesem Abschnitt werden Vor- und Nachteile vorlagenbasierter Dokumenterzeugungssysteme benannt. Dazu werden zwei verschiedene Arten dieser Systeme am Beispiel erläutert.

2.5.1 Eigenschaften

Bekannterweise ist die Umsetzung eines Corporate Design für entstehende Dokumente durch das Verwenden von Vorlagen recht einfach, da hier das Design vom Inhalt getrennt gehalten wird. Jedoch ist die Annahme, dass Vorlagen eine reine Trennung von Aussehen und Inhalt ermöglichen, genauer betrachtet nicht ganz korrekt. Tatsächlich werden bei der Verwendung von Vorlagen drei Inhaltsmengen unterschieden:

Design/Layout Design und Layout der Vorlage umfassen alle Vorgaben bezüglich des Aussehens der späteren Dokumente. Darunter zählen neben Struktur und Farbgebung die genaue Positionierung von Inhalten, nicht aber diese selbst.

Statische Inhalte Statische Daten sind eine von zwei Informationsquellen, die Inhalte für die Dokumente liefern. Sie sollten vor Verwendung von der jeweiligen Fachabteilung auf Richtigkeit geprüft werden. Zur Vorlagenentwicklung feststehende Textbestandteile und Bilder bilden den inhaltlichen Rahmen der Vorlagen.

Dynamische Inhalte Die genauere inhaltliche Ausgestaltung und damit auch die Individualisierung geschieht durch dynamische Inhalte zum Zeitpunkt der Dokumentgenerierung. Dabei sollte zwischen Daten unterschieden werden, die automatisch zur Verfügung gestellt werden, und welchen, die manuell eingegeben werden müssen.

Da nur noch dynamische Inhalte beim tatsächlichen Erstellen der Dokumente eingefügt werden müssen, ist hier mit einem geringen Aufwand zu rechnen. Durch eine Anbindung an das Bestandssystem, welches situativ korrekt relevante Informationen aus dem fachlichen Kontext liefern kann, ist der Aufwand noch weiter reduzierbar. Natürlich bedeutet das auch eine gewisse Einschränkung in der Gestaltung der Dokumente, was aber nicht so stark ins Gewicht fällt.

Welche Vorlagen wie eingesetzt werden, hängt von den damit unterstützten Geschäftsprozessen und der Art der darin verwendeten Dokumente ab. (Siehe dazu Abschnitt 2.2.3 „Einordnung in den Workflow des Unternehmens“ auf Seite 8)

Damit die Anzahl der Vorlagen nicht ins Unermessliche steigt, dürfen sie nicht zu spezifisch ausgerichtet sein. Die Entwicklung allgemeiner Vorlagen, die auch mehrere mögliche Ausprägungen abdecken können, bedarf jedoch eines komplizierteren Vorbereitungs- und Konzeptionsprozesses. Deshalb gilt es, im Vorfeld eine optimale Aufteilung der geplanten Dokumentinhalte in statische und dynamische Teile zu finden.

Meist nutzen die Lösungen ein zentrales Repository. So muss eine Vorlage nur einmalig erstellt werden und ist damit auf allen Clients verfügbar. Änderungen werden ebenfalls an zentraler Stelle ausgeführt, was auch den Wartungsaufwand relativ gering hält.

2.5.2 Individuelle Lösung

Eine mögliche Herangehensweise ist die Neuentwicklung eines solchen Systems von Grund auf. Hierbei kann auf alle Bedürfnisse des Kunden exakt eingegangen werden. Änderungen sind im gesamten System problemlos möglich, da alle Komponenten Eigenentwicklungen sind.

Der Entwicklungsaufwand ist dabei natürlich wesentlich höher als bei der Verwendung eines produktbasierten Ansatzes, wie er in Abschnitt 2.5.3 „Generische, produktbasierte Lösung“ auf der nächsten Seite vorgestellt wird. Mit der Anzahl selbst entwickelter Komponenten steigt auch die Verantwortung für das fehlerfreie Arbeiten des zunehmend komplexeren Systems.

Das Output Management System eines großen Versicherungsunternehmens[11] ist eine solche Individualentwicklung. Das komplexe System besteht aus folgenden wichtigen Komponenten, die für die Dokumenterzeugung relevant sind:

Library Manager Im *Library Manager* werden Vorlagen, Formulare und Textbausteine erstellt und verwaltet.

DatenDrehScheibe Die *DatenDrehScheibe* bildet die Variablen des OMS auf die des Bestandssystems ab.

Scout Die Dokumente können vom Sachbearbeiter mit dem Frontend *Scout* erzeugt werden.

Batch Eine automatische Erzeugung von Massendokumenten, die keinen manuellen Eingriff erfordert, wird vom Modul *Batch* erledigt.

Formatierungsservice Die tatsächliche Formatierung der Dokumente bei der Erzeugung oder zur Vorschau wird zentral in diesem Service erledigt.

2.5.3 Generische, produktbasierte Lösung

Im Gegensatz zur gerade vorgestellten individuellen Entwicklung steht die *Adobe LiveCycle Enterprise Suite*. Adobe selbst beschreibt das Softwarepaket wie folgt:

> *Adobe LiveCycle ES* ist eine Gruppe von Softwarelösungen für die Entwicklung intuitiver Anwendungen und Umsetzung effizienter Prozesse, die die Produktivität von Unternehmen und Organisationen nachhaltig steigern. [adl10]

Ein Teil dieser Suite bietet einen generischer Ansatz zur vorlagenbasierten Dokumenterzeugung. Das bedeutet nicht etwa, dass hier ein Stand-Alone System geliefert wird, welches nach seiner Installation sofort eingesetzt werden kann. Vielmehr stellt das Paket eine Grundlage bestehend aus Vorlagendesigner, Rendering-Engine und einer API dar, auf deren Grundlage man ein individuell gestaltetes System aufbauen kann. Abbildung 2.3 „*Adobe LiveCycle* Suite, Komponenten“ auf der folgenden Seite zeigt die Komponenten im Gesamtpaket.

Das Funktionsprinzip ist einfach: Im Designer können Vorlagen grafisch angelegt und bearbeitet werden. Die Ablage der Vorlage erfolgt dann in einem zentralen Repository. Bei der Dokumenterzeugung werden über eine XML-Datenstruktur dynamische Inhalte zugeführt. Ergebnis ist ein fertiges Dokument im PDF-Format.

Der Vorteil produktbasierter Lösungen besteht im Allgemeinen darin, dass man auf bereits vorhandene Produktbestandteile aufbauen und so den Eigenentwicklungsaufwand verringern kann.

[11] wird hier aus Vertraulichkeitsgründen nicht namentlich benannt

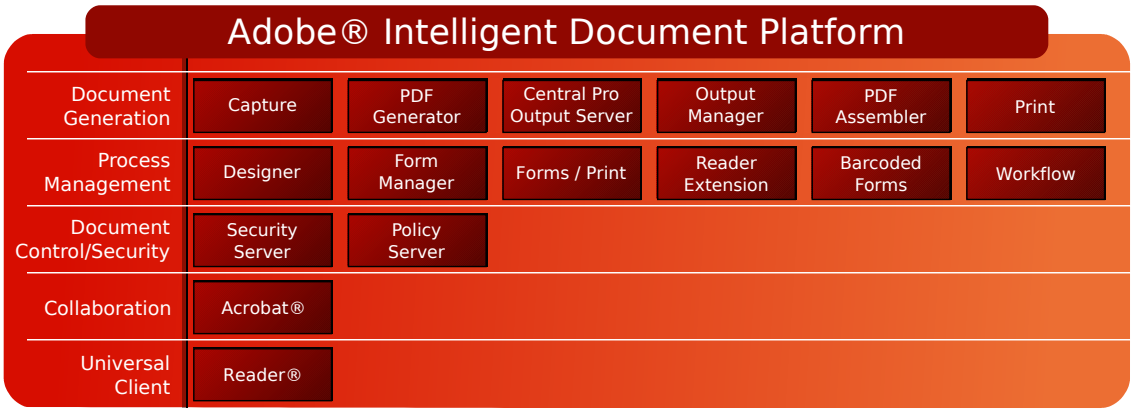

Abbildung 2.3: *Adobe LiveCycle* Suite, Komponenten

Zwischen beiden Ansätzen sind auch Mischformen denkbar. Welche der genannten Ansätze für eine konkrete Situation am besten passt, ist von vielen Faktoren abhängig. Deren Darstellung ist jedoch nicht Gegenstand der Arbeit.

2.6 Dokumentgenerator DokGen

Die Firma msg systems ag betreut zur Zeit ein Kundenprojekt, in welchem Vorlagen zur Dokumenterzeugung eingesetzt werden. Aus Sicht der obigen Darstellung handelt es sich dabei um eine Mischform. Die Anwendung nutzt als Basis einen generischen Ansatz. Trotz der bereits gut ausgereiften Funktionalität der *Adobe LiveCycle ES*, wie in Abschnitt 2.5.3 „Generische, produktbasierte Lösung“ auf der vorhergehenden Seite kurz angerissen, ergaben sich in Hinblick auf die Vorstellungen des Kunden jedoch einige funktionale Lücken beim Einsatz der Software. Diese wurden durch einen individuellen Aufsatz gefüllt.

2.6.1 Erweiterung des einfachen Vorlagenkonzepts

Mit der Grundfunktionalität des Adobe-Produkts war es bereits möglich, editierbare PDF-Dokumente zu erzeugen. Der Kundenwunsch, vorgefertigte Textbausteine bei der Erzeugung zu nutzen, offenbarte eine der funktionalen Lücken. Um diese zu beseitigen, wurde eine Erweiterung des herkömmlichen Vorlagenkonzepts eingeführt. Textfelder, die auf der Vorlage existieren, können wahlweise mit Freitext oder vorgegebenen Textbausteinen befüllt werden. Das bietet gegenüber der reinen Freitextvariante den Vorteil, dass neben dem Corporate Design, welches durch die Vorlagen erreicht wird, auch ein Corporate Wording/Legal Wording möglich ist.

Um dabei die Flexibilität nicht zu stark einzugrenzen, können in den Textbausteinen und Freitextbestandteilen Variable verwendet werden. Diese entsprechen inhaltlich den gleichnamigen Textfeldern.

Die notwendigen Informationen zu Textfeldern, Textbausteinen, Vorlagen und der Verknüpfungen dieser untereinander müssen erfasst und verwaltet werden. Zur Erfassung wird die in Abbildung 2.4 „Datenbank, Darstellung der Struktur“ gezeigte Struktur verwendet. Um dabei die korrekte Zuordnung der Textbausteine zu den vorhandenen Textfeldern zu gewährleisten, muss bei der Entwicklung der Vorlagen eine bestehende Namenskonvention eingehalten werden.

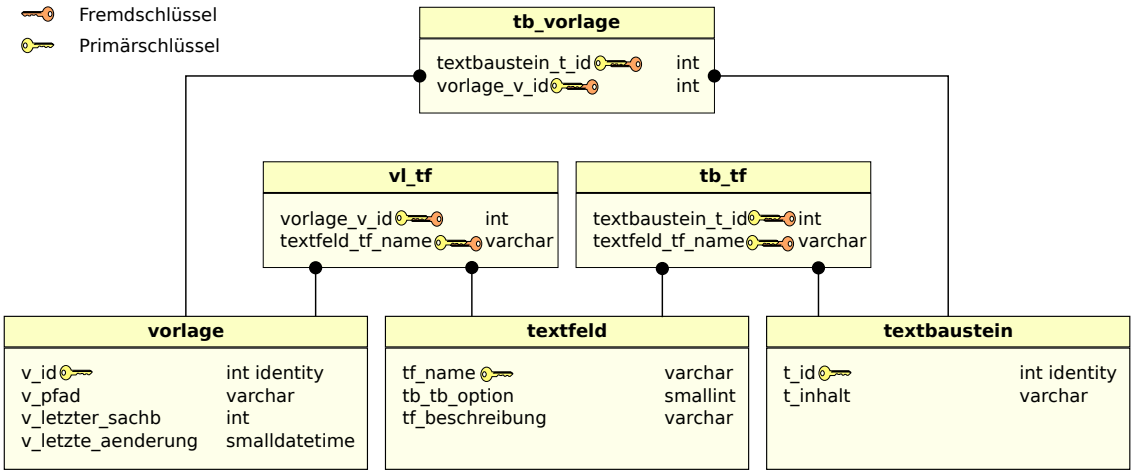

Abbildung 2.4: Datenbank, Darstellung der Struktur

2.6.2 Benutzeroberfläche

Die *Adobe*-Lösung bietet zwar die Möglichkeit, dynamischen Inhalt in eine Vorlage zu rendern, jedoch ist die Eingabe dieser Daten nur per XML möglich. Eine Oberfläche, die der Endbenutzer verwenden könnte, um diese Daten zu erfassen, existiert dazu jedoch nicht.

Diese funktionale Lücke wird durch den *DokGen* behoben. Seine zweigeteilte Oberfläche bietet auf der linken Seite die Möglichkeit, passende Textbausteine für die Befüllung des Dokuments auszuwählen. Auf der rechten Seite befindet sich eine Vorschauanzeige des Dokuments, die Änderungen der Textbausteine zeitgleich übernimmt. Freitexte können direkt in dieser Vorschauanzeige eingetragen werden.

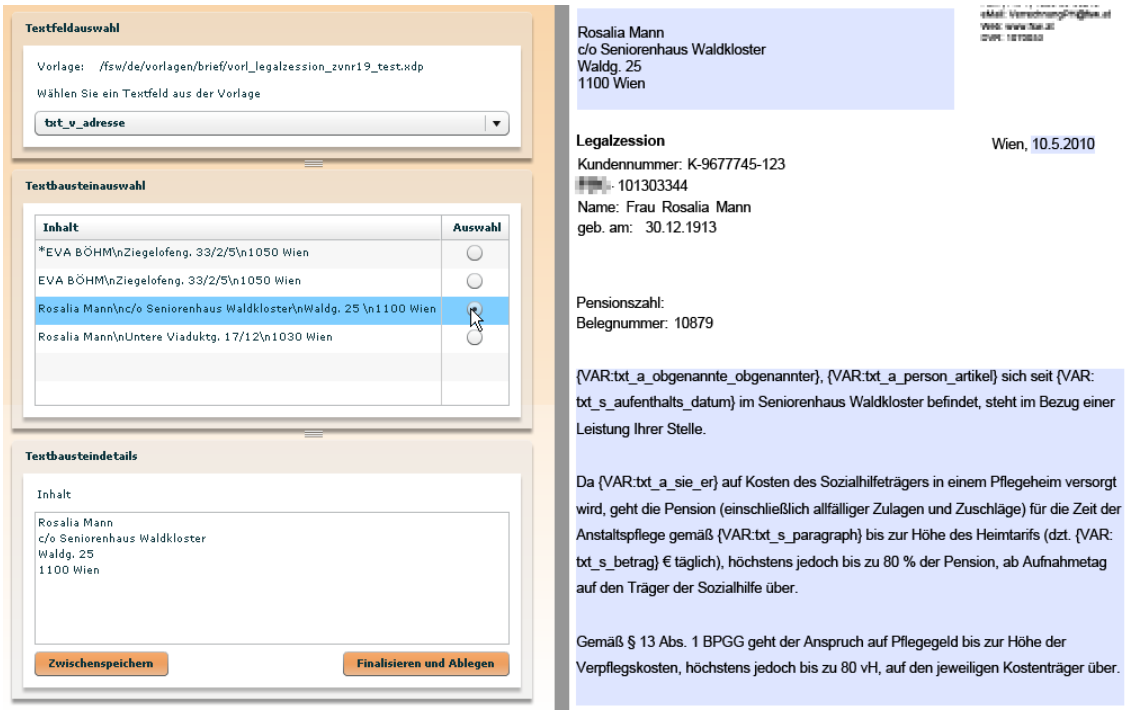

Abbildung 2.5: *DokGen*, Oberfläche

Bei den beiden Oberflächenteilen handelt es sich um zwei separate Elemente auf einer gemeinsamen HTML-Seite. Die Webseite als Container kann als Mittler zwischen den beiden Komponenten verwendet werden, um eine bidirektionale Kommunikationsbeziehung aufzubauen. Diese ist notwendig, um die Inhalte von Vorschau und Programm synchron zu halten.

Die Anwendung wird als Rich Internet Application (kurz RIA) entwickelt: Sie muss folglich nicht lokal installiert werden, sondern kann innerhalb des Firmenintranets im Webbrowser gestartet werden. Trotzdem bietet sie die Funktionalitäten einer traditionellen Desktop-Applikation[12]. Siehe dazu Abbildung 2.5 „*DokGen*, Oberfläche“.

2.6.3 Kommunikation mit dem bestehenden System

Weiterer wichtiger Punkt ist die Anbindung des Dokumentgenerators an das bestehende System des Kunden. Aus diesem heraus sollte der Dokumentgenerator aufrufbar sein. Das lässt sich recht einfach realisieren, da nur eine URL im Webbrowser aufgerufen werden muss. Diese Funktion wurde im bestehenden System implementiert.

Die mit den Vorlagen zu verbindenden Daten, die im Kundensystem existieren, müssen bei der Dokumenterzeugung abgreifbar sein. Dazu zählt unter anderem der fachliche Kontext. Um dies umzusetzen, wurde ein unidirektionaler Kommunikationsansatz

[12] nach [net10]

gewählt. Relevante Datensätze für das entstehende Dokument werden in einer XML-Datenstruktur an den Dokumentgenerator weitergegeben. Eine bidirektionale Kommunikation wird hier deswegen nicht bevorzugt, da sie zum einen neue Abhängigkeiten schafft und zum anderen der Zugriff auf die Originaldaten des Bestands nicht erwünscht ist.

2.6.4 Unterstützte Anwendungsfälle

Durch das System wird eine Reihe von Anwendungsfällen (Use Cases) unterstützt. Die Wichtigsten sollen hier vorgestellt werden:

2.6.4.1 Anwendungsfall-Gruppe: Vorlagenentwicklung

Akteur dieser Use Cases ist der Vorlagenentwickler. Er erstellt und wartet die Vorlagen des Unternehmens, wie in Abbildung 2.6 „Designprozess im Überblick“ grob dargestellt wird.

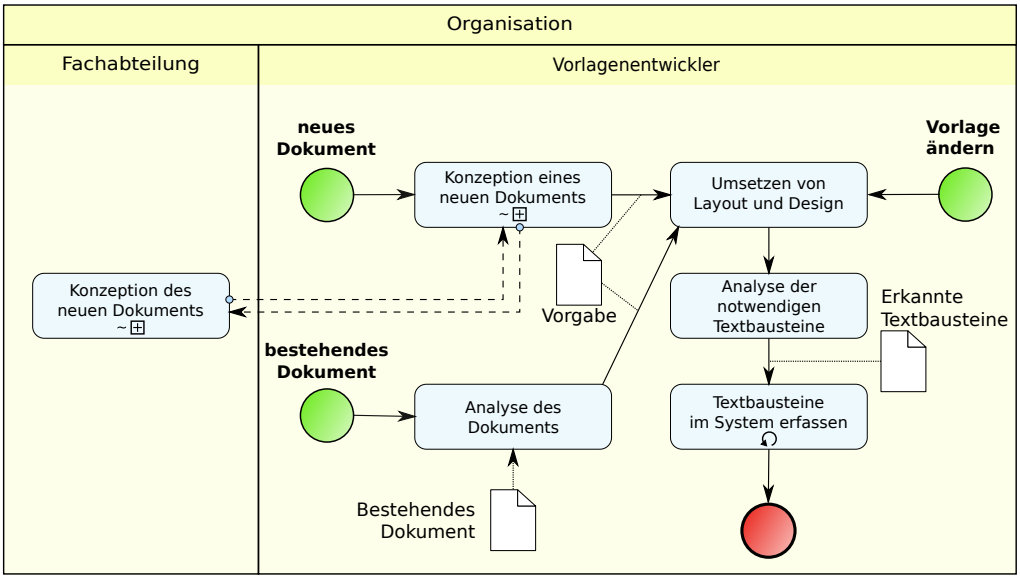

Abbildung 2.6: Designprozess im Überblick

Entwicklung eines bestehenden Dokuments Es existiert bereits ein Geschäftsdokument, welches bisher ohne die Verwendung von Vorlagen produziert wurde. Der Vorlagendesigner muss das bestehende Dokument als Grundlage nehmen, um eine entsprechende Vorlage zu erstellen. Ihm obliegt dabei zu analysieren, welche Bestandteile des Dokuments statisch und welche dynamisch realisiert werden. Nach dem eigentlichen Vorlagendesign muss er den entstandenen Textfeldern auf der Vorlage Textbausteine zuordnen. Dazu ist es notwendig, anhand meh-

rerer Beispieldokumente und in theoretischer Überlegung zu bestimmen, welche Textvarianten nötig und möglich sind. Diese muss er anschließend - sofern noch nicht existent - im System als Textbausteine anlegen und zuordnen.

Entwicklung eines neuen Dokuments Es soll eine Vorlage zu einem Geschäftsdokument erstellt werden, welches es bisher noch nicht gab. Grundlage für die Arbeit kann hier nur eine Konzeption zu dem neuen Dokument sein. Abhängig von der Genauigkeit dieser Konzeption ist die Intensität der Rücksprachen, die gehalten werden müssen. Im Übrigen gleicht dieser Use Case dem vorigen.

Änderung einer bestehenden Vorlage In der Regel liegen konkrete Änderungswünsche vor. Diese müssen zuerst im Designer umgesetzt werden. Haben sich dabei Textfelder auf der Vorlage verändert, so müssen diese Änderungen auch für die Zuordnung in der Datenbank übernommen werden. Anschließend können Modifikationen an Textbausteinen ebenfalls in der Datenbank vorgenommen werden. Darunter zählen Anlegen und Löschen genauso wie auch die Änderung von Textbausteinen und deren Zuordnung.

