Institut für Rechnergestützte Ingenieursysteme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3377

„Process Engine“ als Werkzeug der Automatisierung der Projektabwicklung in Unternehmen

Yangyang Gao

Studiengang:	Informatik
Prüfer:	Univ-Prof. Hon-Prof. Dr. Dieter Roller
Betreuer:	Dipl.-Inf. Felix Baumann Dipl.-Inf. Akram Chamakh
begonnen am:	26. Juli 2012
beendet am:	25. Januar 2013
CR-Klassifikation:	F.3.2, H.2.8, H.4.1

Kurzfassung

Das auf Basis von Web-Anwendungen implementierte Benutzeroberflächen-Konzept ermöglicht die Realisierung von Konsistenzprüfungen, eine unkomplizierte Fehlerbehandlung bei Daten-Inkonsistenz und die schrittweise Integration der entwickelten Dokumentengesteuerten Projektabwicklung in die bisherige Arbeitsumgebung der Firma Innovations-Solutions AG (ISAG). Zusammen mit dem strukturierten und erweiterbaren Datenbankschema werden die Fehlermöglichkeiten verringert und der Fehlerentdeckungsaufwand sowie der Arbeitsaufwand zur Projektabwicklung reduziert. Das neu entwickelte kombinierte Konzept einer gesamtheitlichen Prozess-Steuerung der Projektabwicklung, die in einzelnen Projektabschnitten eine Dokumenten-Steuerung zulässt, basiert auf Bonita Open Solution und Apache Tomcat und bindet die als Web-Anwendungen implementierten Benutzeroberflächen ein. Die dafür definierten Softwaretests sind optimal auf die unterschiedlichen Eigenschaften der hier verwendeten Werkzeuge abgestimmt. Die pilothafte Implementierung der kombinierten Projektabwicklung zeigt neue Möglichkeiten zu strukturierter und effizienter Projektabwicklung in kleinen und mittleren Unternehmen.

Abstract

The implemented user interface concept based on web application enables the realization of consistency checks as well as an easy error treatment in case of data inconsistency, and enables the stepwise integration of the document-controlled project handling into the actual environment of the company Innovations-Solutions AG (ISAG). In combination with the structured and expandable scheme of data base, the error risk is lowered, and the effort for error detection as well as the amount of work for the project handling is reduced. The newly developed combined concept of an holistic approach of process controlling in the project handling, that allows a document-controlled approach in certain project steps, bases on Bonita Open Solution and Apache Tomcat, and integrates the user interfaces that are implemented as web application. The special designed software tests are optimally adjusted to the different properties of the tools which are used here. The experimental implementation of the combined project handling shows new possibilities for more structured and more efficient project handling in small and medium companies.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

API	Application programming Interface
BE	Zeitarbeiterabrechnung
BOS	Bonita Open Solution
BPM	Business Process Management oder Business Process Modelling
BPMN	Business Process Model and Notation
BPMN 2.0	Business Process Model and Notation 2.0
CASE	Computer Aided Software Engineering
CATIA	Computer Aided Three-dimensional Interactive Application
DIN	Deutsches Institut für Normung
GUI	Grafische User Interface
GWT	Google Web Toolkit
HSSF	Horrible Spreadsheet Format
HTML	HyperText Markup Language
HWPF	Horrible Word Processor Format
ISAG	Innovations Solution AG
IT	Informationstechnologie
JDK	Java Development Kit
JPDL	jBPM Process Definition Language
JSP	Java Server Page
KISBPM	keep it simple BPM
LGPL	Lesser General Public License
MA	Wochenbericht
OMG	Object Management Group
PDF	Portable Document Format
PE	Process Engine
RG	Gesamte Rechnung
TB	Tagesbericht
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
VB	Visual Basic

VBA ..... Visual Basic for Application
WFMS ..... Workflow Management System
WSDL ..... Web Service Description Languages
XML ..... Extensible Markup Language
XPDL ..... XML Process Definition Language
XSSF ..... XML Spreadsheet Format
XWPF ..... XML Word Processor Format

1 Einleitung

In Industrieunternehmen ist die Abwicklung von Projekten heutzutage eine typische Aufgabe. In der DIN-Norm 69001 (Deutsches Institut für Normung) wurde der Begriff Projekt als ein Vorhaben definiert, das im Wesentlichen durch eine Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist. Quelle: Artikel 3.44, Seite 11, DIN 69001-5:2009-01. Prinzipiell kann jede komplexe Aufgabe mit zeitlich veränderlichen Zielen, Voraussetzungen oder Randbedingen als ein Projekt bezeichnet werden. Eine solche Projekt-Aufgabe muss einen definierten Anfang und ein definiertes Ende haben, das spätestens dann einsetzt, wenn nach der Ausführung dieser Aufgabe das angeforderte Ziel erreicht wird [Sch73, S.17]. Die Projektabwicklung bezeichnet die Abarbeitung eines Projektes. Ein typisches Beispiel ist die Bearbeitung eines Auftrages. Jeder Auftrag unterliegt einer bestimmten Zeitvorgabe, innerhalb derer das Ziel erreicht werden sollte. Die gesamtheitliche Bearbeitung eines Auftrages einschließlich Akquise und Rechnungsstellung eines Auftrages wird als Abwicklung eines Projektes gesehen.

1.1 Vorgehensweise für die Projektabwicklung

Es gibt zwei grundlegend unterschiedliche Vorgehensweisen für die Projektabwicklung: die Dokumenten-gesteuerte und die Prozess-gesteuerte Projektabwicklung. Die Dokumentengesteuerte Vorgehensweise ist dadurch gekennzeichnet, dass die eingegangenen und zur Bearbeitung anstehenden Dokumente festlegen, welcher Schritt in der Projektabwicklung als nächstes durchgeführt wird. Diese Vorgehensweise ist insbesondere bei Projektabläufen relativ einfacher Struktur geeignet, deren Eingangsdaten weder zeitlich gemeinsam noch sequenziell vorliegen sondern in zufälliger Reihenfolge eintreffen können und auch so abgearbeitet werden sollten. Aufgrund der Einfachheit starten viele neugegründete Unternehmen mit diesem Verfahren. Meist verwenden sie es auch dann weiter, wenn die Projektstrukturen komplexer werden und diese Methode somit weniger gut geeignet ist. Die Dokumenten-gesteuerte Vorgehensweise der Projektabwicklung kann in Papierform oder auch in elektronischer Form durchgeführt werden. So kann beispielsweise die Projektabwicklung mit einer Anzahl von Excel-Dateien durchgeführt werden. Solche Excel-basierten Vorgehensweisen haben in der Projektabwicklung viele Vorteile. Beispielsweise gewährleistet dies die notwendige Flexibilität, auf die Tabellenkalkulationen jederzeit und überall zugreifen zu können. Viele Berechnungsfunktionen können direkt in der Tabellenkalkulation erstellt werden. Durch diese Funktionen können also Daten einfach berechnet werden und außerdem ist die Bedienerfreundlichkeit sehr gut. Die erforderlichen Excel-Kenntnisse können in den

meisten Fällen bei den Mitarbeitern vorausgesetzt werden. Die Nachteile dieser Vorgehensweise sind Fehlerquellen durch Redundanzen in der Datenhaltung, Fehlereingaben oder auch durch versehentliches Löschen von Daten/Dateien oder Modifizieren von Formeln. Die für diese Excel-basierte Vorgehensweise typische, in der Regel recht komplexe Datei- und Dokumentenstruktur erfordert durch das sequenzielle Datei-Öffnen, Daten-Eintragen und Datei-Schließen einen hohen manuellen Arbeitsaufwand.

Die Prozess-gesteuerte Projektabwicklung ist dadurch charakterisiert, dass ein vorgegebener Prozessablaufplan festlegt, welcher Schritt in der Projektabwicklung als nächstes durchgeführt wird. Das Festlegen eines geeigneten Prozessablaufplans erfordert eine genaue Kenntnis der zu steuernden Projektabläufe. Das Anpassen eines solchen Prozessablaufplans an geänderte Projektanforderungen ist in den meisten Fällen aufwändig und schwierig, in manchen Fällen sogar unmöglich. In der Übergangszeit, in der die laufenden Projekte noch nach dem bisherigen Prozessablaufplan abgearbeitet werden, die Neuprojekte jedoch bereits nach dem neuen Ablaufplan, können Fehler zu Datenverlusten oder Projektverzögerungen führen. Deshalb zögern die meisten Unternehmen den Umstieg auf die Prozess-gesteuerte Projektabwicklung bzw. die Anpassung des Prozessablaufplans so lange wie möglich hinaus.

1.2 Motivation

Die vor wenigen Jahren gegründete Firma Innovations-Solutions AG (ISAG) arbeitet bisher mit einer Dokumenten-gesteuerten Projektabwicklung. Bisher wurde die korrekte Abfolge von Schritten in der Projektabwicklung hauptsächlich durch das Wissen und die Erfahrung der Mitarbeiter der ISAG sowie durch einige relativ allgemeine Projekt-Ablaufpläne gewährleistet. Diese Vorgehensweise ist jedoch fehleranfällig, da sie die strikte Einhaltung der Abarbeitungs-Reihenfolge nicht gewährleisten kann. Darüber hinaus erfolgt die Datenhaltung in teilweise redundanten Excel-Dokumenten, deren Formeln in ungeschützten Tabellenfeldern abgelegt sind. Es gibt in der ISAG derzeit keine Mechanismen, die die Konsistenz der neu einzugebenden Daten mit den bereits in den Excel-Dokumenten vorhandenen Daten prüft, wodurch die Gefahr von Fehlern sowohl durch einzugebende Projektdaten als auch durch direkte Eingabefehler relativ hoch ist. Darüber hinaus nehmen mit zunehmendem Auftragsvolumen der ISAG die täglich zu verarbeitenden Datenmengen linear zu und somit auch die obengenannten Fehlermöglichkeiten. Gleichzeitig steigt bei erhöhtem Datenvolumen auch der Aufwand zur Entdeckung jedes einzelnen Fehlers linear an. Somit steigt der Gesamtaufwand zur Fehlerentdeckung bei unveränderter Arbeitsweise mit dem Datenvolumen ungenau an. Eine Weiterentwicklung der Prozessabwicklung mit den Schwerpunkten:

- Verringerung der unterschiedlichen Fehlermöglichkeiten.
- Reduktion des Fehlerentdeckungsaufwandes ist also dringend erforderlich. Darüber hinaus ist eine Reduktion des Arbeitsaufwandes zur Projektabwicklung aus Effektivitätsgründen sehr wünschenswert.

Die Firma Innovations-Solutions AG (ISAG) hat sich deshalb entschieden, zu diesem Thema in Kooperation mit dem Institut für Rechnergestützte Ingenieursysteme (IRIS)[1] eine Diplomarbeit durchzuführen.

1.3 Ziel der Arbeit

Das Ziel dieser Diplomarbeit ist es, die unterschiedlichen Fehlermöglichkeiten zu verringern und den Fehlerentdeckungsaufwand sowie den Arbeitsaufwand zur Projektabwicklung zu reduzieren. Ein wesentlicher Stellhebel zur Reduktion dieser Fehlermöglichkeiten ist eine strukturierte und redundanzfreie Ablage der Projektdaten in einer Datenbank, mit der dann effektiv Konsistenzprüfungen der einzugebenden Projektdaten durchgeführt werden können. Die Verwendung von auf die Projektdaten abgestimmten Eingabe-Datenmasken reduziert insbesondere die Gefahr von Eingabefehlern. Die Gesamtheit dieser Maßnahmen hat bereits ein relativ großes Potential zur Effizienzsteigerung. Deshalb möchte die Firma ISAG von der Excel-basierten Vorgehensweise auf eine datenbankbasierte Vorgehensweise umsteigen. Die Abbildung 1.1 stellt den Umstieg der Vorgehensweise dar. Dafür sind eine redundanzfreie, Datenbank-geeignete und erweiterbare Projektdatenstruktur sowie dazu passende Benutzeroberflächen zu entwickeln. Weiterhin müssen Konnektoren zwischen den Benutzeroberflächen und der Datenbank erstellt werden.

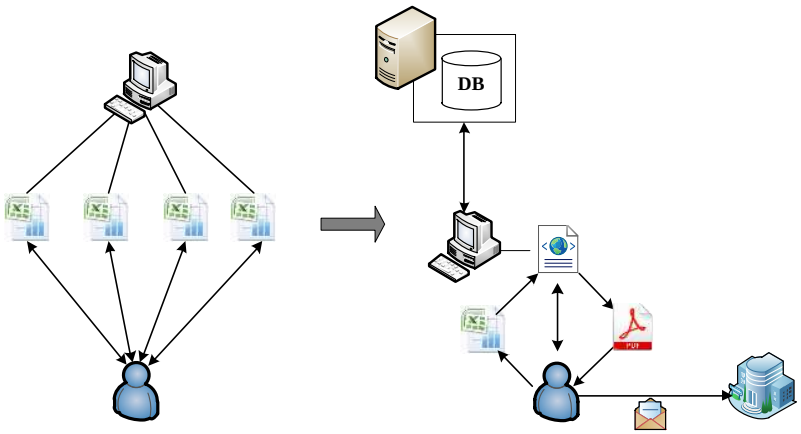

Abbildung 1.1: Umstieg von der Excel-basierten Vorgehensweise auf die datenbankbasierte Vorgehensweise

Jeden Tag werden in der ISAG die Daten zahlreicher Tagesberichte zur Weiterverarbeitung in Excel-Dateien eingetragen. Diese Daten beschreiben zum Beispiel die Arbeitszeiten der

[1] http://www.iris.uni-stuttgart.de/

Mitarbeiter und die Projekte, über die diese Arbeitszeiten abgerechnet werden. Zur Erstellung von Wochenberichten und Rechnungen werden dann genau diese Daten benötigt. Der Prozessabschnitt von der Eingabe der Tagesbericht-Daten bis zur Bereitstellung der Daten für die Wochenberichte und Rechnungen ist sehr arbeitsintensiv und fehleranfällig, weshalb dort ein großes Verbesserungspotential liegt. Eine Aufgabe dieser Diplomarbeit ist deshalb die Erstellung von Benutzeroberflächen zur Eingabe der Daten der Tagesberichte sowie deren Konsistenz-Prüfung. Weiterhin müssen die Daten in einer noch zu definierenden Datenstruktur einer Datenbank gespeichert und anschließend wieder von der Datenbank auf die bisher zur Rechnungserstellung verwendeten Excel-Dateien oder PDF-Dateien exportiert werden.

Eine andere Form der Effizienzsteigerung ist die Reduktion des Einlern-Aufwandes der Mitarbeiter, die Reduktion der Anforderung an die Mitarbeiter hinsichtlich des Projektablauf-Wissens sowie die Zuordnung einzelner Projektaufgabenbereiche zu einer Person oder einer Personengruppe. Diese Anforderungen werden üblicherweise durch eine Prozess-gesteuerte Projektabwicklung erfüllt und im Allgemeinen durch eine sogenannte „Process Engine“ realisiert. Die Beziehung zwischen der Datenbank, der Process Engine und den Benutzeroberflächen wird in der Abbildung 1.2 dargestell.

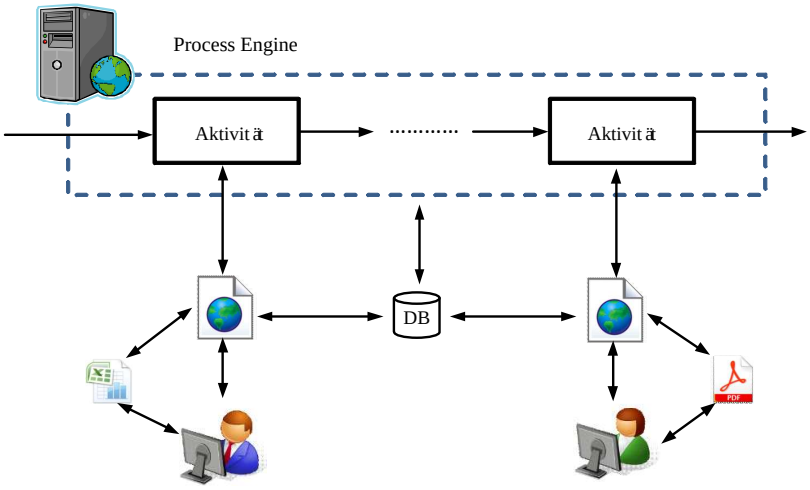

Abbildung 1.2: Beziehung zwischen den Benutzeroberflächen, der Process Engine und der Datenbank

Die auf dem Markt befindlichen Software-Produkte zur Prozess-Steuerung sind jedoch nur für stark Prozess-orientierte Projektabläufe geeignet und somit für die Abläufe in der ISAG nicht zielführend anwendbar, siehe Diplomarbeit von Baumann [Bau12]. Die zweite Aufgabe dieser Diplomarbeit ist deshalb die Untersuchung der Anwendungsmöglichkeiten einer „Process Engine“ als das Werkzeug für die Automatisierung der Projektabwicklung in der ISAG.

1.4 Aufbau der Arbeit

In dieser Diplomarbeit gibt es folgende Kapitel:

Kapitel 1 – Einleitung: Hier werden die Motivation (Kap. 1.2) und Ziele (Kap. 1.2) dieser Diplomarbeit dargelegt.

Kapitel 2 – Grundlagen: Hier werden benötigte Begriffe und Definitionen dieser Diplomarbeit erklärt.

Kapitel 3 – Anforderung und Designanalyse: Hier werden Anforderungen (Kap. 3.1) von der Firma ISAG und Ideen (Kap. 3.3, Kap.3.4, Kap.3.5) für die Entwicklung der Benutzeroberflächen vorgestellt.

Kapitel 4 – Implementierung: Funktionen der Benutzeroberflächen und welche Bedienungen gebraucht werden, sowie die Architektur des Programms, werden beschrieben.

Kapitel 5 – Process Engine: Hier werden zwei Process Engine Bonita Open Solution (Kap. 5.3) und Activiti (Kap. 5.4) untersucht; wie Benutzeroberflächen mit der Process Engine zusammen verbunden werden, wird beschrieben.

Kapitel 6 – Definition und Implementierung der Programmtests: Hier werden Funktionen der Benutzeroberflächen durch den Black-Box-Test (Kap. 6.2) und die JUnit (Kap. 6.3) getestet; Testergebnisse werden in der Tabelle dargestellt.

Kapitel 7 – Bewertung der erarbeiteten Ergebnisse: Schwierigkeiten (Kap. 7.4) werden für die Entwicklung der Benutzeroberflächen aufgelistet; nach dem Umstieg der Vorgehensweise wird der Aufwand (Kap. 7.3) evaluiert.

Kapitel 8 – Zusammenfassung und Ausblick: Zuerst wird zusammengefasst und dann ein Ausblick gegeben.

2 Grundlagen

Dieses Kapitel stellt wichtige Begriffe und Definitionen für diese Diplomarbeit dar, auch wird ein Überblick über die Grundlagen dieser Arbeit gegeben. Hier werden „Process Engine“, Prozessmodelle, „Business Process Model and Notation“ usw. vollständig vorgestellt.

2.1 Geschäftsprozess und Workflow

Ein Geschäftsprozess stellt eine Folge von funktionalen zusammenhängenden Aktivitäten dar, die nach bestimmten Anforderungen der Kunden oder auf ein Unternehmensziel hin ausgeführt werden. Aktivitäten liegen in einem logischen Zusammenhang vor und sind inhaltlich abgeschlossen [SZ05, S.51-53]. Ein Workflow beschreibt einen formal beschriebenen teilautomatisierten oder automatisierten Geschäftsprozess [Gad12, S.41].

2.2 Geschäftsprozessmanagement

Im Englischen wird Geschäftsprozessmanagement als Business Process Management (BPM) bezeichnet. Ziel des Geschäftsprozessmanagements ist Geschäftsprozesse der Unternehmen zu gestalten, zu steuern und zu überwachen [All05, S.12]. Die Informationstechnologie (IT) hat eine unteilbare Beziehung mit dem Geschäftsprozessmanagement, denn durch die IT wird nicht nur für einen effizienten Ablauf der Geschäftsprozesse gesorgt sondern auch für deren Entwicklung. IT-Lösungsanbieter und IT-Berater bieten Softwarewerkzeuge an, z. B. BPM-Tools [SS08, S.29-30]. Damit werden Geschäftsprozesse durch Softwarewerkzeuge analysiert, modelliert, gesteuert, automatisiert usw.

2.3 Werkzeuge für das Geschäftsprozessmanagement

Es gibt unterschiedliche Softwarewerkzeuge für das Geschäftsprozessmanagement, wobei die Funktionen große Unterschiede aufweisen. In der Abbildung 2.1 werden Einsatzmöglichkeiten der Softwarewerkzeuge für das Geschäftsprozessmanagement vorgestellt. Verschiedene auf dem Markt befindliche Softwarewerkzeuge können bezüg-

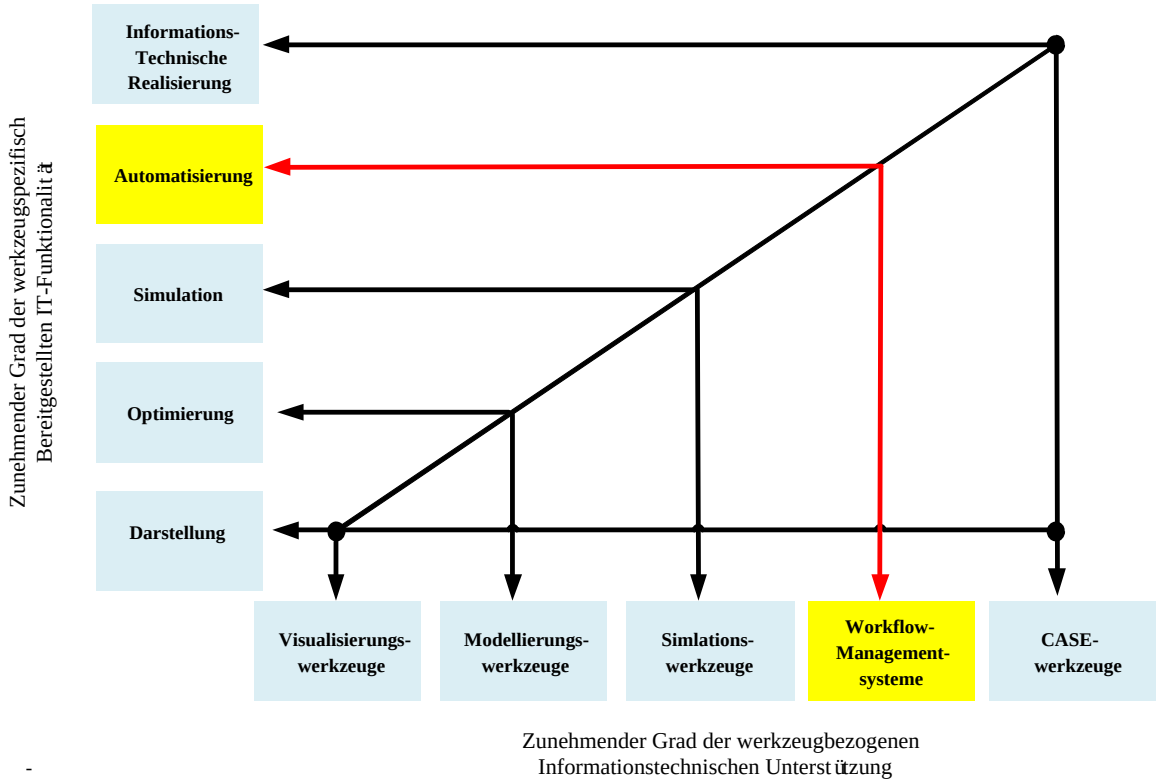

Abbildung 2.1: Prozessmanagement-Werkzeuge [NS02, S.201-210]

lich ihrer Funktionen in Visualisierungswerkzeuge, Modellierungswerkzeuge, Simulationswerkzeuge, Workflow-Management-Systeme und Computer Aided Software Engineering (CASE)-Werkzeuge unterteilt werden. Alle Werkzeuge unterstützen die Darstellung von Geschäftsprozessen. Modellierungswerkzeuge und Simulationswerkzeuge unterstützen die Optimierung und die Simulation von Geschäftsprozessen. Die Automatisierung von Geschäftsprozessen wird durch Workflow-Management-Systeme unterstützt, die Entwicklung und der Test von Informationssystemen werden durch CASE-Werkzeuge realisiert [Gad12, S.104]. Um Geschäftsprozesse besser entwickeln und analysieren zu können, integriert Workflow-Management-Systeme viele Funktionen, d. h. Workflow-Management-Systeme unterstützt nicht nur die Automatisierung der Geschäftsprozesse, sondern auch die Modellierung, die Simulation und die Ausführung. Die Automatisierung der Projektabwicklung basiert dabei auf Workflow-Management-Systemen.

2.4 Workflow Management System

Ein Workflow Management System (WFMS) ist ein Softwaresystem zum Steuern des Arbeitsflusses. Das Workflow-Management-System besteht aus folgenden Komponenten:

- *Buildtime*-Komponente: um Geschäftsprozesse zu modellieren und zu definiteren.
- *Runtime*-Komponente: um Geschäftsprozesse auszuführen, zu kontrollieren oder zu überwachen [LR00, S.62-63].

In dem Workflow-Management-System ist die zentrale Softwarekomponente die Workflow Engine bzw. Process Engine. Es gibt keine Unterschiede in dem Konzept zwischen der Process Engine und der Workflow Engine. Unterschiede kann es nur von Produkt zu Produkt geben, z. B. kann ein IBM WebSphere Process Server[1] anderes benannt werden als die Activiti Workflow Engine. Die Automatisierung der Projektabwicklung verwendet die Process Engine.

2.5 Process Engine

Die Business Process Engine oder kurz die Process Engine führt Geschäftsprozesse aus. Die Voraussetzung für die Ausführung ist, dass Geschäftsprozesse in Prozessmodellen, die alle technischen Details beinhalten, definiert werden. Die Process Engine kann nicht nur den Datenfluss und den Kontrollfluss steuern, sondern auch viele weitere Funktionen anbieten. Z. B. können in der Process Engine verschiedene Versionen eines Prozessmodells gleichzeitig verarbeitet werden, Daten von Prozessinstanzen gesammelt oder der aktuelle Status von Prozessinstanzen erhalten werden, damit eine fehlerhafte Prozessinstanz abgebrochen wird. Verschiedene Versionen eines Prozessmodells bezeichnen, dass ein Prozessmodell von einigen Tagen bis zu mehreren Monate entwickelt werden kann, zu jedem Zeitpunkt kann das Prozessmodell verändert werden; nach der Veränderung auch wird die Version dieses Prozessmodells auch geändert. Wenn ein Prozessmodell verarbeitet wird, kann die alte Version dieses Prozessmodells gleichzeitig verarbeitet werden [FR10, S.106]. Danach werden Geschäftsprozesse auf der Process Engine installiert bzw. deployed. Process Engine führt Prozessmodellen aus, die in der Notation Business Process Model and Notation 2.0 (BPMN 2.0) formuliert werden. D. h. Process Engine kann die BPMN 2.0 ausführen. Mit der BPMN 2.0 wurde eine explizite Ausführungssemantik definiert, die durch Extensible Markup Language (XML) beschrieben werden. Alle technischen Details zur Ausführung eines Prozesses werden in der XML gespeichert, wie z. B. die Aufgabe der Aktivität, das Attribute von der Aktivität usw. Die Ausführungssemantik wird durch die Process Engine ausgeführt. Eigentlich ist die wesentliche Funktion einer Process Engine die Koordination der Abläufe, die in den installierten Prozessmodellen definiert sind [LS].

[1] http://www-01.ibm.com/software/integration/wps/

2.6 Prozessmodell

Als Prozessmodell wird die realistische Abbildung von Geschäftsprozessen aufgefasst. Prozessmodelle beschreiben zeitlich-logische Abfolgen von Aktivitäten. Geschäftsprozesse werden durch Modelle analysiert, entworfen oder dokumentiert. Außerdem unterstützen die Kommunikation über Geschäftsprozesse [Fot10, S.65]. Prozessmodelle können im BPMN erstellt werden.

2.7 Business Process Model and Notation

Business Process Model and Notation (BPMN) ist eine einheitliche Sprache zur Beschreibung von Geschäftsprozessen. Durch BPMN können Geschäftsprozesse mit gerafischen Formen modelliert werden. Im Gegensatz zu anderen Beschreibungssprachen werden Prozessstrukturen in BPMN deutlich dargestellt und außerdem wird die Semantik der Symbole klar definiert. BPMN benötigt eine Notation. Eine Notation bestimmt, *mit welchen Symbolen die verschiedenen Elemente von Prozessen dargestellt werden, was sie bedeuten und wie sie miteinander kombiniert werden können* [Allo9, S.8]. Die erste Version der BPMN wurde bei Stephen A.White von IBM entwickelt. Im Jahr 2006 wurde die BPMN 1.0 durch Object Management Group (OMG)[2] standardisiert, BPMN 2.0 wurde von OMG im Jahr 2011 veröffentlich [OMG11]. Verwendung von BPMN braucht man keine Lizenz, mache BPMN-Tools sind gar kostenlos [Silo9, S.3]. In dieser Arbeit wurden Prozessmodelle durch BPMN 2.0 erstellt.

[2]http://www.omg.org/

2.8 Datenbankbasierte Webanwendungen

Eine Webanwendung ist ein Softwareprogramm, das auf einem Webserver ausgeführt wird. Webanwendungen müssen auf einem Server aufgestellt werden [Bau08, S.4-6]. Eine datenbankbasierte Webanwendung ist dadurch gekennzeichnet, dass Daten durch den Server in die Datenbank gespeichert oder von der Datenbank gelesen werden. Die Abbildung 2.2 zeigt die Komunikation zwischen der Datenbank, dem Server und dem Web-Client.

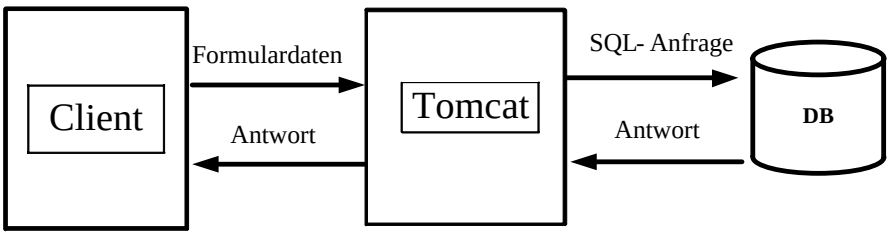

Abbildung 2.2: Datenbankbasierte Webanwendungen

Ein Service bzw. Tomcat[3] bekommt Formulardaten von dem Web-Client. Mit Hilfe der SQL-Sprache wird die Datenbank aufgerufen und danach eine Antwort zurück den Web-Client übermittelt. Die Firma ISAG möchte datenbankbasierte Webanwendungen bzw. eine datenbankbasierte Arbeitsweise für die Bearbeitung des Tagesberichtes.