2.6.4.2 Anwendungsfall-Gruppe: Dokumenterstellung

Nachdem nun die Vorlagen existieren, kann der Sachbearbeiter wie in Abbildung 2.7 „Dokumenterzeugung im Überblick“ gezeigt Dokumente erzeugen.

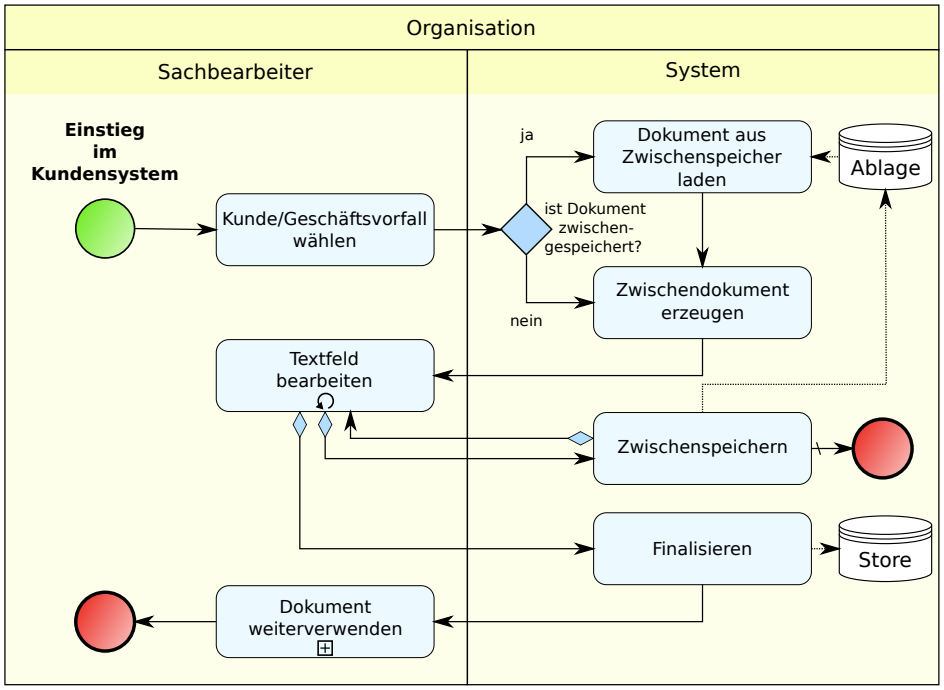

Abbildung 2.7: Dokumenterzeugung im Überblick

Erstellen eines Dokuments Ein Geschäftsdokument soll erzeugt werden. Dazu wählt der Sachbearbeiter im Bestandssystem die Art des Dokuments sowie den inhaltlichen Kontext. Anschließend wird automatisch die Oberfläche des *DokGen* mit der entsprechenden Parametrisierung gestartet. Hier kann der Sachbearbeiter die genaue inhaltliche Ausgestaltung des Dokuments durch die Auswahl von Textbausteinen und die Eingabe von Freitexten festlegen. Anschließend kann die resultierende PDF-Datei abgespeichert und gedruckt oder verschickt werden. Bis dahin bearbeitbare Textfelder im Dokument werden an dieser Stelle versiegelt, sodass die Unveränderbarkeit sichergestellt ist.

Wiederaufnahme der Bearbeitung Wurde das Erstellen des Dokuments durch das Zwischenspeichern unterbrochen, so kann das zwischengespeicherte Dokument wiederaufgenommen werden. Das Bestandssystem bietet dem Sachbearbeiter eine Liste mit relevanten Dokumenten für die Wiederaufnahme an. Nachdem er eines davon ausgewählt hat, wird der Dokumentgenerator gestartet. Wurde der Arbeitsstand vollständig geladen, kann die Bearbeitung im unterbrochenen Use Case „Erstellen eines Dokuments“ fortgesetzt werden.

2.6.5 Schwachstellen der bisherigen Lösung

In den ersten drei Use Cases wird die Arbeit mit der Datenbank angesprochen. Hier liegt eine Schwachstelle des bisherigen Systems. Die Verwaltung der relevanten Information zur Arbeit mit Textbausteinen ist noch nicht durch die Software unterstützt. Folglich muss der Vorlagenentwickler hier direkt mit der Datenbank interagieren. Das birgt das Risiko von unbemerkten Fehlern. In der Vielzahl der existierenden Einträge ist es schwer, die Übersicht zu behalten. Schnell werden Einträge, die noch geändert werden müssen, übersehen. Dem ist die Verwendung von ID’s in der Datenbank nicht unbedingt zuträglich.

Ein weiterer Schwachpunkt ist die fehlende Unterstützung des Vorlagenentwicklungsprozesses. Im Moment werden Änderungen an Vorlagen und Textbausteinen sofort produktiv übernommen. Ein Staging oder ähnliches existiert nicht. Es ist jedoch erforderlich, die Vorlagen und Textbausteine einem Review zu unterziehen.

3 Fachliche Spezifikation der Anforderungen des TEd

Im letzten Kapitel wurden einige Schwachstellen der bisherigen Lösung *DokGen* offengelegt. Um diese zu beheben, soll eine neue Anwendung entstehen. Der sogenannte Textbausteineditor (kurz *TEd*) soll die komfortable Verwaltung der Textbausteine ermöglichen und den Entwicklungsprozess der Vorlagen unterstützen. In diesem Kapitel soll der genaue fachliche Umfang des *TEd* festgelegt werden.

3.1 Analyse von Vorlagen

Bisher wurden alle Informationen zu Vorlagen, Textfeldern und Textbausteinen per Hand in die Datenbank eingetragen. Dazu musste der Vorlagenentwickler eine Liste von Textfeldern anlegen, die er in der Vorlage verwendet. Diese Textfelder hat er anschließend in der Datenbank angelegt, falls sie noch nicht existierten, und anschließend in einem zweiten Schritt der jeweiligen Vorlage zugeordnet. Wurden Textfelder wieder von einer Vorlage entfernt, so musste auch per Hand die entsprechende Verbindung zwischen Textfeld und Vorlage in der Datenbank gelöst werden.

Die Vorlagen liegen im Format XDP vor. Dieses XML-basierte Format enthält bereits alle nötigen Informationen, um die Zuordnung von Textfeldern zu den jeweiligen Vorlagen automatisieren zu können. Diese müssen dazu aber noch „ausgelesen“ werden. Die Strukturanalyse der Vorlagen übernimmt diese Aufgabe.

Primäres Ergebnis der Strukturanalyse einer Vorlage ist die Liste mit Textfeldern, die auf der Vorlage und den verwendeten Fragmenten eingebaut sind. Die Namen finden sich nicht nur in der Zuordnung der Textfelder wieder, sondern auch in der XML-Datenstruktur, die vom bestehenden System Daten an den *DokGen* übersenden wird.

Bedingt durch den hierarchischen Aufbau der Vorlage aus Fragmenten muss die Analyse rekursiv auf alle in einer Vorlage referenzierten Fragmente angewendet werden. So entsteht nebenher noch eine Liste mit referenzierten Fragmenten, die für die korrekte Verwendung der Vorlage notwendig sind.

Optional dazu kann die Vorlage während der Strukturanalyse auf bestimmte Fehler untersucht werden. So besteht die Möglichkeit zu prüfen, ob die Benennung eines Textfeldes und deren Eigenschaften zur geltenden Namenskonvention passen.

3.2 Textbausteinverwaltung

Nicht für alle Textfelder ist eine Befüllung mit Textbausteinen vorgesehen. Die verschiedenen Arten von Textfeldern ergeben sich aus der verwendeten Namenskonvention.

Festgelegte Textfelder bekommen ihre Belegung durch einen festen Wert, der vom aufrufenden System als Parameter übergeben wird. Eine Auswahl an Werten in der Oberfläche ist hier weder nötig noch möglich.

Normale Textfelder haben das Verhalten, welches im Rahmen dieser Arbeit relevant ist. Sie werden mit Textbausteinen aus der Datenbank befüllt, die das Hauptaugenmerkt der Textbausteinverwaltung sind.

Virtuelle Textfelder sind eine Erweiterung dazu. Im Gegensatz zu normalen Textbausteinen werden die Werte virtueller Textbausteine durch das Bestandssystem übertragen.

Eingabefelder können bei der Gestaltung von elektronischen Formularen eingesetzt werden. Sie bleiben auch nach dem Finalisieren noch bearbeitbar. Da diese Felder dazu gedacht sind, manuell ausgefüllt zu werden, werden sie nicht mit Textbausteinen gefüllt.

Nachdem ein kleiner Überblick über die tatsächliche Verwendung von Textbausteinen gegeben wurde, kann nun betrachtet werden, was die Verwaltung alles leisten muss. Neben den CRUD[13]-Operationen ist hier die Zuordnung der Textbausteine zu den Textfeldern wichtig.

Anlegen/Bearbeiten Das Anlegen und Bearbeiten von Textbausteinen ist trivial, da ein Textbaustein nichts weiter ist als eine normale Zeichenkette. Da die Benennung der Variablen sich aus den im System bekannten Textfeldern ergibt, kann das Einfügen von Variablen bei der Eingabe eines Textes unterstützt werden.

Löschen Das Löschen von Textbausteinen beinhaltet eine Schwierigkeit, die schnell übersehen werden könnte. Ein Textbaustein kann von mehreren Textfeldern verwendet werden. Das unbedachte Löschen kann hier ungewollte Auswirkungen auf andere Vorlagen haben. Vor dem Löschen eines Textbausteins muss also sichergestellt werden, dass solche Nebenwirkungen nicht auftreten können.

Zuordnen Aus der Menge der entstandenen Textbausteine wird nun jedem Textfeld eine Untermenge davon zugeordnet. Hierbei besteht die Möglichkeit, einen Textbaustein einer oder mehrerer Vorlagen exklusiv zuzuweisen. Diese Zuweisung muss im Programm genau so möglich sein, wie auch die Anzeige eines Hinweises, falls ein Textbaustein zwar dem aktuellen Textfeld zugeordnet ist, er auf der aktuellen Vorlage jedoch nicht verwendet werden kann.

[13] Create, Read, Update, Delete

3.3 Vorgehensmodell

Es ist im Sinne der Qualitätssicherung nicht praktikabel, dass Änderungen an Vorlagen, Textfeldern und Textbausteinen sofort im produktiven System aktiv werden. So liegt es im Interesse des Unternehmens, dass Vorlagen und davon verwendete Bestandteile einem geregelten Ablauf unterliegen. Dieser stellt sicher, dass nur Vorlagen, die den Ansprüchen des Unternehmens genügen, bewusst zur produktiven Verwendung freigegeben werden.

Zu den Ansprüchen des Unternehmens, deren Erfüllung garantiert werden soll, gehören unter anderem:

- Korrektes Layout im Corporate Design
- Rechtskonforme Formulierungen - Legal Wording
- Fehlerfreiheit in Rechtschreibung und Grammatik
- Fachliche Korrektheit der entstehenden Dokumente

Das Vorgehensmodell muss vom Textbausteineditor geeignet unterstützt werden. Um diese Unterstützung zu implementieren, ist es jedoch notwendig, den Prozess im Vorfeld genauer zu spezifizieren. Dies geschieht im nächsten Kapitel.

3.4 Integration in den Entwicklungsablauf

Die Entwicklung einer Vorlage bedient sich mehrerer Anwendungen. Angefangen mit dem Vorlagendesigner und dem entstehenden Textbausteineditor bis hin zur Testoberfläche, welche aus dem Umsystem heraus gestartet werden kann. Um den Arbeitsablauf so flüssig wie möglich zu gestalten, ist es ratsam, darauf zu achten, dass möglichst wenige Medienbrüche entstehen.

Bei der Entwicklung des TEd muss folglich darauf geachtet werden, dass sich das Tool so in den Ablauf integriert, dass dadurch kein Medienbruch entsteht. Idealerweise könnte der Textbausteineditor sogar den vorhandenen Medienbruch zwischen Entwicklung und Test einer Vorlage überbrücken und somit beseitigen.

4 Analyse und fachliche Spezifikation des Entwicklungsprozesses

Die Wiederverwendung vorhandener Methoden und Strukturen ist ein wichtiges Prinzip der Softwareentwicklung. So könnten auch die vorhandenen Vorgehensmodelle der Softwareentwicklung auf weitere Gebiete übertragen werden.

Auf den ersten Blick sieht es zwar nicht so aus, als ob man die Entwicklung von Dokumentvorlagen mit dem Softwareentwicklungsprozess vergleichen könnte. Genauer betrachtet, besteht eine Dokumentvorlage aber - ähnlich wie Software - auch aus mehreren, zumeist untereinander abhängigen Komponenten (Texte, Grafiken, Skripte, Fragmente, Textbausteine). Eine Qualitätsprüfung ist ebenfalls unabdinglich. Wenngleich auch die verbundenen Aufwände bei der Vorlagenentwicklung kleiner sind als bei Design und Erstellung von Software-Komponenten, Funktionen und Datenmodellen, so ähneln sich doch beide Prozesse.

Daraus ergibt sich der Gedanke, die Vorgehensmodelle der Softwareentwicklung auf die Vorlagenentwicklung anzuwenden. Um ein geeignetes Vorgehensmodell zu finden, werden im Folgenden einige dieser bestehenden Vorgehendmodelle in Hinsicht auf die Anwendbarkeit auf Vorlagenentwicklung betrachtet.

4.1 Vorgehensmodelle

In den Anfangstagen der Softwareentwicklung wurde nach dem Prinzip Code&Fix vorgegangen. Das bedeutet nichts weiter als das einfache Drauflosprogrammieren. Fehler, die sich im Betrieb der Softwarelösung fanden, wurden nachträglich behoben. Dabei verlor die ohnehin nur schwach strukturierte Software meist noch weiter an Struktur, was die Fehlerbehebung in aller Regel erschwerte.

So erkannte man die Notwendigkeit, der Softwareentwicklung einen festgelegten organisatorischen Rahmen zu geben. Dieser Rahmen wird von den Vorgehensmodellen der Softwareentwicklung beschrieben. Er besteht unter anderem aus folgenden Festlegungen[14]:

- Reihenfolge des Arbeitsablaufes (Entwicklungsstufen, Phasenkonzepte)
- Durchzuführende Aktivitäten
- Definition von Teilprodukten
- Fertigstellungskriterien

[14] [Bal98], S. 98

- Notwendige Mitarbeiterqualifikationen
- Verantwortlichkeiten und Kompetenzen
- Anzuwendende Standards, Richtlinien, Methoden und Werkzeuge.

Im Laufe der Zeit ist eine Vielzahl verschiedener solcher Vorgehensmodelle entstanden, welche die unterschiedlichsten Ausprägungen der einzelnen Festlegungen beinhalten. Diese können in ein Raster aus Vorgangsflexibilität und Phasen- bzw. Iterationsprinzip eingeordnet werden. In Abbildung 4.1 „Einordnung der Vorgehensmodelle, angelehnt an [pma07], zitiert nach [Bil09]“ wird diese Einordnung beispielhaft gezeigt.

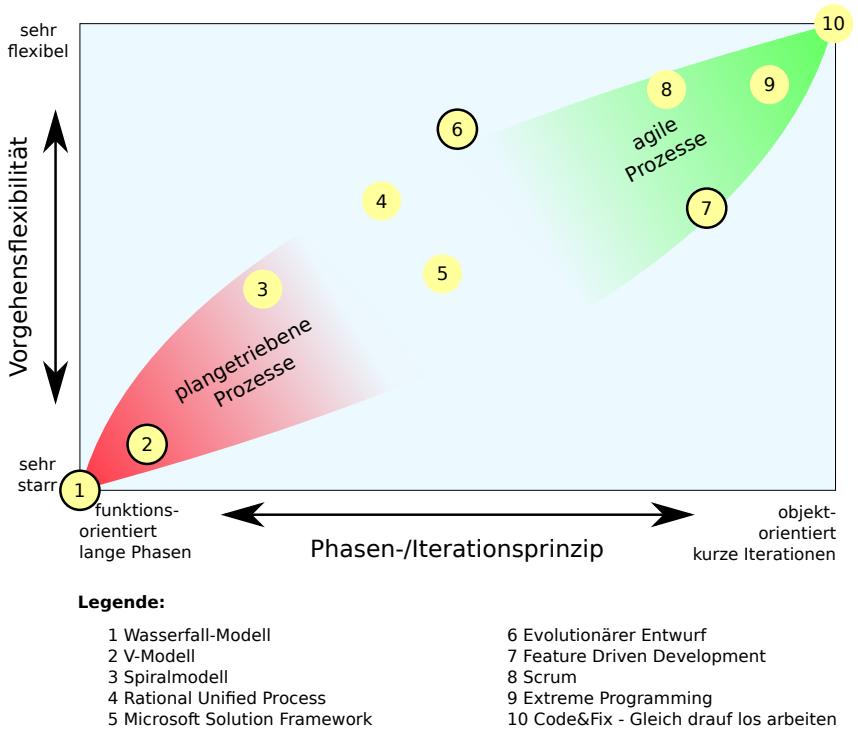

Abbildung 4.1: Einordnung der Vorgehensmodelle, angelehnt an [pma07], zitiert nach [Bil09]

Für die weitere Betrachtung werden eher plangetriebene Modelle herangezogen. Die davon in der Abbildung vorhandenen sind markiert.

4.2 Anforderungen

Das Ergebnis dieser Betrachtung soll die Beschreibung eines Vorgehensmodells sein, welches dann von Sachbearbeiter und Vorlagenentwicklern im Geschäftsalltag angewendet werden muss. Dazu ist es essentiell, die Vorstellungen des Kunden in die Betrachtung mit einzubeziehen.

Laut Projektanforderung muss eine entstehende Vorlage vier definierte Stadien durchlaufen, wobei es nicht notwendig ist, mehr als eine Version einer Vorlage pro Stadium vorzuhalten:

dev: Vorlagen befinden sich während ihrer Entwicklung in diesem Stadium.

test: Umfasst Vorlagen, die auf Funktionsfähigkeit getestet werden sollen. Dieses Zwischenteststadium wird in engem Zusammenhang mit der Entwicklung stehen.

qm: Eine tiefergreifende Form des Tests ist das Quality Management. Hier werden die Vorlagen und ihre Textbausteine auf fachliche und rechtliche Korrektheit geprüft. Man kann auch von einer Art Abnahmeprüfung sprechen, nach der die Vorlage produktiv gesetzt werden kann.

prod: Nach dem Bestehen des QM befindet sich eine Vorlage im Stadium „produktiv“. Die Dokumenterzeugung bedient sich ausschließlich der produktiven Vorlagen.

4.3 Betrachtung der Konzepte

4.3.1 Klassisches Wasserfall-Modell

Das erste aus den beschriebenen Mängeln hervorgegangene Vorgehensmodell ist das von Benington 1956 entwickelte Stagewise Model. Seine Phasen werden sequenziell durchlaufen. Der Benutzer wird nur zu Beginn und am Ende mit der Software konfrontiert. Dieses Modell erweist sich als sehr starr und unflexibel, da zum einen der Start einer Phase die vollständige Beendigung der vorhergehenden Phase voraussetzt und zum anderen keine Rückschritte zur Korrektur vorgesehen sind. [swl06]

Das Wasserfall-Modell ist eine Weiterentwicklung dieses Modells. Bei Betrachtung der Abbildung 4.2 „Darstellung Wasserfall-Modell nach [Bal98]“ auf der nächsten Seite wird deutlich, woher der Name stammt. Das Stagewise Model wird um Rückkopplungsschleifen zwischen den einzelnen Stufen erweitert. Hierbei ist die Einschränkung zu beachten, dass nur direkt benachbarte Stufen rückgekoppelt werden können. Das soll teuren stufenübergreifenden Arbeitsschritten vorbeugen.[15]

Die Abarbeitung ist jedoch immer noch von sequentieller Natur, sodass eine Aktivität, ungeachtet der Bewegungsrichtung im Wasserfall, erst beginnen kann, wenn die vorige abgeschlossen wurde.

[15] [Bal98], S.99

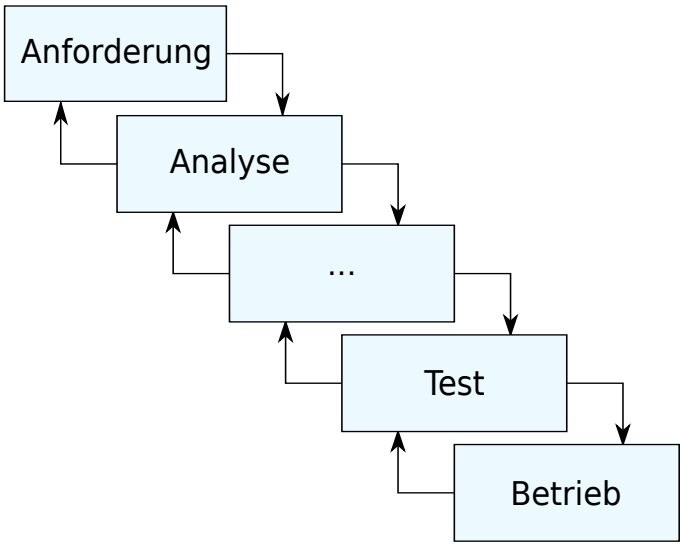

Abbildung 4.2: Darstellung Wasserfall-Modell nach [Bal98]

Die Interaktion mit dem Auftraggeber wurde nicht erweitert. Auch ist es nicht immer möglich, alle Phasen in vollem Umfang und wie gefordert sequenziell auszuführen. Trotzdem hat das Wasserfall-Modell eine weite Verbreitung gefunden. Dadurch konnte es wesentlich zu einem disziplinierten, sichtbaren und kontrollierbaren Prozessablauf beitragen.[16] Außerdem bildet es eine Grundlage für weitere Vorgehensmodelle.

Der geringe Managementaufwand, der mit dem Wasserfall-Modell verbunden ist, wäre ideal für einen leichtgewichtigen Entwicklungsprozess. Fehlende Möglichkeiten der Qualitätssicherung schließen dieses Vorgehensmodell aber grundsätzlich aus, da diese wichtige Anforderung nicht erfüllt wird.

4.3.2 V-Modell

Die Grundidee des V-Modells ist die Erweiterungen des Wasserfall-Modells um Aspekte der Qualitätssicherung. So sind Verifikation und Validation Bestandteile des V-Modells. Die Unterscheidung der beiden Begriffe sollte beachtet werden. Verifikation ist die Überprüfung der Übereinstimmung zwischen dem Softwareprodukt und seiner Spezifikation. Es wird also die korrekte Umsetzung der Spezifikation betrachtet. Durch Validation wird die Eignung eines Produkts in Bezug auf seinen Einsatzzweck bewertet. Es wird also festgestellt, ob das Produkt seinen Einsatzzweck erfüllen kann.[17] Das Ergebnis dieser Erweiterung lässt sich wie in Abbildung 4.3 „Darstellung V-Modell nach [Bal98]“ auf der folgenden Seite in Form eines V darstellen.

[16] [Bal98], S. 101
[17] [Bal98], S. 101/102

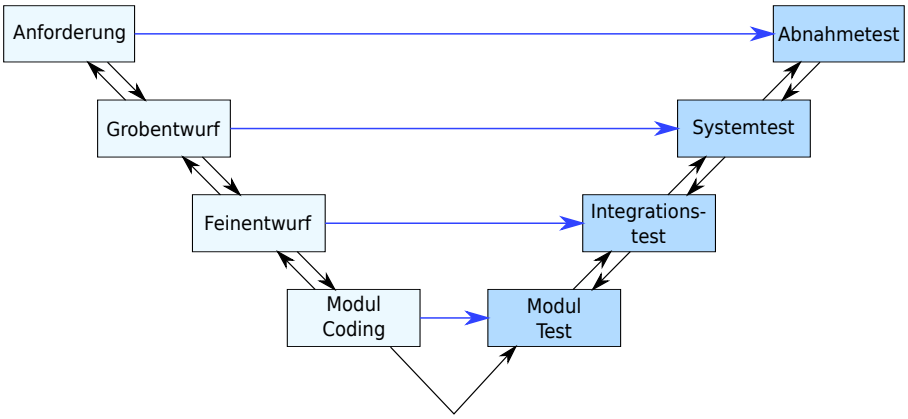

Abbildung 4.3: Darstellung V-Modell nach [Bal98]

Den aktuellsten Stand der Kunst stellt das V-Modell XT dar, welches 2005 als Ablösung und damit auch Verbesserung des V-Modell 97 entstand. Durch Erfahrungen und Kritik, die mit dem alten Modell gesammelt werden konnten, wurden Verbesserungsvorschläge erarbeitet, die die Akzeptanz und die effektive Nutzung des neuen Modells verbesserten.[18] So stellte das V-Modell XT Entwicklungsstandard für IT-Systeme der öffentlichen Hand in Deutschland dar. [vxt06]

Der Schutz vor wirtschaftlichen Verlusten kann in diesem Modell durch die Aspekte der Qualitätssicherung deutlich verbessert werden. Die vorhandenen Projekttypen des V-Modell XT, die komplexe Softwareentwicklung im Zusammenspiel von Auftraggeber und Auftragnehmer abdecken, scheinen für die Vorlagenentwicklung zu umfangreich.

4.3.3 Prototyping

Bei den traditionellen Vorgehensmodellen existiert ein wichtiges Problem, das nicht sofort ins Auge fällt: Der Auftraggeber ist nur zu Beginn und am Ende des Prozesses involviert. Nachdem er seine Anforderungen spezifiziert hat, tritt er in den Hintergrund. Erst zur Fertigstellung des Produkts wird er wieder aktiv. Meist sind aber die Auftraggeber nicht in der Lage, ihre Anforderungen explizit und vollständig zu definieren, da sie einen anderen Blick als das Entwicklerteam haben, folglich auch andere Dinge voraussetzen. Die wechselseitige Kommunikation zwischen Entwickler und Auftraggeber kann beiden Vorteile bringen.[19]

Hier schafft das Prinzip des Prototyping Abhilfe. Es unterstützt die frühzeitige Erstellung lauffähiger Prototypen des zukünftigen Produkts. Mit ihrer Hilfe können praktische Erfahrungen gesammelt, sowie Anforderungen und Entwicklungsprobleme geklärt werden. Ebenfalls helfen sie als Diskussionsbasis bei der Entscheidungsfindung.[20]

[18] [vxt09], S. 7
[19] [Bal98], S.114
[20] [Bal98], S. 115

Betrachtet man den exemplarische Programmaufbau in Abbildung 4.4 „Prototyping nach [Bal98]“, so erkennt man die zweidimensionale Ausprägung des Programms. Dies ermöglicht die Anfertigung zweier verschiedener Prototypenarten. Ein horizontaler Prototyp beinhaltet dabei eine Ebene, wie zum Beispiel die Benutzeroberfläche. Ein vertikaler Prototyp hingegen implementiert ausgewählte Teile auf allen betroffenen Ebenen.[21]

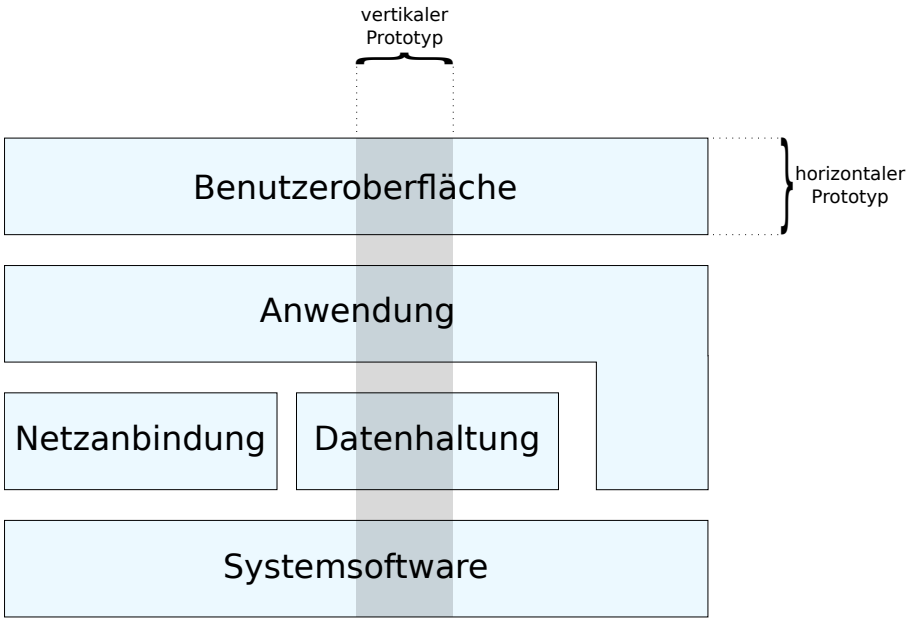

Abbildung 4.4: Prototyping nach [Bal98]

Durch den frühzeitigen Einsatz von Prototypen und das damit verbundene schnelle Feedback sinkt das Entwicklungsrisiko. Falsche Ansätze und Abweichungen zwischen Produktanforderung und tatsächlicher Vorstellung werden früh erkannt. Eine Integration in andere Modelle ist möglich. Die Anfertigung von Prototypen erhöht allerdings den Entwicklungsaufwand.[22] Prototypen können in der Vorlagenentwicklung verwendet werden. Die reine Vorlage ohne Unterbau von Textbausteinen, Benennung von Textfeldern und Verwendung von Skripten wäre ein horizontaler Prototyp. Ein einzelnes Textfeld mitsamt Unterbau wäre ein vertikaler Prototyp.

4.3.4 Inkrementelles/Evolutionäres Modell

Die bereits aufgeführten Vorgehensmodelle haben eine Gemeinsamkeit: Das Produkt steht dem Auftraggeber erst dann zur Verfügung, wenn es fertiggestellt ist. Das führt zu relativ langen Wartezeiten. Meist bilden sich auch erst bei tatsächlichem Produkteinsatz neue Anforderungen und Wünsche, die zu Entwicklungsbeginn, oder bei Ansicht eines Prototyps noch nicht bestanden. Dies führte zum Ansatz des evolutionären Mo-

[21] [Bal98], S. 116
[22] [Bal98], S. 119

dells, welches eine stufenweise Entwicklung des Produkts beschreibt. Grundlage ist die sogenannte Nullversion, die nur den festgelegten Produktkern enthält. Das Ändern der Funktionalität wie auch Pflegeaktivitäten werden als neue Version betrachtet.[23]

Der Auftraggeber kann so in kürzeren Abständen einsatzfähige Produktversionen erhalten. Eine Version muss dabei in ihrer Architektur so gestaltet werden, dass eine Erweiterung möglich ist, ohne unnötigen Aufwand zu betreiben oder gar die Architektur neu zu entwerfen. Ausgegangen von der Tatsache, dass sich eine Dokumentvorlage im Verlaufe der Zeit ändern kann, wäre dieser Ansatz hier anwendbar. Eine Kombination mit dem Prototyping ist möglich.

4.3.5 Feature Driven Development

Feature Driven Development ist im Gegensatz zu den anderen bisher betrachteten Vorgehensmodellen eine agile Methode. Gegenüber anderen agilen Methoden wie Extreme Programming oder auch Scrum besitzt sie dennoch einen recht hohen Grad an Prozessbeschreibung.[24]

FDD lässt sich in fünf Einzelprozesse einteilen, von denen die ersten drei sequenziell für das Gesamtprojekt und die beiden letzten für jedes Feature einzeln ausgeführt werden.[25] Die Prozesse werden in Abbildung 4.5 „Ablauf des Feature Driven Development, nach [Gyg04]“ dargestellt.

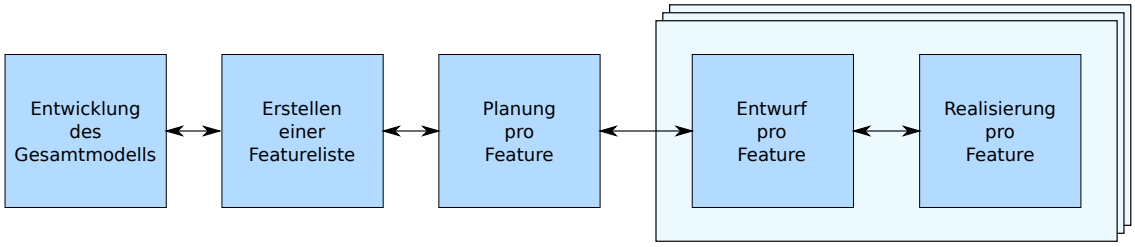

Abbildung 4.5: Ablauf des Feature Driven Development, nach [Gyg04]

Entwicklung eines Gesamtmodells Der Kunde sollte eine grobe Vorstellung von dem haben, was das System leisten soll, sodass das Gesamtmodell entstehen kann, welches sich aus verschiedenen Teilmodellen zusammenfügt. Detaillierte Vorstellungen zu den Features sind aber noch nicht nötig. Auch wird in diesem Schritt Fachwissen gesammelt.

[23] [Bal98], S. 120
[24] [Bil09], S. 25
[25] [Gyg04], Seiten 5-9

Erstellen einer Featureliste Von den Resultaten des ersten Prozesses ausgehend entsteht nun eine Featureliste. Hier wird auch eine Gliederung in sogenannte Feature Sets und eine Klassifizierung nach Wichtigkeit vorgenommen.

Planung pro Feature Hier wird der zeitliche Rahmen für die Fertigstellung der Feature Sets festgelegt. Zusätzlich kommt es zu einer Verteilung von Klassenverantwortlichkeiten auf die Entwickler.

Entwurf pro Feature Unter Nutzung des gesammelten Fachwissens wird hier ein Entwurf für das Feature entstehen. Dazu gehören unter anderem Ablaufdiagramme und Schnittstellenspezifikationen.

Implementierung pro Feature Hierunter zählen neben dem reinen Implementieren auch Codeinspektion und Unit Tests. Am Ende kann das Feature freigeschaltet werden.

Das so konsequente Auseinandernehmen der Vorlagen ist nicht realisierbar. Auch fehlen in diesem Modell projektweite Testszenarien, die eine Qualitätssicherung im Ganzen gewährleisten oder auch die Möglichkeit, nachträglich Änderungswünsche zu berücksichtigen. Etwas abstrakter gedacht, könnte man aber die Textfelder mit darunterliegenden Textbausteinen als eine Art Feature betrachten. Die Textbausteine werden für jedes Textfeld separat verwaltet. Das Grundprinzip könnte also Anwendung finden.

4.4 Ergebnis der Betrachtung

Die Betrachtung konnte keines der Vorgehensmodelle als das explizit auf den nach kundenspezifischen Anforderungen geschneiderten Prozess der Vorlagenentwicklung passende identifizieren. Zwar umfasste die Betrachtung nur einen kleinen Ausschnitt aus der Vielfalt der vorhandenen Vorgehensmodelle zur Softwareentwicklung, doch lässt sich daraus folgern, dass die vorhandenen Modelle nicht Eins zu Eins für die Entwicklung von Vorlagen übernommen werden können.

Jedoch enthalten diese Modelle einige Ansätze, Prinzipien und Strukturen, die aus dem jeweiligen Gesamtmodell herausgegriffen auf die Vorlagenentwicklung gut anwendbar sind. Diese zu kombinieren, wäre ein Ansatz für die Spezifikation eines geeigneten Vorgehensmodells.

In Abschnitt 4.3.2 „V-Modell“ auf Seite 30 wurde die neueste Version des V-Modells, das sogenannte V-Modell XT, kurz angesprochen. XT steht hierbei für eXtreme Tailoring, also extremes Zuschneiden auf die jeweiligen Bedürfnisse des Anwenders. Das Modell beschreibt damit einen Weg universeller Einsatzmöglichkeiten,[26] der als Beschreibungswerkzeug für die Spezifikation des Entwicklungsprozesses für Vorlagen herangezogen werden kann.

[26] [HH08], S. 1

4.5 Grundlagen des V-Modell XT

Bevor jedoch eine Spezifikation unter Zuhilfenahme des V-Modells XT angefertigt werden kann, müssen die Grundbestandteile dieses Modells, die zur Modellierung notwendig sind, bekannt sein. Sie werden in diesem Abschnitt kurz vorgestellt.

4.5.1 Vorgehensbaustein

Ein wichtiger Grundbestandteil ist der Vorgehensbaustein. Er kapselt Produkte, Aktivitäten und Rollen, die für die Erfüllung bestehender Aufgabenstellungen relevant sind und damit inhaltlich zusammengehören.[27] Aktivitäten werden von bestimmten Rollen ausgeführt und haben Produkte als Ergebnis. In Abbildung 4.6 „Aufbau eines Vorgehensbausteines nach [HH08]“ wird dieser Sachverhalt noch einmal grafisch dargestellt.

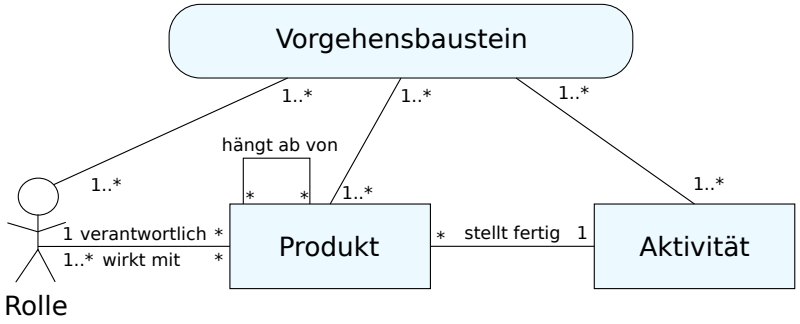

Abbildung 4.6: Aufbau eines Vorgehensbausteines nach [HH08]

Der Begriff Produkt könnte leicht zu der Annahme führen, es würde sich hier nur auf Endprodukte bezogen werden. Das ist ein Trugschluss. Auch Zwischenergebnisse werden als Produkt bezeichnet.

Die modularen und aufeinander aufbauenden Vorgehensbausteine bilden den wesentlichen statischen Inhalt im V-Modell XT. Sie können nach eigenen Bedürfnissen angepasst und erweitert werden.[28]

[27] [HH08], S. 6
[28] [HH08], S. 7

4.5.2 Projektdurchführungsstrategie

Betrachtet man die Vorgehensbausteine, so wird man feststellen, dass durch diese allein noch keine Reihenfolge einzelner Aktivitäten festgelegt ist. Diesen grundlegenden Rahmen für die geordnete, zeitliche und inhaltliche Durchführung findet sich in den Projektdurchführungsstrategien in Form einer Abfolge von Entscheidungspunkten.[29]

Der erfolgreiche Abschluss eines Projektabschnittes wird durch einen solchen Entscheidungspunkt markiert. Ein Entscheidungspunkt erfordert dabei die Fertigstellung einer bestimmten Menge von Produkten. Dabei wird nach dem Prinzip Ende-vor-Ende vorgegangen. Das bedeutet, dass ein Projektabschnitt erst beendet werden kann, wenn der vorige beendet wurde. Der Beginn eines Projektabschnitts wird nicht eingeschränkt.[30]

Damit bildet die verwendete Projektdurchführungsstrategie, wie es in Abbildung 4.7 „Aufbau einer Projektdurchführungsstrategie nach [HH08]“ veranschaulicht wird, den dynamischen Inhaltsbestandteil des V-Modell XT.

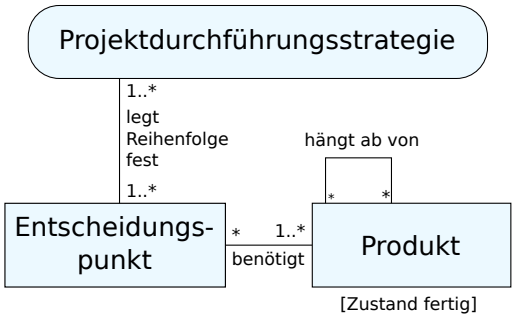

Abbildung 4.7: Aufbau einer Projektdurchführungsstrategie nach [HH08]

4.5.3 Projekttyp

Der Projekttyp führt nun den dynamischen und den statischen Anteil des Modells zusammen. Er definiert, welche Projektdurchführungsstrategie in Verbindung mit welchen Vorgehensbausteinen für das Projekt verwendet wird. Siehe dazu Abbildung 4.8 „Aufbau eines Projekttyps nach [HH08]“ auf der folgenden Seite. Hier sollte angemerkt werden, dass das Produkt, welches im statischen wie auch im dynamischen Anteil vorhanden ist, die Schnittstelle zwischen den beiden Elementen darstellt.[31]

[29] [HH08], S. 8
[30] [HH08], S. 8
[31] [HH08], S. 8

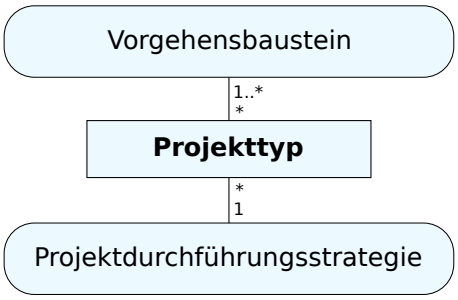

Abbildung 4.8: Aufbau eines Projekttyps nach [HH08]

4.5.4 Übersicht

Setzt man die drei Bestandteile nun zusammen, so entsteht das in Abbildung 4.9 „Grundlegender Aufbau des V-Modell XT“ gezeigte Gesamtkonstrukt.

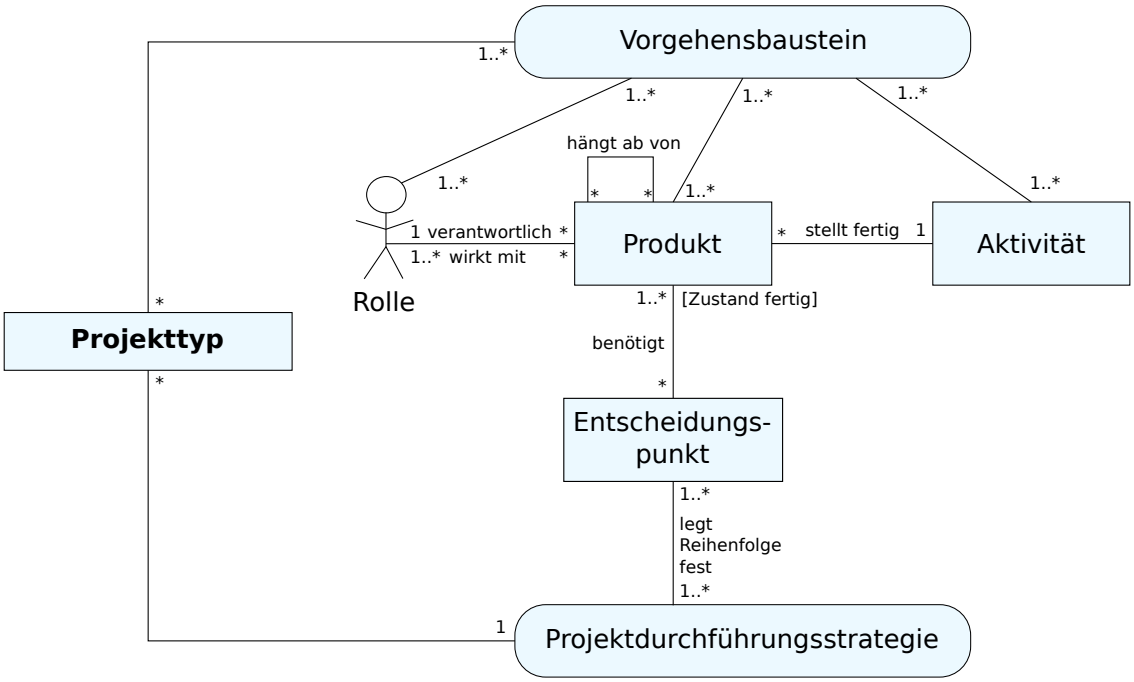

Abbildung 4.9: Grundlegender Aufbau des V-Modell XT

4.6 Spezifikation des Vorlagenentwicklungsprozesses

Die Spezifikation des Vorlagenentwicklungsprozesses wird in zwei Schritten ablaufen. Zuerst werden die Gedanken der einzelnen Konzepte aus dem Abschnitt 4.3 „Betrachtung der Konzepte“ auf Seite 29 kombiniert. Danach folgt die formale Spezifikation mit den Mitteln des V-Modell XT, welche in Abschnitt 4.5 „Grundlagen des V-Modell XT“ auf Seite 35 kurz beschrieben wurden.