[3]http://tomcat.apache.org/

3 Anforderung und Designanalyse

In diesem Kapitel wird zunächst auf die bisherige Projektabwicklung in der Firma ISAG eingegangen. Darauf aufbauend werden Anforderungen an eine zukünftige Projektabwicklung formuliert. Es werden unterschiedliche Benutzeroberflächen-Konzepte entworfen und im Hinblick auf die vorab formulierten Anforderungen analysiert. Durch den Vergleich der jeweiligen Vor- und Nachteile wird das geeignetste Benutzeroberflächen-Konzept identifiziert.

3.1 Bisherige Projektabwicklung bei der Firma ISAG

Die Firma ISAG realisiert die Projektabwicklung bisher mit einer Excel-basierten Arbeitsweise. Die Außendienst-Mitarbeiter erstellen jeden Tag Tagesberichte, die beschreiben, welche Mitarbeiter an welchem Datum und Einsatzort mit welchem Ansprechpartner unter welcher Auftragsnummer gearbeitet haben. Weiterhin wird die Arbeitszeit jedes Mitarbeiters in dem jeweiligen Tagesbericht dokumentiert. Diese Tagesberichte in Papierform werden dann in mehrere Excel-Dateien übertragen und so der weiteren Excel-basierten Projektabwicklung zugänglich gemacht.

Die Daten des Tagesberichts werden für die Erstellung von Rechnungen, Wochenberichten und Zeitarbeiterabrechnungen benötigt. Da es derzeit noch keine zentrale Datenbank gibt, müssen alle erforderlichen Daten in die für den jeweiligen Zweck erstellten Excel-Dateien eingetragen werden. Dadurch entstehen Datenredundanzen und im Fehlerfall Inkonsistenzen der Daten. Werden diese Fehler nicht vorab bemerkt, bekommen die Kunden fehlerhafte Rechnungen oder Zahlungen, was die Geschäftsbeziehungen belasten kann. Darüber hinaus binden die bei Reklamationen notwendigen Aktivitäten zur Fehlersuche und -behebung zunehmend wertvolle personelle Ressourcen.

3.2 Anforderungen an zukünftige Projektabwicklung

Die zukünftige Prozessabwicklung sollte also die Schwerpunkte a) Verringerung der Fehlermöglichkeiten und b) Reduktion des Fehlerentdeckungsaufwandes haben. Darüber hinaus ist eine Reduktion des Arbeitsaufwandes zur Projektabwicklung aus Effektivitätsgründen sehr wünschenswert, siehe Kapitel 1.

Um die Gefahr von Dateninkonsistenzen zu verringern, ist die Erstellung einer redundanzfreien Datenbank eine bekannte und bewährte Vorgehensweise. Die Kommunikation des

Benutzers mit der Datenbank erfolgt dann üblicherweise über eine oder mehrere Benutzeroberflächen.

Anforderungen an die zu erstellenden Benutzeroberflächen sind:

1. Reduktion der Eingabe-Fehlermöglichkeiten.

   Die ersten Fehler können beim manuellen Ausfüllen der Tagesberichte durch die Außendienst-Mitarbeiter oder auch beim Übertragen der Daten vom eingereichten Tagesbericht entstehen, z. B. bei der Auftragsnummer, dem Einsatzort, dem Ansprechpartner oder auch beim Namen des Außendienst-Mitarbeiters. Diese Daten gehören zu den allgemeinen Projektdaten und sind damit einerseits bereits vorhanden und andererseits unabhängig von dem einzelnen Tagesbericht. Die Eingabe dieser Daten über Auswahlmenüs vermeidet die Falscheingabe dieser Daten bei der Eingabe in die Datenbank. Eine bisher ebenfalls mögliche Fehlerquelle war, dass einzelne Excel-Felder mit Formeln manuell mit Werten überschrieben werden, wodurch die Gesamtfunktion des betreffenden Excel-Blatts eingeschränkt oder zerstört wurde. Durch die Verwendung einer Eingabemaske kann diese Fehlerquelle ausgeschlossen werden.

2. Prüfung der Konsistenz der eingereichten Tagesberichte.

   Redundanzen können entstehen, wenn der Tagesbericht versehentlich nochmals in die Datenbank eingegeben wird, obwohl er bereits in der Datenbank vorhanden ist. Fehlermöglichkeiten sind weiterhin, dass im Auswahlmenü ein falsches Datum oder eine falsche Auftragsnummer selektiert wird. Weiterhin können Arbeitszeiten mehrfach oder falsch ausgefüllt sein. Dabei können die Fehler bereits im eingereichten Tagesbericht enthalten sein oder auch erst bei der Eingabe passieren. Eine wirksame Möglichkeit, diese Fehler zu bemerken, ist, die in die aktuelle Benutzeroberfläche eingetragenen Daten vor dem Übernehmen in die Datenbank mit den bereits in der Datenbank befindlichen Daten zu vergleichen und auf Plausibilität hin zu prüfen. Wird eine Unplausibilität bemerkt, können die Mitarbeiter den Fehler korrigieren. So können keine inkonsistenten Daten in die Datenbank importiert werden.

3. Reduktion des manuellen Aufwandes.

   Mehrfach benötigte Daten sollten nicht mehrfach in die jeweiligen Excel-Dateien eingegeben werden müssen. Der damit verbundene Aufwand zur Bearbeitung der Excel-Datei, z. B. Öffnen, Speichern der Schließen der Excel-Datei, ist zu reduzieren. Durch einen oder wenige Buttons der Oberfläche sollten die Daten automatisch nach der Überprüfung in die Datenbank importiert oder in andere Datenformate exportiert werden können. Darüber hinaus reduziert die oben beschriebene Eingabekontrolle effektiv den Aufwand für das andernfalls erforderliche Gegenprüfen der eingegebenen Daten bzw. den Aufwand für das Suchen von Fehlern im Falle von Reklamationen.

Randbedingungen für die zukünftige Projektabwicklung sind:

1. Reibungsloses Einfügen in die bestehenden Vorgehensweisen.

   Bisherige Formulare, sowohl für die Dateneingabe (Tagesberichte) als auch für die Rechnungsstellung und Berichtserstattung, sollen weitgehend erhalten bleiben. Das

Design der Eingabemaske sollte also dem Design des Tagesbericht-Formulars entsprechen, um den Einarbeitungsaufwand zu reduzieren und einen optischen Vergleich der eingegebenen Daten mit dem eingereichten Tagesbericht zu ermöglichen. Weiterhin sollen die in der Datenbank vorhandenen Daten in bereits bestehende und aktuell verwendete Excel-Listen exportiert werden können, damit ein gleitender Übergang von manuellem auf automatisches Ausfüllen dieser Excel-Listen ermöglicht wird. Erst wenn alle aktuellen Projekte in der Datenbank vollständig enthalten sind, können die weiteren manuellen Auswertungen sowie die Rechnungsstellung und Berichterstattung direkt über die Datenbank abgewickelt werden. Bis dahin muss sich die zukünftige Projektabwicklung reibungslos in die bisherige Arbeitsweise einfügen.

2. Manuelle Korrigierbarkeit in der Einführungsphase.

   Während der Einführungsphase müssen entdeckte Fehler in den Programmen zur Projektabwicklung oder in den eingegebenen Daten vollständig manuell korrigierbar sein. Zum Einen muss die Projektabwicklung so gestaltet sein, dass Korrekturen an den Schnittstellen möglichst einfach sind. Zum Anderen muss ein Durchgriff auf die Datenbank vorhanden sein, so dass Korrekturen des Datensatzes, für den es gegebenenfalls noch keine Benutzeroberfläche gibt, ermöglicht werden.

3. Optimale Wartbarkeit und Erweiterbarkeit.

   Es muss gewährleistet sein, dass neue oder korrigierte Benutzeroberflächen sofort und flächendeckend allen Benutzern zur Verfügung stehen. Alte Benutzeroberflächen dürfen ab Aktivierung der neuen Oberflächen nicht mehr aufrufbar sein. Diese Anforderungen müssen so realisiert sein, dass der Administrator nur minimalen Aufwand für diese Wartung bzw. Erweiterung hat, die im besten Falle sogar unabhängig von der Anzahl der Benutzer bzw. der verwendeten Rechner ist.

Die Abbildung 3.1 zeigt die von der Firma ISAG erwartete Datenbank-basierte Arbeitsweise für die Auswertung der Daten des Tagesberichtes und deren weiterer Verwendung.

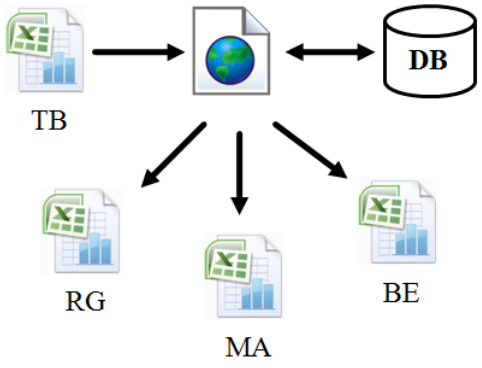

Abbildung 3.1: Ablaufdiagramm für die Bearbeitung der Tagesberichte

Die Daten aus dem Tagesbericht (kurz: TB) werden als Eingaben in die Datenbank importiert. Programme überprüfen Daten der Tagesberichte und importieren entweder Daten in die Datenbank oder exportieren Daten aus Excel-Dateien. Ausgabe-Excel-Dateien sind die gesamte Rechnung (kurz: RG), die Wochenberichte (kurz: MA) und die Zeitarbeiterabrechnungen (kurz: BE). Die Firma möchte also im ersten Schritt Benutzeroberflächen für die Kommunikation mit der Datenbank haben, wobei als Ergebnis die obengenannten Ausgabe-Excel-Dateien mit aktuellen Daten befüllt werden.

3.3 Analyse unterschiedlicher Benutzeroberflächen-Konzepte

Ausgehend von den im vorigen Kapitel genannten Anforderungen an die zu erstellenden Benutzungsoberflächen sind mehrere Realisierungen möglich, die sich unterschiedlich weit von den bisherigen Excel-Datei-basierten Vorgehensweisen entfernen.

3.3.1 Benutzeroberfläche auf Basis von Microsoft Excel

Die Firma ISAG hatte sich einst für die Projektabwicklung auf Basis von Excel-Dateien entschieden, weil Microsoft Excel ein leistungsstarkes, flexibles und weit verbreitetes Tabellenkalkulationsprogramm ist. In Excel können Daten einfach analysiert und präsentiert werden und außerdem haben die meisten Mitarbeiter schon Excel Kenntnisse. Es ist also naheliegend, die bisherigen Excel-Programme um die notwendigen Funktionalitäten mittels eingebetteten Programmcodes zu erweitern.

3.3.1.1 Grundidee

Die Grundidee ist also, die Benutzeroberflächen-Funktionen direkt in den Excel-Dateien zu programmieren. Eine häufig verwendete Excel-Programmiersprache ist VBA (Visual Basic for Application). VBA ist eine spezielle Programmiersprache für Anwendungen wie z. B. Microsoft Office und Computer Aided Three-dimensional Interactive Application[1] (CATIA). Diese Programmiersprache ist eine Light-Version von Visual-Basic (VB), die als Makro-Sprache für Microsoft Office und andere die als Makro-Sprache für Microsoft Office und andere Anwendungen eingesetzt wird [VBA]. Die Makros werden in VBA geschrieben und in einer Excel-Datei abgelegt. In der Excel-Datei Tagesbericht (TB) können alle erforderlichen Makros definiert werden, durch die Daten der Excel-Tabelle in die Datenbank importiert sowie Fehlereingaben überprüft werden.

[1] http://www.3ds.com/products/catia/welcome/

Listing 3.1 Erstellung eines Untermenüs von VBA

```
1 Sub Macro1()
2 Columns("A:A").Select
3 With Selection.Validation.Delete.Add Type:=xlValidateList,
4 AlertStyle:=xlValidAlertStop,
5 Operator:= _xlBetween, Formular1:="AAA,BBB,CCC,DDD..."
6 End With
7 End Sub
```

3.3.1.2 Vorteile und Nachteile

Es gibt folgende Vorteile für diese Realisierungsart:

- Benutzerfreundlichkeit: Die Datenbearbeitung basiert auf der Excel-Tabelle, Mitarbeiter benötigen keine zusätzlichen Programmkenntnisse, sondern nur die Grundkenntnisse von Microsoft Excel.
- Flexibilität: Microsoft Excel bietet viele Berechnungsfunktionen, durch diese Funktionen Daten einfach berechnen können.
- Erweiterbarkeit: VBA ist für modulare Programmierung entwickelt, d. h. das Problem wird in einzelne Module aufgeteilt, die jeweils eine genaue Schnittstellenspezifikation aufweisen [HV07, S.45]. Der Entwickler kann einfach ein Makro für die neue Funktion definieren. Die Auflistung 3.1 zeigt uns beispielhaft, wie ein Untermenü in den Excel-Feldern durch VBA definiert werden kann.

  Angenommen, es wird der Mitarbeitername in der Spalte „A“ eingefügt, dann bezeichnet das Formular 1 alle Mitarbeiternamen, z. B. „AAA“, „BBB“, „CCC“. Daraufhin wird für jedes Feld dieser Spalte ein Untermenü erstellt und die Mitarbeiternamen, die in dem Untermenü gespeichert sind, können ausgewählt werden. Es gibt noch weitere Methoden, um Untermenüs zu erstellen. Zuerst wird eine Spalte ausgewählt, um das Untermenü zu definieren. In Microsoft Excel gibt es ein Daten-Menü. Wenn die Gültigkeit (Excel-2003) oder die Datenüberprüfung (Excel-2007 und Excel 2010) in diesem Menü ausgewählt wird, wird ein Fenster geöffnet. In den Einstellungen dieses Fensters gibt es ein „Zulassen“; falls die „Liste“ ausgewählt wird, wird eine Quelle angenommen. In der Quelle wird der Mitarbeitername mit einem Semikolon eingefügt. Wenn im „OK“ geklickt wird, wird das Untermenü in der ausgewählten Spalte erstellt.
- Testbarkeit: Makroprogramme werden einfach getestet; bei Fehlern werden Programme abgebrochen und im Editor dargestellt.

Nachteile dieser Realisierungsart sind:

- Die mehrfache Implementierung: In Microsoft Excel muss VBA in einem Makro verwendet werden, weil Microsoft Excel nur ein Makro ausführen kann. Makros dienen VBA-Programmen bei den meisten Microsoft-Anwendungsprogrammen. In jede Excel-Datei müssen viele Makros konsistent implementiert werden, damit wird beispielsweise die Datenüberprüfung fehlerfrei funktioniert.

- Die komplizierte Speicherung: Wenn Excel-Dateien und Makros zusammengespeichert werden, müssen die verwendeten Datentypen konsistent gehalten werden. In Excel-2003 kann der jeweilige Datentyp einfach als Microsoft Excel 97-2003-Arbeitsmappe ausgewählt werden. Im Gegensatz zu Microsoft Excel 2003 wird der Datentype in Microsoft Excel 2007 oder Microsoft Excel 2010 als Excel-Arbeitsmappe mit Makros ausgewählt. Das sind für Benutzer, die lediglich Excel-Grundkenntnisse haben, relativ große Schwierigkeiten.
- Die nicht-intuitive Bedienung: Wenn Microsoft Excel 2003 mit den Makros geöffnet wird, müssen Makros aktiviert werden. Ohne die manuelle Aktivierung der Makros wird das Sicherheitswarnungs-Makro ausgeführt.
- Die aufwändige Aktualisierung: Wenn die alte Excel-Vorlage nicht mehr verwendet werden soll und eine neue Excel-Vorlage erstellt wird, müssen alle Makros neu definiert werden; die alten Makros können nämlich nicht in der neuen Excel-Vorlage angewendet werden.
- Außerdem werden Daten, die in der komplizierten Excel-Tabelle gespeichert werden, aufwendig in die Datenbank importiert oder von der Datenbank exportiert werden.

Nach diesen grundsätzlichen Überlegungen liegen die Vor- und Nachteile dieser Umsetzungsvariante bereits ausreichend detailliert vor, so dass eine testweise Realisierung für den beabsichtigten Variantenvergleich keine weiteren habhaften Erkenntnisse bringt.

3.3.2 Benutzeroberfläche auf Basis von Microsoft Excel und Java-Programmen

Die zweite Idee für das Design der Benutzeroberfläche ist, ein Microsoft Excel-Formular lediglich als Eingabemaske zu verwenden, die Daten-Manipulation jedoch mit Java-Programmen durchzuführen. Das heißt, die Daten werden zunächst manuell in die Excel-Datei eingetragen. Im Anschluss daran werden die Daten eines Tagesberichtes durch Java-Programme kontrolliert und in die Datenbank übernommen. Die Hauptaufgabe der Programme sind dabei die Eingabeüberprüfung sowie Datenimport und -export.

Dieses Vorgehen lehnt sich stark an die bisherige Arbeitsweise an und zeigt in der ersten Betrachtung keine gravierenden Nachteile. Deshalb wird eine Test-Realisierung durchgeführt, die weitere Erkenntnisse zu den Eigenschaften dieser Realisierungsart bringen soll.

Folgende Funktionen müssen zur Eingabeüberprüfung realisiert werden:

1. Überprüfen, ob die Daten der Excel-Tabelle vollständig eingetragen sind, also ob z. B. das Datum, die Auftragsnummer, der Mitarbeitername usw. ausgefüllt sind.
2. Überprüfen, ob die eingetragenen Daten bereits in der Datenbank existieren.
3. Überprüfen, ob es Konsistenzfehler gibt, wenn Daten eines Tagesberichtes in die Datenbank importiert oder aktualisiert werden, sowie Ermöglichen von entsprechenden Korrekturen. Dazu wird eine Status-Information mit diesen Daten in der Datenbanktabelle abgebildet. Falls z. B. Daten eines Tagesberichtes in die Datenbank neu geschrieben

werden, ist der Status der Daten „neu“ und wird in der Datenbanktabelle als „neu“ bezeichnet. Falls Daten dieses Tagesberichtes beim Konsistenztest in Ordnung sind und so in die Datenbank übernommen werden, wechselt der Status der Daten auf genehmigt (bzw. geprüft), was in der Datenbanktabelle als „gen“ bezeichnet wird. Sollte sich dennoch ein Fehler in diesen Daten herausstellen, dann können diese Daten korrigiert und einfach erneut geprüft und in die Datenbank geschrieben werden, wobei der Status auf „genehmigt“ bleibt. Wenn die Daten dieses Tagesberichtes in die entsprechenden Ausgabedateien exportiert werden, wechselt der Status der Daten zu „exportiert“, was in der Datenbanktabelle als „exp“ bezeichnet wird. Falls bereits exportierte Daten dieses Tagesberichtes nachträglich korrigiert werden, wechselt der Status der Daten auf „korrigiert“, was in der Datenbanktabelle als „kor“ bezeichnet wird. Bei einem nachfolgenden Export-Vorgang werden die korrigierten Daten den Ausgabedateien dann nicht hinzugefügt, sondern die bereits in die Ausgabedateien geschriebenen Daten korrigiert.

Daten Import/Export:

1. Daten der Excel-Datei müssen in die Datenbank geschrieben oder Daten in der Datenbank aktualisiert werden können.
2. Daten müssen von der Datenbank in eine Excel-Datei oder PDF-Datei exportiert werden können.

Es zeigt sich bereits an dieser Stelle, dass das Abfangen von Eingabefehlern sowie die Umsetzung der entsprechenden Korrekturmöglichkeiten einen weitaus größeren Aufwand darstellen als der eigentliche Datenimport und -export. Diese Funktionen für diese Excelbasierte Benutzeroberfläche können mit den folgenden vier Konvertern realisiert werden, wobei darauf geachtet wird, dass auch alle möglichen Bedienungsfehler abgefangen werden.

3.3.2.1 Eintragen von Daten eines neuen Tagesberichtes in ein Formular

Die Hauptaufgabe für den Konverter 1 ist die Erstellung eines Tagesberichtes (kurz „TB_neu“) mit Hilfe eines Tagesbericht-Formulars. In der Abbildung 3.2 wird das Flussdiagramm für den Konverter 1 dargestellt. Es gibt eine Excel-Vorlage für den Tagesbericht (kurz „*TB_Vorlage.xls*“). Mitarbeiter benutzen das Tagesbericht-Formular (kurz „*TB_neu.xls*“) und geben die Daten in *TB_neu.xls* ein. *TB_neu.xls* muss natürlich in einem bestimmten Ordner bereitgestellt werden. Zuerst überprüft das Programm, ob dieser Ordner existiert. Falls dieser Ordner nicht existiert, wird eine Fehlermeldung gegeben und Konverter 1 abgebrochen. Falls dieser Ordner existiert, überprüft das Programm, ob *TB_Vorlage.xls* existiert und schreibgeschützt ist. Wenn *TB_Vorlage.xls* nicht existiert oder nicht schreibgeschützt ist, wird eine Fehlermeldung gegeben und Konverter 1 abgebrochen. Wenn *TB_Vorlage.xls* existiert und schreibgeschützt ist, überprüft das Programm, ob *TB_neu.xls* existiert und falls es nicht existiert, wird *TB_neu.xls* erstellt. Danach werden zwei Zweige in dem Diagramm zusammengesetzt. Das Programm überprüft, ob *TB_neu.xls* schon geöffnet wird. Wenn *TB_neu.xls* geöffnet wird, wird eine Fehlermeldung gegeben und Konverter 1 abgebrochen, anderenfalls

öffnet *TB_neu.xls* mit dem Programm Excel. In diesem Schritt wird der Konverter 1 beendet und der betreffende Mitarbeiter befüllt die Daten in der Datei *TB_neu.xls*. Anschließend speichert und schließt er die *TB_neu.xls*.

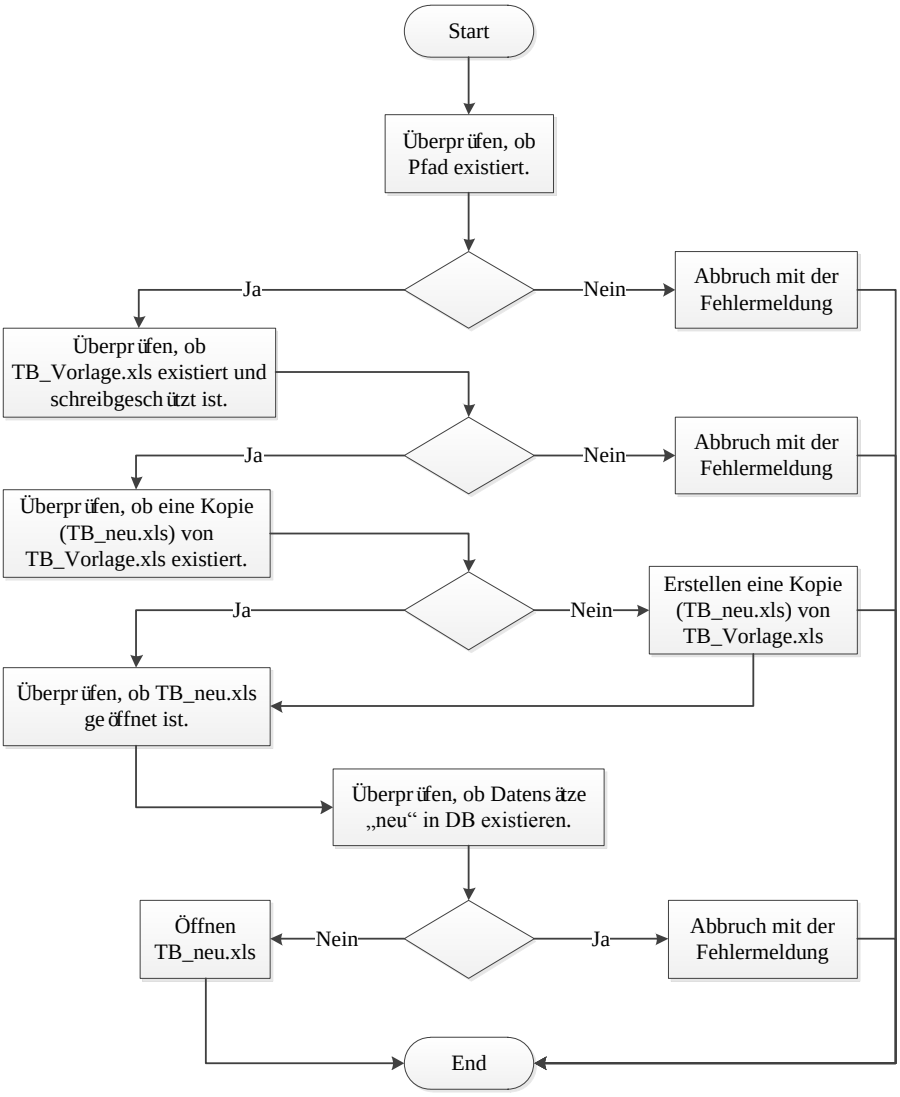

Abbildung 3.2: Flussdiagramm für die Erstellung eines Tagesberichtes (TB_neu)

3.3.2.2 Daten eines neuen Tagesberichts in die Datenbank importieren

Aufgaben des Konverters 2 (kurz „TB_Imp“) sind, die Datenkonsistenz mit der Datenbank zu überprüfen, Fehlereingaben von der Excel-Datei zu überprüfen sowie die Daten der Excel-Datei in die Datenbank zu importieren oder in der Datenbank zu aktualisieren.

Der Anfangsschritt ist ähnlich wie beim Konverter 1. Nachdem das Programm überprüft hat, ob *TB_neu.xls* schon geöffnet wird, wird eine Abfrage gestellt, ob *TB_neu.xls* die wesentlichen Daten enthält. Falls *TB_neu.xls* keine wesentlichen Daten enthält, wird angenommen, dass die Eingabe fehlerhaft war und Konverter 2 beendet. Andernfalls überprüft das Programm, ob *TB_neu.xls* schon komplett ausgefüllt ist. Wenn *TB_neu.xls* nicht komplett ausgefüllt ist, wird Konverter 1 gestartet. Andernfalls überprüft das Programm, ob *TB_neu.xls* erneut geöffnet wurde. Wenn *TB_neu.xls* aktuell nicht geöffnet ist, überprüft das Programm, ob die Daten dieser Excel-Datei schon in der Datenbank existieren. Falls sie schon in der Datenbank gespeichert wurden, liegt eine Inkonsistenz vor. In diesem Fall werden die gespeicherten Daten von der Datenbank aus *TB_db.xls* exportiert. *TB_db.xls* ist eine Kopie von *TB_Vorlage.xls*.

Bevor Daten aus *TB_db.xls* exportiert werden, überprüft das Programm, ob *TB_db.xls* existiert. Wenn *TB_db.xls* existiert, wird diese Datei gelöscht und eine neue *TB_db.xls* durch das Programm erstellt. Dann werden zur besseren Benutzerfreundlichkeit die Daten, die zur Inkonsistenz geführt haben, in *TB_db.xls* mit roter Farbe gekennzeichnet. Mitarbeiter können nun diese zwei Excel-Dateien vergleichen und herausfinden, welche Datei Eingabefehler enthält und entsprechend die Daten korrigieren. Über ein Abfrage-Fenster wird abgefragt, ob die Daten aus dem neuen Tagesbericht direkt in die Datenbank überschrieben werden sollen. Wenn der Mitarbeiter „ja“ auswählt, liegt der Fall von inkorrekten Daten in der Datenbank vor und es werden die Daten aus dem neuen Tagesbericht direkt in der Datenbank aktualisiert. Falls Daten von *TB_neu.xls* nicht in der Datenbank existieren, werden sie direkt in die Datenbank importiert. Schließlich wird *TB_neu.xls* gelöscht. Wenn der Mitarbeiter „nein“ auswählt, liegen Fehler in dem neuen Tagesbericht vor und es müssen die Daten von *TB_neu.xls* korrigiert werden. Dazu wird der Konverter 2 beendet und der Konverter 1 gestartet. Die Abbildung 3.3 zeigt das Flussdiagramm für den Konverter 2.

3.3.2.3 In der Datenbank enthaltene neue Tagesberichte durch Mitarbeiter überprüfen und genehmigen

Aufgaben des Konverters 3 (kurz „TB_Prf“) sind die Überprüfung, ob neue Tagesberichtsdaten in der Datenbank existieren, deren Korrektheit durch einen weiteren Mitarbeiter festgestellt und ob deren Status dann auf „genehmigt“ gesetzt werden muss. Der Konverter 3 ist ähnlich wie der Konverter 1. Das Programm überprüft, ob *TB_gen.xls* existiert. Falls diese Datei existiert, wird *TB_gen.xls* durch das Programm gelöscht, dann kopiert das Programm *TB_Vorlage.xls* zu *TB_gen.xls*. Falls diese Datei nicht existiert, wird *TB_gen.xls* sofort erstellt. Danach werden Datensätze überprüft. Wenn es keine Datensätze mit der Type „neu“ gibt, wird eine Fehlermeldung für die Mitarbeiter dargestellt und der Konverter 3 beendet. Sonst

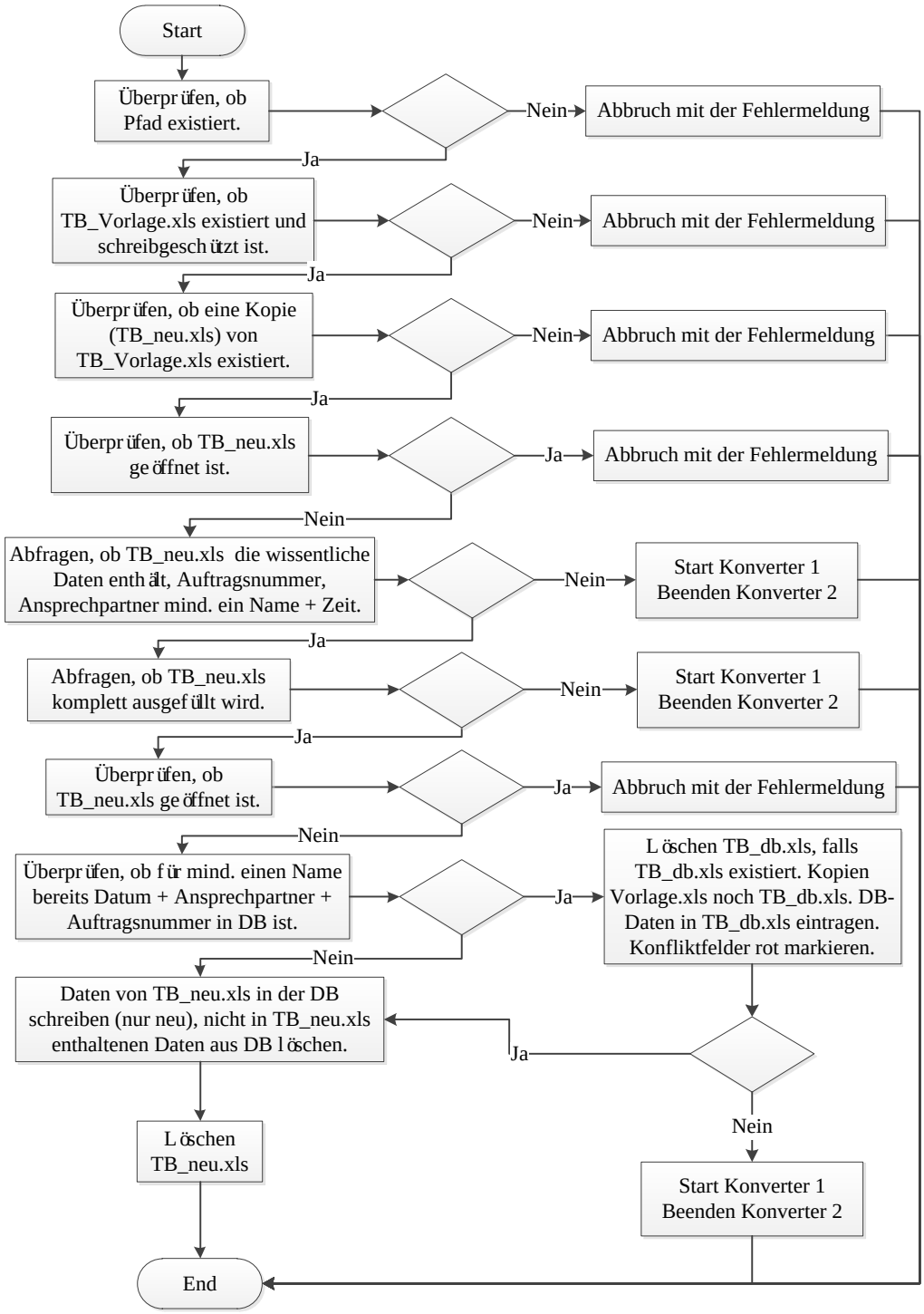

Abbildung 3.3: Flussdiagramm für den Import eines Tagesberichtes (TB_Tmp)

werden Datensätze des ältesten Tages von der Datenbank aus *TB_gen.xls* exportiert; diese Excel-Datei wird automatisch geöffnet.

In diesem Schritt prüfen Mitarbeiter Daten von *TB_gen.xls* ggf. werden die Daten korrigiert, schließlich wird *TB_gen.xls* gespeichert und geschlossen. Die Abbildung 3.4 zeigt das Flussdiagramm für den Konverter 3 an.

3.3.2.4 Konsistenz-Überprüfung der „genehmigten“ Tagesberichte mit der Datenbank und anschließender Datenimport

Aufgaben der Konverter 4 (Kurz: „TB_Gen“) sind, die genehmigten Daten auf Konsistenz mit den bereits in der Datenbank befindlichen Daten hin zu prüfen. Im Falle der Konsistenz werden diese Daten gegebenenfalls in der Datenbank aktualisiert und auf den Status „genehmigt“ gesetzt. In der Abbildung 3.5 wird das Flussdiagramm für den Konverter 4 dargestellt.

Der Konverter 4 ist ähnlich wie der Konverter 2. Zuerst überprüft das Programm, ob *TB_gen.xls* existiert, die durch den Konverter 3 erstellt wird. und ob *TB_Vorlage.xls* existieren. Wenn diese Dateien nicht existieren, wird eine Fehlermeldung dargestellt und der Konverter 4 beendet. Danach werden wesentliche Daten geprüft. Falls *TB_gen.xls* keine wesentlichen Daten enthält, wird der Konverter 4 beendet und Konverter 2 gestartet. Andernfalls überprüfen Programme, ob die Daten von *TB_gen.xls* mit den Datenbankdaten identisch sind, weil Mitarbeiter Daten in Konverter 3 korrigieren können. Falls die Daten nicht identisch sind, wird der Programmschritt des Konverters 2 durchgeführt, die Daten werden von der Datenbank aus *TB_db.xls* exportiert, Konfliktfelder werden mit rot markiert, die Daten von der *TB_gen.xls* in die Datenbank überschrieben; Daten, die nicht in TB_gen.xls enthalten sind, werden dabei aus der Datenbank gelöscht. Falls Daten identisch sind, werden Daten von der *TB_gen.xls* in die Datenbank aktualisiert. Nachdem die Daten in der Datenbanktabelle aktualisiert worden sind, werden die Datensätze als „gen“ bezeichnet. Im Konverter 4 entscheidet der jeweilige Mitarbeiter, ob die Daten von *TB_gen.xls* in die Datenbank aktualisiert werden. Falls der Mitarbeiter „ja“ auswählt, werden die Daten von *TB_gen.xls* korrigiert und *TB_gen.xls* geschlossen. Anderenfalls wird Konverter 4 beendet.

3.3.2.5 Vorteile und Nachteile

Vorteile dieser Idee sind die Nähe zur bisherigen Arbeitsweise und die für den Benutzer relativ einfache Bedienung, weil die meiste Dateneingabe und -korrektur in Microsoft Excel durchgeführt wird. Excel-Dateien werden durch den Konverter in die Datenbank importiert oder von der Datenbank exportiert. Aber die Flexibilität und die Erweiterbarkeit sind sehr schlecht, z. B. wenn Daten eines Tagesberichtes von der Datenbank aus Excel exportiert werden, wird immer der Datensatz des jeweils programmierten Tagesberichtes für den Benutzer dargestellt. In der Vorlage des Tagesberichtes gibt es zudem viele verbundene Felder. Um die Daten dieser Felder in die Datenbank zu importieren, muss das Tool Apache POI verwendet werden, das in Kapitel 4.2 besprochen wird. Falls ein neues verbundenes Feld

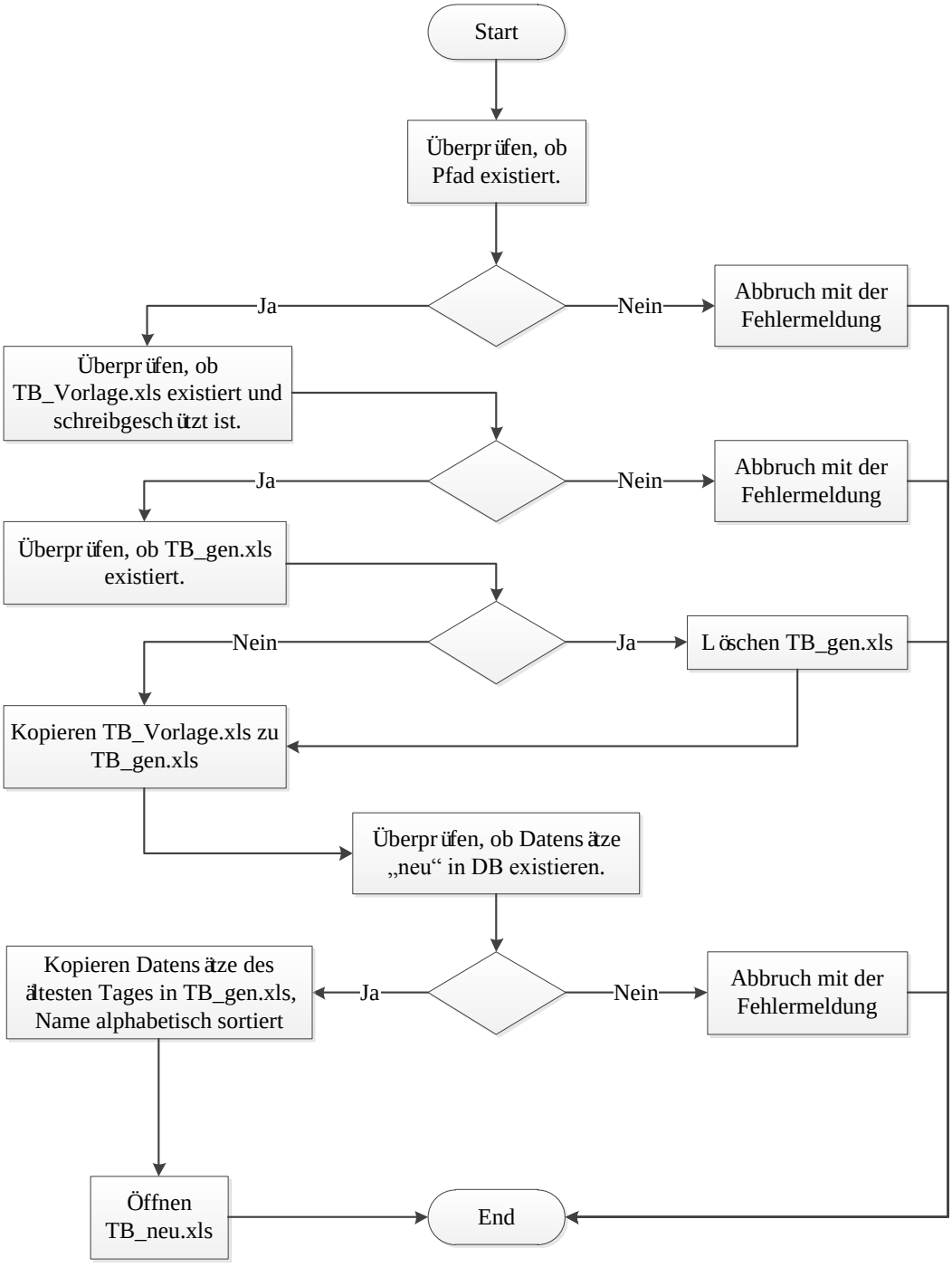

Abbildung 3.4: Flussdiagramm für den Export eines Tagesberichtes (TB_Prf)

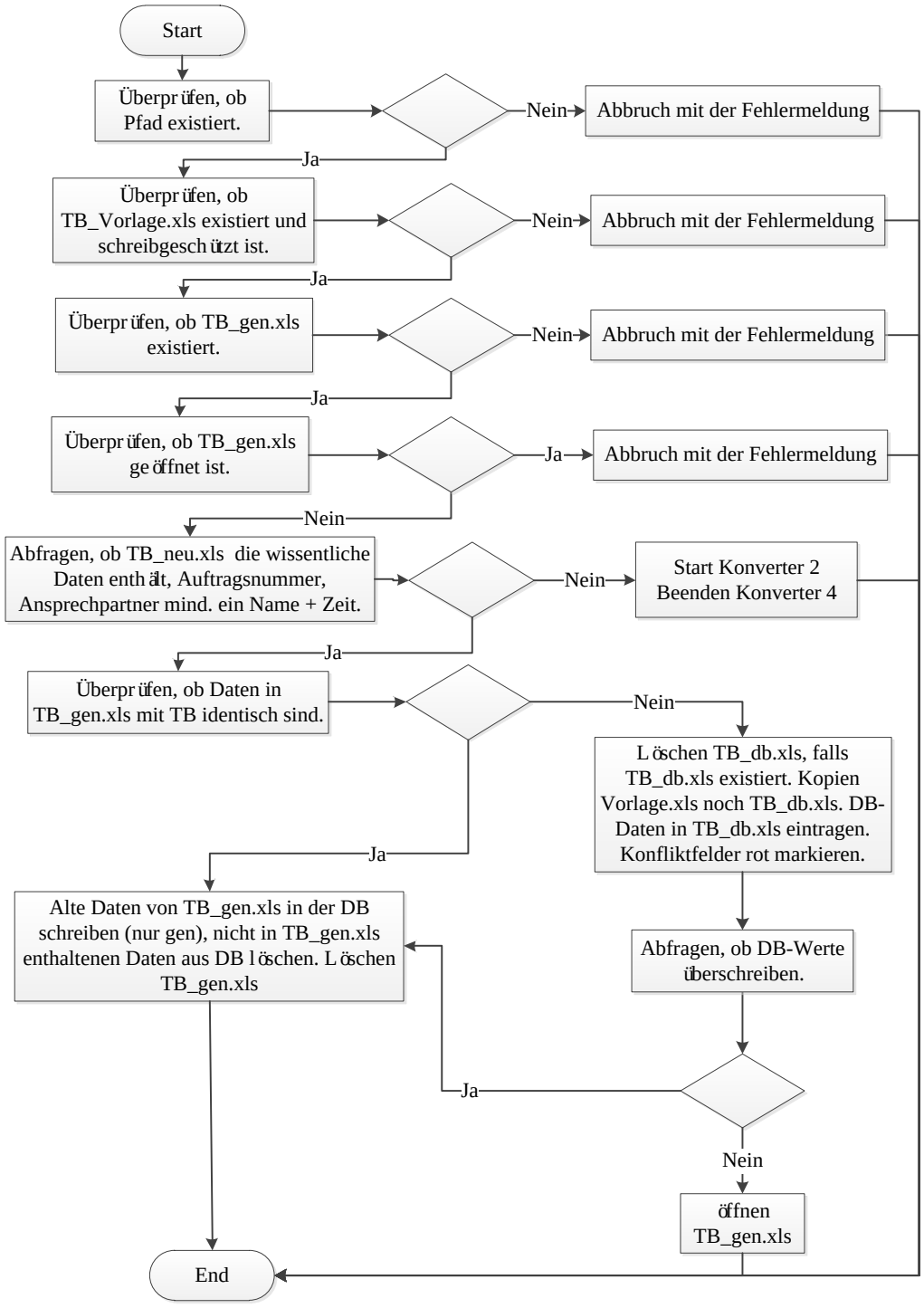

Abbildung 3.5: Flussdiagramm für die Konsistenz-Überprüfung eines Tagesberichtes (TB_Gen)

in der Excel-Vorlage erzeugt wird, muss der größte Teil des Konverters neu programmiert werden. Außerdem ist in der Firma ISAG keine konsequente Durchgängigkeit gegeben. So wurde zum Beispiel der Tagesbericht mit Microsoft Excel 2003 erstellt, die gesamte Rechnung wird jedoch mit Microsoft Excel 2007 durchgeführt. Apache POI bietet zwei verschiedene Methoden für Datenimport/ -export auf die unterschiedlichen Microsoft Excel-Versionen. Wenn beispielsweise Bilder in Microsoft Excel 2007 gespeichert werden, werden sie nur umständlich in die Datenbank importiert, weil Apache POI für Microsoft Excel 2007 noch nicht vollständig entwickelt wird.

3.3.3 Benutzeroberfläche auf Basis von Webanwendungen

Die dritte Realisierungs-Idee für das Design der Benutzeroberfläche ist, Webformulare zu definieren. So kann z. B. ein Tagesbericht durch das Webformular beschrieben werden. Das Webformular bezeichnet eine Interaktion zwischen Benutzereingaben und dem Web-Service oder der Datenbank. Webformulare werden durch Java Server Page (JSP) geschrieben. Jede Benutzeroberfläche kann als eine Webseite betrachtet werden, d. h. Benutzeroberflächen werden mit Hilfe eines beliebigen verfügbaren Webbrowsers für die Benutzer dargestellt. In diese Benutzeroberflächen tragen die Benutzer die Daten ein und klicken anschließendden Button „Speicher“. Auf diesen Befehl hin werden die Daten dann entsprechend ihres Status in die Datenbank importiert.

In den Benutzeroberflächen sollen folgende Funktionen realisiert werden.

- Eingabefehler überprüfen.
- Daten in die Datenbank importieren.
- Datenbank auf Konsistenz prüfen.
- Daten in der Datenbank aktualisieren.
- Daten aus der Datenbank exportieren, bzw. Daten aus Excel-Dateien oder PDF-Dateien exportieren

Wenn Daten des Tagesberichtes in die Datenbank importiert oder aktualisiert werden, haben diese Daten auch einen Status. Werden z. B. Daten in der Datenbanktabelle geschrieben, wird der Status dieser Daten als „neu“ oder als „gen“ bezeichnet bzw. der Status des Tagesberichtes als „neu“ oder „gen“ abgebildet. Wenn Daten in der Datenbank aktualisiert werden, wird der Status dieser Daten als „gen“ oder „exp“ modifiziert bzw. der Status des Tagesberichtes als „exp“ oder „gen“ abgebildet. Damit gibt es vier wichtige Funktionen für die Bearbeitung des Tagesberichtes. In dem folgenden Kapitel werden sie vorgestellt.

3.3.3.1 Funktion „neu Tagesbericht erstellen“

Die Funktion „neu TB erstellen“ beschreibt, wie Daten in die Datenbank importiert werden und wie Eingabefehler überprüft werden. Wenn der Button „neu TB erstellen“ in der Benutzeroberfläche „*Hauptfenster.jsp*“ angeklickt wird, wird die Benutzeroberfläche „*Tagesbericht.jsp*“ geöffnet. Die Daten werden dann vom Benutzer in diese Benutzeroberfläche eingetragen. Nach dem Anklicken des entsprechenden Buttons werden die Daten geprüft und in der Datenbank gespeichert. Die Abbildung 3.6 zeigt die unterschiedlichen Funktionen der Buttons in dieser Benutzeroberfläche.

Wenn der Button „Speichern“ angeklickt wird, wird vom Programm überprüft, ob die eingetragenen Daten schon soweit vollständig sind, dass sie in der Datenbank eindeutig identifiziert werden können. Falls Daten nicht vollständig genug eingetragen sind, werden die mindestens zu befüllenden Felder gelb markiert. Andernfalls werden die Daten in die Datenbank importiert und der Status dieser Daten in der Datenbanktabelle auf „neu“ gesetzt.

Wird der Button „Speichern&&Prüfen“ angeklickt, wird vom Programm ebenfalls zuerst überprüft, ob Daten schon soweit vollständig sind, dass sie in der Datenbank eindeutig identifiziert werden können. Danach wird vom Programm überprüft, ob die eingegebenen Daten schon in der Datenbank existieren. Wenn die Daten schon in der Datenbank existieren, werden die Felder der Konflikt-Daten mit rot markiert. Andernfalls werden die Daten in die Datenbank importiert und der Status dieser Daten in die Datenbanktabelle auf „gen“ gesetzt.

Wenn der Button „Abbrechen“ angeklickt wird, wird diese Benutzeroberfläche lediglich geschlossen und alle eingegebenen Daten werden verworfen.

3.3.3.2 Funktion „gespeicherte Tagesberichte korrigieren“

Diese Funktion beschreibt, wie Daten eines Tagesberichtes überprüft und korrigiert werden, deren Status „neu“ ist. Mit dieser Funktion werden diese Daten aus der Datenbank in die Benutzeroberfläche geschrieben, so dass sie dort überprüft, korrigiert und ergänzt oder sogar komplett gelöscht werden können. Anschließend können die Daten wieder als „neue“ Daten in die Datenbank geschrieben werden oder auch auf Eingabefehler sowie Konsistenz hin geprüft und als „genehmigte“ Daten in der Datenbank gekennzeichnet werden.

Wenn der Button „gespeicherte TB korrigieren“ in der Benutzeroberfläche „*Hauptfenster.jsp*“ angeklickt wird, wird die Benutzeroberfläche „*search_neu.jsp*“ geöffnet. Ein bestimmter Tagesbericht wird ausgewählt. Nach dem Anklicken des Buttons „Öffnen“ werden die Daten dieses Tagesberichtes von der Datenbank auf die Benutzeroberfläche „*Tagesbericht_kor.jsp*“ übermittelt. Nach dem Anklicken des entsprechenden Buttons werden die Daten geprüft und in der Datenbank aktualisiert. Die Abbildung 3.7 zeigt die Funktionen der unterschiedlichen Buttons in der Benutzeroberfläche „*search_neu.jsp*“ und Benutzeroberfläche „*Tagesbericht_kor.jsp*“.

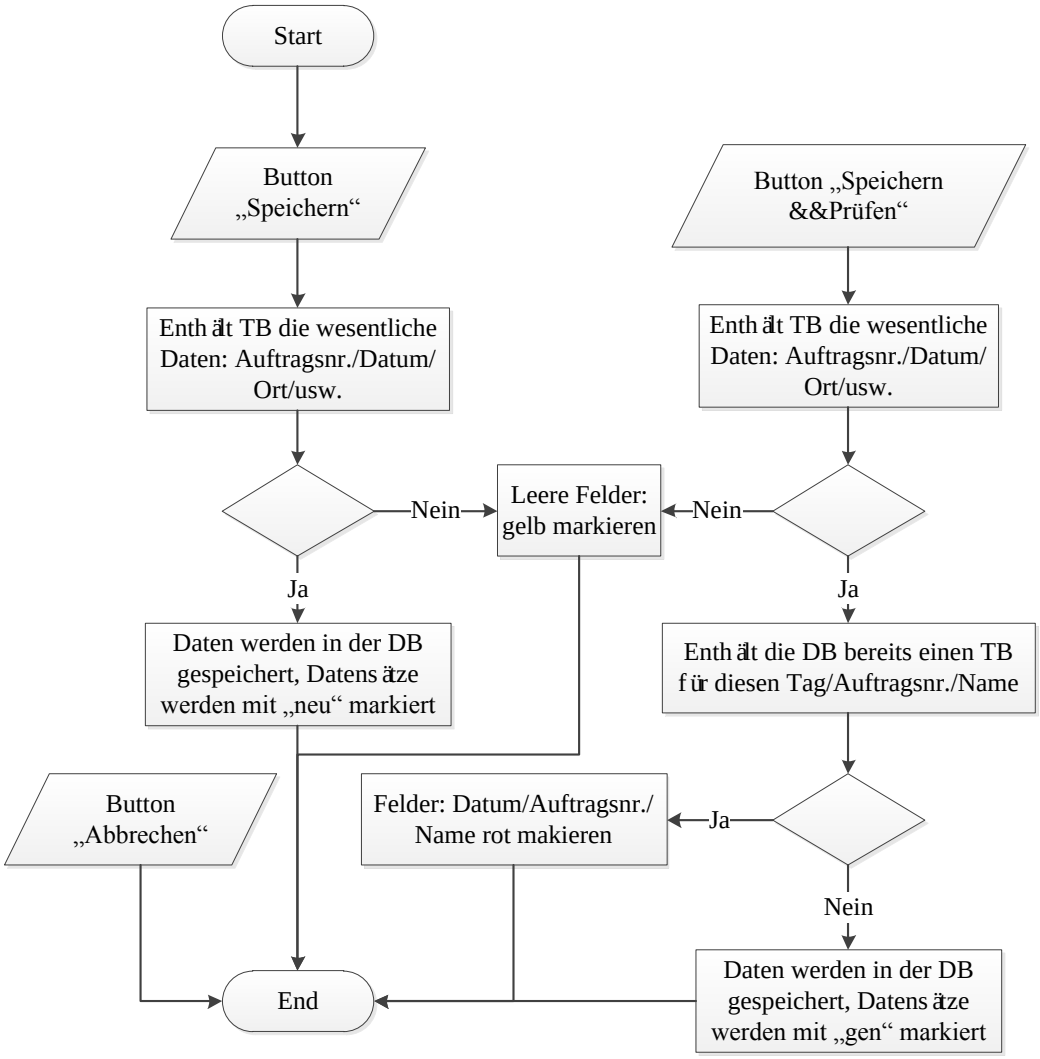

Abbildung 3.6: Flussdiagramm für die Funktionsweise der unterschiedlichen Buttons in der Benutzeroberfläche *„Tagesbericht.jsp“*

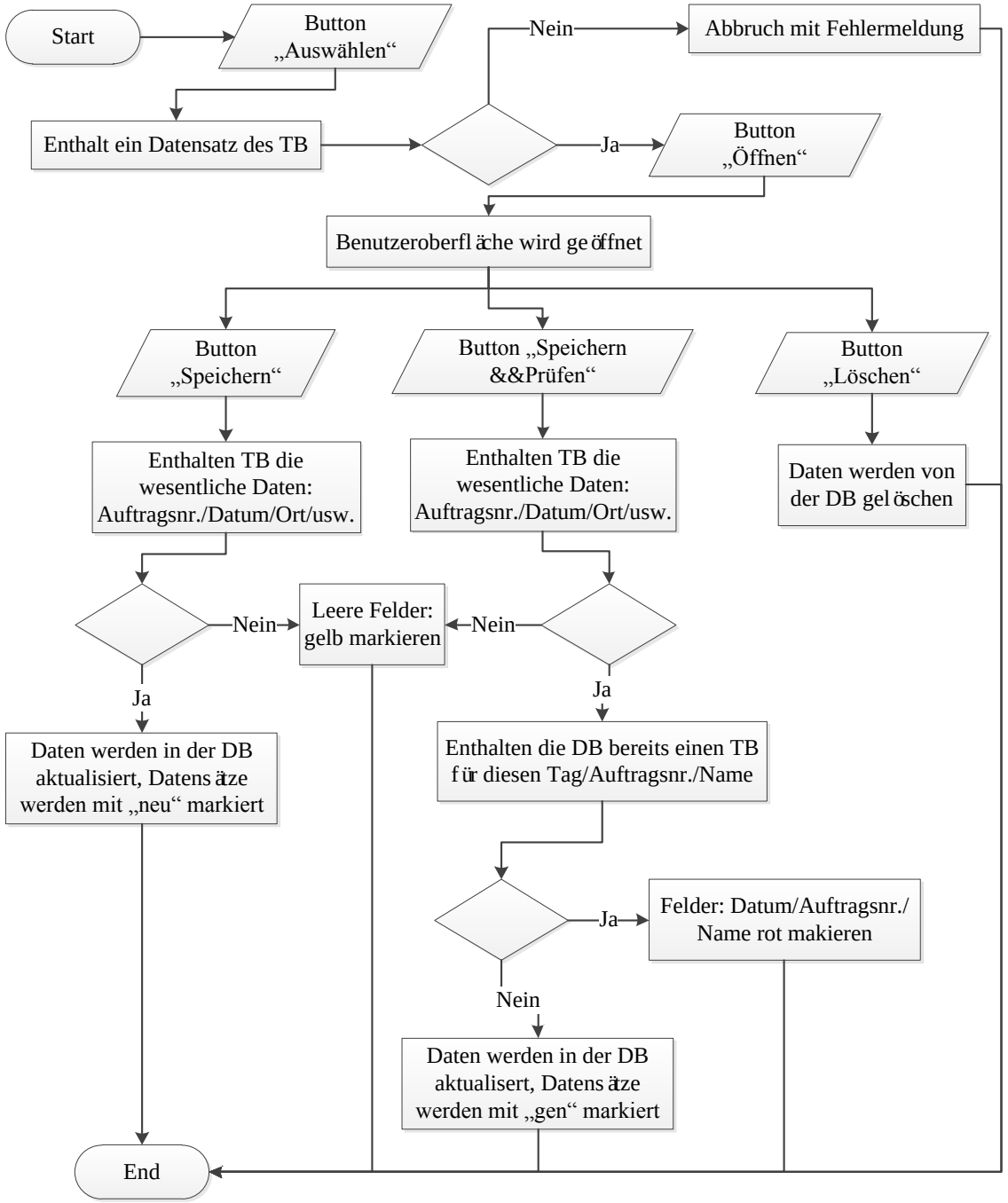

Abbildung 3.7: Flussdiagramm für die Funktionsweise der unterschiedlichen Buttons in der Benutzeroberfläche *„search_neu.jsp“* und Benutzeroberfläche *„Tagesbericht_kor.jsp“*

Wenn der Button „Auswählen“ angeklickt wird, wird durch das Programm geprüft, ob ein Tagesbericht durch die Auswahl der Auftragsnummer, des Datums, der Ansprechpartner, des Einsatzortes und die Identifikation des Tagesberichtes identifiziert werden kann, der den Status „neu“ hat. Wenn es keinen solchen Tagesbericht gibt, wird eine Fehlermeldung ausgegeben. Wenn ein entsprechender Tagesbericht identifiziert wird, kann der Button „Öffnen“ angeklickt werden, woraufhin die Daten dieses Tagesberichtes in der Benutzeroberfläche „*Tagesbericht_kor.jsp*“ dargestellt werden und zur Überprüfung durch den Benutzer bereitstehen.

In der Benutzeroberfläche „*Tagesbericht_kor.jsp*“ ist die Funktionsweise des Buttons ähnlich zu den Buttons der Benutzeroberfläche „*Tagesbericht.jsp*“. Weil die Benutzer in dieser Situation vielleicht noch nicht alle erforderlichen Daten des Tagesberichtes haben oder die Daten nicht vollständig überprüfen können, benötigen sie eine Möglichkeit zum ungeprüften Abspeichern der Daten. Hierzu wird der Button „Speichern“ in dieser Benutzeroberfläche bereitgestellt. Nach dem Anklicken des Buttons „Speichern“ prüft das Programm wiederum, ob die Daten ausreichend vollständig eingetragen sind. In diesem Fall werden der Mitarbeitername und die Arbeitszeit geprüft, weil das Datum, die Auftragsnummer usw. die Auswahlkriterien waren und deshalb in dieser Benutzeroberfläche schreibgeschützt sind. Danach werden die Daten in der Datenbanktabelle aktualisiert, wobei der Status dieser Daten in der Datenbank unverändert bleibt.

Wenn die Daten eines Tagesberichtes schon vollständig und geprüft sind, kann der Benutzer den Button „Speichern&&Prüfen“ anklicken. Nach der Überprüfung der Vollständigkeit und der Datenkonsistenz werden die Daten in der Datenbank aktualisiert und der Status dieser Daten in der Datenbanktabelle von „neu“ zu „gen“ geändert.

Weil bei dem Button „Speichern“ die Daten mit dem Status „neu“ nicht auf Konsistenz zur Datenbank hin geprüft werden, kann es vorkommen, dass Daten eines Tagesberichtes mehrfach als „neue“ Daten in der Datenbank stehen. Weiterhin kann es passieren, dass das Datum oder die Auftragsnummer im neuen Tagesbericht falsch eingegeben wurden. Diese Daten sind in der Korrekturfunktion nicht editierbar und die zugehörigen Tagesberichtseingaben somit nutzlos. Diese Tagesberichtseingaben können mit dem Button „Löschen“ komplett aus der Datenbank gelöscht werden.