4.6.1 Lose Spezifikation

Das V-Modell mit seiner symmetrischen Anordnung von Entwicklung und Qualitätssicherung dient als Grundlage für die Spezifikation. Das Entwicklungs-Stadium wird hier den beiden Qualitätssicherungs-Stadien gegenüberstehen.

Die aus Planung und Ausführung bestehende Entwicklungsphase umfasst neben der eigentlichen Vorlage auch die Textfelder und deren Textbausteine. Diese Einteilung kann man wie beim Feature-Driven-Development auch als eine Einteilung in sogenannte Features sehen.

Vorgabe war, dass das Test-Stadium eng mit der Entwicklung verbunden ist. Das wird dadurch erreicht, dass hier Prototyping integriert wird. So werden jeweils Prototypen aus der Entwicklung an diese Testphase weitergereicht werden. Auf dieser Ebene ist damit sehr schnelles Feedback möglich.

Nach dem Abnahmetest des Qualitätsmanagements ist die Vorlage fertig und wird damit produktiv verwendet. Damit ist die Entwicklung aber noch nicht abgeschlossen. Änderungswünsche erfordern eine Rückkopplung zum Entwicklungsstadium. An das inkrementelle Modell angelehnt, entsteht durch diese Rückkopplung jeweils eine neue Version.

4.6.2 Formale Spezifikation

Aus den vorhergehenden Überlegungen leiten sich eine Reihe von Produkten, Aktivitäten und Rollen ab, welche die Grundlage für die Spezifikation nach dem V-Modell XT bilden. Diese werden hier beschrieben.

4.6.2.1 Produkte

Vorlagenspezifikation Spezifikation über das spätere Aussehen der Vorlage. Hierin wird festgelegt, welche Textelemente, Textfelder und Grafiken auf der Vorlage enthalten sein werden und wie diese angeordnet und formatiert sein sollen. Zur Formatierung zählen auch Farb- und Schriftartzuordnungen.

Textfeldspezifikation Hier wird neben dem verwendeten Datentyp eines Textfeldes festgehalten, welche Textbausteine diesem später zur Verfügung stehen sollen.

Vorlagendesignprototyp Die Umsetzung der Vorlagenspezifikation resultiert in einem Vorlagendesignprototyp. Dieser muss noch keine funktionierenden Textfelder enthalten.

Textfeldprototyp Ein mit Textbausteinen und notwendigen Skripten untersetztes Textfeld auf einer Vorlage wird Textfeldprototyp genannt. Dazu muss die Vorlage selbst noch nicht fertiggestellt sein.

Testergebnis Das Testergebnis bezieht sich auf einen Zwischentest, also auf einen konkreten Prototyp. Auf deren Grundlage können Korrekturen durchgeführt werden.

Abnahmetestergebnis Der Abnahmetest liefert das Abnahmetestergebnis. Es zeigt an, ob der Abnahmetest bestanden wurde. Ist dies nicht der Fall, so enthält er eine Liste von Mängeln, die vor der endgültigen Freigabe noch behoben werden müssen.

Produktivversion Wurde der Abnahmetest bestanden, so wird der Prototyp, der getestet wurde, zur Produktivversion. Die Produktivversion wird zur Dokumenterzeugung verwendet.

Feedback Während der Verwendung von Vorlagen im produktiven Betrieb kann Feedback entstehen. Das umfasst praktische Änderungswünsche wie auch aus dem Umgang resultierende Verbesserungsvorschläge.

4.6.2.2 Aktionen

Vorlage spezifizieren Hier wird festgelegt, welche Elemente die Vorlage einmal enthalten soll und wie diese Elemente aussehen und angeordnet werden. Ergebnis ist die Vorlagenspezifikation.

Textfeld spezifizieren In diesem Schritt wird ausgewählt, welchen Datentyp und welche Textbausteine ein Textfeld zugeordnet bekommen soll. Ergebnis ist die Textfeldspezifikation.

Vorlage designen Die Vorlagenspezifikation wird in dieser Aktivität im Designer schrittweise in eine Vorlage umgesetzt. Ergebnis ist ein Vorlagendesignprototyp.

Textbausteine verwalten In der Textbausteinverwaltung werden die spezifizierten Textbausteine für ein Textfeld angelegt und diesem zugeordnet. Ergebnis ist ein Textfeldprototyp.

Zwischentest Ein Prototyp, egal ob Textfeldprototyp oder Vorlagendesignprototyp, wird getestet. Ergebnis ist ein Testergebnis.

Abnahmetest Ist ein Prototyp so umfänglich geworden, dass alle Elemente enthalten und fertiggestellt sind, so erfolgt der Abnahmetest. Ergebnis ist das Abnahmetestergebnis und bei erfolgreichem Bestehen des Abnahmetests die Produktivversion.

Vorlage verwenden Die Produktivversion der Vorlage wird im täglichen Geschäftsbetrieb verwendet, um Dokumente zu erzeugen. Ergebnis ist das Feedback.

4.6.2.3 Rollen

Sachbearbeiter Der Sachbearbeiter verwendet die Vorlage, die aus der Entwicklung hervorgeht. Sein Feedback kann Grundlage für Veränderungen und Verbesserungen der Vorlage sein.

Vorlagendesigner Der Vorlagendesigner ist zuständig für Spezifikation, Design der Vorlage und für Textbausteinverwaltung. Kleinere Zwischentests fallen ebenfalls in den Bereich der Vorlagenentwickler.

QM-Team Das Qualitäts-Management-Team übernimmt den Abnahmetest, welcher die Vorlagen tiefgehend prüft.

4.6.2.4 Vorgehensbausteine

Fasst man diese Produkte, Aktionen und Rollen im Gesamtkonzept zusammen, so ergeben sich die folgenden Vorgehensbausteine als statischer Bestandteil des Modells:

Vorgehensbaustein(Aktivität)	**Rolle**	**Erzeugte Produkte**
Vorlage spezifizieren	Vorlagenentwickler	Vorlagenspezifikation
Textfeld spezifizieren	Vorlagenentwickler	Textfeldspezifikation
Vorlage designen	Vorlagenentwickler	Vorlagendesignprototyp
Textbausteine verwalten	Vorlagenentwickler	Textfeldprototyp
Zwischentest	Vorlagenentwickler	Testergebnis
Abnahmetest	QM-Team	Abnahmetestergebnis, Produktivversion
Vorlage verwenden	Sachbearbeiter	Feedback

Tabelle 4.1: Vorgehensbausteine

4.6.2.5 Projektdurchführungsstrategie

Zur Bildung des dynamischen Anteils in Form einer Projektdurchführungsstrategie liegen die in Tabelle 4.2 gelisteten Entscheidungspunkte zu Grunde:

Entscheidungspunkt	**Notwendige Produkte**
Vorlage spezifiziert	Vorlagenspezifikation
Textfeld spezifiziert	Textfeldspezifikation
Vorlage designt	Vorlagendesignprototyp
Textfeld erfasst	Textfeldprototyp
Test durchgeführt	Testergebnis
Abnahmetest bestanden	Produktivversion
Vorlage verwendet	Feedback

Tabelle 4.2: Entscheidungspunkte

Die Reihenfolge der Entscheidungspunkte ist der Abbildung 4.10 „Vorgehensmodell zur Vorlagenentwicklung“ auf der folgenden Seite zu entnehmen.

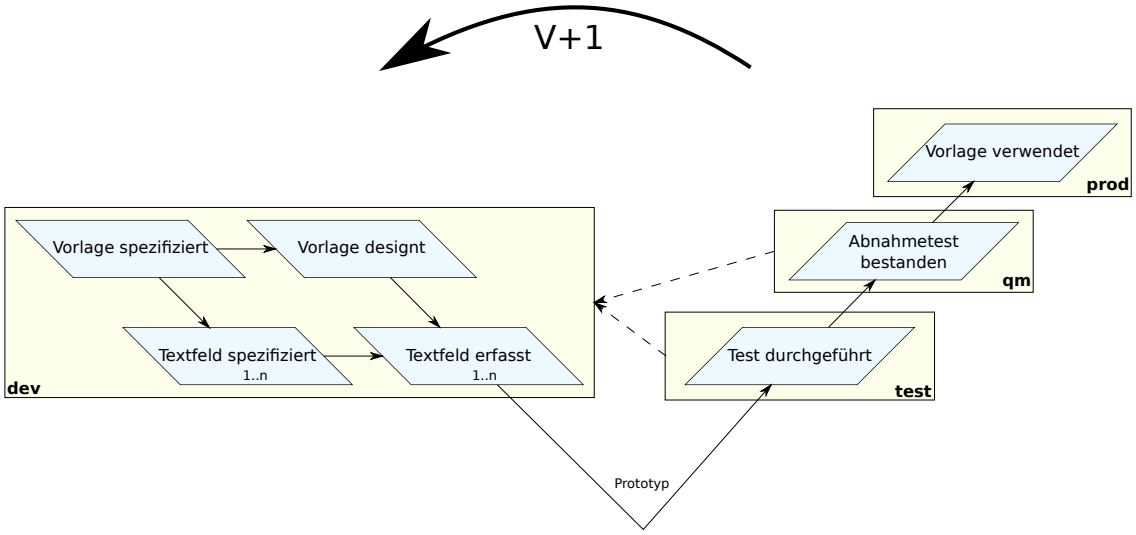

Abbildung 4.10: Vorgehensmodell zur Vorlagenentwicklung

4.6.2.6 Projekttyp

Diese Projektdurchführungsstrategie wird mit den Vorgehensbausteinen zu einem neuen Projekttyp **Vorlagenentwicklungsprojekt** kombiniert.

4.7 Fachliche Umsetzung des Modells

Nachdem ein passendes Vorgehensmodell spezifiziert wurde, folgt nun die Umsetzung einer technischen Unterstützung dieses Modells. Hier sind zwei Dinge von Bedeutung: Die Umsetzung der verschiedenen Stadien im Kundensystem und die Realisierung der Freigabe einer Vorlage von einem Stadium zu einem anderen.

4.7.1 Umfang einer Vorlage

Für die Verwendung von Dokumentvorlagen, welche sich im Laufe ihrer Entwicklung in einer Reihe verschiedener Stadien befinden, zwischen denen sie transferiert werden, muss eine entsprechende Abbildung dieser Stadien im Modell gefunden werden. Dazu wird zunächst festgehalten, aus welchen Bestandteilen eine Vorlage effektiv besteht.

Neben der Vorlage selbst, die als XDP-Datei im Repository des Servers liegt, sind auch die darin enthaltenen (referenzierten) Fragmente und Ressourcen wie beispielsweise Bilder notwendig. Diese sind ebenfalls als Dateien im Repository vorhanden. Dazu kommen noch die Metainformationen, die über Textfelder, Textbausteine und deren Verknüpfung in der Datenbank erfasst sind.

Zur Untermalung sind in Abbildung 4.11 „Aufbau eine Vorlage“ Fragmente rot, Ressourcen blau und Textfelder grün markiert.

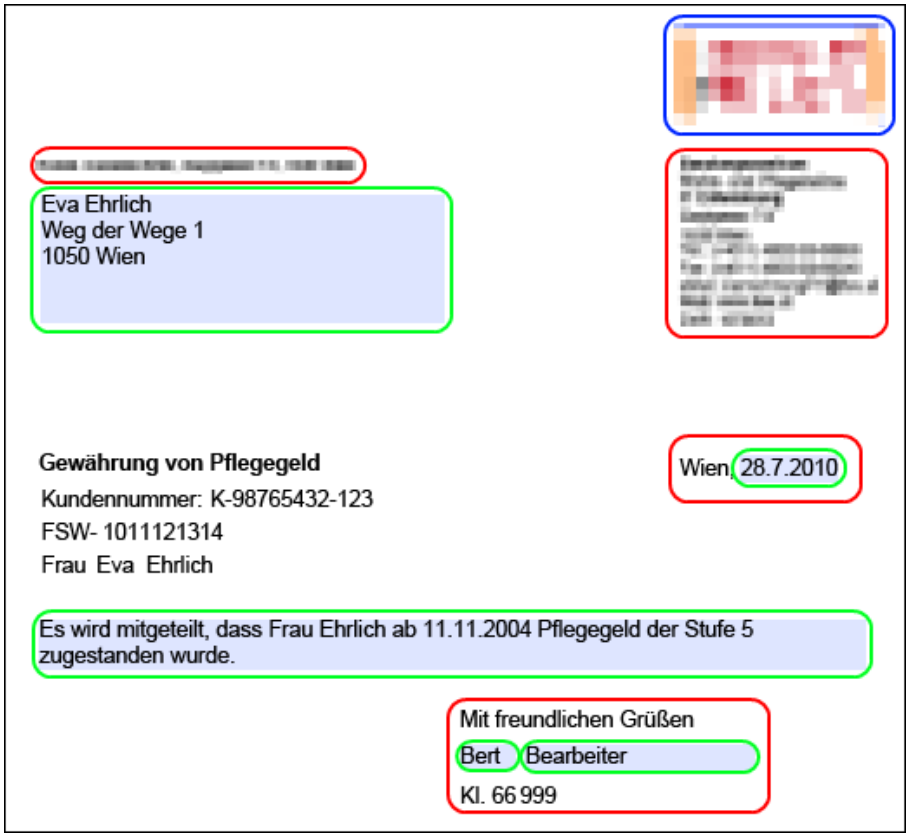

Abbildung 4.11: Aufbau eine Vorlage

4.7.2 Umsetzung der verschiedenen Stadien

Die Stadien, wie sie in Abbildung 4.12 „Statusdiagramm“ auf der folgenden Seite gezeigt werden, müssen sowohl in der Datenbank als auch im *LiveCycle* Repository umgesetzt werden.

Im Repository ist das trivialerweise dadurch möglich, dass für jedes Stadium ein Ordner angelegt wird, in dem jeweils der gesamte Dateibaum mit Vorlagen, Fragmenten und anderen Ressourcen vorhanden ist. In der Datenbank werden die bestehenden Tabellen zur Ablage der zusätzlichen Information um eine Spalte, die das Stadium erfasst, erweitert. So kann der Datenbestand einer Vorlage für mehrere Stadien im System vorgehalten werden. Warum dies notwendig ist, wird im folgenden Abschnitt dargestellt.

Effektiv kann sich dabei in jedem Stadium genau eine Version jeder Vorlage befinden. Die Weitergabe der Daten zwischen den Stadien erfolgt nur in einer Richtung. Folgende einfache tabellarische Darstellung soll das „Wandern“ einer Version (durch Buchstaben A bis D dargestellt) durch die verschiedenen Stadien veranschaulichen. Zum besseren

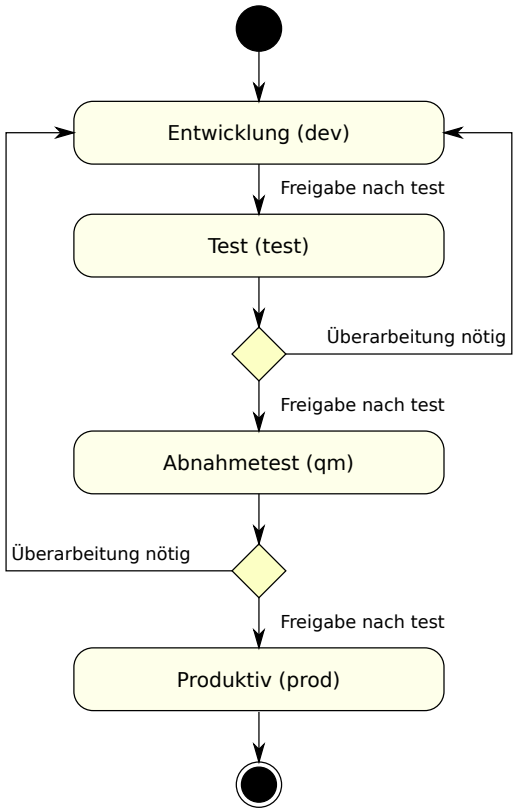

Abbildung 4.12: Statusdiagramm

Verständnis wird in der Spalte „Aktion“ nochmals in Kurzform angegeben, welche Veränderungen stattgefunden haben. Eine neue Version wird dabei immer aus der Version erstellt, die sich davor im Stadium *dev* befunden hat.

	dev	**test**	**qm**	**prod**	**Aktion**
1	A				A erstellt
2	A	A			A nach test freigegeben
3	B	A	A		A nach qm freigegeben, B erstellt
4	C	B	A	A	A nach prod und B nach test freigegeben, C erstellt
5	C	C	A	A	C nach test freigegeben
6	D	C	C	A	C nach qm freigegeben, D erstellt
7	D	D	C	C	D nach test und C nach prod freigegeben
8	D	D	D	D	D über qm nach prod freigegeben

Durch die Veränderung in Repository und Datenbankschema wird es notwendig, an der bestehenden Anwendung *DokGen* einige Änderungen durchzuführen, damit diese mit den neuen Strukturen fehlerfrei arbeiten kann.

4.7.3 Realisierung der Freigabe

Die Freigabe einer Vorlage ist die Übertragung von getätigten Änderungen von einem bestimmten Stadium in das nachfolgende. Dabei ist zu beachten, dass manche Änderungen auch Auswirkungen außerhalb der eigentlichen Vorlage haben, wie es zum Beispiel bei mehrfach verwendeten Fragmenten der Fall ist. Damit das System nicht automatisiert zu umfängliche Freigaben durchführt, obliegt dem Benutzer die Entscheidung, welche Änderungen freigegeben werden sollen.

Um Änderungen selektiv freigeben zu können, ist es notwendig, über die Feststellung der Unterschiede zweier Stadien herauszufinden, welche Änderungen seit der letzten Freigabe vorgenommen wurden. Würde bei der Freigabe der Datenbestand lediglich mit einem neuen Status gekennzeichnet, wäre es nicht möglich, Unterschiede zur letzten Freigabe festzustellen. Folglich wird bei der Freigabe eine Kopie angelegt, wodurch der Datenbestand einer Vorlage für mehrere Stadien im System gehalten wird.

Nachdem die Änderungen festgestellt wurden, kann der Nutzer in einer Auswahl entscheiden, welcher er davon in der Freigabe berücksichtigen will. Mögliche Änderungen sind dabei:

- Vorlage/Fragment/Ressource verändert
- Vorlage/Fragment/Ressource angelegt
- Vorlage/Fragment/Ressource gelöscht
- Textfeldeigenschaften verändert
- Textfeld angelegt
- Textbaustein erstellt und einem Textfeld zugeordnet
- Textbaustein verändert
- Textbaustein gelöscht
- Exklusivzuordnung eines Textbausteins verändert

Diese Änderungen werden dann durch Kopieroperationen im Repository und in der Datenbank ausgeführt. Im Repository kann durch vorhandene Mittel mit geringstem Aufwand eine Historie geführt werden. Da das in der Datenbank nicht in dieser Form möglich ist und das Führen einer Historie nicht vorausgesetzt wird, kann in der Datenbank darauf verzichtet werden, eine Historie zu führen.

5 Technische Konzeption

Die Umsetzung dieser fachlichen Anforderungen wird nun technisch konzipiert. Die wichtigsten Punkte der Konzeption werden in diesem Kapitel behandelt.

5.1 Ablaufumgebung

Damit sich die neue Lösung sowohl optisch als auch technisch an die bisherige anpasst, wird als Programmiersprache für den Textbausteineditor *TEd Adobe Flex* verwendet. Für den zentralen Zugriff auf Datenbanken und die Funktionen des *Adobe LiveCycle* Servers wird dessen Webservice-Interface genutzt.

5.1.1 Adobe Flex

Mit Adobe Flex können sehr leistungsfähige und optisch ansprechende Oberflächen entwickelt werden. Diese lassen sich mit Hilfe von Adobe Flash Player oder Adobe AIR auf allen gängigen Browsern, Desktops und Betriebssystemen ausführen. Bei Flex handelt es sich um ein kostenloses Open-Source-Framework. [ado10]

Dabei nutzt das Framework zwei Sprachen:

MXML ist eine auf XML basierte Oberflächenbeschreibungssprache, welche vom Compiler in äquivalente Actionscriptklassen umgesetzt wird. Oberflächen können in Flex aber auch direkt im Actionscript-Quellcode angelegt werden. Durch die hierarchische Darstellung von XML lassen sich die aus ineinander geschachtelten Komponenten bestehenden Oberflächen jedoch wesentlich intuitiver gestalten.

Actionscript ist eine objektorientierte und eventbasierte Programmiersprache, die auf dem ECMA-Script-Standard basiert. Im Gegensatz zu Java werden in Actionscript auch primitive Datentypen als Klassen behandelt. Komponenten, die in der MXML-Beschreibung eine ID zugeordnet bekommen haben, können via Actionscript über diese angesprochen und verwendet werden.

5.1.2 LiveCycle Server

Es gibt eine Menge von Funktionen, deren Auslagerung in einen Server der Anwendung Vorteile bringt. Darunter zählen der Zugriff auf Datenbank, Vorlagen-Repository und das Rendering von PDF-Dokumenten.

Hierzu eignet sich der *LiveCycle*-Server, der die oben genannten Funktionen bereits implementiert und als Servicebausteine anbietet. Funktionalität, die noch in keinem Baustein verfügbar ist, kann über den Zugriff auf die unterliegende API mit Java ergänzt werden. Die Servicebausteine können dann zu beliebigen Abläufen kombiniert werden, welche unter anderem über eine Webserviceschnittstelle angesprochen werden können. (Siehe dazu Abbildung 5.1 „Prozessmodellierung im LiveCycle Server“.)