3.3.3.3 Funktion „freigegebene Tagesberichte korrigieren“

Diese Funktion ist ähnlich wie die Funktion „gespeicherte TB korrigieren“. Durch diese Funktion können die Daten eines bereits freigegebenen, aber dennoch zu korrigierenden Tagesberichts in der Datenbank aktualisiert werden. Hierzu werden die Daten aus der Datenbank auf die Benutzeroberfläche „*Tagesbericht_exp.jsp*“ übertragen, wo die entsprechenden Korrekturen durchgeführt werden können. Zusätzlich gibt es die Möglichkeit, alle in der Eingabemaske befindlichen Daten, also auch die derzeit nicht in der Datenbank speicherbaren Fehlerberichte, in eine PDF-Datei zu exportieren. Nach dem Anklicken des Buttons „Exportieren&&Prüfen“ werden die Daten in der Datenbank aktualisiert. Der Status des Tagesberichtes bleibt jedoch unverändert „gen“. Dann wird die Abfrage gestellt, ob Daten in

die PDF-Datei exportiert werden sollen. Wenn der Benutzer „ja“ auswählt, wird der Button „PDF-erstellen“ aktiviert. Nach dem Anklicken dieses Buttons wird der Tagesbericht in eine PDF-Datei exportiert und diese PDF-Datei anschließend automatisch geöffnet. Die Abbildung 3.8 zeigt die Funktionen der unterschiedlichen Buttons in den Benutzeroberflächen „*search_kor.jsp*“ und „*Tagesbericht_exp.jsp*“.

3.3.3.4 Funktion „freigegebene Tagesberichte exportieren“

Diese Funktion beschreibt, wie Daten von der Datenbank in die Gesamtrechnungen (RG) exportiert werden. Die Gesamtrechnungen enthalten die gesamten Arbeitszeiten am gleichen Einsatzort und mit dem gleichen Ansprechpartner bezogen auf eine bestimmte Auftragsnummer. Nach dem Anklicken des Buttons „Auswählen“ werden die Datensätze der Tagesberichte, die den gleichen Einsatzort, den gleichen Ansprechpartner und die gleiche Auftragsnummer haben, in der Benutzeroberfläche „*search_exp.jsp*“. Die Abbildung 3.9 zeigt die Funktionen der unterschiedlichen Buttons in der Benutzeroberfläche „*search_exp.jsp*“.

Falls es keine solcher Datensätze gibt, wird eine Fehlermeldung angezeigt. Wenn es jedoch einen entsprechenden Datensatz gibt, kann der Benutzer den Button „Excel-Öffnen“ anklicken. Die Daten werden dann von der Datenbank in die Excel-Datei (RG) exportiert, der Status der exportierten Tagesberichte wird auf „exp“ gesetzt und zur Überprüfung die Export-Datei automatisch geöffnet.

3.3.3.5 Vorteile und Nachteile

Die großen Vorteile für diese Idee sind die hohe Verfügbarkeit und die optimale Wartbarkeit und Erweiterbarkeit der Web-basierten und somit auf einem Server zentral abgelegten Programme. So können die Funktionen der Benutzeroberfläche jederzeit weiterentwickelt oder korrigiert werden. Neue Funktionen können beliebigen Benutzeroberflächen hinzugefügt werden, da es nach dem Update auf dem Server keine ladbaren alten Versionen der Oberflächen mehr gibt. Die im Servlet-Container vorhandenen neuen Versionen der Benutzer-Oberflächen stehen sofort allen Anwendern zur Verfügung, Alte Versionen der Benutzer-Oberflächen wurden nach Löschen aus dem Servlet-Container automatisch aus dem gesamten System entfernt. Die Flexibilität ist ebenfalls sehr gut, da die Benutzeroberflächen durch beliebige Webbrowser dargestellt und somit auf den unterschiedlichsten Geräten aufgerufen werden können, z. B. Desktop-PC, Laptop-PC, Tablet-PC und Smartphones.

Diesen wertvollen Vorteilen steht der Einmal-Aufwand der Benutzer gegenüber, die sich in die neuen Funktionen der Benutzeroberfläche einlernen müssen.

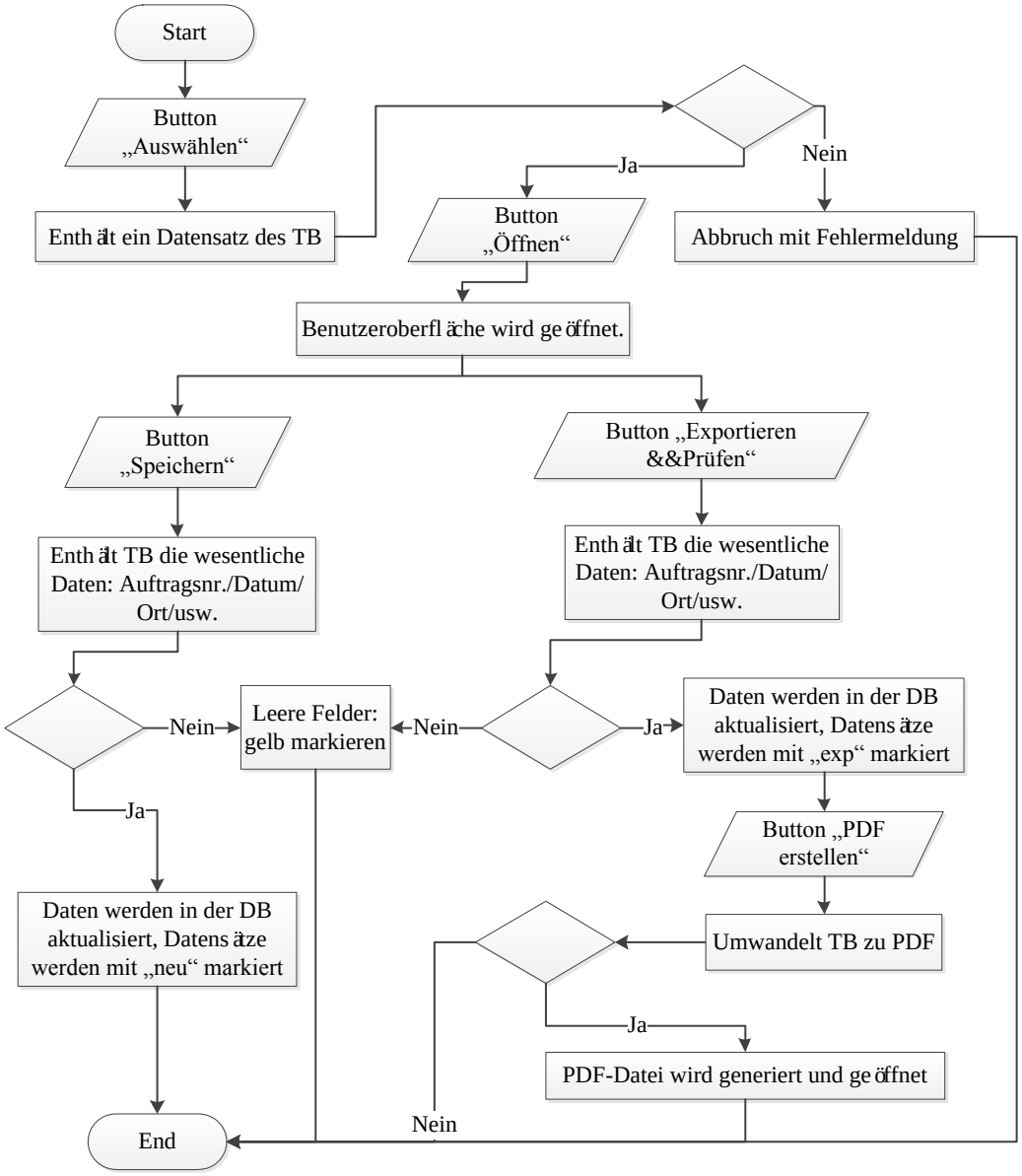

Abbildung 3.8: Flussdiagramm für die Funktionsweise der unterschiedlichen Buttons in der Benutzeroberfläche *„search_kor.jsp“* und Benutzeroberfläche *„Tagesbericht_exp.jsp“*

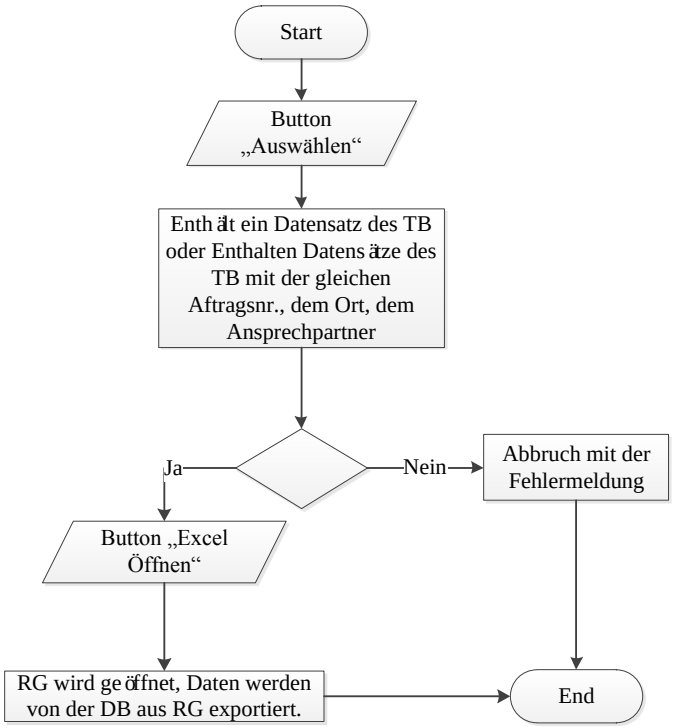

Abbildung 3.9: Flussdiagramm für die Funktionsweise der unterschiedlichen Buttons in der Benutzeroberfläche *„search_exp.jsp“*

3.4 Auswahl des optimalen Benutzeroberflächen-Konzepts

Auf der Basis der in Kapitel 3.2 aufgeführten Anforderungen an die zu erstellenden Benutzeroberflächen sowie der Randbedingungen für die zukünftige Projektabwicklung werden die beschriebenen drei Benutzeroberflächen-Realisierungsarten bewertet.

Die Benutzeroberfläche auf Basis von Microsoft Excel reduziert die Eingabe-Fehlermöglichkeiten durch entsprechende Eingabemasken und den manuellen Aufwand durch die Verwendung einer Datenbank. Die Prüfung der Konsistenz ist zwar prinzipiell programmierbar, jedoch nur mit sehr hohem Aufwand. Das Einfügen in bestehende Vorgehensweisen und die manuelle Korrigierbarkeit sind sehr gut, die Wartbarkeit und Erweiterbarkeit sind jedoch sehr schlecht.

Die Benutzeroberfläche auf Basis von Microsoft Excel und Java-Programmen reduziert ebenfalls die Eingabe-Fehlermöglichkeiten durch entsprechende Eingabemasken und den manuellen Aufwand durch die Verwendung einer Datenbank. Die Prüfung der Konsistenz ist hier programmierbar, jedoch ist der Aufwand dafür hoch. Das Einfügen in bestehende

Vorgehensweisen und die manuelle Korrigierbarkeit sind sehr gut. Die Wartbarkeit und Erweiterbarkeit sind zwar besser, als bei der vorigen Variante, aber dennoch schlecht.

Die Benutzeroberfläche auf Basis von Microsoft Excel und Java-Programmen reduziert ebenso wie die anderen Varianten die Eingabe-Fehlermöglichkeiten durch entsprechende Eingabemasken und den manuellen Aufwand durch die Verwendung einer Datenbank. Die Prüfung der Konsistenz ist hier deutlich einfacher und mit relativ geringem Aufwand programmierbar. Das Einfügen in bestehende Vorgehensweisen und die manuelle Korrigierbarkeit sind wie bei den vorigen Varianten sehr gut. Die Wartbarkeit und Erweiterbarkeit sind hingegen sogar hervorragend, ebenso wie die Einsetzbarkeit auf unterschiedlichsten Geräten und Betriebssystemen.

Die dritte Realisierungsvariante Benutzeroberfläche auf Basis von Webanwendungen bietet alles, was die anderen Varianten gut können und das bei relativ geringem Programmieraufwand für die Konsistenzprüfung und hervorragenden Eigenschaften hinsichtlich der Einsetzbarkeit, Wartbarkeit und Erweiterbarkeit. In dieser Diplomarbeit wird deshalb die Realisierungsvariante Benutzeroberfläche auf Basis von Webanwendungen weiterentwickelt.

4 Implementierung

In diesem Kapitel wird das Design der Benutzeroberfläche auf Basis von Webanwendungen ausgestaltet. Die Programme werden unter dem Projektnamen „ISAG“ implementiert. Mit Hilfe des UML-Diagramms (Unified Modeling Language)[1] werden die Programmstruktur und die Bedeutung der Klassen aufgezeigt. In dem Projekt „Test“ werden die realisierten Funktionen getestet.

4.1 Datenbankstruktur

Um das Programm auch in einem frühen Programmierstand lauffähig zu gestalten, muss die bei F. Baumann [Bau12] beschriebene Datenbank auf das absolut notwendige Minimum reduziert werden. Andererseits müssen neue Tabellen und Tabelleneinträge hinzugefügt werden, damit die neuen Funktionen übersichtlich und redundanzfrei implementiert werden können. Wichtig ist dabei, dass die funktional zusammengehörenden Daten auch in der Datenbank möglichst direkt miteinander verknüpft sind. Weiterhin ist darauf zu achten, dass die Datenstruktur möglichst rückwirkungsfrei ist. Wird zum Beispiel für ein Projekt der Ansprechpartner geändert, so muss auch in diesem Fall die Konsistenz für die alten Datensätze gewahrt bleiben. Tagesberichte gehören als Scan direkt zu den Projektdaten, während die im Tagesbericht aufgeführten Mitarbeiterstunden zum jeweiligen Mitarbeiter gehören. Die neue Datenbankstruktur gliedert sich somit in 4 wichtige Teile: die Personendaten, die Projektdaten zusammen mit dem Angebot, die Firmendaten und schließlich die personellen Ressourcen mit der Zeiterfassung.

- Der Teil Personen:

  Dieser Teil ist unabhängig von den anderen Teilen. In diesem Teil gibt es Tabellen, die nur die Personeninformationen betreffen. Wenn Personeninformationen von der Oberfläche eingetragen werden, werden Informationen in diesen Tabellen gespeichert. Einer Person muss dabei nicht notwendigerweise eine Firma zugeordnet werden.

- Der Teil Firmen:

  In diesem Teil werden die Firmeninformationen gespeichert. Dieser Teil ist abhängig von den Personen, weil jede Firma mindestens einen Ansprechpartner benötigt. Allerdings müssen einer Firma nicht notwendigerweise ein Projekt oder personelle Ressourcen zugeordnet sein.

[1] http://www.uml.org/

- Der Teil Projekte und Angebote:

  In diesem Teil werden die Projektinformationen wie Auftragsnummern für den Tagesbericht und die möglichen Einsatzorte abgelegt. Dieser Teil ist somit abhängig von den Firmen und den Personen sowie, bei erteiltem Auftrag, von den Ansprechpartnern und Mitarbeitern der jeweiligen Firma. Von den personellen Ressourcen ist dieser Teil unabhängig, da ein Projekt bereits existieren kann, bevor dafür Ressourcen fest eingeplant sind.

- Der Teil personelle Ressourcen mit Zeiterfassung:

  Dieser Teil beschreibt die Arbeitszeit der Mitarbeiter. Er ist abhängig von den anderen drei Datenteilen, da eine Zeiterfassung ohne die Daten Personen, Firmen und Projekte nicht sinnvoll ist.

In der Abbildung 4.1 wird das ER-Diagramm für die Datenbankstruktur dargestellt, die roten Quadrate bezeichnen die vier Teile. Wenn also ein neues Projekt mit einer neuen Firma und neuen Personen in die Datenbank eingegeben werden soll, müssen die Daten in der obengenannten Reihenfolge (als Primary Key Einträge) eingegeben werden, um nachfolgend die Referenzierungen durch die entsprechenden Foreign Keys aufbauen zu können. Mit dieser Vorgehensweise kann man konsequent die Eingabe vollständiger Datensätze fordern, ohne die Flexibilität der Datenbank dabei zu reduzieren.

Wenn Daten eines Tagesberichts in die Datenbanktabellen geschrieben werden, werden zwei wichtige Tabellen verwendet. In der Tabelle „Tagesbericht“ werden das Datum, die Auftragsnummer, der Ansprechpartner, der Einsatzort und der Status des Tagesberichtes gespeichert. Der Primary Key des Tagesberichtes wird automatisch inkrementiert und dient dazu, in den TagesberichtEintragungen referenziert zu werden. Die Arbeitszeit und der Mitarbeitername werden in der Tabelle „TagesberichtEintragung“ gespeichert. Diese zwei Tabellen werden mit 1 zu n abgebildet, weil ein Tagesbericht genau ein Datum, eine Auftragsnummer, einen Einsatzort und einen Ansprechpartner hat, aber im Allgemeinen mehrere Mitarbeiternamen sowie deren Arbeitszeiten beinhaltet. Nach der Erweiterung der Tabellen um diese Einträge ist die Datenbankstruktur für die Programmierung der notwendigsten Benutzeroberflächen-Funktionen geeignet. Darüber hinaus ist sie im Zuge der Realisierung weiterer Oberflächen-Funktionen leicht erweiterbar, ohne dass dafür die Struktur oder die Benennungen geändert werden müssten.

4.2 Lesen und Schreiben der Microsoft Office-Dateien durch Apache POI

Wenn Daten von Microsoft Excel in die Datenbank importiert werden oder Daten von der Datenbank auf Microsoft Excel exportiert werden, wird Apache POI[2] verwendet. Apache POI

[2] http://poi.apache.org/

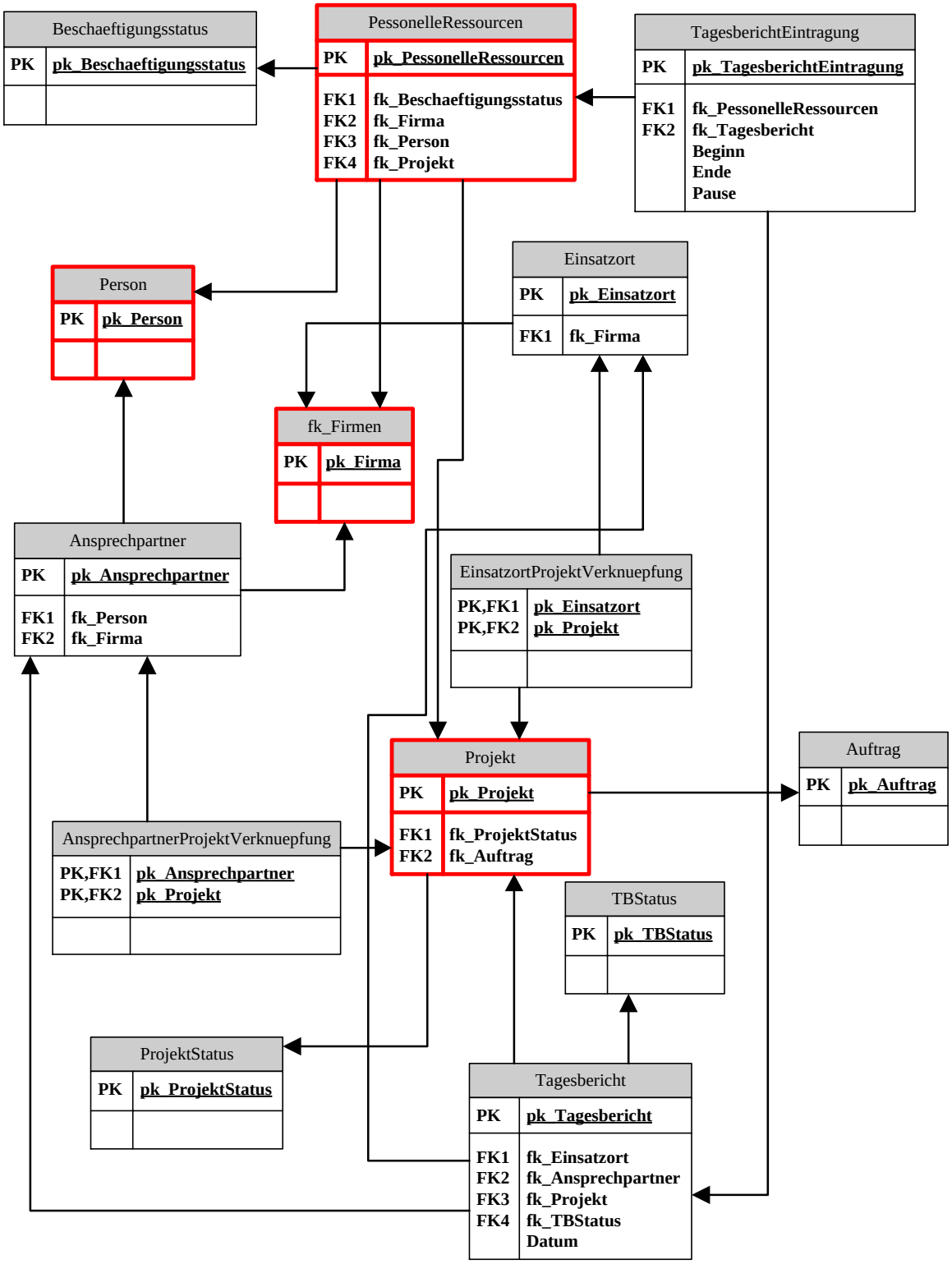

Abbildung 4.1: ER-Diagramm für die Datenbankstruktur

ist eine Open Source Java API[3] zum Lesen und Schreiben für das Dateiformat von Microsoft Office. In dieser Diplomarbeit wird die Version der POI 3.8 verwendet. Apache POI bietet folgende Komponenten für das Lesen und das Schreiben von der Microsoft Excel-Datei und der Microsoft Word-Datei POI [POI]:

- HSSF (Horrible Spreadsheet Format): HSSF ist eine Java API, um Daten von den Microsoft Excel 97(-2007)-Dateien zu verarbeiten.
- XSSF (XML Spreadsheet Format): XSSF ist eine Java API, um Daten von den Microsoft Excel (.xlsx)-Dateien zu verarbeiten.
- HWPF (Horrible Word Processor Format): HWPF ist eine Java API, um Daten von den Microsoft Word 97(-2007)-Dateien zu verarbeiten.
- XWPF: (XML Word Processor Format): XWPF ist eine Java API, um Daten von den Microsoft Word (.xlsx)-Dateien zu verarbeiten.

4.2.1 Import Daten von Microsoft Excel 2003 in die Datenbank

Jedes Excel-Dokument kann mit dem HSSF-Objekt oder dem XSSF-Objekt abgebildet werden. Die Abbildung 4.2 stellt die Beziehungen zwischen den HSSF-Objekten und dem Microsoft Excel 2003 Spreadsheet dar.

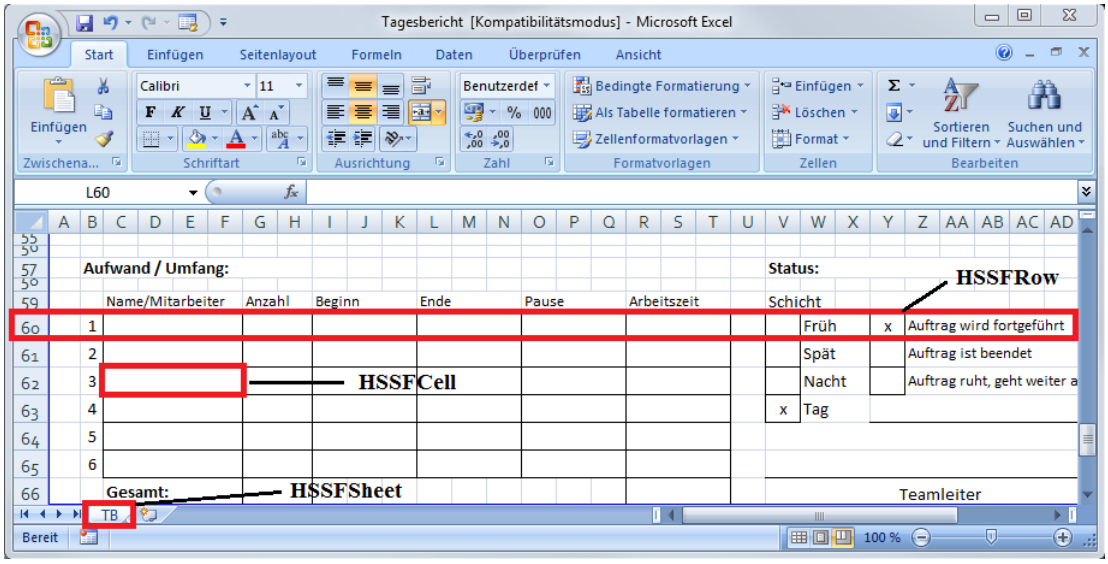

Abbildung 4.2: Beziehung zwischen den HSSF-Objekte und dem Spreadsheet

Ein Excel-Dokument ist eine Arbeitsmappe. Immer dann, wenn die HSSFWorkbook in einem Programm aufgerufen wird, wird eine Arbeitsmappe in Excel bearbeitet. Jede Arbeitsmappe

[3]http://poi.apache.org/apidocs/index.html

enthält eine oder mehrere Spreadsheets. Wenn ein Programm das HSSFSheet aufruft, wird ein Spreadsheet verarbeitet. Jedes Spreadsheet hat im Allgemeinen mehrere Zeilen. Wenn Zeilen des Spreadsheets angesprochen werden sollen, dann muss das HSSFRow verwendet werden. Um die Zellen des Spreadsheets zu bearbeiten, rufen Programme die HSSFCell auf. Bei Microsoft Excel 2007 oder Microsoft Excel 2010 wird das XSSFWorkbook verwendet, das Prinzip ist ähnlich wie Excel 2003 ist. In der Abbildung 4.3 wird die Funktionsweise der POI-HSSF für das Lesen des Excel-Dokumentes dargestellt.

Es gibt in dieser schematischen Darstellung insgesamt 3 Schleifen. In der ersten Schleife werden alle Spreadsheets eines spezifizierten Workbooks gelesen. Die zweite Schleife liest alle Zeilen eines spezifizierten Spreadsheets, Daten der Zellen einer spezifizierten Zeile werden durch die dritte Schleife ausgelesen. Im ersten Schritt wird ein *POIFSFileSystem* Objekt erstellt, damit wird das spezifizierte Excel-Dokument gelesen. Durch das *POIFSFileSystem* Objekt wird ein *HSSFWorkbook* Objekt konstruiert. Dieses *HSSFWorkbook* Objekt stellt das Excel Dokument (z. B. einen TB) dar.

Im zweiten Schritt wird die Anzahl der Spreadsheets durch den Aufruf der Funktion `getNumberOfSheets()` bestimmt. Anschließend gibt die Funktion `getPhysicalNumberOfRows()` die Anzahl der physikalischen definierten Zeilen von einem Spreadsheet zurück. Nach dem Aufruf der Funktion `getRow()` wird jede Zeile des Spreadsheets gelesen, die Zellen jeder Zeile werden durch mehrmalige Aufrufe der Funktion `getCell()` ausgelesen. Weil es z. B. im Tagesberichtsformular viele verbundende Zellen gibt, muss eine eigene Funktion `getMergedRegionValue()` aufgerufen werden. Diese Funktion beschreibt, wie Daten von den verbundenden Zellen gelesen werden. In dieser Funktion wird zuerst die Anzahl der verbundenden Zellen bestimmt.

Anschließend wird die Grenze der jeweiligen verbundenden Zelle durch den Aufruf der Klasse *CellRangeAddress* bestimmt. Dann wird jede Zelle des Spreadsheets überprüft, ob sie zu einer verbundenden Zelle gehört. Wenn sie zu einer verbundende Zelle gehört, wird die eigene Funktion `getCellValue()` aufgerufen, durch die die Inhalte der Zelle gelesen werden. Schließlich werden gelesenen Inhalte in einer Liste gespeichert und durch den Aufruf weiterer Funktionen Daten der Liste in die Datenbank importiert.

4.2.2 Export Daten von der Datenbank auf Microsoft Excel 2003

Die Funktionsweise der POI-HSSF für das Schreiben in das Excel-Dokument ist ähnlich wie das Lesen aus dem Excel-Dokument. Die Abbildung 4.4 zeigt, wie Daten von der Datenbank in das Excel-Dokument exportiert werden.

Von der Oberfläche werden die Auftragsnummer, der Ansprechpartner und der Einsatzort festgelegt. Dann wird zunächst ein *POIFSFileSystem* Objekt erstellt, mit dem die angegebene Excel-Vorlage (RG) gelesen wird. Durch den Aufruf der Funktion `getSheetAt()` wird das Spreadsheet für die gesamte Rechnung bestimmt. Danach werden zwei eigene Klassen *IdSelect* und *ExcelExportDataSelect* aufgerufen, um Mitarbeiternamen und Arbeitszeiten von der Datenbank auszuwählen und diese Daten in eine Liste zu speichern. Dann werden die

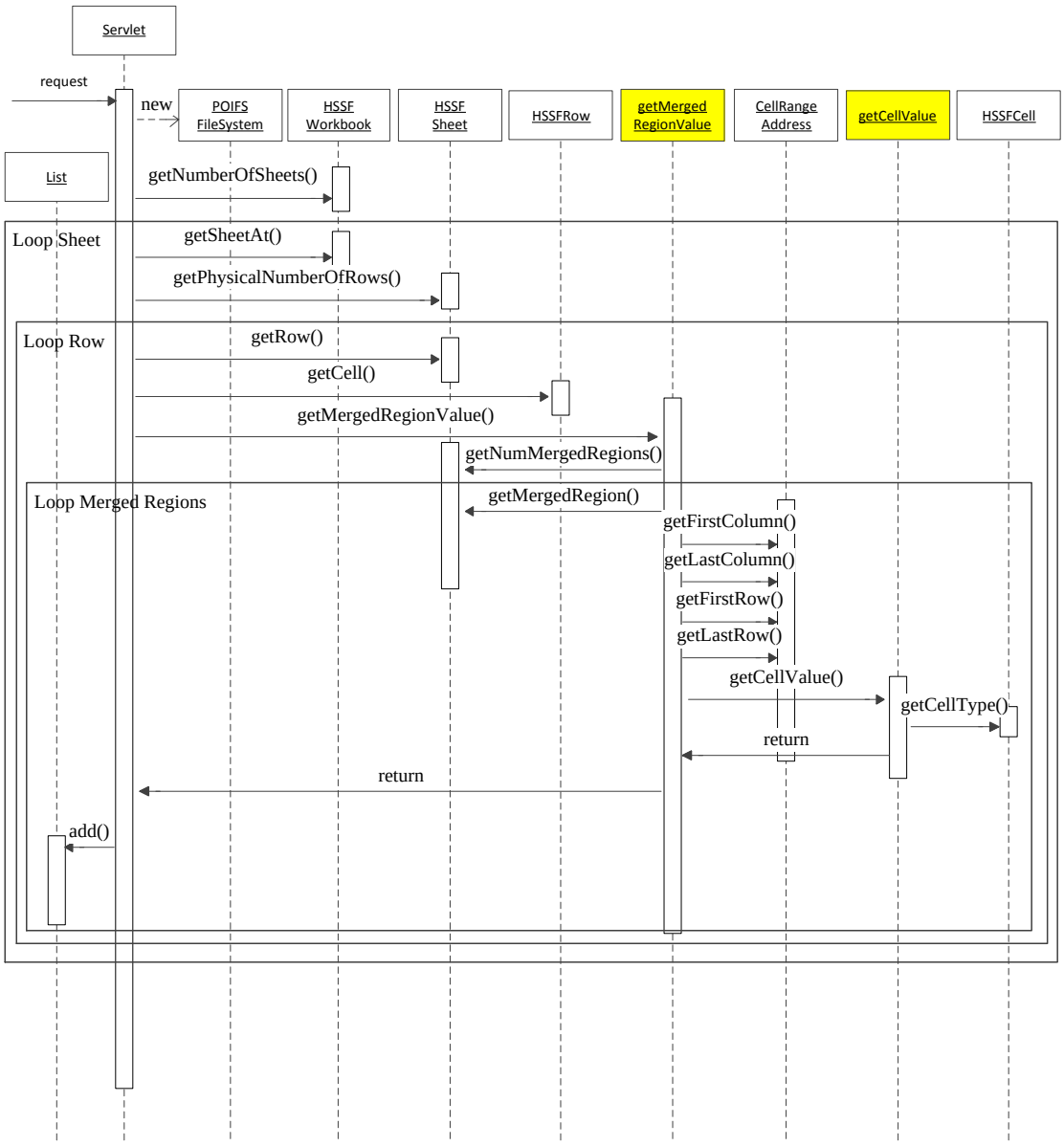

Abbildung 4.3: Sequenzdiagramm für das Lesen des Excel-Dokumentes durch Apache POI-HSSF

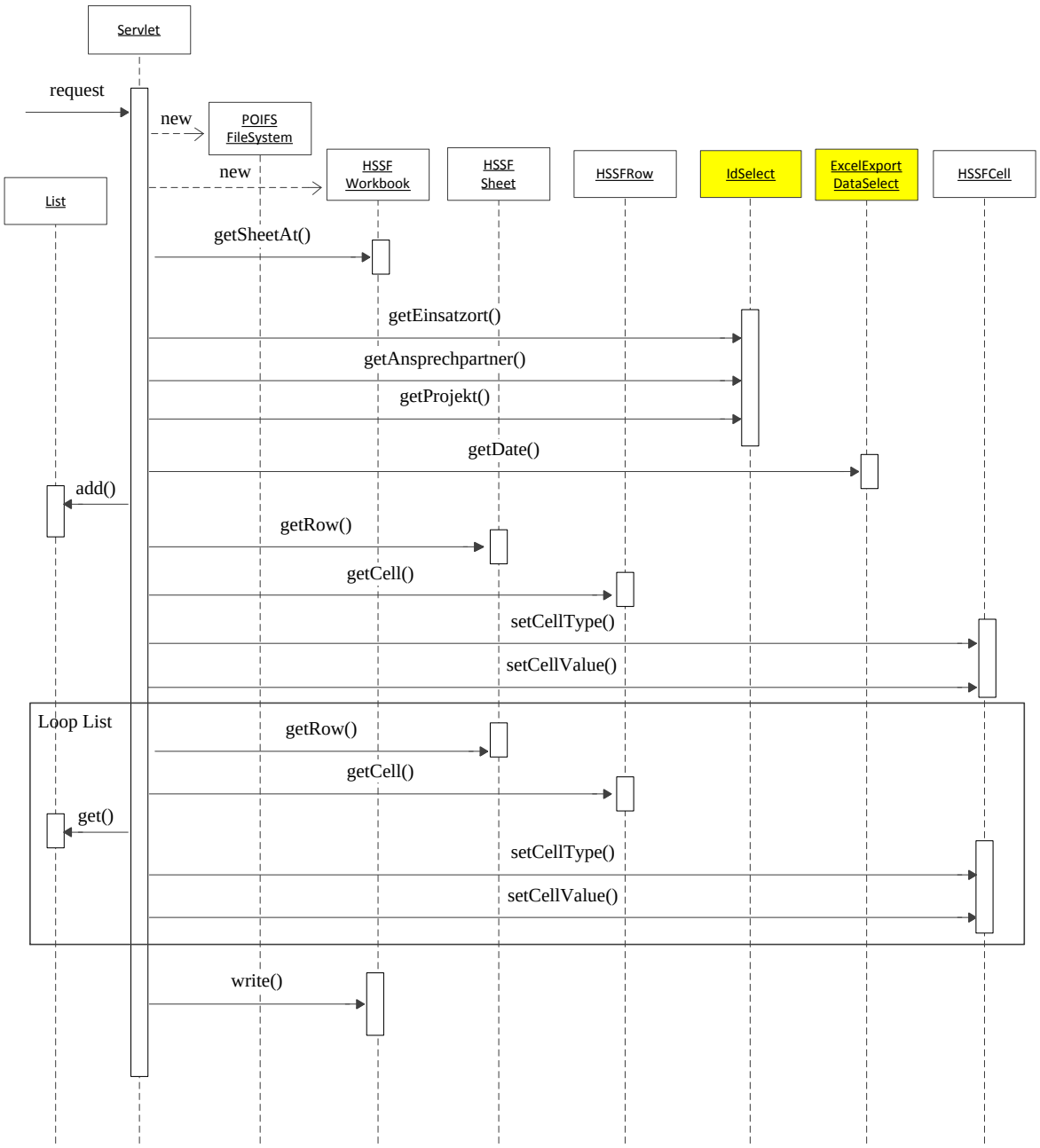

Abbildung 4.4: Sequenzdiagramm für das Schreiben in das Excel-Dokument durch Apache POI-HSSF

Zellen des Spreadsheets gelesen. Die Auftragsnummer, der Ansprechpartner und der Einsatzort werden durch den Aufruf der Funktionen `SetCellType()` und `SetCellValue()` in das Spreadsheet geschrieben. Die Mitarbeiternamen und die zugehörigen Arbeitszeiten werden aus der erstellten Liste gelesen und durch mehrmalige Aufrufe der Funktion `SetCellType()` sowie `SetCellValue()` in das Spreadsheet geschrieben. Schließlich wird dieses Spreadsheet über einen *OutputStream* ausgegeben.

4.3 Funktionsweise der Benutzeroberflächen

Die Funktionsweise der Benutzeroberflächen wird durch das UML-Diagramm dargestellt. Die Sprache UML ist eine vereinheitlichte Modellierungssprache und wurde von der OMG standardisiert. Sie dient zu der Visualisierung einer grafischen Notation, der Spezifizierung der formalen Sprachen, Konstruktion und Dokumentation von Modellen für komplexe Softwaresysteme [BRJ06]. Durch das entsprechende Diagramm werden die Beziehungen zwischen unterschiedlichen Klassen grafisch dargestellt und die angewendeten Methoden der Klassen erklärt. Das UML-Diagramm beschreibt auch, wie die Daten zwischen den Benutzeroberflächen und der Datenbank übermittelt werden. In der Benutzeroberfläche *„Hauptprogramm-Fenster.jsp“* gibt es vier Buttons, jeder Button bildet eine Funktion ab.

4.3.1 Funktionsweise für „neu Tagesbericht erstellen“

Nach dem Anklicken des Buttons „neu TB erstellen“ wird die Benutzeroberfläche *„Tagesbericht.jsp“* geöffnet. In der Abbildung 4.5 wird das Sequenzdiagramm für diese Benutzeroberflächen beschrieben. Dieses Diagramm stellt dar, wie die in jQuery[4] und JavaScript[5] geschriebenen Funktionen in *„Tagesbericht.jsp“* aufgerufen werden. Alle Überprüfungsfunktionen werden in *„Tagesbericht.jsp“* mit JavaScript geschrieben. Diese Funktionen überprüfen zum Beispiel, ob ein Tagesbericht ausreichend vollständig ausgefüllt ist. Alle Berechnungsfunktionen werden in der *tagesbericht.js* in jQuery geschrieben, durch die die Tabellenkalkulations-Funktionen realisiert werden.

[4] http://jquery.com/
[5] http://www.w3schools.com/js/default.asp

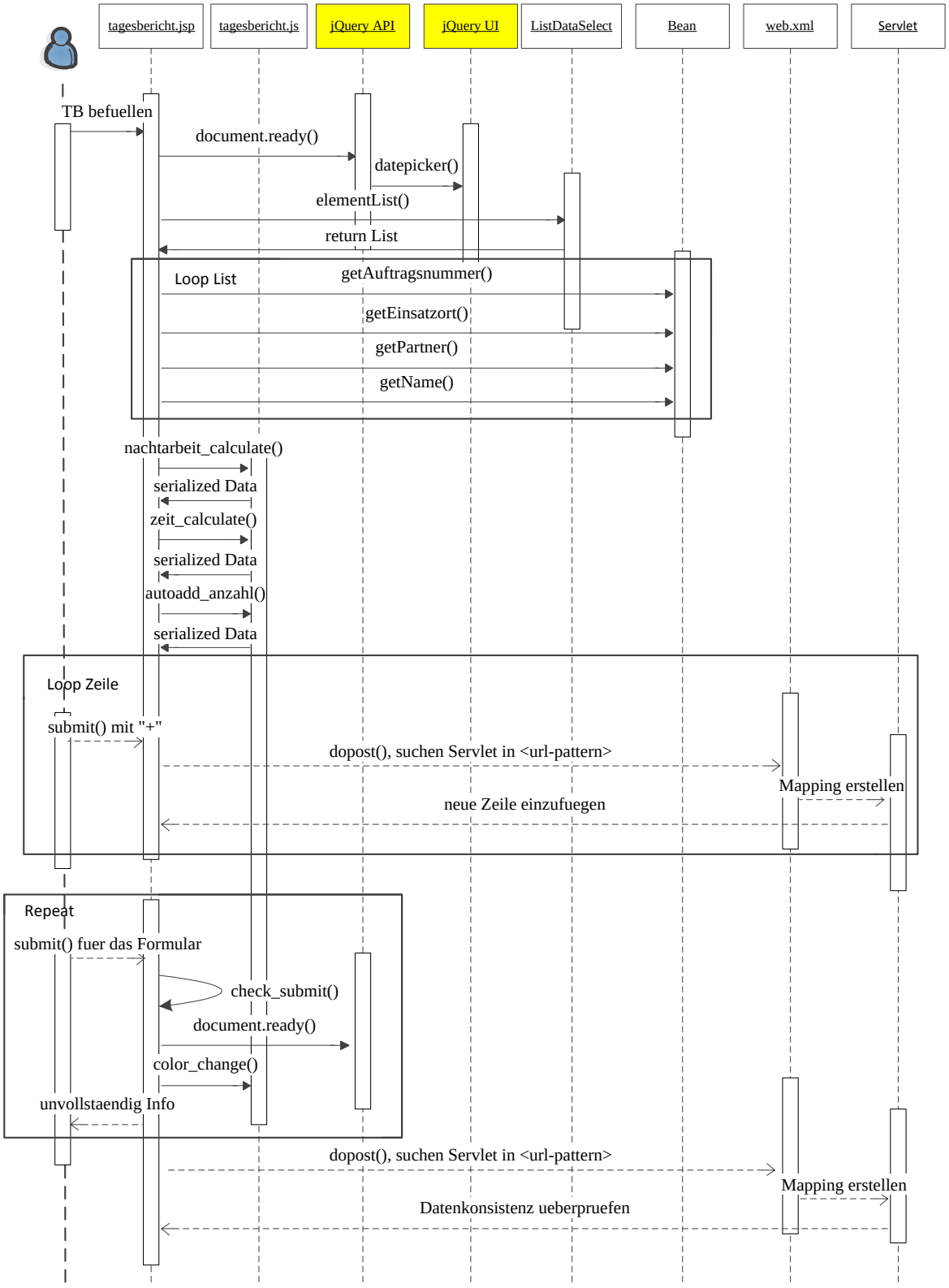

Abbildung 4.5: Sequenzdiagramm für die Funktionen der Benutzeroberflächen

4.3.1.1 Überprüfungsfunktion und Berechnungsfunktion

Das Datum wird nicht von der Tastatur eingegeben, sondern nur in einem Kalender ausgewählt. Dazu wird die Funktion `jQuery(document).ready()` aufgerufen, die diese Funktion beschreibt. Wenn Bilder, Tabellen und andere statistische Inhalte in der Seite fertig geladen sind, wird die Funktion `datepicker()` durchgeführt, die bei jQuery UI (User Interface) angeboten wird. In dieser Funktion können das Datumsformat und die Sprache definiert werden sowie Zeiträume beschränkt werden. Das Datumsformat wird als „D,dd,mm,yy“ bzw. der Wochentag, der Tag, der Monat, das Jahr definiert. In der Benutzeroberfläche *„Tagesbericht.jsp“* werden einige Pakete importiert, die die Funktion `datepicker()` anbieten. Diese Pakete werden im Anhang A.3 aufgeführt.

Auftragsnummern, Einsatzorte, Ansprechpartner und Mitarbeiternamen werden durch eine Unterliste ausgewählt. Wenn sie noch nicht in der Unterliste enthalten sind, können sie durch die Tastatur eingegeben werden. Durch Aufrufe der Klasse *ExcelListDataSelect* und des Pakets Bean werden die entsprechenden Daten von der Datenbank extrahiert. Dann werden diese Daten in den HTML-Elementen[6] eingefügt. Wenn also z. B. Auftragsnummern in dem <select> Tag eingefügt werden, wird eine For-Schleife verwendet. Die entsprechenden Auftragsnummern werden zwischen dem <option> Tag und dem </option> Tag eingegeben, womit alle Auftragsnummern in der Unterliste gespeichert werden.

Wenn die Daten nicht vollständig ausgefüllt werden oder einen Konflikt mit der Datenbank haben, werden die ausgefüllten Daten an die Benutzeroberfläche zurückgegeben und die entsprechenden Fehlermeldungen dargestellt. Weil es in dem <select> Tag kein Attribut *„value“* gibt, wird nach der Einreichung des Formulars eine Uniform Ressource Locator (URL) generiert. Eine URL beinhaltet alle Namen der HTML-Elemente und die entsprechenden Werte der HTML-Elemente von der aktuellen Seite *„Tagesbericht.jsp“*. Das heißt, in dem URL ist nur der Name des <select> Tages enthalten, das ausgewählte Datum muss in einem versteckten <input> Tag gespeichert werden. Die Funktion `jQuery(document).ready()` wird verwendet, um das ausgewählte Datum durch den Aufruf der Funktion `jQuery("\#Data option[value=" "]").attr("selected","selected")` an den <select> Tag zu übertragen. Der Wert des <select> Tages wird nach der Fehlermeldung an die Benutzeroberfläche *„Tagesbericht.jsp“* zurückgegeben.

In den Benutzeroberflächen müssen Daten von mehreren Tabellen berechnet werden. Durch den Aufruf der Funktion `nachtarbeit_calculate()` werden die erforderlichen Berechnungen durchgeführt. Durch diese Funktion werden die Berechnung der „Nacharbeit“, die Berechnung der „gesperrten n.i.O“, die Berechnung der „Fehlerhaften Teile“ und die Berechnung der „Stückzahl geprüft“ erfolgreich realisiert. Berechnungsergebnisse werden automatisch dargestellt. Wenn Daten von der Benutzeroberfläche eingetragen werden, ist der Typ der Eingaben ein String. In der Funktion `nachtarbeit_calculate()` muss der String in einen Integer umwandelt werden, erst dann kann diese Berechnungsfunktion richtig funktionieren. Bei der Berechnung der Arbeitsstunden muss die Funktion `spilt()` aufgerufen

[6] http://www.w3schools.com/html/default.asp

werden, weil Arbeitsstunden durch einen Doppelpunkt getrennt werden. Die getrennten Teile werden in einem Array gespeichert. Wenn all das erledigt ist, werden Ergebnisse in der Benutzeroberfläche angezeigt.

Die Funktion `autoadd_anzahl()` realisiert, dass in das Feld „Anzahl der Mitarbeiter“ automatisch „1“ eingetragen wird, wenn ein Mitarbeitername aus der Unterliste ausgewählt wird. Das Auswählen aus der Unterliste wird festgestellt, indem der Wert des <input> Tag des Mitarbeiternamens überprüft wird. Wenn der Wert nicht leer ist, wird der Wert „1“ am <input> Tag in das Feld „Anzahl der Mitarbeiter“ eingegeben.

In der Benutzeroberfläche gibt es einen Button „+“. Wenn dieser Button angeklickt wird, wird eine neue Zeile in der Tabelle „Aufwand/Umfang“ eingefügt. Um diese Funktion zu realisieren, muss ein Zähler erstellt werden. Der aktuelle Wert dieses Zählers wird in dem versteckten <input> Tag gespeichert. Nach dem Anklicken des Buttons „+“ wird die Benutzeroberfläche „*Tagesbericht.jsp*“ aufgerufen und zum aktuellen Wert des Zählers wird „1“ addiert. In der Benutzeroberfläche „*Tagesbericht.jsp*“ wird die Tabelle durch eine For-Schleife geschrieben. Die Obergrenze der Schleife ist die Anzahl der Zeilen und ist gleich der Gesamtanzahl des Zählers. Nach dem Aufruf wird die Obergrenze der Schleife erhöht und damit eine neue Zeile eingefügt.

In der Benutzeroberfläche werden die Eingaben überprüft, ob die Daten vollständig ausgefüllt sind oder einen Konflikt mit der Datenbank haben. Wenn die Vollständigkeit geprüft wird, wird ein Attribute „*onsubmit*“ in dem <form> Tag verwendet. Wenn das Formular aktiviert wird, wird die Funktion `check_submit()` zuerst aufgerufen. Diese Funktion überprüft, ob alle notwendigen Felder ausgefüllt oder redundante Werte eingetragen wurden. Falls alle Felder korrekt befüllt sind, werden die Daten in das Servlet übertragen. Andernfalls wird eine Fehlermeldung ausgegeben und die Konflikt-Felder durch den Aufruf der Funktion `color_change()` mit gelber Farbe markiert. Weiterhin wird die boolesche Variable „falsch“ zurückgeliefert und das Absenden abgebrochen. Die Überprüfung der Datenkonsistenz wird durch ein Servlet realisiert. In dem Servlet werden die eingetragenen Daten und die Daten, die in der Datenbanktabelle gespeichert sind, verglichen. Wenn zwei Datensätze gleich sind, werden die Zeilennummern der Tabelle „*Aufwand/Umfang*“ von dem Aufruf des Formulars in einen *Stringbuffer* gespeichert. Diese Nummern werden als die Werte des Parameters betrachtet und mit dem Namen des Parameters „*error*“ der URL hinzugefügt. Nach der Aktualisierung der Benutzeroberfläche „*Tagesbericht.jsp*“ kann ein verstecktes <label> Tag durch `request.getparameter()` diese Nummern aus der URL auslesen. Dann wird das Attribut „*onload*“ in dem <body> Tag verwendet und die Funktion `color_change()` wird aufgerufen. In dieser Funktion werden die Zeilennummern durch den Aufruf der Identifikation des <label> Tages bestimmt. Damit werden das Feld des Datums, der Auftragsnummer, des Ansprechpartners, das Feld des Einsatzorts und die Zeilen der Tabelle „*Aufwand/Umfang*“ mit roter Farbe markiert.

4.3.1.2 Funktionen für den Datenimport

In der Funktion „neu Tagesbericht erstellen“ wird ein eigenes Servlet „*AufwandServlet*“ verwendet. Die Abbildung 4.6 zeigt die Beziehung zwischen der Klasse *AufwandServlet* und anderen Java-Klassen an.

In der Klasse *Aufwandservlet* gibt es mit `doGet()` und `doPost()` zwei Funktionen. Diese zwei Funktionen beschreiben die am häufigsten verwendeten Requesttypen von http. In der Funktion `doGet()` sollen Ressourcen vom Server angefordert werden. Wenn z. B. ein Link des Browsers angeklickt wird oder eine URL in den Browser eingegeben wird, werden die entsprechenden HTML-Formulare zurückgegeben. Die Funktion `doPost()` wird für die Formularangabe verwendet. Diese Methode wird also benutzt, wenn die Daten in die Datenbank geschrieben oder in der Datenbank geändert werden. Weil die Funktion das *Aufwandservlet* für das Schreiben von Daten in die Datenbank ist, wird die Funktion `doPost()` verwendet.

Die Klasse Aufwandservlet ruft viele Klassen auf:

1. Die Klasse *Tagesbericht* und die Klasse *TagesberichtEintragung* definieren den Typ des Datums, der Arbeitszeit usw.
2. Die Klasse *TBDataInsert* beschreibt, wie Daten in die Datenbank geschrieben werden. In dieser Klasse gibt es zwei Funktionen. Durch den Aufruf der Funktion `insertNeuTB()` soll der Status des Tagesberichtes als „neu“ in die Datenbanktabelle geschrieben werden. Durch den Aufruf der Funktion `insertGenTB()` wird der Status des Tagesberichtes als „gen“ in die Datenbanktabelle geschrieben.
3. Die Klasse *IdSelect* bezeichnet, wie die Identifikation der Auftragsnummer, des Einsatzortes, des Mitarbeiternamens usw. von der Datenbank ausgewählt wird.
4. Die Klasse *TBNeuSelect* beschreibt, wie Daten eines Tagesberichtes von der Datenbank ausgewählt werden und der Status dieses Tagesberichts in der Datenbanktabelle als „neu“ gekennzeichnet wird. Beispielsweise ruft die Klasse *Aufwandservlet* die Funktion `getTBNeuId()` auf; damit wird die Identifikation des Tagesberichtes ausgelesen. Diese Identifikation wird später in der Datenbanktabelle „*TagesberichtEintragung*“ eingetragen.
5. Die Klasse *TBGenSelect* beschreibt, wie Daten eines Tagesberichtes von der Datenbank ausgewählt werden und der Status dieses Tagesberichts in der Datenbanktabelle mit „gen“ gekennzeichnet wird. Beispielsweise ruft die Klasse *Aufwandservlet* die Funktion `getTagesberichtGen()`. Daraufhin werden die Daten aller Tagesberichte in einer Liste gespeichert. Später werden sie mit den Eingabedaten verglichen, um die Datenkonsistenz mit der Datenbank zu prüfen.
6. Die Klasse *DbConnection* beschreibt, wie die Datenbankverbindung gehandhabt wird. Durch den Aufruf der Funktion `getDbConnection()` wird die Verbindung mit der Datenbank aufgebaut und durch den Aufruf der Funktion *close*() wird die Verbindung wieder gelöst.

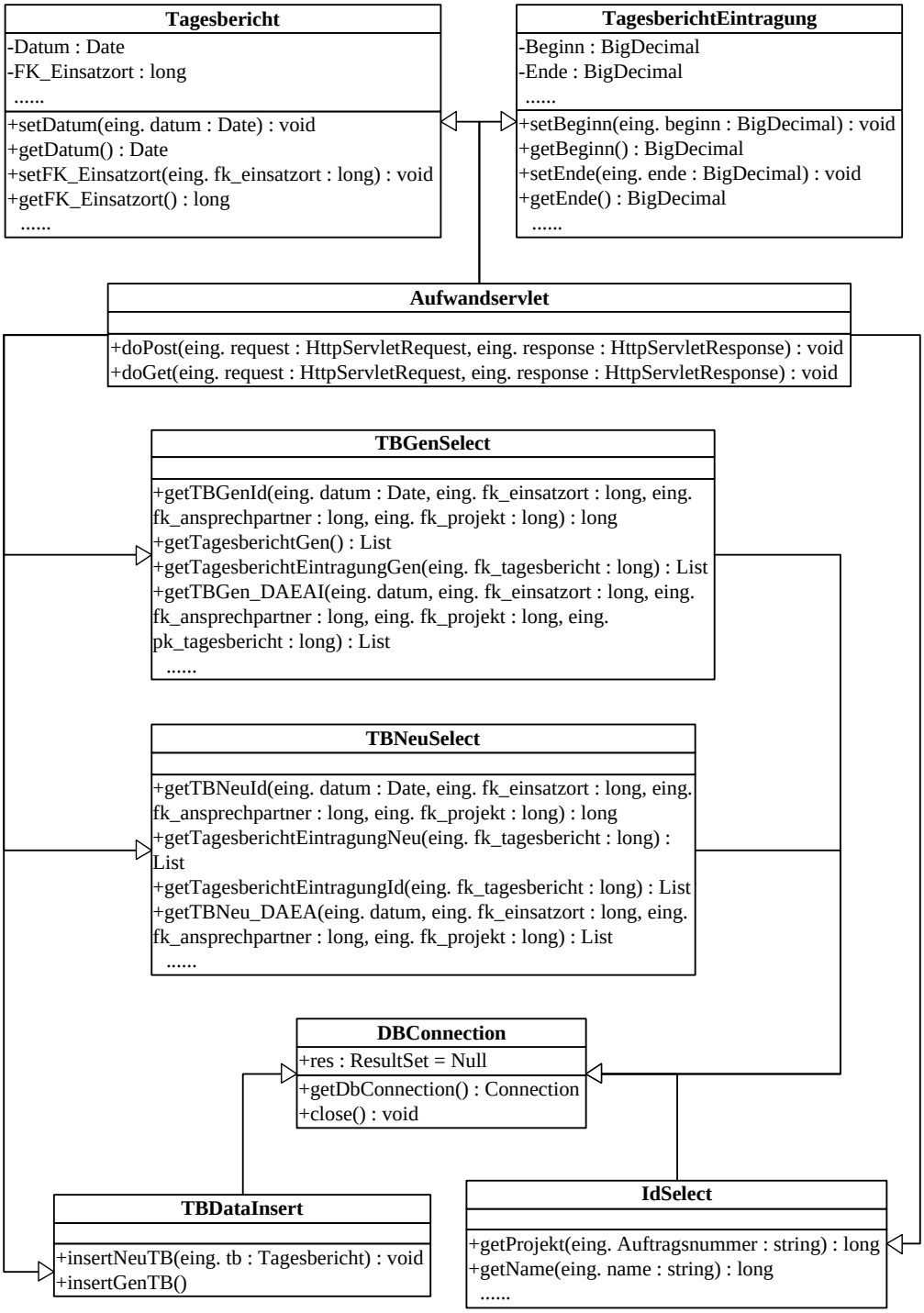

Abbildung 4.6: Klassendiagramm für die *AufwandServlet*

4.3.2 Funktionsweise für „gespeicherte Tagesberichte korrigieren“

4.3.2.1 Funktionen für die Datenauswahl

Wenn der Button „gespeicherte TB korrigieren“ angeklickt wird, wird eine neue Benutzeroberfläche „*Search_frame_neu.jsp*“ geöffnet. Diese Benutzeroberfläche wird durch den <frame> Tag in zwei Teile unterteilt. In dem oberen Teil „*Search_neu.jsp*“ sollen Auswahlbedingungen eingegeben werden, durch diese Bedingungen kann ein bestimmter Tagesbericht ausgewählt werden. Nach dem Anklicken des Buttons „Auswählen“ werden Auswahlergebnisse in dem unteren Teil „*Search_table.jsp*“ durch eine Tabelle aufgelistet. Um die Funktion des Buttons „Auswählen“ zu realisieren, soll ein eigenes Servlet „*SearchNeuServlet*“ verwendet werden. In der Abbildung 4.7 wird die Beziehung zwischen der Klasse *SearchNeuServlet* und den anderen Java-Klassen dargestellt.

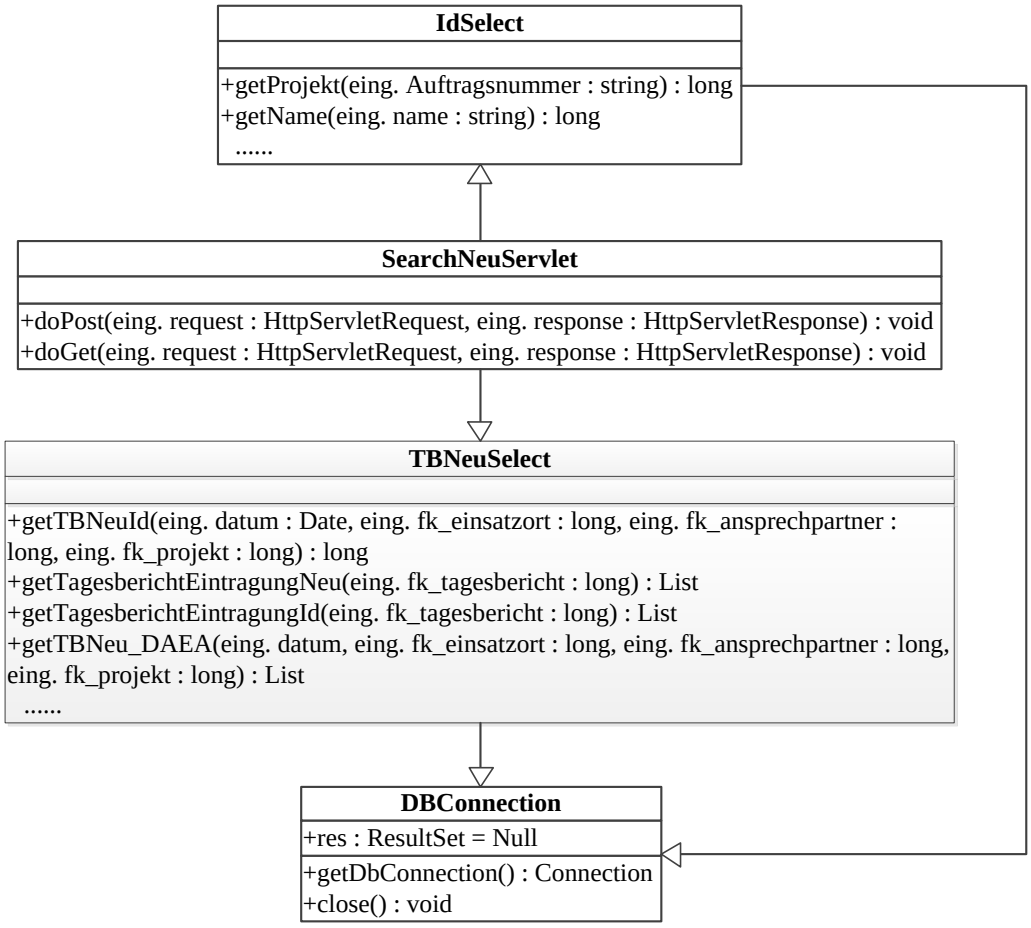

Abbildung 4.7: Klassendiagramm für die *SearchNeuServlet*

Die Klasse *SearchNeuServlet* hat mit `doGet()` und `doPost()` zwei Funktionen, die bereits in der Klasse *AufwandServlet* erklärt wurden. Ein Datensatz des Tagesberichts kann durch Eingaben unterschiedlicher Auswahlbedingungen in der Tabelle identifiziert werden. So gibt es z. B. in der Klasse *TBNeuSelect* eine Funktion `getTBNeu_DAEA()`, die durch das *SearchNeuServlet* aufgerufen wird. Diese Funktion beschreibt, wie ein Datensatz des Tagesberichts aus der Datenbank durch die Eingabe des Datums, der Auftragsnummer, des Ansprechpartners und des Einsatzorts ausgewählt werden kann. Es kann sein, dass ein Datensatz des Tagesberichtes bereits durch das Datum oder auch erst durch mehrere Bedingungen eindeutig identifizierbar ist.