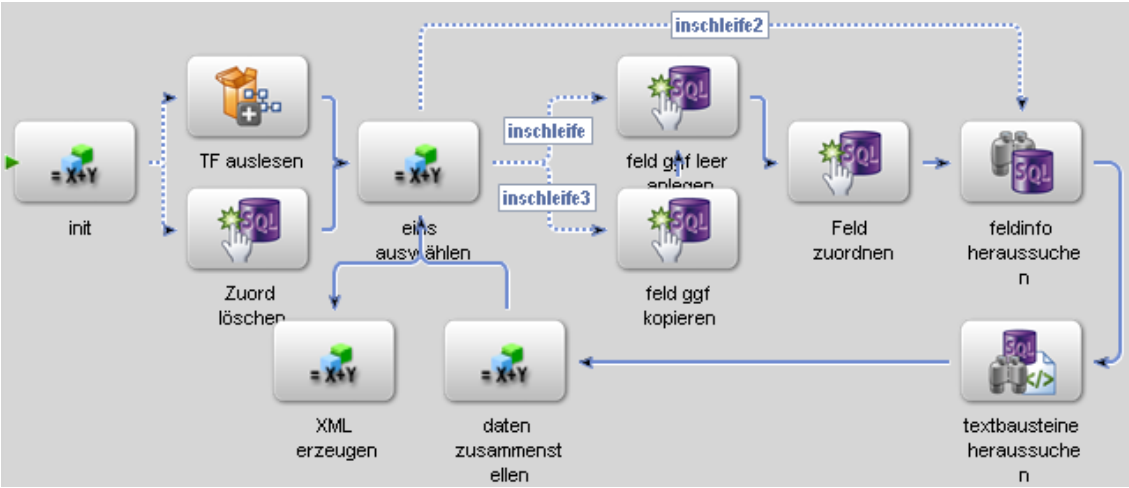

Abbildung 5.1: Prozessmodellierung im LiveCycle Server

5.1.3 Cairngorm-Framework

Das Open Source Framework *Cairngorm* von Adobe stellt dem Entwickler eine Implementierung des Model-View-Controller-Entwurfsmusters bereit. Die damit eingeführte Trennung von Datenmodell, Oberfläche und Geschäftslogik erhöht die Wartbarkeit des entstehenden Programms. Wichtige Komponenten dieses Frameworks sind:

ModelLocator Hierbei handelt es sich um eine global verwendete Klasse, die das Datenmodell beinhaltet. Der Zugriff auf das Datenmodell ist folglich nur über den *ModelLocator* möglich.

View Zum View gehören Klassen, die die grafische Oberfläche bilden. Sie werden durch Data-Binding an den *ModelLocator* und damit an das Datenmodell angebunden.

Event Mithilfe von Events werden Aktionen im Programm angestoßen. Sie dienen gleichzeitig als Träger für notwendige Informationen zum Aufruf der jeweiligen Aktion.

FrontController Über den *FrontController* wird die Abbildung von *Events* auf *Commands* realisiert.

Command Die *Commands* stellen die Aktion dar, die von den *Events* angestoßen wird. Sie können Änderungen am Datenmodell vornehmen. Es besteht die Möglichkeit, Teilaufgaben an *Business Delegates* weiterzuleiten.

Business Delegate Delegates werden zum Beispiel eingesetzt, um Webserviceaufrufe zu kapseln. Dabei muss die genaue Kenntnis über Art und Weise des Aufrufs nur im *Delegate* bekannt sein.

5.1.4 Erweiterung des Frameworks

Während der Nutzung des Frameworks sind einige Dinge aufgefallen, deren Umsetzung einer Erweiterung bedurften:

Es ist immer wieder das Problem aufgetreten, dass eine Reihe von Aktionen auf Grund von Abhängigkeiten nicht parallel, sondern sequentiell stattfinden musste. Das ist zwar realisierbar, jedoch nur mit relativ großem Aufwand. Deshalb wurde eine Erweiterung der Grundklassen Event und Command zu Job und JobCommand vorgenommen, die die Definition von Abhängigkeiten erlauben. So kann einem Job beim Absenden eine Menge anderer Jobs mitgegeben werden, die vorher abgeschlossen werden müssen. Im Gegenstück, dem JobCommand, muss entweder bestätigt werden, dass der Job abgeschlossen oder abgebrochen wurde. Im Falle eines Abbruchs werden abhängige Jobs ebenfalls abgebrochen. Für diese Bestätigung existieren entsprechende Methoden.

Im Testbetrieb ist es sehr hilfreich zu wissen, welche Schritte im Programm durchlaufen werden. Das wird aber standardmäßig nicht angezeigt. Da die Geschäftslogik in den Jobs liegt, bietet es sich hier an, Start und Beendigung von Jobs zu protokollieren.

Im Weiteren ist bei der Betrachtung der bereits bestehenden Projekte aufgefallen, dass die Delegates, welche die Webserviceaufrufe kapseln, einen hohen Anteil an identischem Code aufweisen. In Hinsicht auf Wartbarkeit ist das bedenklich. So ist eine Delegateklasse entstanden, die diesen identischen Code zusammenfasst. Die einzelnen Klassen vereinfachen sich dadurch maßgeblich.

5.2 System-Übersicht

Im Folgenden werden die wichtigsten Eckpunkte des Systems beschrieben. Die Art und Weise, nach der sie zu diesem zusammengesetzt werden, erkennt man am besten in der Darstellung von Abbildung 5.2 „Gesamtaufbau des Systems“ auf der folgenden Seite. Auch wird hier das Zusammenspiel der Komponenten sichtbar.

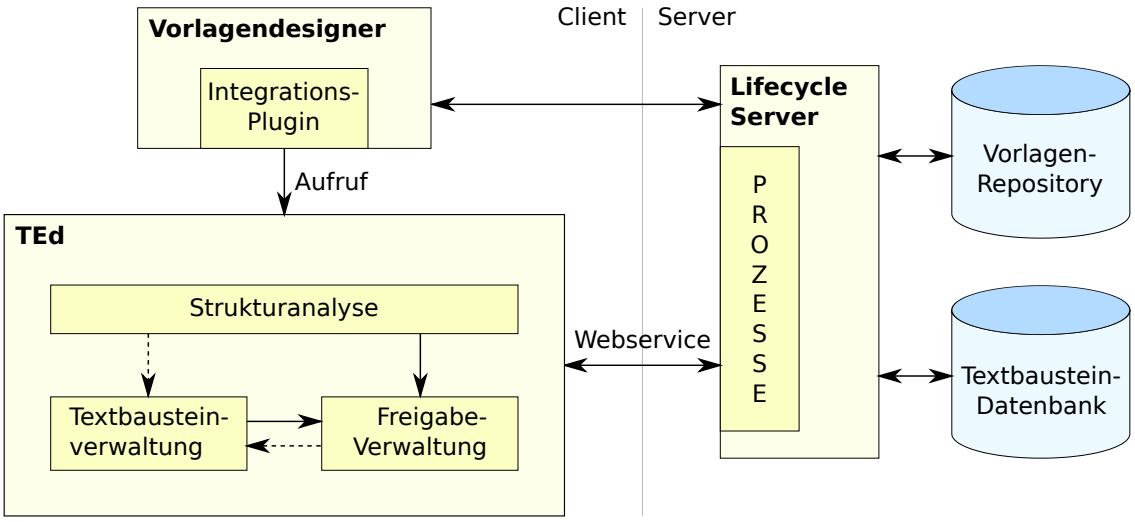

Abbildung 5.2: Gesamtaufbau des Systems

5.3 Strukturanalyse

Für die Analyse der XML-Struktur ist es notwendig, die jeweilige Vorlage in das Programm zu laden. Ist dies geschehen, so können die komfortablen Möglichkeiten von Actionscript genutzt werden, um auf das XML zuzugreifen. Dazu ist es im Vorhinein notwendig, herauszufinden, welche Elemente der XML-Struktur die Informationen enthalten, die für die Analyse wichtig sind.

5.3.1 Analytisch relevante Inhalte

Es muss also festgestellt werden, wie Textfelder, verlinkte Bilder und Fragmente in die Vorlage eingebunden sind. Aus der Spezifikation des Formats ergeben sich folgende Strukturen:

Fragment: Fragmente werden jeweils als sogenanntes Unterformular eingebunden: `<subform usehref="Dateiname#ReferenzierteSubform"/>` [32] Einzige Nutzinformation ist der referenzierte Dateiname. Der Name des Subforms selbst ist unerheblich.

Skript-Fragment: Ähnlich wie bei normalen Fragmenten verhält es sich beim Script-Fragment. Einziger Unterschied ist, dass diese direkt als Skript eingebunden werden: `<script usehref="Dateiname#ReferenziertesSkript"/>` [33]

[32] [xfa08], S. 750, 758
[33] [xfa08], S. 736, 737

Dynamischer Bestandteil: Dynamische Bestandteile, also die Textfelder, die bei der Dokumenterzeugung ausgefüllt werden, haben folgende Entsprechung im XML: `<field name="Bezeichner"/>` [34] Hier ist nur der Name des Feldes für die Analyse von Interesse.

Statischer Bestandteil: Statische Bestandteile, also die Textfelder, die schon beim Design feststehen, werden wie folgt modelliert: `<draw name="Bezeichner"/>` [35] In der Analyse werden sie nur eine untergeordnete Rolle spielen.

Bild: Das Bild ist zwar ebenfalls ein statischer Inhalt, erhält jedoch eine Sonderbehandlung, da hier zusätzlich der Dateiname des Bildes ausgelesen werden muss: `<draw><value><image href="Dateiname"/></value></draw>` [36]

5.3.2 Zugriff mittels XPath

Diese Strukturen müssen nun im Dokument gefunden und anschließend ausgelesen werden. Dabei ist zu beachten, dass die gesuchten Elemente verschieden tief in der Hierarchie des XML liegen können. Dank der XPath-Implementierung in Actionscript muss hier kein manuelles Durchschreiten des gesamten Dokuments erfolgen.

Ist die genaue Position des Elements innerhalb des XML nicht bekannt, so kann der Nachfolge-Accessor-Operator (..) verwendet werden. Dieser durchläuft die Struktur rekursiv. Die Ergebnisse gibt er als *XMLList* zurück, die dann weiterverarbeitet werden kann. [as307]

So können beispielsweise alle Draw-Elemente im XML mit folgendem Befehl gefunden werden: `content..*::draw`

5.3.3 Vorgehen

In Abbildung 5.3 „Ablauf der Analyse“ auf der nächsten Seite wird der Ablauf bildlich dargestellt.

Wie bereits erwähnt, besteht eine Vorlage nicht nur aus einer einzelnen Datei. Folglich muss der Analyseprozess auch auf alle mit der Vorlage verlinkten Fragmente angewendet werden, um wirklich eine ganzheitliche Erfassung der Vorlage zu gewährleisten. Bereits durchlaufene Dateien sollten vermerkt werden, so dass keine Datei mehrmals eingelesen und analysiert wird.

[34] [xfa08], S. 636
[35] [xfa08], S. 600
[36] [xfa08], S. 600, 605, 660, 661

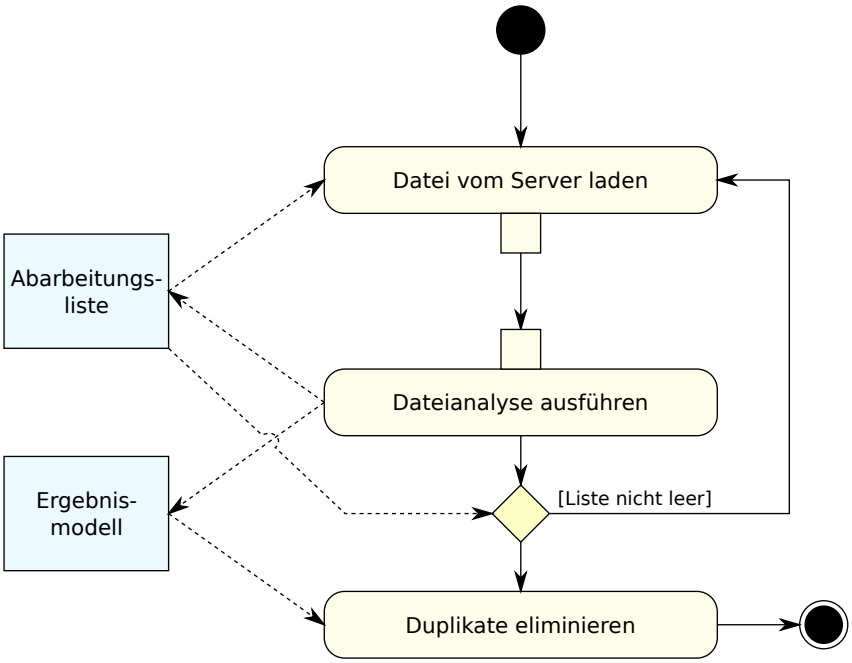

Abbildung 5.3: Ablauf der Analyse

Es kann durchaus vorkommen, dass ein Fragment mehrfach verwendet wird oder dass sich ein Textfeld in verschiedenen verwendeten Fragmenten befindet. In diesem Fall dürfen die Einträge zu Referenz beziehungsweise Textfeld aber nicht mehrfach in den Analyseergebnissen vorkommen. Es ist also notwendig, die Ergebnisse der Analyse in einer Darstellung bereitzustellen, in der Duplikate eliminiert wurden.

5.3.4 Datenmodell

Anhand dieser Vorüberlegungen kann nun ein Datenmodell entwickelt werden, welches die transienten Ergebnisse der Analyse fasst. Grundtyp dieser Überlegung ist die Klasse *XDP*. Die Klasse *Vorlage* ist lediglich eine Ableitung, die noch zusätzliche Informationen aus der Datenbank enthält. Siehe dazu Abbildung 5.4 „Datenmodell der Analyseergebnisse“ auf der folgenden Seite.

Es ist erkennbar, dass eine Instanz der Klasse *XDP* Referenzen zu Ressourcen und anderen XDP’s - in diesem Fall Fragmenten - enthält. Zusätzlich werden die Textfelder als Elemente verwaltet. Die Benennung dieser zwei Klassen orientiert sich an den Bezeichnern, welche im XML verwendet werden.

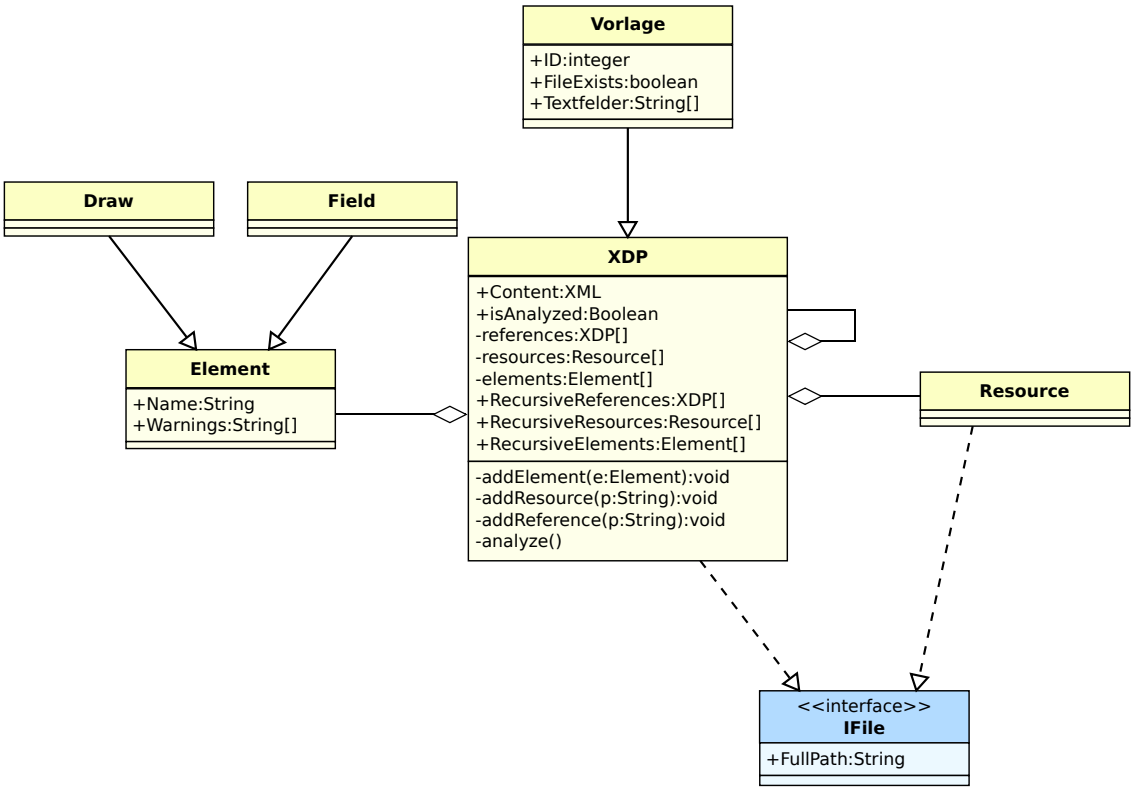

Abbildung 5.4: Datenmodell der Analyseergebnisse

5.4 Textbausteinverwaltung

Die Textbausteinverwaltung ist einer der Kernpunkte der Anwendung. Die dazu notwendige Umsetzung der Stadien für Textfelder und Textbausteine in der Datenbank ist nicht so trivial wie die Umsetzung im Repository. Deshalb wird sie hier noch kurz erläutert.

5.4.1 Realisierung der Stadien

In der Datenbank werden die bestehenden Tabellen zur Ablage der zusätzlichen Information um eine Spalte, die das Stadium erfasst, erweitert. Diese neue Spalte muss zum Primärschlüssel der Haupttabellen gehören, damit die Eindeutigkeit und Zusammengehörigkeit der Textbausteine, Textfelder und Vorlagen über verschiedene Stadien hinweg weiterhin gewährleistet werden kann. Zur Erfassung der Stadien wird eine weitere Schlüsseltabelle eingeführt. Daraus ergibt sich eine Erweiterung des bisherigen Datenbankmodells in Abbildung 2.4 „Datenbank, Darstellung der Struktur“ auf Seite 17. Das Ergebnis der Erweiterung wird in Abbildung 5.5 „Datenbankschema nach Anpassung“ auf der folgenden Seite dargestellt.

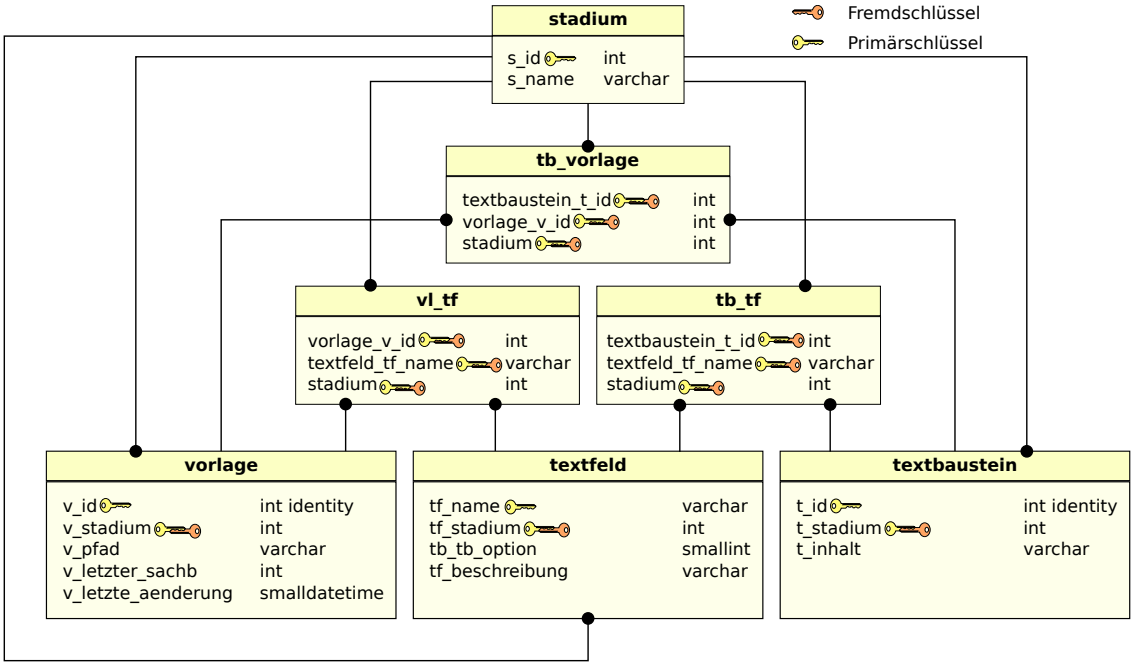

Abbildung 5.5: Datenbankschema nach Anpassung

Nach der Einführung von verschiedenen Stadien, in denen sich eine Vorlage befinden kann, kann die Einschränkung festgelegt werden, dass das Bearbeiten der Vorlage und damit auch der Textbausteine nur im ersten - dem Entwicklungsstadium - stattfindet. Hier sei nochmals erwähnt, dass innerhalb eines Stadiums keine verschiedenen Versionen einer Vorlage existieren.

5.4.2 Vorgehen

Der Prinzipielle Ablauf der Textbausteinverwaltung wird in Abbildung 5.6 „Ablauf der Textbausteinverwaltung“ auf der nächsten Seite dargestellt.

5.4.3 Automatische Zuordnung von Vorlagen und Textfeldern

Die Zuordnung von Vorlagen und verwendeten Textfeldern ist eine Aufgabe, die programmatisch erfolgen sollte. Deshalb wird direkt nach der Analyse der Vorlage die ermittelte Liste von verwendeten Textfeldern durchlaufen. Existiert ein Textfeld noch nicht in der Datenbank, so wird es angelegt. Die Zuordnung der Textfelder zur Vorlage wird ebenfalls anhand dieser Liste rekonstruiert.