4.3.2.2 Funktionen für die Datenaktualisierung

Wenn es nur einen einzigen Datensatz in der Tabelle gibt, kann der Button „Öffnen“ angeklickt werden. Damit werden Daten dieses Tagesberichts in der Benutzeroberfläche *„Tagesbericht_kor.jsp“* dargestellt. Um diese Daten zu prüfen, muss ein eigenes Servlet *„UpdateGenServlet“* verwendet werden. In der Abbildung 4.8 wird die Beziehung zwischen der Klasse *UpdateGenServlet* und den anderen Java-Klassen dargestellt.

Die Klasse *UpdateGenServlet* ruft viele Klassen auf. Die meisten Klassen wurden bereits in dem Kapitel 4.3.1 vorgestellt. Um die Daten eines Tagesberichts in der Datenbank zu aktualisieren, müssen die folgenden zwei Klassen aufgerufen werden.

Die Klasse *TBDataUpdate* beschreibt, wie Daten eines Tagesberichts in der Datenbanktabelle aktualisiert werden. Die Klasse *UpdateGenServlet* ruft zum Beispiel die Funktion `updateTagesberichtGen()` und die Funktion `updateTagesberichtEintragung()` auf. Zuerst wird der Status des Tagesberichtes von „neu“ zu „gen“ geändert. Danach werden die Arbeitszeiten, die Mitarbeiternamen usw. von der Datenbanktabelle *„TagesberichtEintragung“* aktualisiert. Die Auftragsnummer, der Einsatzort, der Ansprechpartner und das Datum sind die Identifikationsmerkmale des Tagesberichts und dürfen deshalb nicht geändert werden. Aus diesem Grund sind diese vier Daten in der Benutzeroberfläche *„Tagesbericht_kor.jsp“* schreibgeschützt. Wenn auch nur eine davon falsch eingegeben wurde, muss der betreffende Tagesbericht komplett aus der Datenbank gelöscht werden.

Die Klasse *TBDataDelete* bezeichnet, wie die Daten eines Tagesberichts aus der Datenbanktabelle gelöscht werden. Durch den Aufruf der Funktion `deleteTBNeu()` werden alle Daten des Tagesberichtes von der Datenbank gelöscht, wenn der Status dieses Tagesberichtes „neu“ ist. Jeder Tagesbericht hat eine Identifikation. Wenn die Daten eines Tagesberichts in die Datenbank geschrieben werden, wird diese Identifikation in der Datenbanktabelle *„Tagesbericht“* als Primärschlüssel automatisch inkrementiert. Wenn diese Identifikation von der Datenbanktabelle gelöscht wird, werden alle Daten dieses Tagesberichtes aus der Datenbank gelöscht. Die Mitarbeiternamen und deren Arbeitszeiten werden durch den wiederholten Aufruf der Funktion „`deleteTagesberichtEintragung()`“ jeweils zeilenweise aus der Datenbanktabelle *„TagesberichtEintragung“* gelöscht.

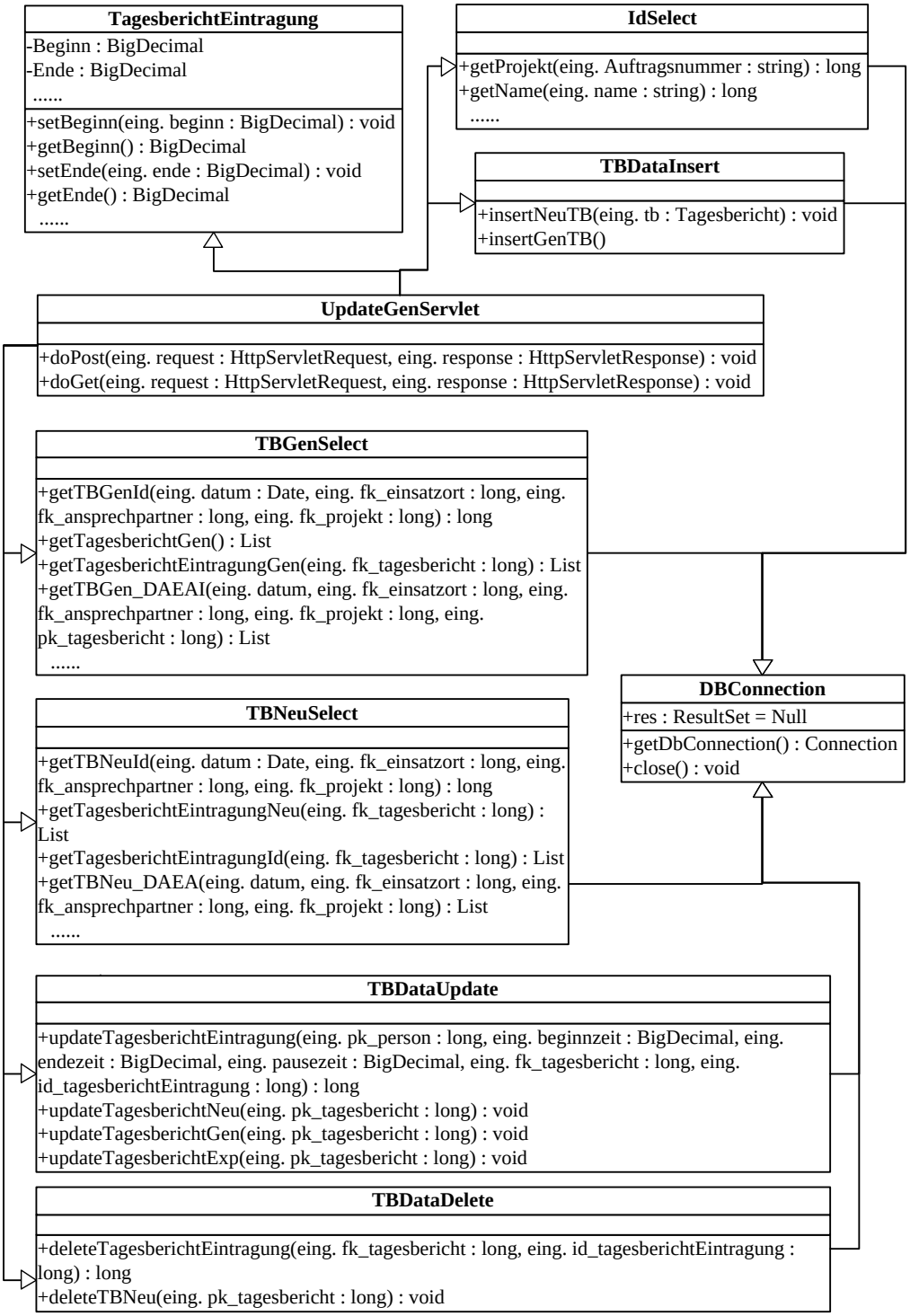

Abbildung 4.8: Klassendiagramm für die *UpdateGenServlet*

4.3.3 Funktionsweise für „freigegebene Tagesberichte korrigieren“

Wenn der Button „gespeicherte TB korrigieren“ angeklickt wird, wird die Benutzeroberfläche „*Search_frame_gen.jsp*“ geöffnet. Diese Benutzeroberfläche ist ähnlich wie die Benutzeroberfläche „*Search_frame_neu.jsp*“. Durch den <frame> Tag wird sie in zwei Teile unterteilt. In dem oberen Teil „*Search_gen.jsp*“ müssen die Auswahlbedingungen eingegeben werden. Durch diese Bedingungen kann ein bestimmter Tagesbericht ausgewählt werden, der den Status „gen“ hat. Nach dem Anklicken des Buttons „Auswählen“ werden Auswahlergebnisse in dem unteren Teil „*Search_table.jsp*“ in einer Tabelle aufgelistet. Wenn in der Tabelle nur noch ein einziger Datensatz ist, kann der Button „Öffnen“ in dem Teil „*Search_gen.jsp*“ angeklickt werden, Daten über diesen Datensatz werden in der Benutzeroberfläche „*Tagesbericht_frei.jsp*“ angezeigt.

Der Tagesbericht wird schließlich in Portable Document Format (PDF) Dateien umwandelt, wodurch der Tagesbericht direkt ausgedruckt und auf dem Rechner gespeichert werden kann. In der Tat kann die Benutzeroberfläche durch geeignete Software direkt in eine PDF-Datei konvertiert werden, z. B. mit Adobe Acrobat XI[7] oder PDFCreator[8]. Diese Methode ist nicht gut geeignet, weil es in vielen Schritten durchgeführt werden muss. Soll beispielsweise die Webseite zuerst gespeichert werden, dann wird erst die Software geöffnet und anschließend die Webseite zu PDF konvertiert. Wenn auch nur ein einziger Schritt Fehler aufweist, muss die ganze Arbeit wiedergeholt werden. Eine alternative Methode ist, aus der gesamten Benutzeroberfläche durch ein geeignetes Programm automatisch eine PDF-Datei zu generieren. Weil die Benutzeroberfläche basierend auf HTML geschrieben wird, kann iText[9] verwendet werden. Die Programmbibliothek iText dient zur dynamischen Verarbeitung von PDF-Dateien. Das zugehörige Projekt wurde von Bruno Lowagie an der Universität Gent entwickelt [Low10]. Von der Webseite iText[10] kann die Programmbibliothek itextpdf-5.3.2 heruntergeladen werden.

In der Benutzeroberfläche „*Tagesbericht_fei.jsp*“ gibt es einen Button „PDF erstellen“. Durch den Aufruf des Servlet „*PDFServlet*“ kann die Benutzeroberfläche direkt in eine PDF-Datei konvertiert werden. In Abbildung 4.9 wird das Klassendiagramm der *PDFServlet* dargestellt.

Weil die generierte PDF-Datei nicht auf dem Server gespeichert wird, wird die Funktion `doPost()` verwendet. Die Benutzeroberfläche kann in sechs Tabellen unterteilt werden. Titel des Tagesberichtes z. B. werden durch die eigene Funktion `createTable1()` erstellt. Durch die eigene Funktion `createTable2()` wird eine Tabelle für das Datum, die Auftragsnummer usw. erstellt.

Die Abbildung 4.10 zeigt, wie das PDF-Dokument durch das Programm iText erstellt wird.

[7] http://www.adobe.com/de/products/acrobatstandard.html
[8] http://sourceforge.net/projects/pdfcreator/
[9] http://itextpdf.com/
[10] http://itextpdf.com/download.php

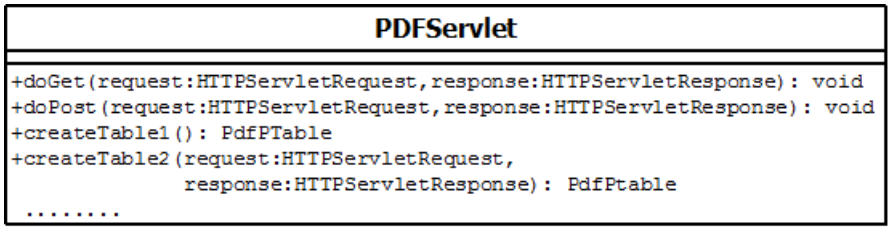

Abbildung 4.9: Klassendiagramm für die *PDFServlet*

Im ersten Schritt werden ein Dokument und ein *ByteArrayOutputStream* deklariert. Dann werden die Zeiger auf diese Datenelemente vom sogenannten *PdfWriter* an die Schreibe-Funktion übergeben. Danach werden die Bilder und Tabellen von der Schreibe-Funktion erstellt und im *ByteArrayOutputStream* abgelegt.

Im zweiten Schritt wird eine API Image verwendet. Durch den Aufruf der Funktion `getInstance()` wird eine Instanz eines Bildes bereitgestellt. Die Instanz eines Bilds ist eine Art Zeiger auf das Bild. Dann wird die Funktion `setAbsoluteposition()` und die Funktion `scaleAbsolute()` angewendet, die die Position des Bildes und die Größe des Bildes in dem PDF Dokument festlegt.

Im dritten Schritt wird ein API *Document* verwendet, ein neues Dokument wird erstellt und geöffnet und Bilder in das Dokument eingefügt.

Im vierten Schritt wird die eigene Funktion `createTable()` aufgerufen, um Tabellen zu erstellen. In dem Programm gibt es insgesamt sechs eigene Funktionen für die Erstellung der Tabelle, die einander sehr ähnlich sind. Beispielsweise wird in der Funktion `createTable()` die API *PdfPTable* aufgerufen. Durch den Aufruf der Funktion `setHorizontalAlignment()` wird die horizontale Ausrichtung der Table in dem PDF-Dokument festgelegt und durch den Aufruf der Funktion `setTotalWidth()` wird die Breite der Tabelle aus der absoluten Spaltenbreite bestimmt. Die Funktion `setLockedWidth()` ist eine boolesche Funktion. Wenn deren Wert „wahr“ ist, wird die Grenze der Tabelle festgelegt. Durch den Aufruf der Funktion `setWidths()` werden die relativen Breiten der Tabelle bestimmt. Nach der Erstellung der Parameter der Tabelle werden die Zellen der Tabelle erstellt. Die Methode `setHorizontalAlignment()` legt die horizontale Ausrichtung der Zelle an der Tabelle fest. Durch den Aufruf der Funktion `disableBorderSide()` wird die Grenze auf der angegebenen Seite deaktiviert. Die Funktion `setBackgroundColor()` und die Funktion `setBorderColor()` legen die Hintergrundfarbe der Zelle und die Farbe der Grenze der Zelle fest. Durch den Aufruf der Funktion `setFixedHeight()` wird die Höhe der Zelle festgelegt. Wenn in der Tabelle eine Zelle aus zweier kleine Zellen besteht, wird die Funktion `setColspan()` verwendet, um die Zellen zusammenzusetzen. Schließlich werden die Zellen durch den Aufruf der Funktion `add()` zur der Tabelle addiert und die erstellte Tabelle zurück an das Servlet gesendet.

Im fünften Schritt wird die Funktion `writeSelectedRows()` aufgerufen, wodurch die Tabellen mit der absoluten Position in dem PDF-Dokument dargestellt werden.

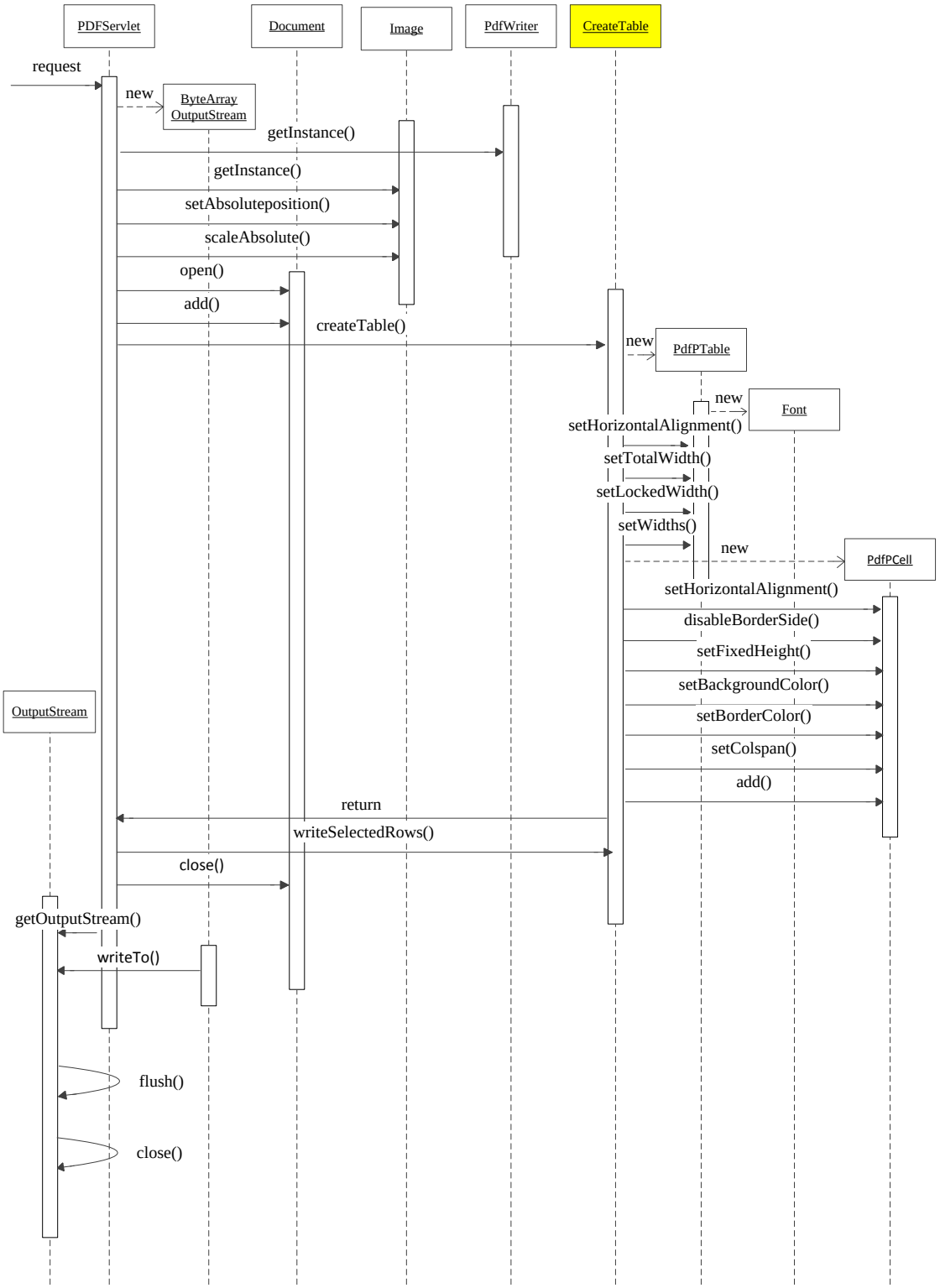

Abbildung 4.10: Sequenzdiagrmm für die Generierung eines PDF-Dokumentes

In der Benutzeroberfläche *„Tagesbericht_frei.jsp“* werden zwei Servlets aufgerufen. Eine ist das *PDFServlet* und die andere das *UpdateExpServlet*. Durch den Aufruf der Funktion `document.forms[0].action = ,,‘‘` kann die Durchführung der Servlets gesteuert werden.

4.3.4 Funktionsweise für „freigegebene Tagesberichte exportieren“

Nach dem Anklicken des Buttons „gespeicherte TB exportieren“ wird die Benutzeroberfläche *„Search_frame_exp.jsp“* geöffnet. Diese Benutzeroberfläche wird durch den <frame> Tag wird in zwei Teile unterteilt. Durch die eingegebenen Auswahlbedingungen werden die Daten nach dem Anklicken des Buttons „Excel öffnen“ in die Gesamtrechnung exportiert. Um Daten in das Excel 2003 Format zu exportieren, wird das Servlet *„ExcelServlet“* aufgerufen, das in Abbildung 4.11 dargestellt ist.

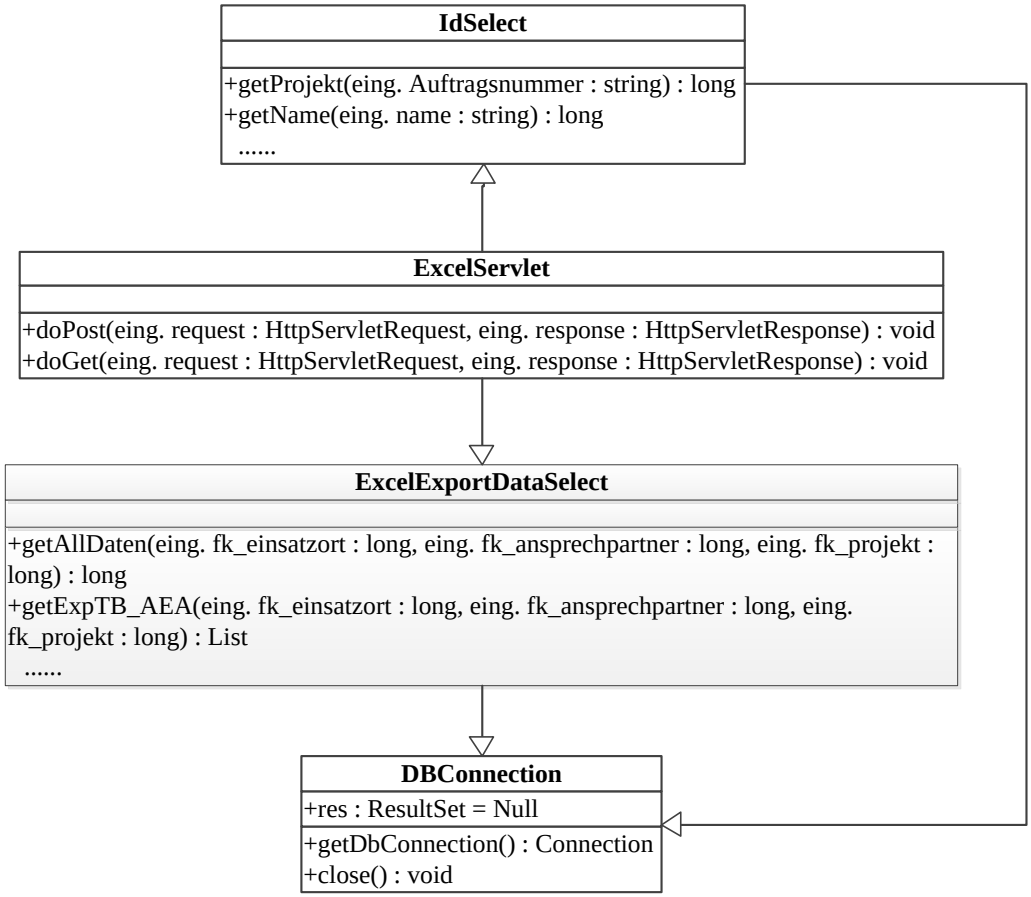

Abbildung 4.11: Klassendiagramm für die *ExcelSevlet*

Die Klasse *ExcelServlet* ruft die Klasse *IdSelect* und die Klasse *ExcelExportDataSelect* auf. In der Klasse *ExcelExportDataSelect* werden die Daten der Tagesberichte, die mit dem Status „exp“ bezeichnet sind, durch den Aufruf der Funktion `getAllDaten()` in einer Liste gespeichert. Die Daten dieser Liste werden später nach Excel exportiert. Das Prinzip des Exports nach Excel wurde bereits in dem Kapitel 4.2.2 vorgestellt.

5 Process Engine

In diesem Kapitel wird ein Konzept für eine gesamtheitliche Prozess-Steuerung der Projektabwicklung entwickelt, die jedoch in einzelnen Projektabschnitten auch eine Dokumenten-Steuerung zulässt. Ein wesentliches Werkzeug dafür ist die gestufte Anwendung von Process Engines und Servlet Engines, wodurch deutlich mehr Flexibilität beim Datenhandling erreicht wird.

5.1 Eigenschaften der Process Engines

Zur strikten Steuerung von Prozess-Abläufen, siehe Kapitel 1, eignen sich sogenannte Process Engines (PE). Diese Process Engines verwalten dabei den jeweiligen Projektstatus und schalten den Projektfluss um jeweils einen Schritt weiter, wenn eine entsprechende Aktivität erfolgreich abgeschlossen wurde. Die Anwendung einer solchen Process Engine auf die Projektabwicklung in der ISAG wurde in der Diplomarbeit „Automatisierung der Projektabwicklung in Unternehmen“ [Bau12] bereits untersucht. Dabei stellten sich mehrere Schwierigkeiten heraus. Die aktuell verfügbaren Process Engines können zwar die Projektstatusdaten gut verwalten, haben jedoch große Defizite bei der Kommunikation mit der Projektdatenbank. Zudem sind manche Aufgaben, zum Beispiel das Eintragen von Tagesberichtdaten in die Datenbank, nur möglich, wenn die entsprechenden Tagesberichte auch vorliegen. Somit ist diese Tätigkeit prinzipiell Dokumenten-gesteuert. Diese Aufgabe in einen für die Process Engines geeigneten und somit Prozess-gesteuerten Ablaufplan zu übersetzen, ist sehr aufwändig und erzeugt sehr komplexe Ablaufpläne. Ebenso aufwändig ist es, die Dokumenten-Steuerung so zu erweitern, dass die strikte Einhaltung der obengenannten, vorgegebenen Reihenfolge bei der Abarbeitung der Prozessschritte gewährleistet ist. Die Realisierung einer Kombination aus Dokumenten- und Prozess-Steuerung bei der Projektabwicklung ist also aus diesen Gedankengängen heraus nicht möglich.

5.2 Konzept einer gestuften Anwendung von Process Engines und Servlet Engines

Der Bruch mit der bisherigen und etablierten Vorgehensweise der Process Engines, nämlich die Projektstatusdaten in der Process Engine zu verwalten, eröffnet neue Möglichkeiten. Zwar wird ein Stück der Kontrolle der Process Engines an die Projektdatenbank abgegeben,

jedoch gewinnt man die wertvolle Möglichkeit hinzu, einige Projektaufgaben Dokumenten-gesteuert durchzuführen. Dies vereinfacht die erforderlichen Prozessablaufpläne erheblich und gestattet z. B. die Dokumenten-gesteuerte Abarbeitung der Tagesberichte, obwohl die Kontrolle der weiteren Prozessschritte weiterhin bei der Process Engine liegt. Für die Umsetzung dieser neuen Vorgehensweise gibt es jedoch noch keine geeigneten Konzepte in der Literatur, weshalb die erforderlichen Konzepte in dieser Diplomarbeit entwickelt werden.

Die bisher erstellten Benutzeroberflächen sind, wie im vorigen Kapitel erläutert, als Webformulare realisiert, die zusammen mit den zugehörigen Konnektoren auf dem Web-Client ausgeführt werden. Die Benutzeroberflächen dienen dem Informationsaustausch zwischen dem Web-Client und dem Benutzer und werden durch die Servlet Engine verwaltet.

In BPMN wird ein Prozessmodell ganz allgemein als eine Gruppe von Aktivitäten definiert. Die Geschäftsprozesse sind somit als Gruppen von Aktivitäten in einem BPMN-Diagramm graphisch modellierbar. Das BPMN-Diagramm beschreibt dabei nicht nur die logischen Beziehungen zwischen den Prozessen, sondern stellt auch den Prozessablauf dar. Die Geschäftsprozesse werden in dieser Realisierung durch eine Process Engine gesteuert. Die Bearbeitung einer Benutzeroberfläche kann dabei als ein Prozessschritt betrachtet werden. Dieser Prozessschritt wird somit abgeschlossen, wenn die zugehörige Benutzeroberfläche geschlossen wird. Die Abbildung 5.1 zeigt uns die für die obengenannte gemeinsame Speicherung der Projekt- und Projektstatusdaten in der Projektdatenbank erforderliche Kommunikation zwischen den zwei Engines, also der Servlet Engine und der Process Engine, sowie den Benutzern.

Hierzu werden die Process Engine und die Servlet Engine zusammenpassend aufgebaut. Die Aktivitäten werden direkt in Formularen bzw. Webformularen definiert. Durch die Bearbeitung der Formulare oder Webformulare werden die Daten zwischen den einzelnen Aktivitäten übermittelt. Auf diese Weise werden die Geschäftsprozesse durch die „Process Engine“ ausgeführt, wobei die Projektdaten und die Projektstatusdaten jedoch zentral in der Projektdatenbank gespeichert werden.

5.3 Auswahl der Process Engine

In der Diplomarbeit „Automatisierung der Projektabwicklung in Unternehmen“ [Bau12] werden unterschiedliche Werkzeuge für die Prozessmodellierung vorgestellt und miteinander verglichen. *Bonitasoft*, *Signaviol* und *Aktiviti* stellen sehr geeignete Werkzeuge dar, um Prozessmodelle zu erstellen und Webformulare für die entsprechenden Aktivitäten zu implementieren. Durch die Bearbeitung der Webformulare werden die Geschäftsprozesse strikt sequenziell durchgeführt. Weil die Steuerung der Geschäftsprozesse und die Ausführung der Benutzeroberflächen verschiedene Engines verwenden, die möglichst reibungslos miteinander arbeiten sollen, werden an die „Process Engine“ folgende Anforderungen gestellt:

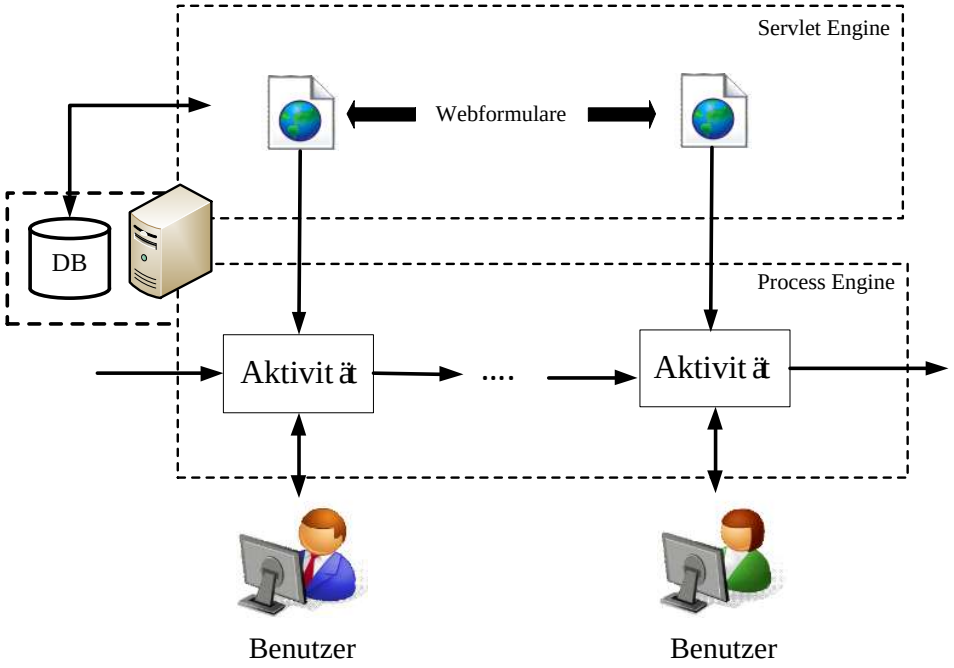

Abbildung 5.1: Beziehung zwischen den zwei Engines und den Benutzer

- Verfügbarkeit: Bietet die PE viele API-Pakete für die Entwicklung und die Änderung von Prozessanwendungen. Das API Technologien zur Verfügung [Rü12, S.88], z. B. Java, Web Service?
- Einbettbarkeit: Kann die PE in verschiedenen Betriebssystemen betrieben werden, z. B. Windows, Linux? Kann die PE mit einer anderen Engine zusammen aufgebaut werden, z. B. Tomcat, Java-Engine?
- Steuerbarkeit: Können die Webformulare oder andere Formulare in die Definition der Aktivitäten einbezogen werden?
- Integrierbarkeit: PE bietet viele Verbindungen mit anderen Services an und kann eine Web-Service bei einer Aktivität aufgerufen werden?
- Testbarkeit: Wenn Prozessmodelle fertig erstellt werden, können für die erstellten Prozessmodelle dann Testfälle implementiert werden? Kann damit die Prozesslogik überprüft werden, z. B. die Überprüfung der Prozesslogik mit dem Modultest bzw. *Unit test*?
- Komplexität: Kann die PE einfach konfiguriert und installiert werden?

- Benutzbarkeit: Unterstützt die PE die BPMN 2.0 Notation? Und kann bei dieser PE die jBPM Process Definition Language (JPDL) oder die XML Process Definition Language (XPDL) eingesetzt werden?

In dieser Diplomarbeit werden die zwei etablierte Open Source Process Engines untersucht, die Bonita Open Solution und die Aktiviti.

5.4 Servlet Engine

Apache Tomcat[1] ist eine Open-Source-Software, die eine Umgebung anbietet, dass Java Server Page in Servlet übersetzt und ausgeführt werden kann. Apache Tomcat wurde von der Apache Software Foundation im Jahr 1999 entwickelt. Die verwendete Version ist 6.0.36 und wird hier als Open-Source Apache-Lizenz genutzt.

5.5 Bonita Open Solution

Bonitasoft ist ein Open-Source-Softwarepaket für die Entwicklung der *Business process modeling* (BPM) und *Workflow*, das eine Process Engine anbietet, mit der der Benutzer die Geschäftsprozesse definieren, ausführen und überwachen kann. BonitaSoft[2] wird beim BonitaSoft S. A. entwickelt und besteht aus 3 Teilen:

- Bonita Studio: um Geschäftsprozesse grafisch zu konstruieren.
- Bonita Execution Engine: um Geschäftsprozesse auszuführen.
- Bonita User Experience: um Geschäftsprozesse zu verwalten.

Die Lizenz ist Licensed under GNU General Public License v2 (LGPL), die aktuelle Version ist Bonita Open Solution (BOS)-5.9.

5.5.1 Konfiguration der Process Engines

Um die erforderlichen Aktivitäten in die Webformular zu implementieren, müssen BOS und der Apache Tomcat Server zusammen aufgebaut werden. Weil die Webformulare in JSP geschrieben werden, werden sie in dem Apache Tomcat Server aufgerufen und verwaltet. Die Prozessausführung wird durch die Bonita Execution Engine gesteuert.

BOS ist eine komplette Open Source BPM Solution mit vielen zusätzlichen Schnittstellen. Unter anderem kann das BOS-Tomcat Paket von der Webseite[3] heruntergeladen werden. Dieses Paket gibt es in der Version BOS 5.8 und Tomcat 6.0.35, es enthält also bereits

[1] http://tomcat.apache.org/index.html
[2] http://www.bonitasoft.com
[3] http://www.bonitasoft.com/products/BPM_downloads

BOS und Tomcat als gemeinsames Paket. Es ist also nicht erforderlich, die zwei Engines speziell zu konfigurieren, sondern es müssen nur einige Variablen in dem File und auf dem System festgelegt werden. Die Konfiguration wird im Anhang A.1 dargestellt. Nach der Konfiguration zweier Engines wird die Beziehung zwischen Benutzer und neue Engine in der Abbildung 5.2 dargestellt.

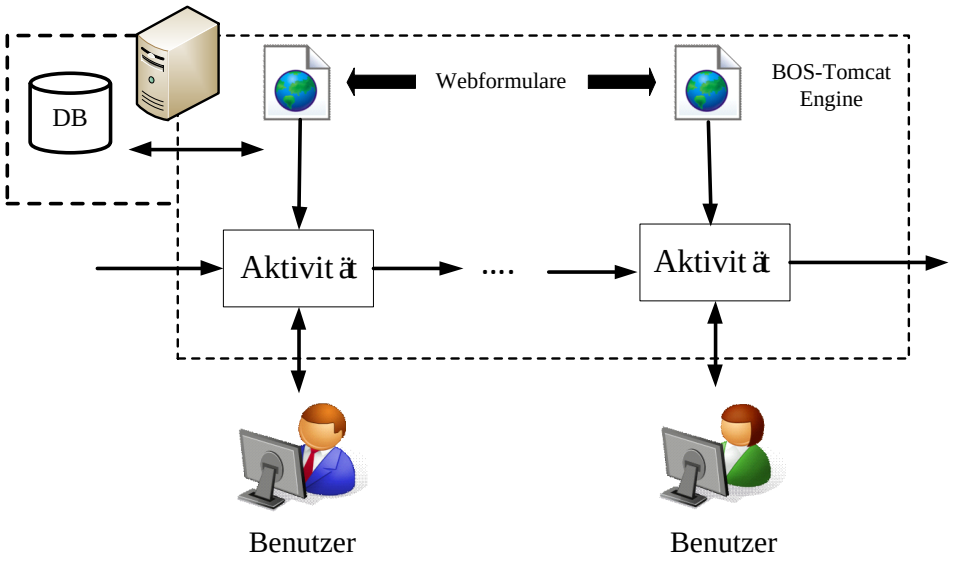

Abbildung 5.2: Beziehung zwischen BOS-Tomcat Engine und Benutzer

Weil BOS und Tomcat bereits als gemeinsames Paket bereitgestellt werden, ist die Einbettbarkeit dieser Process Engine positiv und die Komplexität der Konfiguration des BOS-Tomcat Pakets ist ebenfalls relativ gering.

5.5.2 Darstellung eines Beispiel-Prozessmodells durch Bonita Open Solution

Der Prozess der Bearbeitung des Tagesberichtes wird hier in Bonita Studio modelliert. Bonita Studio unterstützt BPMN 2.0. In der Arbeitspalette können deshalb die BPMN 2.0 Elemente einfach in dem Pool gezogen werden. Prozessmodelle, die durch XPDL dargestellt werden, können ebenfalls in Bonita Studio importiert werden. Die Benutzerfreundlichkeit ist also positiv. In der Abbildung 5.3 wird dieses Prozessmodell dargestellt.

Jede Bearbeitungsvariante des Tagesberichtes kann als ein Prozessschritt betrachtet werden. Der Benutzer kann einen neuen Tagesbericht erstellen, einen ausgewählten Tagesbericht überprüfen, korrigieren oder freigeben sowie einen ausgewählten Tagesbericht exportieren.

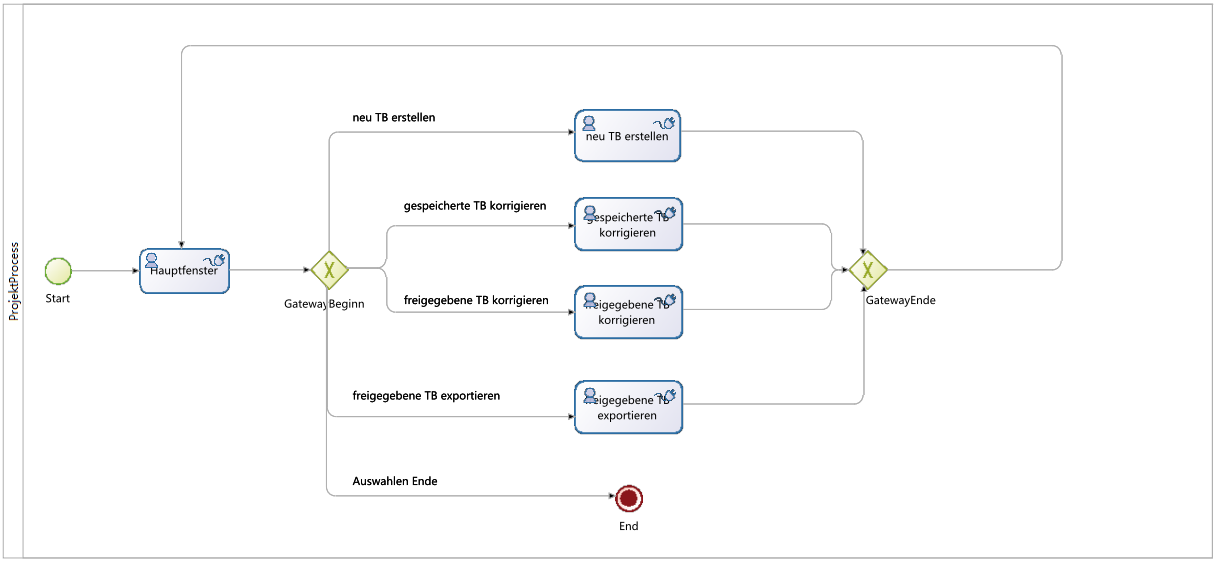

Abbildung 5.3: Beispielprozessmodell für die Bearbeitung des Tagesberichtes

5.5.3 Implementierung der Webformulare

Die Benutzeroberfläche wird durch JSP geschrieben, wobei die Webformulare in den **.jsp*-File definiert werden. Bonita Studio bietet einen Formbuilder genannten Formulargenerator, durch den die Formulare einfach erstellt werden können. Durch URL können dynamische Webseiten, die mit JSP erstellt werden, mit BOS verlinkt werden. URL bezeichnet dabei die Webadresse. Durch URL können diese Webseiten identifiziert und auf sie zugriffen werden. Jedes **.jsp*-File kann auf einer Webseite abbilden, wenn die zugehörige URL im Webbrowser eingetragen ist. Wird das entsprechende **.jsp*-File geöffnet, wird die durch das File definierte Benutzeroberfläche dargestellt. In dem Formbuilder gibt es ein Frame-Widget. Die durch das IFrame-Widget definierten Felder können Web-Adressen, sogenannte Links, als Variable gespeichert werden [BOS]. Die Abbildung 5.4 zeigt, wie ein Link in dem IFrame-Widget definiert werden kann.

In der Kategorie General gibt es die Option Data, in der die eingebettete URL dem Feld *URL of the iFrame* übergeben wird. Damit wird für jede Aktivität eine entsprechende Webseite abgebildet. Die Abbildung 5.5 veranschaulicht den Zusammenhang zwischen der Aktivität und der Webseite.

Weil BOS auch den Designer für die grafische Darstellung des Prozessmodells anbietet und die Formulare einfach in der Aktivität definiert werden können, ist die Benutzerfreundlichkeit sehr hoch.

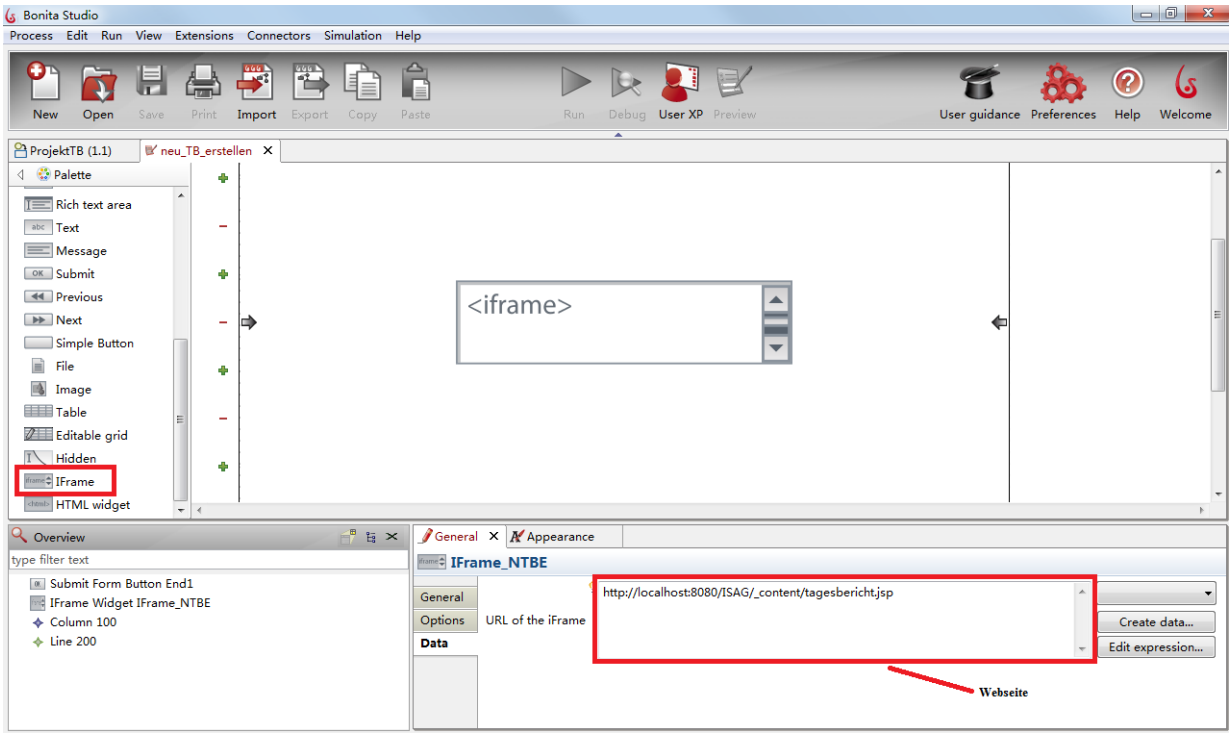

Abbildung 5.4: Definition eines Links in dem IFrame-Widget

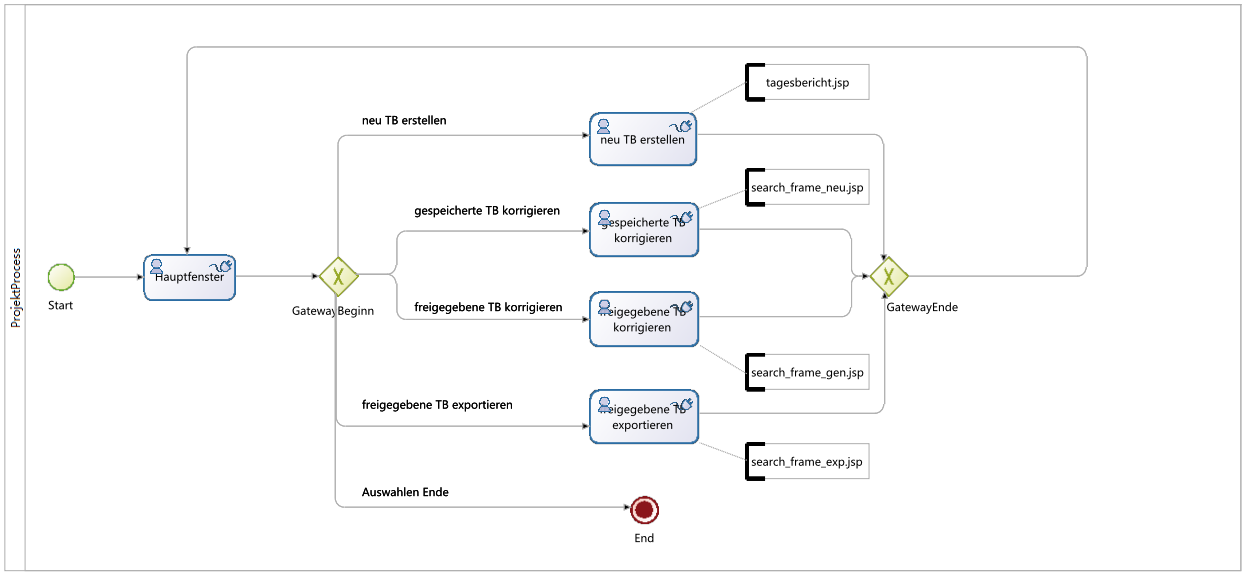

Abbildung 5.5: Abbildung zwischen den Aktivitäten und den Webseiten

5.5.4 Ausführung der Geschäftsprozesse

Um die Geschäftsprozesse auszuführen zu können, müssen die Konnektoren und Variablen definiert werden. Ein Konnektor verbindet einen Prozessschritt oder den Prozess mit externen Informationssystemen [BOS11]. Die Funktion der Variable ist es Informationen zu speichern. In BOS können Variablen als global oder lokal definiert werden. Globale Variablen können im ganzen Prozess verwendet werden, lokale Variablen nur in einer bestimmten Aktivität, in anderen Aktivitäten jedoch nicht. In dem Beispielsprozess wird die Variable als eine Liste definiert, in der es folgende Elemente gibt:

- neuen_TB_erstellen
- gespeicherten_TB_korrigieren
- freigegebenen_TB_korrigieren
- freigegebenen_TB_exportieren
- Beenden

Die fünf Elemente bezeichnen die unterschiedlichen Bearbeitungsvarianten eines Tagesberichts. Weil sowohl der Konnektor als auch die XOR-Gateway diese Variable aufrufen, muss diese Variable als global definiert werden. Die vorher beschriebene Variable soll den Prozessschritt definieren. Bei den von BOS angebotenen Konnektoren gibt es unter dem Konnektor-Typ Bonita den Konnektor *Set Variable*. Durch diesen Konnektor wird die Variable mit dem Prozessschritt verbunden. Wenn der Konnektor dann im Prozess ausgeführt wird, wird diese Variable verwendet [BOS11].

Außerdem bietet BOS einen Konnektor für die Verbindung der Web-Services an, mit dem die Aktivitäten durch Web-Services aufgerufen werden können. Die Integrierbarkeit der BOS ist somit positiv. Durch den Groovy-Editor wird die Bedingung der XOR-Gateway erstellt. Wenn z. B. die Bedingung „*neu_TB_erstellen*“ wahr ist, wird der Aktivität „*neu TB erstellen*“ ausgeführt. Die Abbildung 5.6 zeigt die Oberfläche der Aktivität Hauptfenster.

Es gibt kleine Unterschiede zwischen der Oberfläche der Aktivität und der Benutzeroberfläche, die in JSP programmiert wird. In der Benutzeroberfläche *HauptprogrammFenster.jsp* ist der Typ des Buttons „*button*“. In dem Event „*onclick*“ wird die Funktion `window.open()` ausgeführt, die die entsprechende Webseite nach dem Anklicken des vorgenannten Buttons geöffnet. In der Oberfläche der Aktivität ist der Typ des Buttons hingegen „*radio*“. Durch die Auswahl des betreffenden Elementes, das in der globalen Variablen festgelegt ist, wird dann die entsprechende Aktivität ausgeführt. Die Abbildung 5.7 zeigt, dass nach der Auswahl „*neuen TB erstellen*“ die Aktivität „*neuen TB erstellen*“ ausgeführt wird.

Die Oberfläche der Aktivität „*neu TB erstellen*“ ist identisch zur Benutzeroberfläche „*Tagesbericht.jsp*“. Die Funktionen, die in der Benutzeroberfläche des Kapitels 4 beschrieben und implementiert, also als Programmkode realisiert wurden, können somit nahezu unverändert in der Oberfläche der Aktivität laufen.

Abbildung 5.6: Oberfläche der Aktivität *Hauptfenster*

5.5.5 Verwaltung der Geschäftsprozesse

In der Bonita User Experience kann der Verlauf analysiert werden. Der betreffende Administrator kann zum Beispiel sehen, wer welche Prozesse durchgeführt hat oder wie oft der jeweilige Prozess aufgerufen wurde. Der Bonita-Posteingang listet dazu die abgeschlossenen Schritte auf. In der Abbildung 5.8 wird das Ergebnis eines Prozesslaufes dargestellt.

5.5.6 Überprüfung der Prozesslogik

In BOS wird die Prozesslogik automatisch überprüft. Falls das dargestellte Prozessmodell einen Syntaxfehler hat, wird die Fehlermeldung in der entsprechenden Aktivität, in dem Ereignis oder in dem Gateway dargestellt. Außerdem kann die Funktion des Konnektors getestet werden, während das Prozessmodell noch entworfen oder weiterentwickelt wird. Jedoch können nicht alle Konnektoren vor der Ausführung des Prozesses geprüft werden. Bei der Überprüfung mancher Konnektoren kann allerdings eine Fehlermeldung zurückgegeben werden, weil Bonita Execution Engine noch nicht gestartet wurde, d. h. der Prozess wurde nicht wirklich ausgeführt. In manchen Konnektoren werden nämlich dynamische Werte verwendet. Um diese Tests ohne diese Fehlermeldung durchführen zu können, müssen diese dynamischen Werte durch den Groovy[4] Ausdruck auf statische Werte gesetzt werden. Die Testbarkeit ist bei BOS dennoch positiv. Weil die meisten Fehler durch BOS automatisch gefunden werden, ist es nicht zwingend erforderlich, den Unit-Test zu verwendet.

[4]http://groovy.codehaus.org/

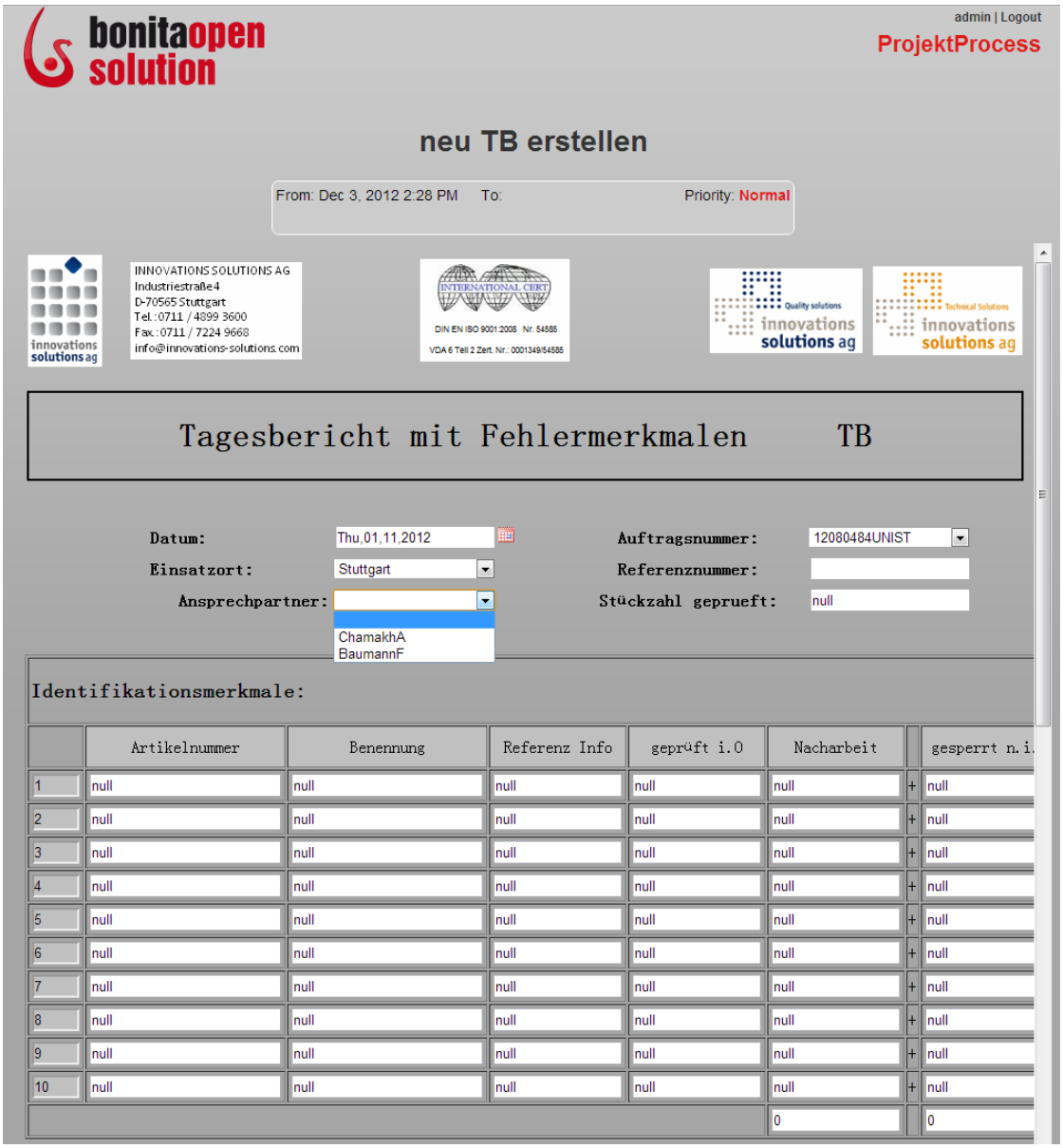

Abbildung 5.7: Oberfläche der Aktivität *neu TB erstellen*

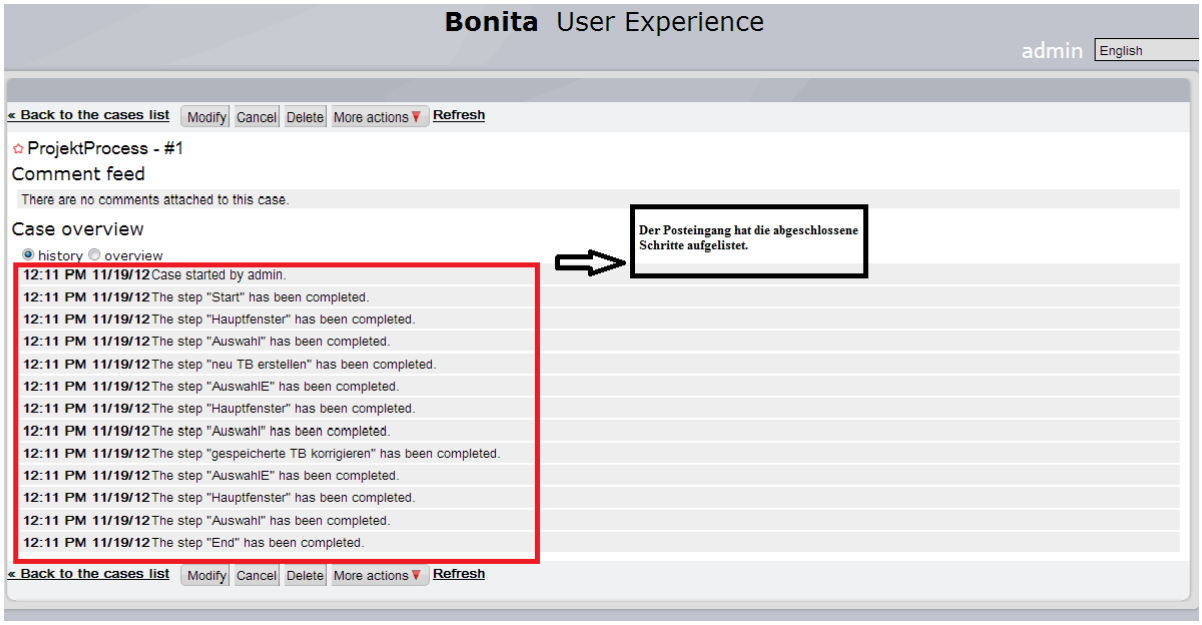

Abbildung 5.8: Ergebnis des Prozessablaufs

5.5.7 Entwicklung der Webanwendung durch Bonita Execution Engine

BOS verwendet die GWT (Google Web Toolkit) Technologie für die Entwicklung der Webanwendung für Prozesse, aber manchmal ist diese Technologie nicht für den Endbenutzer geeignet. Bonita Execution Engine bietet jedoch viele Java-APIs an, durch die die Webanwendung in einem Web-Container entwickelt und durch Bonita Execution Engine ausgeführt werden kann [Sou10, S.1]. In der Abbildung 5.9 werden die wichtigsten APIs in Form von Klassediagrammen dargestellt. Alle APIs werden durch die Klasse *AccessorUtil* aufgerufen, die in dem Paket *org.ow2.bonita.util* definiert ist. Die hier verwendete Version ist Bonita-Client library 5.9 API.

BOS APIs verwendet zwei Objektmodelle [Sou10, S.2-3]: das Objektmodell „Definition“ und das Objektmodell „runtime“. Das Objektmodell „Definition“ beschreibt mit „*ProcessDefinition*“, wie Prozesse definiert werden. Insbesondere Prozesse, die in Java geschrieben werden, werden durch diese „*ProcessDefinition*“ dargestellt. Es enthält die Eigenschaften der Prozesse und die Menge der Instanzen, z. B. die Identifikation des Prozesses, den Name des Prozesses oder auch die Aktivitäten. Das Objektmodell „runtime“ beschreibt die Prozessinstanz bzw. „*Processinstance*“. Die „*Processinstance*“ stellt die Ausführung der Instanz durch den definierten Prozess dar. Es enthält die Eigenschaften der Prozessinstanzen und die Menge der Instanzen, z. B. die Eigenschaften der Prozessinstanzen, Aktivitäten oder Variablen.

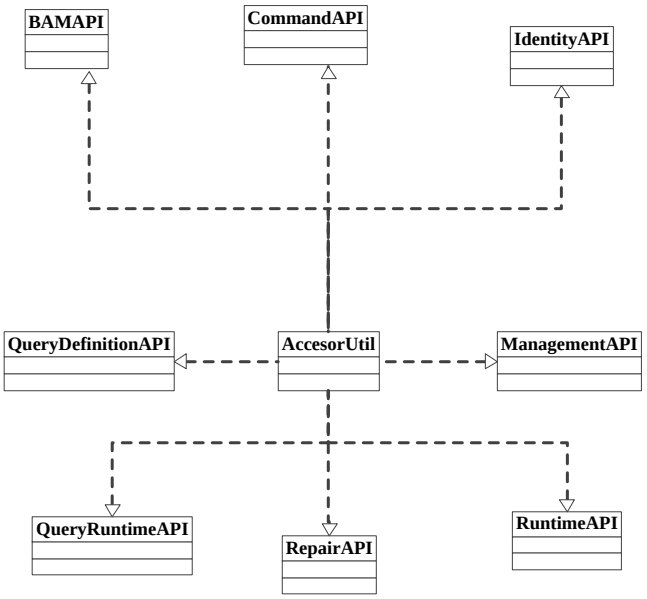

Abbildung 5.9: BOS APIs

- BAMAPI: Eine Operation für die Suchfrage, um die Statistik von laufenden Daten abzufragen.
- CommandAPI: Eine Operation, um verfügbare Befehle in einem bestimmten Prozess oder in der Engine auszuführen.
- IdentityAPI: Eine Operation, um Benutzer zu bearbeiten.
- ManagementAPI: Eine Operation, um Prozesse zu installieren, zu entfernen, zu verwalten usw.
- QueryDefinitionAPI: Eine Operation für die Suchfrage, um „Definition“ Objektmodelle abzufragen.
- QueryRuntimeAPI: Eine Operation für die Suchfrage, um „runtime“ Objektmodelle abzufragen.
- RepairAPI: Eine Operation, um die Ausführung der Prozessinstanz zu verwalten.
- RuntimeAPI: Eine Operation, um „runtime“ Objektmodelle zu modifizieren.