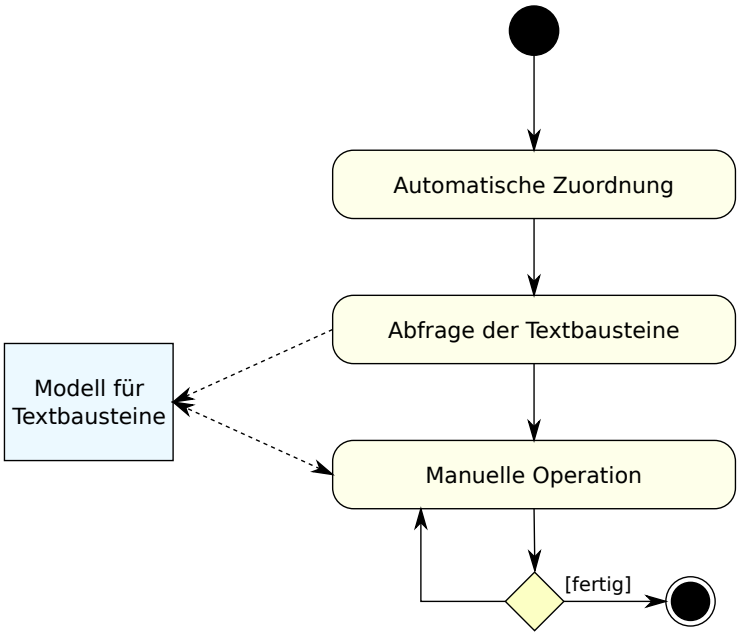

Abbildung 5.6: Ablauf der Textbausteinverwaltung

Dies passiert ohne Zutun des Vorlagenentwicklers im Hintergrund, da manuelle Änderungen in dieser Zuordnung die Konsistenz des Datenbestands beeinträchtigen können. Einzige Tätigkeit, die er hier ergänzend leisten muss, ist das Eingeben einer Beschreibung für neuangelegte Textfelder.

5.4.4 Manuelle Operationen

Zur Verwaltung der Textbausteine wird dem Vorlagenentwickler die Oberfläche aus Abbildung 5.7 „Oberfläche der Textbausteinverwaltung“ auf der folgenden Seite angeboten, in der er jeweils ein Textfeld auswählen kann. Ihm steht dann frei, das Textfeld zu bearbeiten und Textbausteine anzulegen, zu ändern und zu löschen. Die Operationen werden direkt nach ihrer Bestätigung in der Datenbank ausgeführt und in der Anzeige aktualisiert.

Das System lässt die Freiheit, einen Textbaustein auch mehreren Textfeldern zuordnen zu können. Diese Möglichkeit wird nicht ausgeschöpft. Abgesehen von der Erstellung einer zusätzlichen Suchoberfläche über alle vorhandenen Textfelder erhöht sich durch solche Mehrfachzuordnungen die Komplexität der Abhängigkeiten in der Datenbank deutlich.

Textbausteine können exklusiv einer oder mehreren Vorlagen zugeordnet sein. In der entstehenden Oberfläche kann der Vorlagenentwickler entscheiden, ob ein Textbaustein exklusiv zu der Vorlage gehört, die er gerade bearbeitet. Inwiefern der Textbaustein bereits zugeordnet ist, kann in der Anzeigeliste der Textbausteine abgelesen werden.

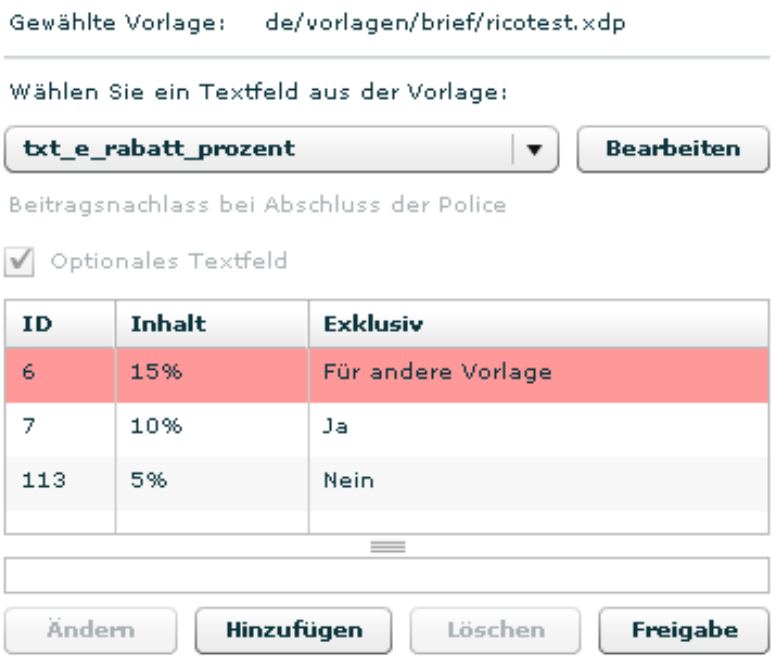

Abbildung 5.7: Oberfläche der Textbausteinverwaltung

Zudem werden Textbausteine, die durch exklusive Zuordnung zu anderen Vorlagen in der aktuellen nicht aktiv sind, farbig hervorgehoben, wie in Abbildung 5.7 „Oberfläche der Textbausteinverwaltung“ zu sehen ist.

5.5 Freigabeverwaltung

5.5.1 Vorgehen

Die Freigabeverwaltung setzt sich aus zwei wichtigen Komponenten zusammen. Als erstes werden alle Bestandteile der Vorlage erfasst und deren Unterschiede festgestellt. Nachdem der Benutzer dann ausgewählt hat, welche der festgestellten Änderungen er freigeben will, werden diese nacheinander freigegeben. Hierbei ist ein Zwischenschritt notwendig, um die Datenintegrität zu wahren. Der Gesamtablauf wird in Abbildung 5.8 „Ablauf der Freigabe“ auf der folgenden Seite dargestellt. Auf Parallelisierung wird in der Abbildung in Hinblick auf die tatsächliche Implementierung verzichtet.

5.5.2 Feststellen der Unterschiede zwischen zwei Stadien

In Abschnitt 4.7.3 „Realisierung der Freigabe“ auf Seite 44 wurde bereits festgelegt, dass der Freigabemechanismus selektiv arbeiten sollte. Dies erfordert als Vorbereitung der Freigabe die Überprüfung der Vorlage auf Änderungen. Diese werden hierbei direkt am Objekt festgestellt, was in drei unterschiedlichen Vergleichsoperationen resultiert.

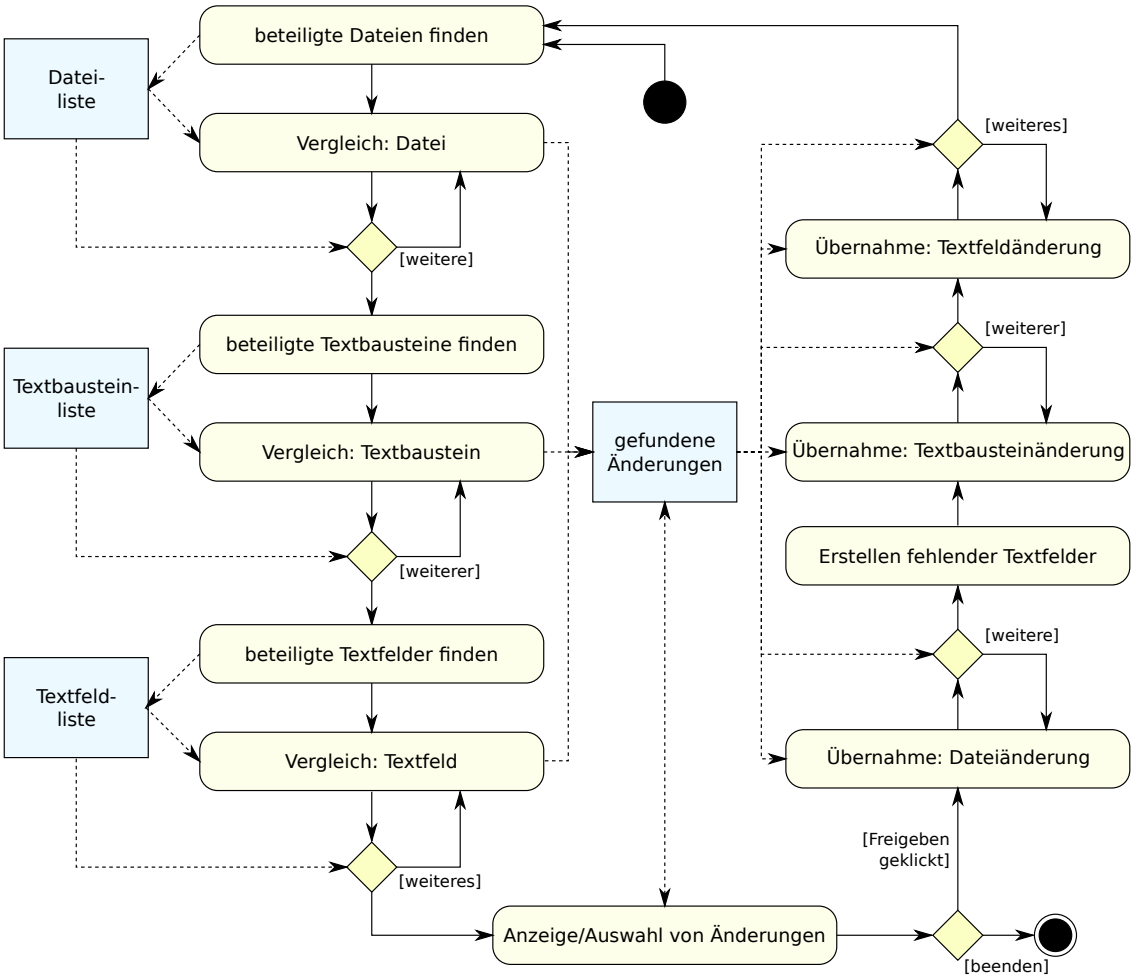

Abbildung 5.8: Ablauf der Freigabe

Es handelt sich dabei um einen vorlagenbezogenen Vergleich, was bedeutet, dass nur Objekte in den Vergleich einbezogen werden, die tatsächlich von der Vorlage referenziert werden.

5.5.2.1 Dateivergleich

Im ersten Schritt werden alle betroffenen Dateien überprüft. Dazu ist es notwendig, beide Vorlagen - die im Quellstadium und die im Zielstadium - zu analysieren. Anschließend wird die Liste mit betroffenen Dateien aus den Vorlagen selbst, den verwendeten Fragmenten und Ressourcen der beiden Vorlagenversionen erstellt.

Die Einträge dieser Listen werden nun der Reihe nach überprüft. So können folgende Unterschiede festgestellt werden:

- Vorlage/Fragment/Ressource verändert
- Vorlage/Fragment/Ressource angelegt

- Vorlage/Fragment/Ressource gelöscht

5.5.2.2 Textbausteinvergleich

Der nächste Schritt ist der Vergleich aller verwendeter Textbausteine. Auch hier werden die Listen von verwendeten Textbausteinen von beiden Vorlagenstadien aus der Datenbank abgerufen.

Bei der Überprüfung ist acht darauf zu geben, dass auch Änderungen an der exklusiven Zuordnung des Textbausteins erkannt werden. Dabei wird nur die Zuordnung betrachtet, die sich auf die aktuelle Vorlage bezieht. Andere Zuordnungen müssen bei der entsprechenden Vorlage freigegeben werden.

So können folgende Unterschiede erkannt werden:

- Textbaustein erstellt und einem Textfeld zugeordnet
- Textbaustein verändert
- Textbaustein gelöscht
- Exklusivzuordnung eines Textbausteins verändert

5.5.2.3 Textfeldvergleich

Der Vergleich bezieht sich ausschließlich auf die im Datenmodell erfassten Metainformationen eines Textfeldes, da Änderungen am tatsächlichen Textfeld auf der Vorlage als Änderung der Vorlagendatei erfasst werden.

Zwar werden bei den Textfeldern auch beide Stadien zur Bildung einer Liste herangezogen, jedoch erfolgt die Prüfung nur auf Unterschiede der Metainformationen existierender Felder, wie zum Beispiel der Feldbeschreibung.

Das Neuanlegen von Textfeldern wird hier nicht erfasst. Das liegt daran, dass das Anlegen von Textfeldern weniger an die tatsächlichen Felder als an die damit verknüpften Textbausteine und Vorlagen gebunden ist.

Textfelder werden folglich beim Anlegen von Textbausteinen und bei der Übernahme von Vorlagen und Fragmenten im Hintergrund automatisch angelegt. Das manuelle Freigeben dieser Änderung könnte hingegen zu Inkonsistenz im Datenbestand führen. Mehr dazu in Abschnitt 5.5.3 „Freigabe“ auf der folgenden Seite.

Das Löschen von Textfeldern wurde vorerst aus der Überlegung ausgelassen. Da Textfelder meist von mehreren Vorlagen verwendet werden, könnte das Übernehmen des Löschens eines Textfeldes bei einer Vorlage viele andere negativ beeinflussen.

Es werden also folgende Unterschiede erkannt:

- Textfeldeigenschaften verändert

5.5.2.4 Ergebnis der Vergleiche

Nachdem alle Operationen ausgeführt wurden, erhält der Vorlagenentwickler in der Freigabeoberfläche eine Übersicht mit erkannten Änderungen. Diese sind nach den eben vorgestellten Bereichen gegliedert. In Abbildung 5.9 „Anzeige von erkannten Änderungen“ wird gezeigt, wie eine Auswahl von Änderungen getroffen werden kann, die im nächsten Schritt freigegeben werden.

Kategorie	Titel	Freigabe
▼ Datei		
verändert	skr_DOCGEN_Utils.xdp	☑
▼ Textbaustein		
verändert	TB7: 10%	☑
verändert	TB6: 15%	☑
hinzugefügt	TB113 5%	☑
entfernt	TB118: kein rabatt	☑
▼ Textfeld		
verändert	txt_e_rabatt_prozent	☑

Abbildung 5.9: Anzeige von erkannten Änderungen

5.5.3 Freigabe

Ist die Auswahl getroffen, so müssen die gewählten Änderungen ins Zielstadium übertragen werden. Dies geschieht abhängig davon, in wievielen verschiedenen Kategorien diese Änderungen liegen, in bis zu vier Schritten.

5.5.3.1 Dateiübernahme

Das Übernehmen einer Dateiänderung wird durch Löschen oder Kopieren einer Datei im Vorlagenrepository realisiert.

Beim Kopieren kann die interne Versionsverwaltung mit geringem Aufwand verwendet werden. So muss festgestellt werden, ob die Zieldatei schon existiert. Ist dass der Fall, so wird eine neue Version dieser Datei mit dem neuen Inhalt angelegt. Existiert sie noch nicht, so wird die Zieldatei als Erstversion erstellt. Vorteil dabei ist, dass so auch auf vorherig freigegebene Dateien zugegriffen werden kann.

5.5.3.2 Anlegen von neuen Textfeldern

Nachdem alle Dateiänderungen übernommen wurden, wird der Analyseprozess auf den Vorlagenstand im Zielstadium angewendet. Die dadurch erkannten Textfelder werden - sofern sie noch nicht existieren - vom System automatisch angelegt. Dabei dienen die Textfeldinformationen aus dem Quellstadium als Grundlage.

5.5.3.3 Textbausteinübernahme

Der Umfang dieser Operation ist abhängig von der Art der Änderung. Dabei stellt das Ändern eines existierenden Textbausteins den kleinsten Umfang dar.

Wird ein Textbaustein neu angelegt, so muss als erstes sichergestellt sein, dass das zugehörige Textfeld existiert. Dieses wird dann gegebenenfalls aus dem Quellstadium übernommen. Nachdem der eigentlichen Textbaustein dann erstellt oder angepasst wurde, erfolgen die Zuordnungen zu Textfeld und gegebenenfalls zur Vorlage. Hier wird nur die Zuordnung zur aktuellen Vorlage betrachtet, da nur diese eine direkte Änderung an der Vorlage darstellt.

Beim Löschen müssen diese Zuordnungen vor dem Entfernen des Textbausteins gelöst werden. Hier ist es notwendig, alle Zuordnungen zu entfernen, da sich der Textbaustein sonst aufgrund von Constraints in der Datenbank nicht entfernen lässt.

5.5.3.4 Textfeldübernahme

Analog zum Vergleich sind auch hier nur die im Datenmodell erfassten Metainformationen eines Textfeldes gemeint. Die Aufgabe, solche Änderungen zu übernehmen, ist trivial. Es muss lediglich der Datenbankeintrag des Textfeldes aktualisiert werden. Änderungen an Zuordnungen des Textfeldes sind nicht notwendig.

5.5.3.5 Aktualisierung der Anzeige

Nachdem alle Aktionen durchgeführt wurden, wird der Vergleichsprozess erneut ausgeführt, um die Anzeige von Unterschieden zu aktualisieren. Dies gibt zum einen eine Kontrollmöglichkeit der erfolgten Freigabe, zum anderen die Chance, vorher abgewählte Änderungen doch noch freizugeben.

5.6 Integration in den Designer

Im praktischen Arbeitsablauf eines Vorlagenentwicklers tritt zum aktuellen Zeitpunkt noch ein Medienbruch auf. Um vom Design in die Textbausteinverwaltung zu wechseln, müsste er sich den Pfad der Vorlage merken, den Textbausteineditor starten und dort die Vorlage auswählen. Im Entwicklungsalltag ist das ein unnötiger Aufwand. Besser ist der Start des *TEd* direkt aus dem *LiveCycle Designer* heraus.

5.6.1 Art der Integration

Hier wäre der Aufruf des Programms aus dem Menü denkbar. Dabei wird im Textbausteineditor die Vorlage geöffnet, die gerade im Editor bearbeitet wird. So muss der Vorlagenentwickler nicht mehr per Hand den Browser starten und zur betreffenden URL navigieren.

Es soll auch ermöglicht werden, dass eine Vorlage im Dateibaum des Repositories zum Bearbeiten gewählt werden kann. Um das umzusetzen, wird ein zusätzlicher Eintrag im Kontextmenü der XDP-Dateien im Vorlagenrepository platziert. Bei dessen Aufruf wird ebenfalls der *TEd* ausgeführt, welchem der Name der betroffenen Vorlage übermittelt wird.

5.6.2 Erweiterbarkeit von Eclipse

Da der Vorlagendesigner *Eclipse*-basiert ist, kann dessen Schnittstelle zur Entwicklung von Plugins verwendet werden. Bei Eclipse handelt es sich um eine offene Plattform, die darauf ausgelegt ist, durch Dritte erweitert zu werden. Mit Ausnahme des *Eclipse*-Kerns sind alle Funktionen, die man in der Entwicklungsumgebung findet, Plugins.

Das Prinzip ist einfach zu verstehen. Der *Eclipse*-Kern oder auch vorhandene Plugins bieten Erweiterungspunkte (sogenannte Extension Points) an. Der Entwickler eines neuen Plugins kann diese Erweiterungspunkte nutzen, um an diesen Stellen seine Erweiterung (auch Extension genannt) anzudocken. Extension Points sind zum Beispiel die Erweiterung des Hauptmenüs, das Hinzufügen von Eigenschaftsseiten zu Dokumenten, sowie eigene Anzeige- und Editierkomponenten.

Die Konfiguration der Erweiterung, also welche Eigenschaften sie hat und welche Extension Points sie wie benutzt, werden in einer Manifestdatei festgehalten. Diese liegt im XML-Format vor. In aktuellen *Eclipse*-Umgebungen existiert hierfür ein Editor, so dass kaum noch im XML-Quellcode gearbeitet werden muss. Diese Konfiguration ergibt zusammengepackt mit benötigten Ressourcen und Klassen das eigentliche Plugin in Form eines Javaarchives, wie in Abbildung 5.10 „Pluginarchitektur Eclipse, angelehnt an [epl08]“ skizziert wird.

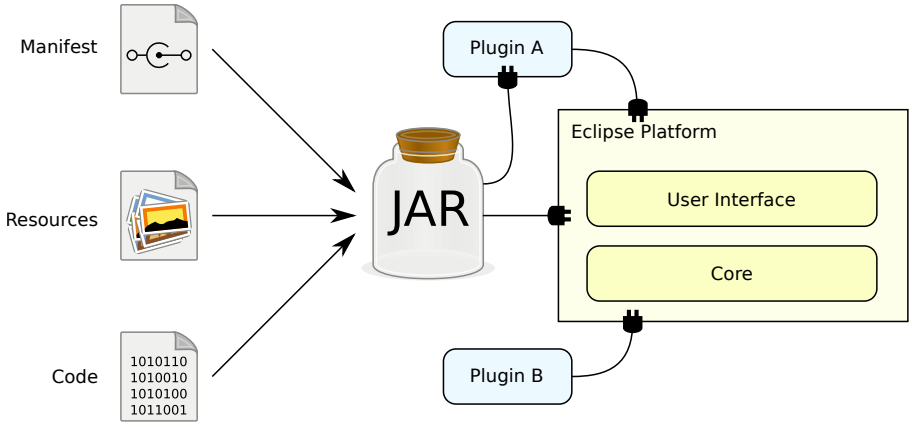

Abbildung 5.10: Pluginarchitektur Eclipse, angelehnt an [epl08]

5.6.3 Konzeption des Plugins

Betrachtet man die Anforderungen, so ergeben sich die folgenden zu verwendenden Extension Points:

org.eclipse.ui.actionsets Über Actionsets kann ein Eintrag im Hauptmenü von *Eclipse* erfolgen. Dieser wird verwendet, um den Textbausteineditor so auszuführen, dass er die aktuell bearbeitete Vorlage öffnet.

org.eclipse.ui.popupmenus Hier verrät der Name bereits, dass das Anlegen von Popupmenüeinträgen gemeint ist. So kann eingestellt werden, dass im Popupmenü der XDP-Dateien der Aufruf des *TEd* möglich ist. Siehe dazu Abbildung 5.11 „Eintrag im Popupmenü des LiveCycle Designers“ auf der nächsten Seite.

org.eclipse.ui.preferencepages Die URL des Textbausteineditors kann nicht fest vorgeschrieben werden, da das Plugin sonst nicht mehr auf ein anders konfiguriertes System übertragbar wäre. Deshalb wird eine Einstellungsseite erzeugt, in der diese URL konfiguriert werden kann.

org.eclipse.core.runtime.preferences Trotzdem kann eine Standardeinstellung für den Kunden als Defaultwert definiert werden.

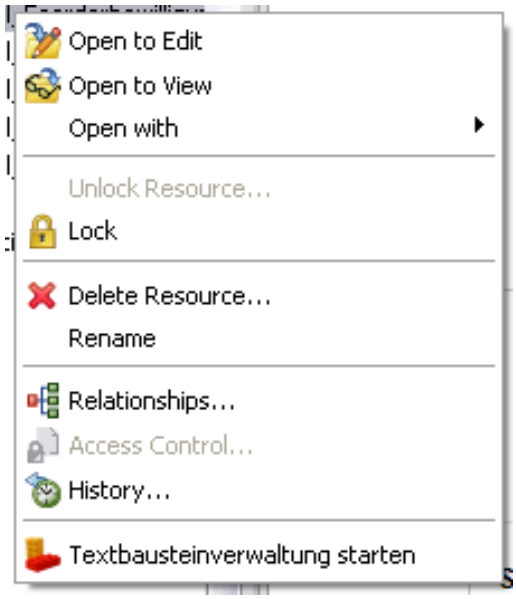

Abbildung 5.11: Eintrag im Popupmenü des LiveCycle Designers

Unabhängig davon, auf welchem Weg der *TEd* gestartet werden soll, muss ein Webbrowser geöffnet werden, der die URL aufruft, über die der Textbausteineditor erreichbar ist. Zwar ist es möglich, ein Programm aus Java heraus zu starten, nur bestünde dabei das Problem, festzustellen, welcher Browser verwendet werden soll. Das kann umgangen werden, indem die *Eclipse*-API verwendet wird. Sie stellt die Funktion bereit, den in *Eclipse* als Standard konfigurierten Browser zu starten. Per Get-Parameter kann die gewünschte Konfiguration an den Textbausteineditor weitergeleitet werden.

6 Entwicklungsumgebung und Softwareverteilung

An die eigentliche Entwicklung grenzen einige Bereiche an, die relativ eng damit zusammenhängen, jedoch nicht explizit zur Entwicklung gehören. Diese werden im folgenden kurz vorgestellt.

6.1 Verwendung einer Entwicklungsumgebung

Die Verwendung einer Entwicklungsumgebung ist nicht zwingend erforderlich. Actionscript und MXML können beide direkt in einem Texteditor erstellt und bearbeitet werden. Das Kompilieren ist auf der Kommandozeile möglich. Jedoch fehlt so der Komfort, den die proprietäre Entwicklungsumgebung *Adobe Flex Builder* bietet.

Gerade bei der Entwicklung von Rich Internet Applications wird der Gestaltung der Oberfläche eine besonders wichtige Rolle zugemessen. Um diese zu gestalten, kann neben der MXML-Quelltext-Ansicht der grafische Designer aus Abbildung 6.1 „Oberflächendesigner des Flex Builder“ verwendet werden.

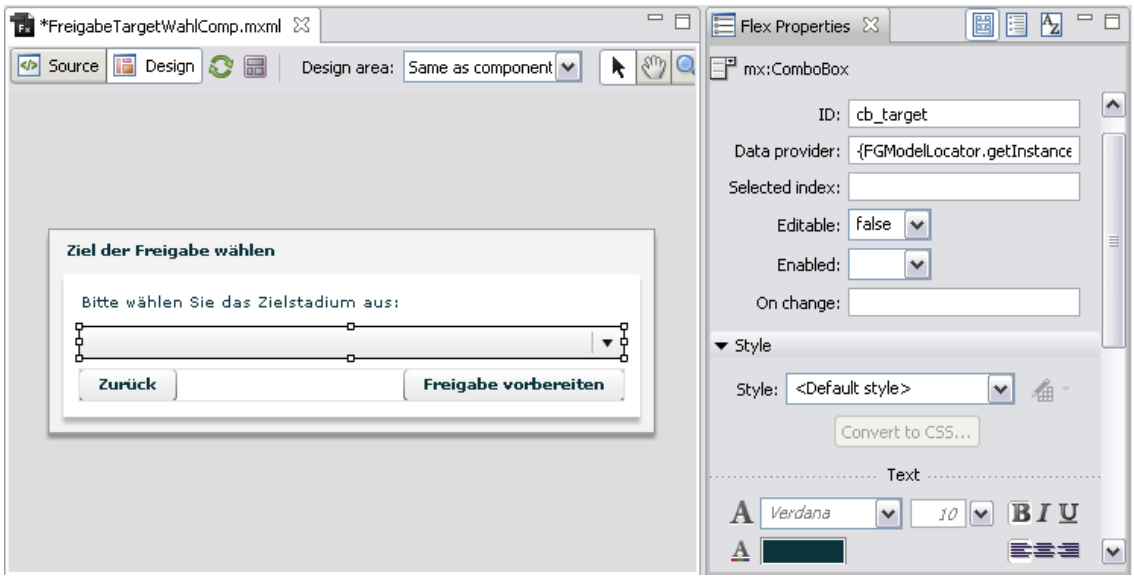

Abbildung 6.1: Oberflächendesigner des Flex Builder

In der Projektansicht in Abbildung 6.2 „Ansicht des Projekts in Flex Builder“ auf der nächsten Seite ist die Ordnerstruktur erkennbar, die aus der Verwendung des Cairngorm-Frameworks resultiert.

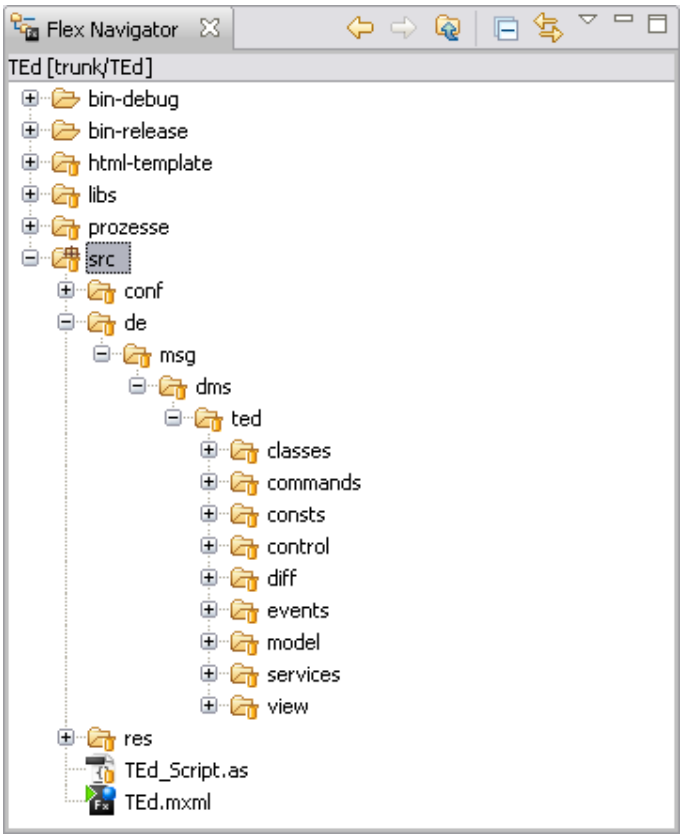

Abbildung 6.2: Ansicht des Projekts in Flex Builder

Zur Versionsverwaltung der Quellcodes wurde Subversion eingesetzt. Da auch *Flex Builder* auf *Eclipse* basiert, kann das entsprechende SVN-Plugin für *Eclipse* auch in den *Flex Builder* integriert werden.

6.2 Softwareverteilung

6.2.1 Textbausteineditor

Die Lösung kann technisch in Server und Client unterschieden werden. Der Client ist dabei als Rich Internet Application realisiert. Vorteil der RIA-Anwendung ist, dass sie nicht bei jedem einzelnen Arbeitsplatzrechner installiert werden muss, sondern an zentraler Stelle im Server eingespielt wird. Von dort aus ist der Textbausteineditor dann von allen Rechnern aus über den Webbrowser verfügbar.

Bei der zentralen Stelle handelt es sich um einen *JBoss* Application Server. Die Anwendung wird mitsamt ihrer Konfiguration in ein sogenanntes Enterprise Archive (EAR) gepackt und auf dem Server in ein dafür vorgesehenes Verzeichnis kopiert. Dieser erkennt, dass das Archiv erneuert wurde und stellt die neuen Inhalte zum Abruf bereit.

6.2.2 Designerintegration

Der Textbausteineditor kann aus dem *Eclipse*-basierten Werkzeug zum Erstellen der Dokumentvorlagen, dem *LiveCycle* Designer, gestartet werden. Zur nahtlosen Integration ist eine Erweiterung des Designers um ein Plugin notwendig. Dieses Plugin muss hingegen auf jedem Arbeitsplatz manuell installiert werden.

Dazu wird eine sogenannte *Update Site* erstellt, ein Ordner, der lokal oder über einen Server erreichbar, das Plugin und zugehörige Metainformationen enthält. Anschließend kann im Designer, wie bereits aus *Eclipse*-Umgebungen bekannt, die *Update Site* hinzugefügt und aus ihr das Update installiert werden.

7 Zusammenfassung und Ausblick

7.1 Erreichte Ergebnisse

Zusammenfassend kann festgestellt werden, dass die gesteckten Ziele im Verlauf der Arbeit erreicht werden konnten. Diese sind in den folgenden zwei Abschnitten nochmals kurz zusammengefasst.

7.1.1 Wissenschaftliches Ergebnis

Im Rahmen der Arbeit hat ein Vergleich von Vorgehensmodellen der Softwareentwicklung stattgefunden, der zum Ziel hatte, das geeignete Modell für die Entwicklung von Vorlagen zu finden.

Da keines der betrachteten Modelle eindeutig den gestellten Anforderungen entsprach, wurde aus den gewonnenen Erkenntnissen des Vergleichs ein angepasstes Vorgehensmodell spezifiziert. Als Rahmen und Beschreibungswerkzeug kam dazu das V-Modell XT zum Einsatz.

Es wurde dabei gezeigt, dass die Erkenntnisse der klassischen Softwareentwicklung durchaus auf andere, ähnliche Gebiete ausgeweitet werden können. Dazu ist die jeweilige Anpassung der Modelle auf die spezifischen Gegebenheiten unabdinglich.

7.1.2 Wirtschaftliches Ergebnis

Im Weiteren ist eine leicht-installierbare, flashbasierte RIA-Anwendung entstanden. Sie bietet dem Vorlagenentwickler technische Unterstützung bei der Textbausteinverwaltung gestalteter Vorlagen.

Das spezifizierte Vorgehensmodell erfordert zur reibungsfreien und mehraufwandsarmen Anwendung eine softwareseitige Unterstützung. Diese wurde in das Programm integriert. Es ist damit möglich, die Änderungen an Vorlage, Textfeldern und Textbausteinen in wenigen Schritten von einem Stadium in ein folgendes freizugeben.

Zur optimalen Integration des neuen Tools in den Arbeitsablauf wurde eine Erweiterung des Vorlagendesigners entwickelt, welche es ermöglicht, den Textbausteineditor direkt aus der Oberfläche des *LiveCycle* Designers heraus aufzurufen.

Das Arbeitsergebnis dieser Anwendungsentwicklung wird dem Kunden zum späteren Einsatz vorgestellt.

7.2 Kritische Wertung

Das entstandene Programm erfüllt die Anforderungen, die daran gestellt werden. Nichts desto trotz gibt es einige wenige Kritikpunkte, die hier erläutert werden.

Durch Änderungen an Vorlagen kann es dazu kommen, dass vorhandene Textfelder, Textbausteine und Fragmente nicht mehr notwendig sind. Das wird zur Zeit vom System noch nicht festgestellt. Unnötige Elemente haben keinen Einfluss auf die Funktionsfähigkeit der Vorlagen. Auch die Belastung der Server durch meist kleine Dateien und einige Datenbankeinträge ist nahezu vernachlässigbar gering. Trotzdem fördern diese Rückstände die Unübersichtlichkeit im System.

Das Freigeben von Änderungen an Fragmenten, Textfeldern und Textbausteinen beeinflusst durch die modulare, mehrfache Verwendung dieser Elemente potentiell auch andere Vorlagen. Die Software bietet bisher noch keine weitere Behandlung dieser Seiteneffekte an. Die dadurch entstehenden Wirkungen können im Test jedoch festgestellt werden.

7.3 Ausblick

Mit der Fertigstellung dieser Arbeit ist der Themenkomplex sowie die Entwicklung der Software noch nicht besiegelt. In vielerlei Hinsicht gibt es Möglichkeiten zur Anknüpfung und Weiterentwicklung.

7.3.1 Wissenschaftlicher Ausblick

Eine Vorlage besteht aus vielen Bestandteilen, was die Portierung einer Vorlage auf ein anderes System, wie es zum Beispiel bei Supportanfragen eines Vorlagenentwicklers zu bestehenden Vorlagen an den Produkthersteller (msg systems ag) geschieht, deutlich verkompliziert. Die Entwicklung eines Transportformats, welches alle für die Funktion einer Vorlage notwendigen Bestandteile erfassen kann, wäre ein Schritt in die Richtung, eine solche Übertragung zu vereinfachen. So könnte eine Vorlage in komplettem Umfang in einer einzelnen Datei mit der Supportanfrage versendet werden.

Es ist zu prüfen, ob die Vorgehensmodelle der Softwaretechnik nicht auch noch in anderen Bereichen neben der Vorlagenentwicklung eingesetzt werden können. Selbst die Anwendung einzelner Grundprinzipien dieser Modelle könnte Vorteile bringen.

7.3.2 Wirtschaftlicher Ausblick

Bezugnehmend auf die erfolgte Wertung ergeben sich zwei mögliche Erweiterungen des Systems:

In der weiteren Entwicklung des Programms ist es in Bezug auf das gezielte Testen der Vorlagen denkbar, bei der Freigabe zusätzlich eine Liste von Wechselwirkungen zu anderen Vorlagen anzuzeigen. Über die Datenbank und die Ergebnisse der Vorlagenanalyse können die nötigen Informationen darüber gewonnen werden.

Ebenfalls wäre ein Tool hilfreich, welches durch Analyse des Gesamtvorlagenbestandes Elemente erkennt, welche nicht mehr verwendet werden. Die Entscheidung, welche davon gelöscht werden können und welche nur temporär unverwendet sind, kann dabei nicht vom Programm getroffen werden. Dieser Wartungsmodus könnte auch stadienbezogen in den Textbausteineditor integriert werden.

Einige getätigte Entwicklungen können in anderen Projekten wiederverwendet werden. Ein Beispiel dafür ist die Analysefunktionalität für Vorlagen, die das Feststellen von Abhängigkeiten zwischen Vorlagen und Fragmenten realisiert. Eine solche Funktion sollte im Produkt *Adobe LiveCycle* Einzug finden.

Anhang A: Reihe von Bildschirmfotos

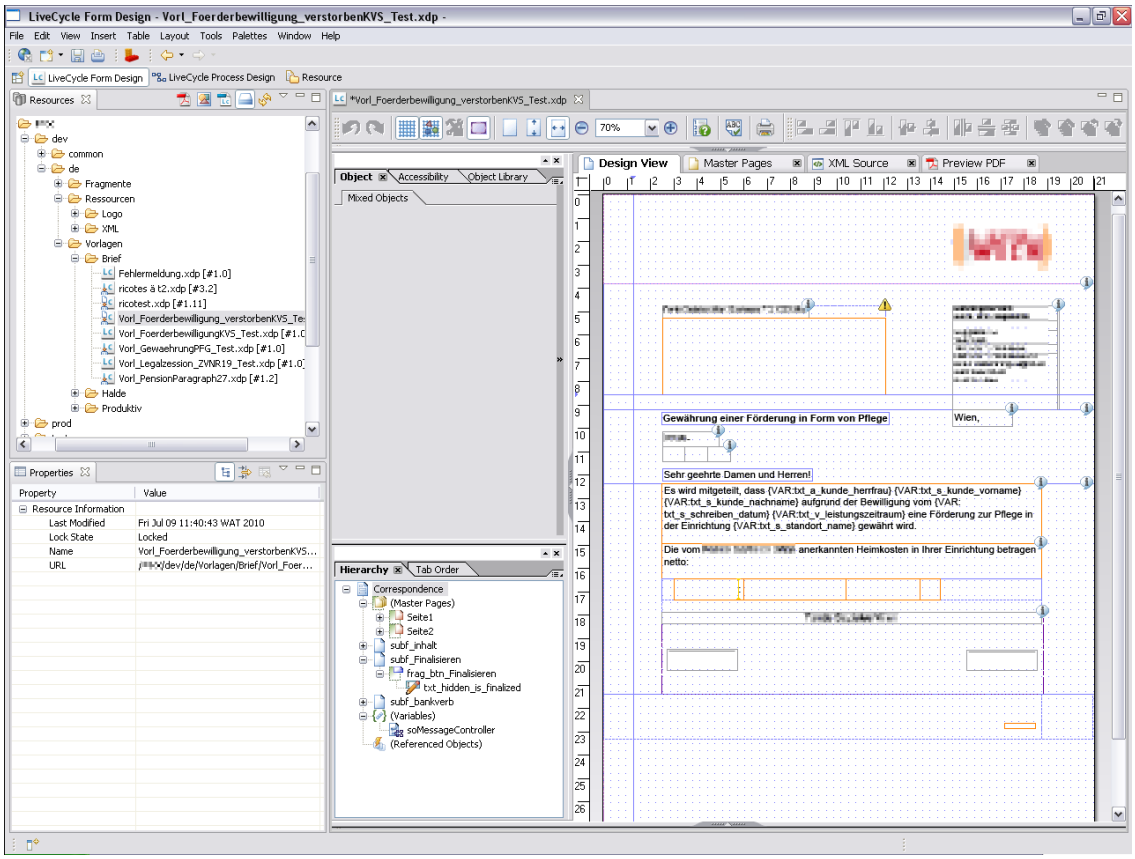

Abbildung A.1: LiveCycle Designer

Abbildung A.2: Starten des TEd über die Integration

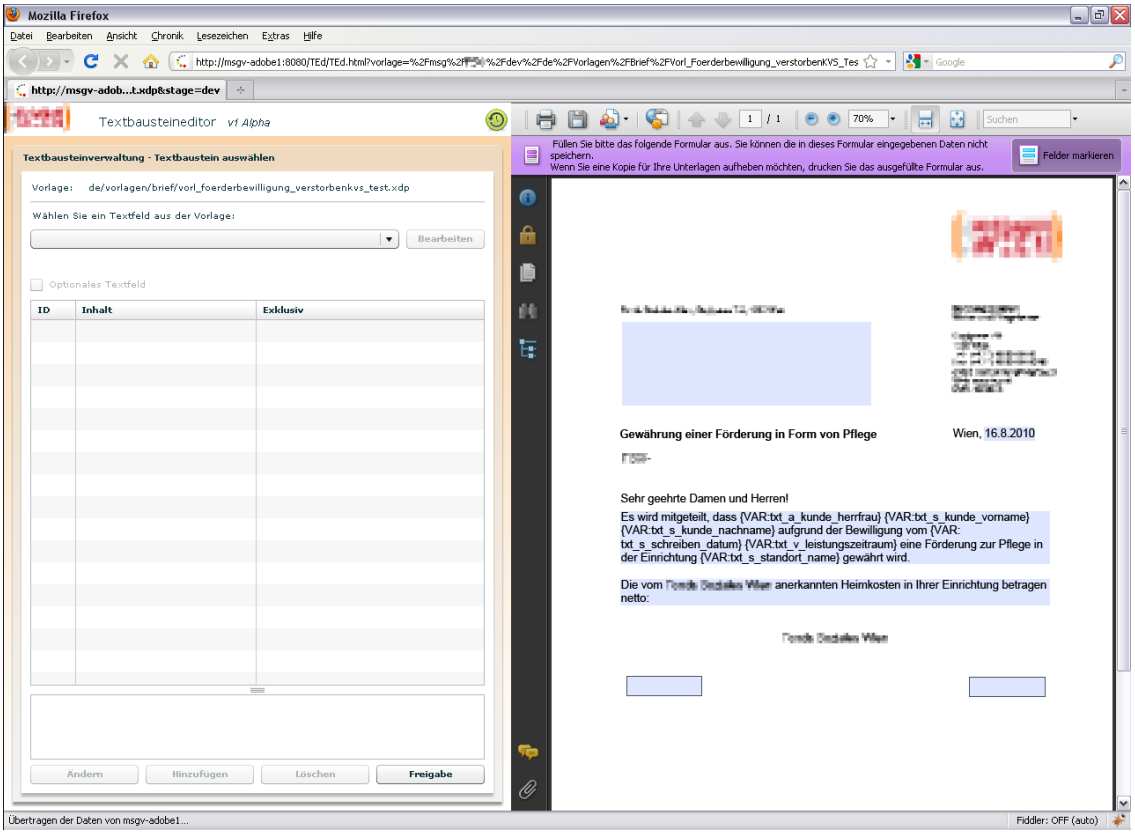

Abbildung A.3: Textbausteineditor

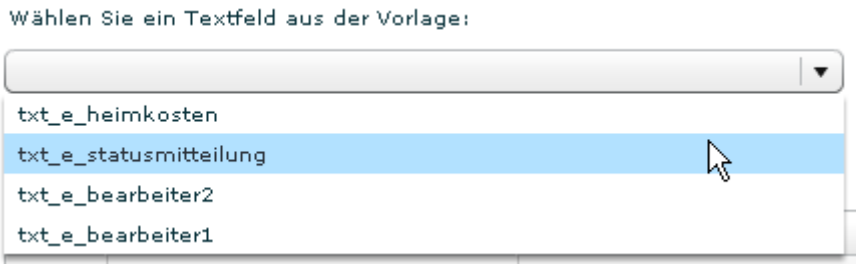

Abbildung A.4: Wahl eines Textfeldes

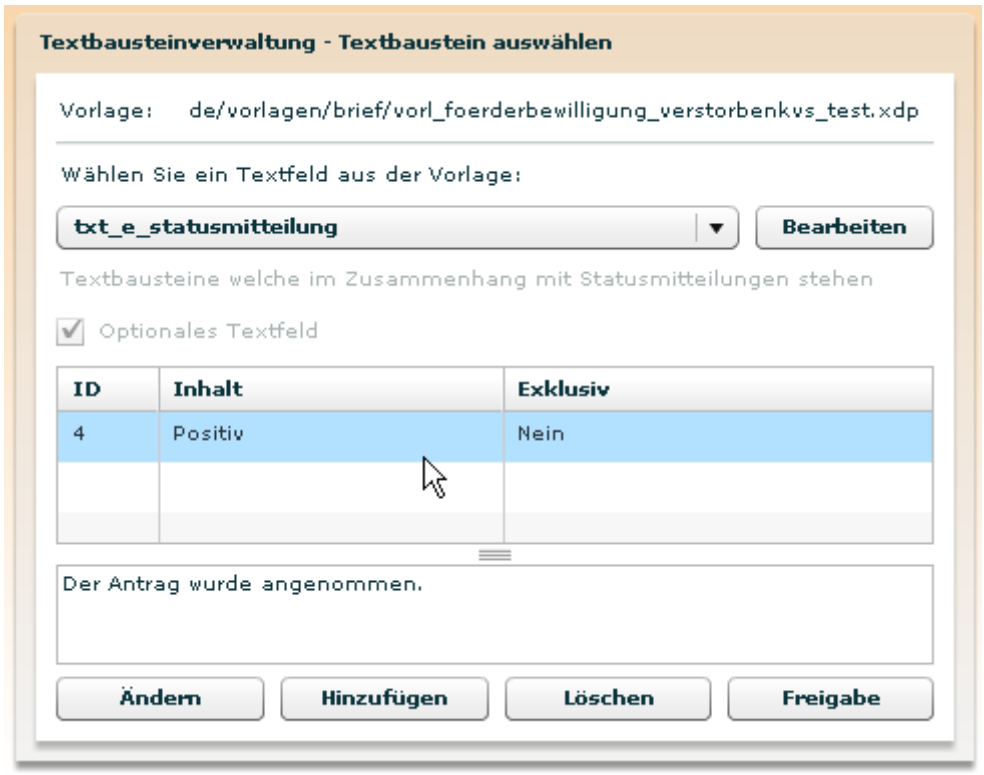

Abbildung A.5: Anzeige der Textbausteine des Textfeldes

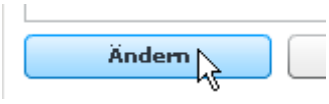

Abbildung A.6: Klick auf Ändern

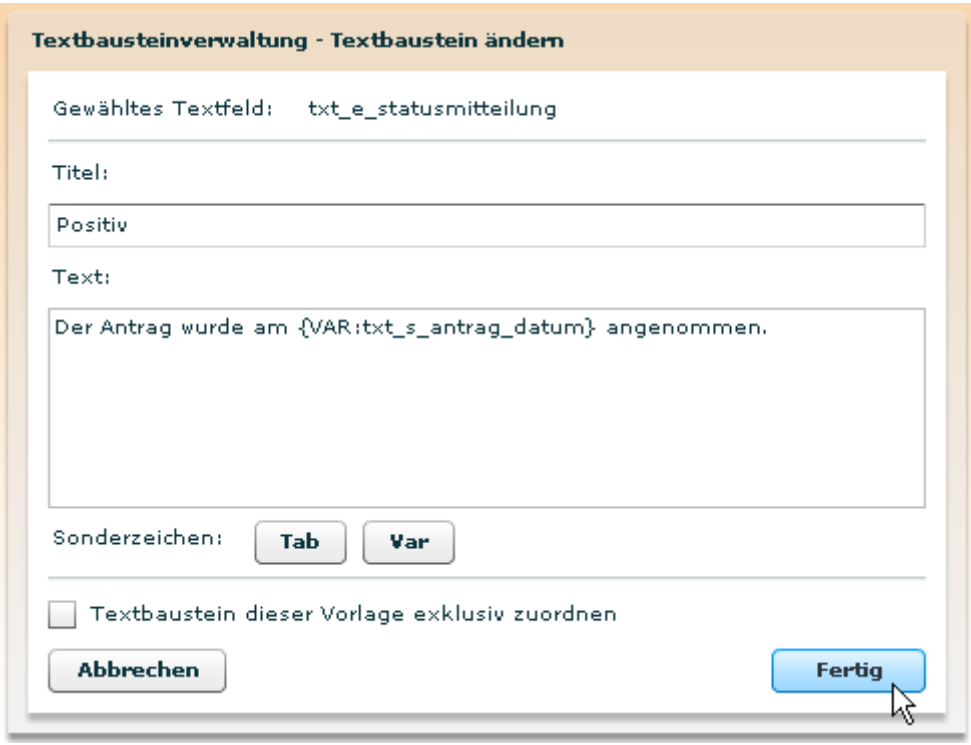

Abbildung A.7: Änderung am Textbaustein

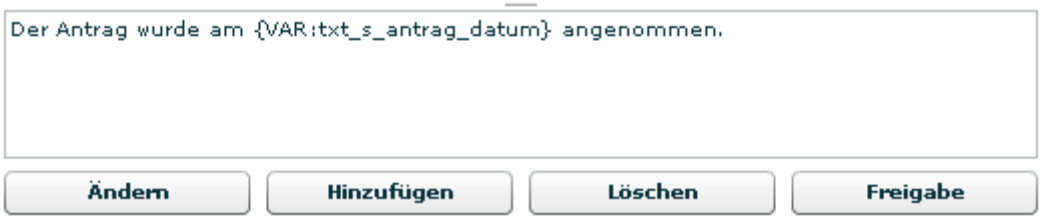

Abbildung A.8: Anzeige des geänderten Textes

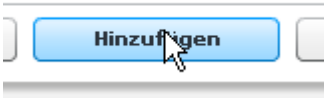

Abbildung A.9: Klick auf Hinzufügen

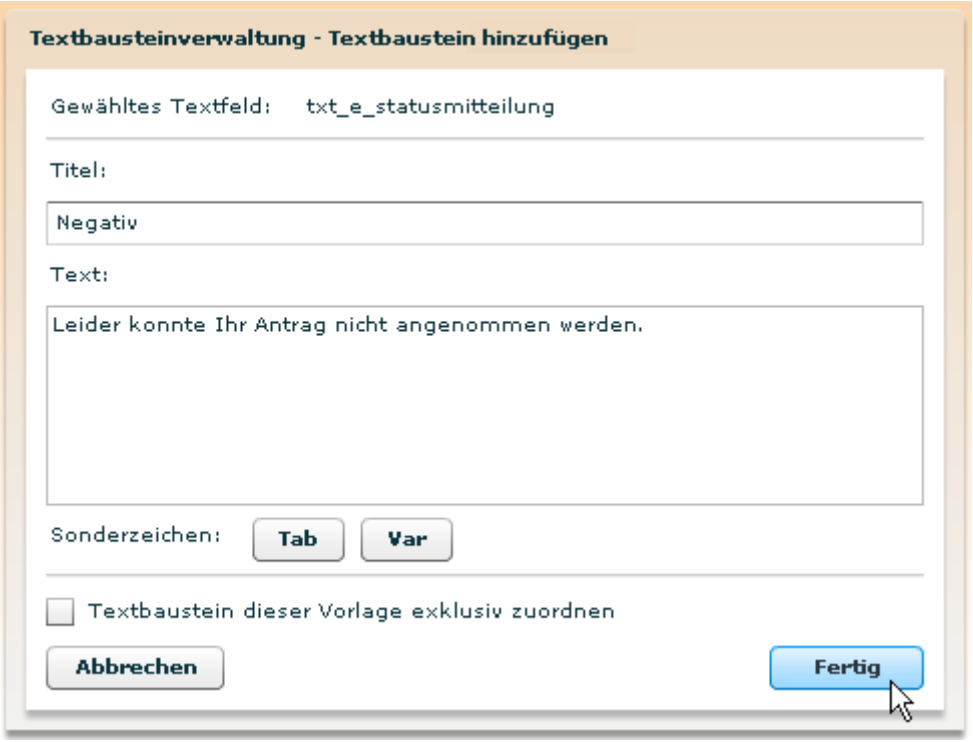

Abbildung A.10: Eingabe der Daten eines neuen Textbausteins

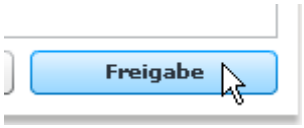

Abbildung A.11: Öffnen der Freigabeverwaltung

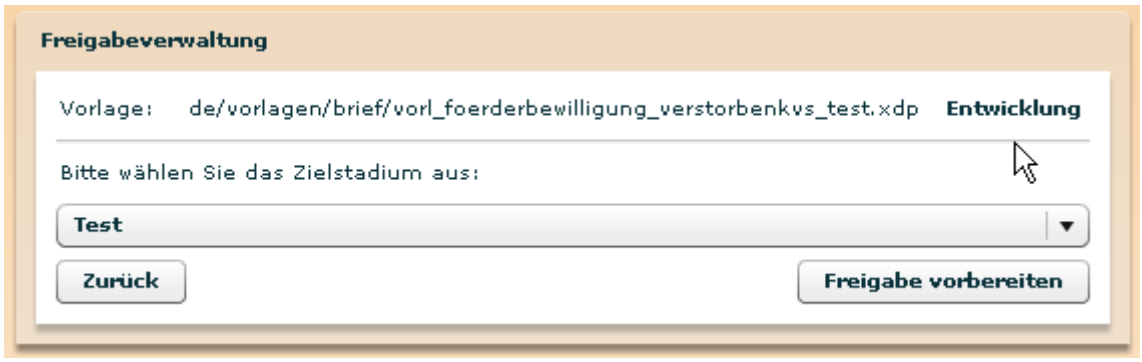

Abbildung A.12: Zielauswahldialog

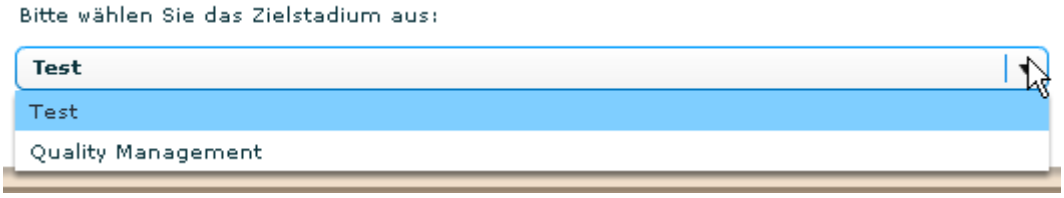

Abbildung A.13: Auswahl eines Zielstadiums

Abbildung A.14: Klick auf Freigabe vorbereiten

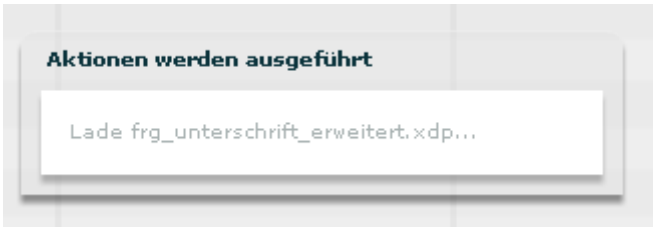

Abbildung A.15: Feststellen der Unterschiede

Abbildung A.16: Anzeige der Unterschiede

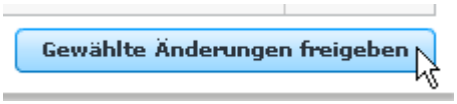

Abbildung A.17: Freigabe auslösen

Abbildung A.18: Freigeben der Unterschiede

Glossar

Constraint In Datenbanksystemen definieren Constraints Bedingungen für Tabellen, die bei allen Operationen eingehalten werden müssen. Beispiel ist der Fremdschlüssel. Es können nur Werte eingefügt werden, für die in der Referenztabelle ein Gegenstück existiert. Ein Wert der Referenztabelle kann im Gegenzug nur entfernt werden, wenn er nicht referenziert wird.

Corporate Design Das Corporate Design ist der Teil der Corporate Identity, der das äußere Erscheinungsbild der Unternehmens umfasst. Darunter zählen unter anderem Geschäftsbriefe und Internetauftritt.

Corporate Identity Als Corporate Identity wird das strategische Konzept zur Gestaltung einer einheitlichen Unternehmenskultur bzw. eines einheitlichen Unternehmensauftritts bezeichnet. Es handelt sich um die Zusammenfassung der dem Unternehmen zu Grunde liegenden Unternehmensphilosophie. Dabei soll die Steigerung der Mitarbeitermotivation durch Identifikation mit dem Unternehmen, eine Positionierung des Unternehmens am Markt und die Steigerung des Bekanntheitsgrades sowie der Kundenzufriedenheit erreicht werden [cil10].

Corporate Wording Corporate Wording bezeichnet als Teil des Corporate Identity die Festlegung einer unternehmensweit einheitlichen Sprachkultur. Darunter fallen beispielsweise Vorgaben zu Verwendung und Vermeidung bestimmter Formulierungen.

Data-Binding Data-Binding ist eine Technologie, die es erlaubt, zwei Datenquellen miteinander zu verbinden, sodass deren Inhalte synchron miteinander gehalten werden. So können Elemente der Oberfläche an das Datenmodell gebunden werden, die sich bei Änderung des Modells automatisch aktualisieren. Die manuelle Aktualisierung der Oberfläche kann so gespart werden. Das Data-Binding von Flex ist unidirektional.

Intelligent Character Recognition ICR kann als eine Erweiterung des Optical Character Recognition angesehen werden. Aufgrund der Position erkannter Textbestandteile kann eine fachliche Zuordnung dieser geschehen. Damit ist auch die Möglichkeit der Plausibilisierung einzelner Werte gegeben.

Legal Wording Innerhalb des Corporate Wording bildet das Legal Wording die Vorgaben zu juristisch relevanten, rechtssicheren Formulierungen.

Medienbruch Erfolgt bei der Übertragung von Informationen innerhalb der Übertragungskette ein Wechsel des Mediums, so wird von einem Medienbruch gesprochen. Medienbrüche bergen die Gefahr der Informationsverfälschung und ziehen eine Verlangsamung der Informationsbearbeitung nach sich [mbr].

Optical Character Recognition Vorzugsweise eingescannte analoge Dokumente werden mit diesem Verfahren behandelt. Dabei werden in der vorhandenen Bilddatei des Dokuments vorkommende Texte auf Grundlage der Erkennung von Buchstaben wiederhergestellt und so der maschinellen Weiterverarbeitung zur Verfügung gestellt.

rendern In diesem Kontext bezeichnet das Rendern die Umsetzung der in der Vorlage getroffenen Festlegungen in ein Dokument. Dabei wird die Vorlage gegebenenfalls um dynamische Inhalte angereichert.

Repository Ein Repository ist ein verwaltetes Verzeichnis. Im Falle des Vorlagenrepositories des *Adobe LiveCycle* Servers handelt es sich dabei um einen Verzeichnisbaum, der über den visuellen Vorlagendesigner und über die Java-API angesprochen werden kann. Bei Änderung der darin enthaltenen Dateien wird für diese eine neue Version angelegt. Auf die Historie kann zugegriffen werden.

XSLT XSLT ist eine Programmiersprache, mit deren Hilfe Umwandlungen zwischen XML-Strukturen beschrieben werden können. So können Umwandlungsregeln definiert werden, die auf ein XML-Quelldokument angewendet einen neuen XML-Baum erzeugen. Die Ausgabe ist nicht auf XML beschränkt.

Literatur- und Quellenverzeichnis

[adl10] *Adobe - Livecycle Enterprise Suite*. http://www.adobe.com/de/products/livecycle, 2010. (Online am 17. Juni 2010).

[ado10] *Open Source Framework | Adobe Flex*. http://www.adobe.com/de/products/flex, 2010. (Online am 27. Mai 2010).

[aii10a] *AIIM - About AIIM*. http://www.aiim.org/AboutAIIM/ECM-ERM-BPM-Association.aspx, 2010. (Online am 22. Juni 2010).

[aii10b] *Enterprise Content Management: ECM Komponenten (Quelle: AIIM / PROJECT CONSULT 2003)*. http://upload.wikimedia.org/wikipedia/de/c/c8/ECM_Komponenten.jpg, 2010. (Online am 11. Juni 2010).

[as307] *Komponenten-Referenzhandbuch für ActionScript 3.0*. http://livedocs.adobe.com/flash/9.0_de/ActionScriptLangRefV3/operators.html, 2007. (Online am 18. Juli 2010).

[Bal98] BALZERT, HELMUT: *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Spektrum, Akademischer Verlag, Heidelberg, Berlin, 1998.

[Bil09] BILEK, SYLVIA: *Geschäftsprozessanalyse und Konzeption zur Prozessverbesserung unter Verwendung agiler Methoden und Praktiken bei einer mittelständischen Internetagentur*. Diplomarbeit, Hochschule Mittweida - University of Applied Sciences, 2009.

[BSB09] BÖHN, MARTIN, MICHAEL SCHIKLANG und STEFANIE BERGMANN: *Output Management - Systeme für Output Management im Vergleich*. BARC, Würzburg, 2009. Leseprobe.

[Bun05] BUNDESMINISTERIUM DES INNERN: *DOMEA-Konzept - Organisationskonzept 2.1 Dokumentmanagement und elektronische Archivierung im IT-gestützten Geschäftsgang*. Schriftenreihe des KBSt, Band 61, 2005.

[cil10] *Rechtslexikon - Corporate Identity*. http://www.juraforum.de/lexikon/corporate-identity, 2010. (Online am 28. Juli 2010).

[epl08] *Introduction to Eclipse Plugin Development - Plug-in Architecture*. http://www.eclipsepluginsite.com/eclipse-plugin-development/image004.jpg, 2008. (Online am 21. Juli 2010).

[GK08] GEIGER, WALTER und WILLI KOTTE: *Handbuch Qualität - Grundlagen und Elemente des Qualitätsmanagements: Systeme - Perspektiven*. Vieweg+Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden, 5. Auflage, 2008.

[Gyg04] GYGER, DANIEL: *Feature Driven Development - Agile vs. klassische Methoden der Software-Entwicklung*. http://www.ifi.uzh.ch/rerg/fileadmin/downloads/teaching/seminars/seminar_ws0304/08_Gyger_Fdd_Ausarbeitung.pdf, 2003/04. (Online am 2. Juli 2010).

[HGB09] *Handelsgesetzbuch*. http://www.dejure.org/gesetze/HGB, 2009. (Online am 28. Juni 2010).

[HH08] HÖHN, REINHARD und STEPHAN HÖPPER: *Das V-Modell XT - Anwendungen, Werkzeuge, Standards*. Springer-Verlag, Berlin, Heidelberg, New York, 2008.

[hmc10] *Definition Dokument*. http://hmc-cp.de/def/def_doku.htm, 2010. (Online am 10. Juni 2010).

[iso00] *DIN EN ISO 9000:2000-12, Qualitätsmanagementsysteme - Grundlagen und Begriffe. Dreisprachige Fassung.* Beuth Verlag GmbH, Berlin, 2000.

[jdk10] *Was ist Enterprise Content Management (ECM)?* http://www.jdk.de/de/cms/ecm-enterprise-content-management/ecm-definition.html, 2010. (Online am 11. Juni 2010).

[LSL08] LIEBHART, DANIEL, GUIDO SCHMUTZ und MARCEL LATTMANN: *Business Communication Architecture Blueprint - Leitfaden zur Konstruktion von Output Management Systemen*. Carl Hanser Verlag, München, 2008.

[mbr] *Definition: Medienbruch - Wirtschaftslexikon*. http://wirtschaftslexikon.gabler.de/Definition/medienbruch.html. (Online am 10. August 2010).

[msg10] *msg systems ag - Wir über uns*. http://msg.de/unternehmen.0.html, 2010. (Online am 31. Mai 2010).

[net10] *.NET-Glossar*. http://zfs.informatik.rwth-aachen.de/glossar/r.aspx, 2010. (Online am 22. Juni 2010).

[pir06] *PIRONET NDH Newsletter*. http://www.pironet-ndh.com/pb/site/pndh-website-site/resource/content/newsletter/2006-09/PNDH-Newsletter-2006-09-Interview.html, 2006. (Online am 16. Juni 2010).

[pma07] *Einordnung agiler Vorgehensmodelle*. http://www.pmaktuell.org/uploads/PMAktuell-200701/036-Wissen.jpg, 2007. (Online am 29. Juni 2010).

[rep10] *Wikipedia - Reportgenerator*. http://de.wikipedia.org/wiki/Reportgenerator, 2010. (Online am 04. August 2010).

[Rig09] RIGGERT, WOLFGANG: *ECM – Enterprise Content Management*. Vieweg+Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden, 1. Auflage, 2009.

[swl06] *SwtWiki - Prozessmodelle*. http://www.imn.htwk-leipzig.de/~weicker/pmwiki/pmwiki.php/Main/Prozessmodelle, 2006. (Online am 30. Juni 2010).

[vxt06] *V-Modell XT - Übersicht über das V-Modell XT*. http://v-modell.iabg.de/index.php?option=com_content&task=view&id=25&Itemid=46, 2006. (Online am 30. Juni 2010).

[vxt09] *V-Modell XT*. http://v-modell.iabg.de/dmdocuments/V-Modell-XT-Gesamt-Deutsch-V1.3.pdf, 2009. (Online am 30. Juni 2010).

[w3s10] *XSL-FO Tutorial*. http://www.w3schools.com/xslfo/default.asp, 2010. (Online am 16. Juni 2010).

[xfa08] *XML Forms Architecture (XFA) Specification - Version 2.8*. http://partners.adobe.com/public/developer/en/xml/xfa_spec_2_8.pdf, 2008. (Online am 17. Juli 2010).

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 20. August 2010