Auf der Webseite[5] werden viele APIs dargestellt. Mit diesen APIs können verschiedenste, kundenspezifische Benutzeroberflächen erstellt werden, die dann in die Prozessmodelle integriert werden können. Damit ist die Verfügbarkeit positiv.

[5]www.bonitasoft.org/docs/javadoc/bpm_engine/5.9/

5.6 Activiti

Activiti[6] ist eine funktional reduzierte („lightweight“) Workflow- und BPM-Plattform, deren Kernkomponente die Process Engine ist. Diese Process Engine bietet Möglichkeiten für die Ausführung des Prozessmodells, das durch BPMN 2.0 beschrieben wird und die Erstellung neuer Workflow-Aufgaben. Activiti verwendet die Apache-Lizenz, wodurch alle Java-Anwendungen lauffähig sind. Die aktuelle Vision ist Activiti 5.11. In der Abbildung 5.10 werden Komponenten von Activiti dargestellt.

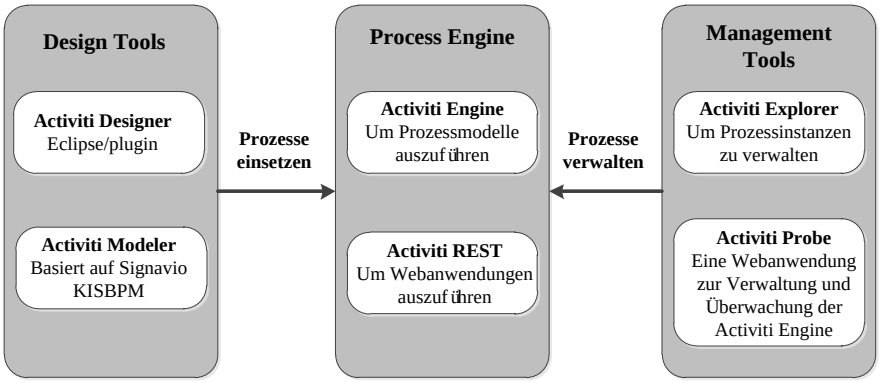

Abbildung 5.10: Komponenten der Activiti [Rad12, S.4]

Activiti bietet folgende wichtige Komponenten und Anwendungen.

- Design Tools:
  1. Activiti Modeler ist ein webbasierter grafischer Editor für die Erstellung der Prozessmodelle. Es ist ein Zweig des Projektes von der *Signavio Core Componets*[7]. In der Activiti 5.11 wird der Name als *KISBPM Process Solution* genannt und die Activiti Modeler und die Activiti Explorer werden zusammen aufgebaut.
  2. Activiti Designer ist ein Eclipse Plugin für das Design eines Prozessmodells.
- Process Engine:
  1. Activiti Engine ist die Kernkomponente von Activiti, durch die Prozessmodelle, die durch BPMN2.0 geschrieben werden, ausgeführt werden können.
  2. Activiti REST ist die API für die Webanwendung.

[6]http://activiti.org/
[7]http://code.google.com/p/signavio-core-components/

- Management Tools:
    1. Activiti Explorer ist eine webbasierte Anwendung, die den Zugriff für alle Benutzer bietet, wenn die Activiti Engine aktiv ist. Diese Anwendung enthält das Task-Management, die Darstellung der statistischen historischen Daten und die Darstellung der Prozessinstanzen.
    2. Activiti Probe ist eine Webanwendung zur Administration und Überwachung der Activiti Engine. Es wird in der Version 5.7 mit dem Activiti Explorer zusammen aufgebaut.

Die Prozessmodelle werden durch Design Tools entwickelt und anschließend in der Process Engine eingesetzt. Durch Management Tools können die Prozesse verwaltet werden.

5.6.1 Konfiguration der Process Engines

Wegen der Verwendung der Apache-Lizenz ist die Einbettbarkeit dieser Process Engine positiv und die Komplexität der Konfiguration der zwei Engines ist relativ niedrig. Zuerst kann Tomcat 6.0.36 in der Webseite der Apache heruntergeladen und in einer manuell festzulegenden Partition dekomprimiert werden. Dann wird das ZIP-File der Activiti 5.11 von der Webseite[8] heruntergeladen und dekomprimiert. Danach werden die Dateien *activiti-explorer.war* und *activiti-rest.war* in den Ordner webapps kopiert, der wiederum ein Unterordner des dekomprimierten Files Tomcat 6.0.36 ist. Schließlich werden die Umgebungsvariablen CATALINA_HOME und JAVA_HOME definiert. Weil die Aktiviti 5.11 erst am 05.12.2012 veröffentlicht wurde, wird in dieser Diplomarbeit die Activiti 5.6 und die Tomcat 6.0.32 verwendet. Die Konfiguration der Activiti 5.6 und Tomcat 6.0.32 ist nur wenig komplizierter als die der neueren Versionen. Die Activiti 5.6 wird von der Webseite der Activiti heruntergeladen und dekomprimiert. Allerdings muss ein zusätzliches Werkzeug Apache Ant[9] verwendet werden, wodurch sich Activiti automatisch in Tomcat installiert wird. In dem Anhang A.1 wird die Konfiguration der 2 Engines beschrieben.

5.6.2 Erstellung des Prozessmodells durch Design Tools

Geschäftsprozesse können durch Design Tools modelliert werden. Es gibt zwei Werkzeuge für die Prozessmodellierung. Activiti Designer ist ein Plugin von Eclipse[10]. Durch Activiti Designer können Prozessmodelle unter der Eclipse grafisch erstellt werden. Dieses Plugin wird in der Webseite[11] dargestellt und einfach in Eclipse installiert. Ein weiteres Werkzeug für die Entwicklung der Prozessmodelle ist Activiti Modeler, die durch den Activiti Explorer aufgerufen werden kann. In der Activiti 5.11 bietet der Activiti Explorer die neue Funktion

[8] http://activiti.org/download.html
[9] http://ant.apache.org/
[10] www.eclipse.org/
[11] http://activiti.org/designer/update/

„*Model workspace*“ unter dem Menü „*Process*“ an. In der Abbildung 5.11 wird diese neue Funktion dargestellt.

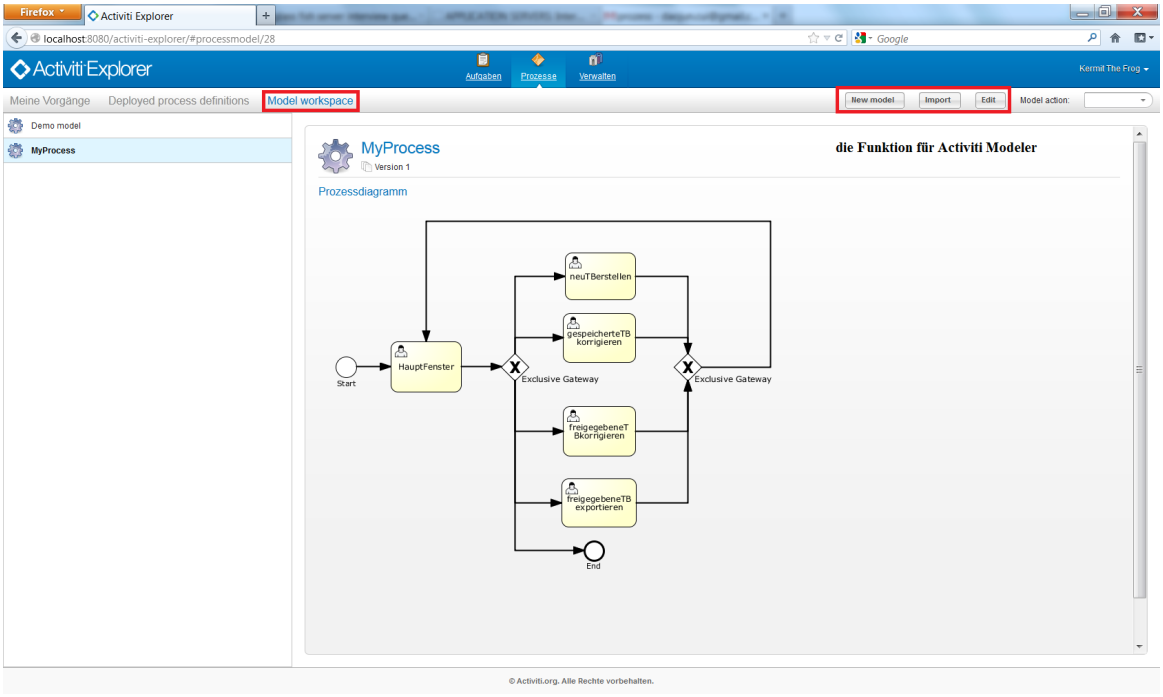

Abbildung 5.11: neue Funktion in Activiti Explorer

Nach dem Anklicken des Buttons „*New model*“ wird der Activiti Modeler geöffnet, der in der Abbildung 5.12 dargestellt ist.

Jedes Element des Prozessmodelles kann durch „Drag“ und „Drop“ von der linken Liste in der zentralen Fläche gezogen werden. Durch die rechte Liste können die Eigenschaften der Elemente definiert werden. Weil Design Tools von Activiti die BPMN 2.0 unterstützen, ist die Benutzbarkeit positiv.

5.6.3 Implementierung der Webformulare

In Aktiviti Modeler gibt es keinen Formulargenerator, d. h. die Oberfläche der Aktivität muss selbst definiert werden, also die Information dieser Oberfläche müssen durch HTML beschrieben und in einem **.form*-File gespeichert werden. Ein **.form*-File ist dabei eine XML-Datei, die für die Beschreibung der Informationen einer GUI (Grafische User Interface) verwendet. Um die Webformulare auf der Aktivität zu implementieren, wird das HTML <iframe> Tag verwendet. Durch den dynamische Webseiten in dem aktuellen HTML-Dokument bzw. in dem **.form*-File eingebettet werden können. Die **.jsp*-File werden dann mit der URL abgebildet, die URL wird durch den HTML <iframe> Tag in das **.form*-File implementiert und

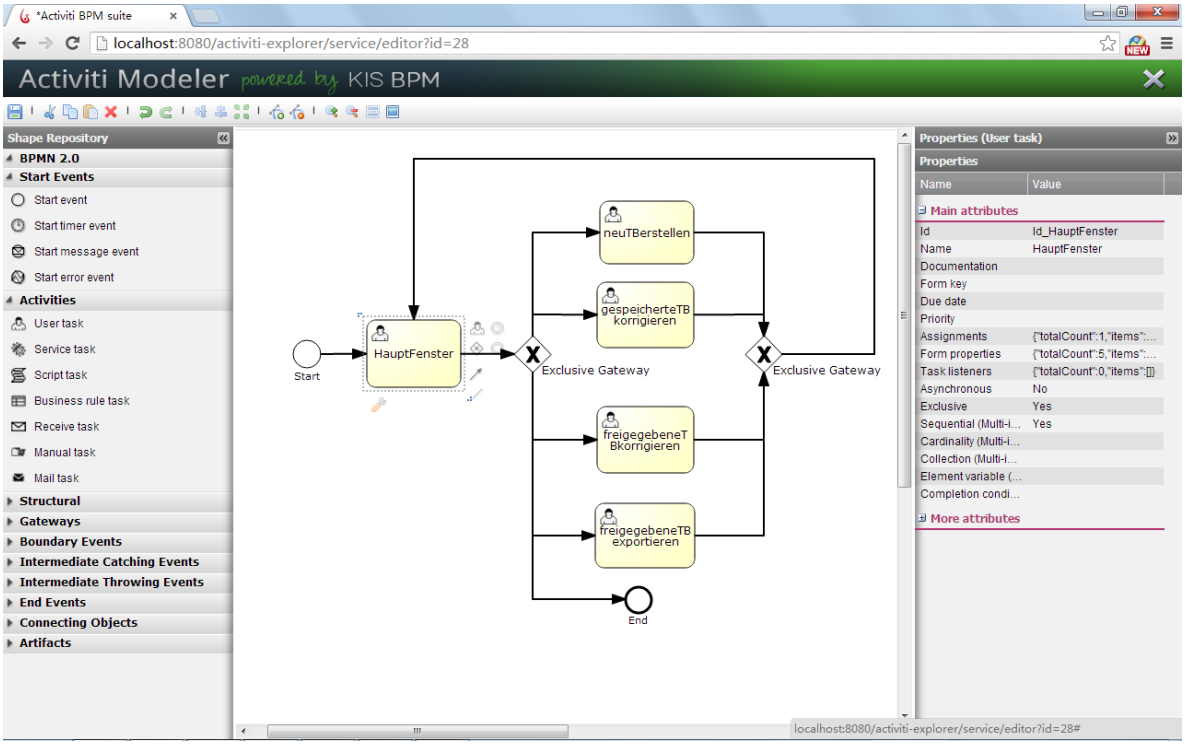

Abbildung 5.12: Erstellung des Prozessmodelles durch Activiti Modeler

schließlich wird das *.*form*-File in der Aktivität verlinkt. Die Abbildung 5.13 zeigt, wie das *.*form*-File in der Aktivität definiert wird.

Unter dem Menü „*Main config*“ gibt es die Option „*Form key*“, mit der das *.*form*-File in dieses Feld implementiert wird. In Activiti Modeler wird die gleiche Methode für die Implementierung des *.*form*-Files verwendet. Weil Activiti spezielle Designer Tools für die grafische Darstellung des Prozessmodelles anbietet, ist die Integrierbarkeit dieser Process Engine positiv. Die Oberflächen der Aktivitäten müssen jedoch durch andere Werkzeuge erstellt werden, bevor sie in die Aktivität implementiert werden. Die Steuerbarkeit ist deshalb negativ.

5.6.4 Ausführung der Geschäftsprozesse

Die Geschäftsprozesse werden in der Activiti Engine ausgeführt. Die Variablen und Bedingungen der XOR-Gateways müssen dabei vorab definiert werden. Die Methode ist ähnlich wie die Definition der Variablen und der Bedingungen in BOS. Beispielsweise werden in Activiti 5.6 die Variablen jeder Aktivität durch „*Form properties*„ definiert. Diese Eigenschaft wird unter dem Menü Form des Activiti Designers dargestellt. Variablen können als „Boolean“, „Date“, „Enum“, „Long“ oder „String“ definiert werden. Die Bedingungen müssen

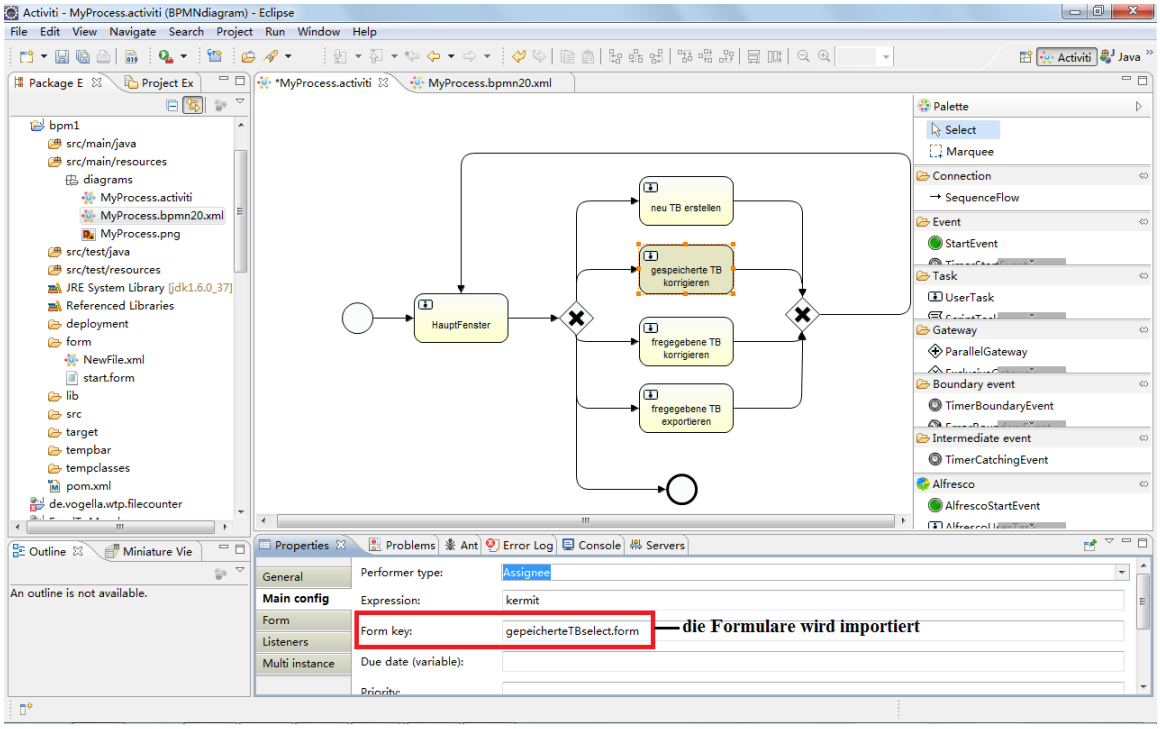

Abbildung 5.13: Implementierung des *.*form*-Files in der Aktivität durch Activiti Designer

mit einem $ Zeichen und in einer Eckklammer gegeben werden. Danach wird das Prozessdiagramm als das „BPMN 2.0 XML“ Format von Activiti Designer exportiert. Anschließend wird dieses exportierte File mit dem *.*form*-File zu einem *.*zip*-File komprimiert und schließlich durch Activiti Probe auf die Activiti Engine hochgeladen. Die Geschäftsprozesse werden durch den Activiti Explorer aktiviert. In der Abbildung 5.14 wird die Oberfläche der Aktivität *„gespeicherte TB korrigieren“* dargestellt.

5.6.5 Verwaltung der Geschäftsprozesse

Durch Activiti Explorer können die Ergebnisse angezeigt werden, z. B. welche Prozessschritte ausgeführt werden oder wer den Prozessschritt durchgeführt hat. In der Abbildung 5.15 ist das Ergebnis des Prozessablaufes dargestellt.

In Activiti Explorer kann das Ergebnis unter dem Menü *„Verwalten“* abgerufen werden. In der Fläche *„Prozessdiagramm“* wird der aktuell auszuführende Prozessschritt angezeigt. In der Fläche *„Aufgaben“* werden der abgeschlossene Prozessschritt und der Prozess-Bevollmächtigte aufgelistet.

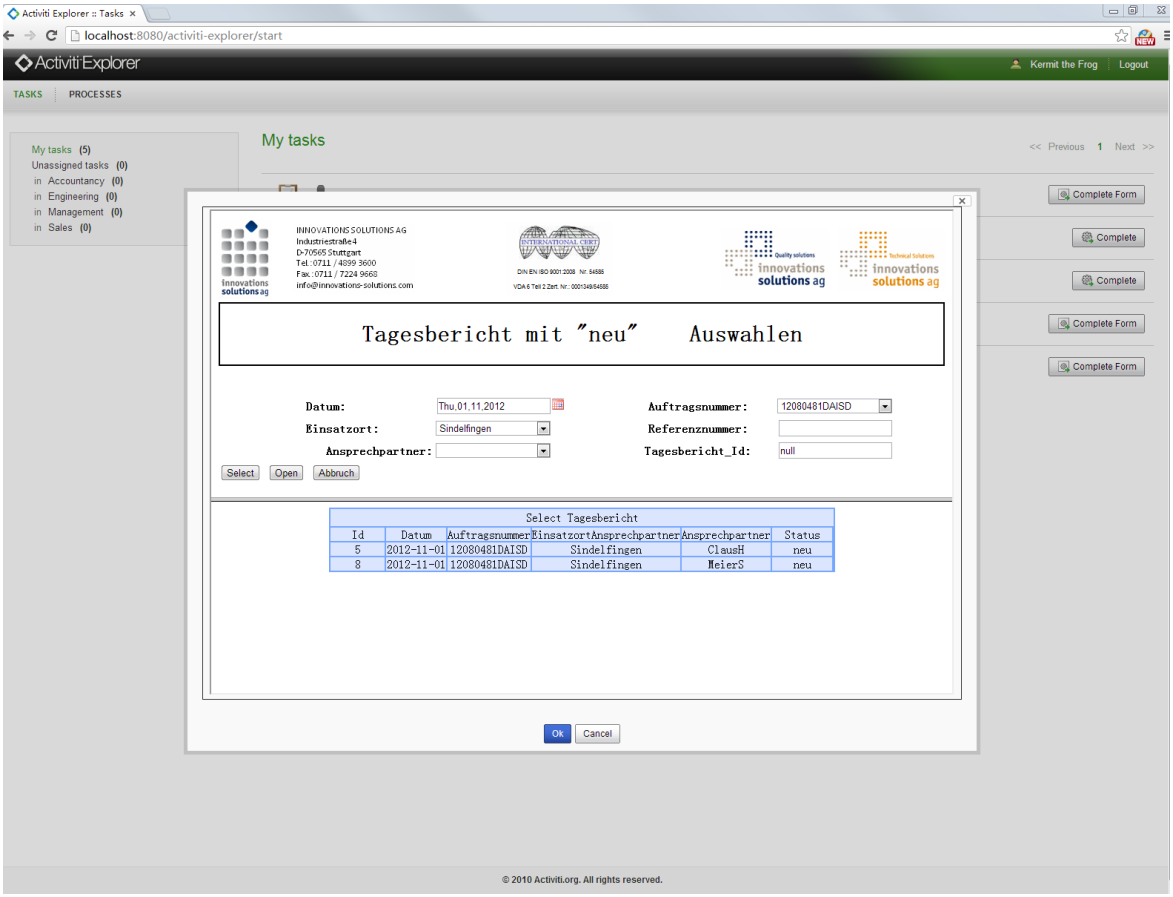

Abbildung 5.14: Oberfläche der Aktivität *„gespeicherte TB korrigieren“*

5.6.6 Überprüfung der Prozesslogik

Weil die Activiti Engine in die Java Engine eingebettet ist, kann der UnitTest für die Überprüfung der Prozesslogik verwendet werden. Activiti unterstützt JUnit 3 und JUnit 4 für den UnitTest. Beispielsweise wird in der JUnit 4 wird die Regel *org.activiti.test.ActivitiRule* angewendet. Durch diese Regel sind die Process Engine und Dienstleistungen für den Entwickler verfügbar. In dem Test kann die Annotation *org.activiti.engine.test.Deployment* verwendet werden, durch die die Default-Konfigurationsdatei einlesen wird. Die Testbarkeit der Activiti ist positiv, weil viele Fehler durch den Unit-Test überprüft werden.

5.6.7 Process Engine in der Webanwendung

Die Activiti Engine bietet viele Java APIs, durch die viele Anwendungen aufgerufen werden können. Beispielsweise kann eine API *ProcessEngine* verwendet werden, In dieser API werden die Informationen der Konfiguration zunächst aufgerufen und in dem File *activiti.cfg.xml*

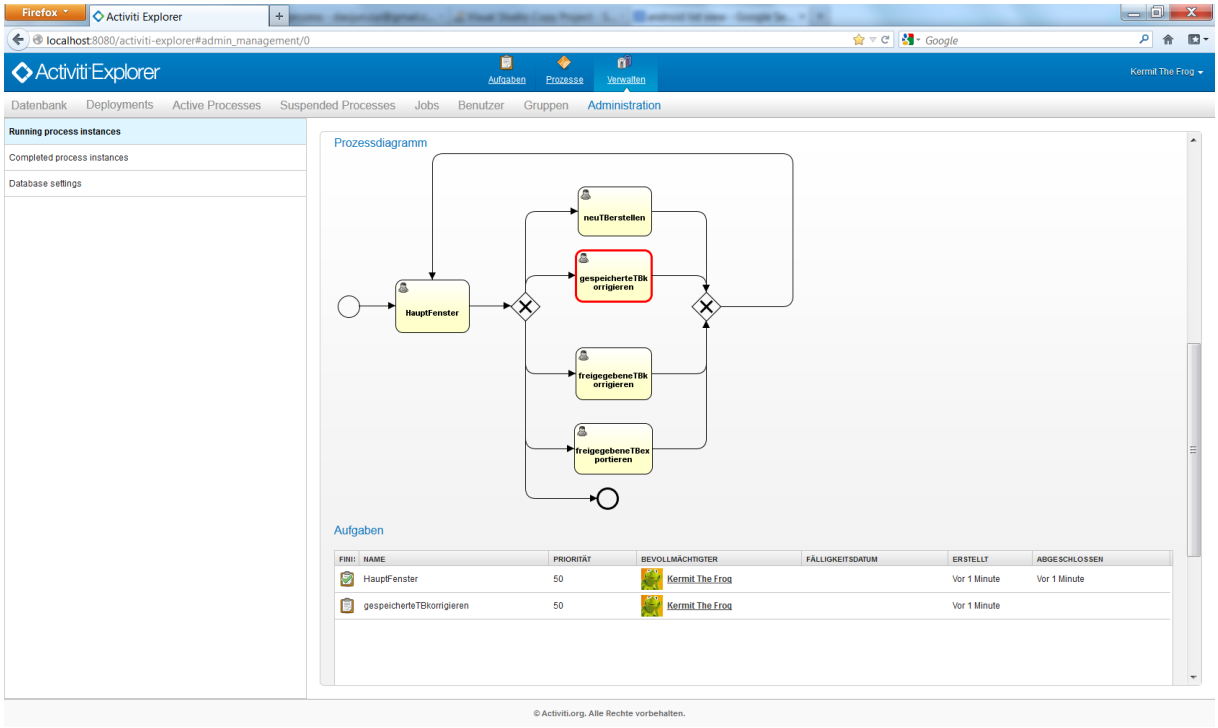

Abbildung 5.15: Ergebnis des Prozessablaufs

gespeichert oder beim der Aufruf der Klasse *ProcessEngineConfiguration* angezeigt. Dann können verschiedene Services durch diese API bedient werden. Die Funktionen der APIs werden unter der Webseite[12] dargestellt. In der Abbildung 5.16 werden die wichtigsten Engine APIs dargestellt.

- FormService:

  FormService bietet Methoden, um die Daten von den Formularen abzurufen, das Formular der Prozessinstanz darzustellen und die Aufgabe zu vervollständigen.

- HistoryService:

  Um Informationen über abgeschlossene Prozessinstanzen abzurufen [Rad12, S.56].

- IdentityService:

  Um Benutzer, Gruppen und die Relation zwischen den Benutzern und Gruppen zu verwalten.

- ManagementService:

  Um die Tabelle der Aktivität abzufragen oder Aufgaben ausführen.

[12]http://activiti.org/javadocs/index.html

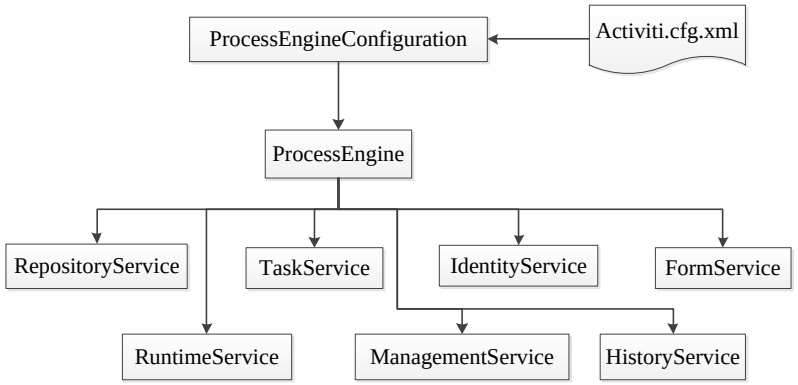

Abbildung 5.16: Activiti APIs [Act]

- RepositoryService:

  Um Prozesse zu implementieren, zu löschen, abzufragen oder abzurufen [Rad12, S.57].

- RuntimeService:

  Um Prozessinstanzen zu starten oder zu suchen.

- TaskService:

  Um die HumanTask zu behaupten, zu übertragen usw.

Diese APIs können in vielen Webanwendungen verwendet werden und die Process Engine kann direkt in das Servlet implementiert werden. Damit können die Benutzeroberflächen, die in JSP geschrieben wurden, einfach mit der Aktivität kombiniert werden. Die Verfügbarkeit der Activiti ist somit positiv. Die Activiti Engine unterstützt den Aufruf der Web-Services. Weil die Prozessmodelle in Activiti Designer als *.*zip*-File gespeichert werden, können die zusätzlichen XML-Elementen zu diesem File einfach hinzugefügt werden, z. B. kann die WSDL-Datei durch den <import> Tag importiert werden. Nach der Definition dieses *.*zip*-Files kann die Aktivität die Web-Services aufgerufen werden, wodurch die Integrierbarkeit auch positiv ist.

5.7 Ergebnis des Vergleichs der Process Engines

Beide Process Engines werden unter dem Betriebssystem Windows 7 Home (32bit) installiert. Sie sind einfach mit der Tomcat konfigurierbar. Die Ausführung -und Verwaltung der Geschäftsprozesse ist bei beiden positiv. Lediglich die Steuerbarkeit also ist bei BOS besser als bei Activiti. Weil die Entwickler nur wenig Kode für die Entwicklung des Prozessmodelles direkt in BOS schreiben müssen und darüber hinaus BOS schnell lernen können, ist die BOS

	Bonita Open Solution	Activiti
Version	5.8	5.6
Servlet Engine	Tomcat 6.0.35	Tomcat 6.0.32
Betriebssystem	Windows 7 Home 32bit	Windows 7 Home 32bit
Einbettbarkeit	Das BOS-Tomcat Paket wird heruntergeladen. Das Batch-File und Umgebungsvariablen werden konfiguriert.	Das Activiti Paket wird heruntergeladen. Tomcat wird durch Ant installiert. Umgebungsvariablen wird konfiguriert.
Komplexität	niedrig	niedrig
Benutzbarkeit	BPMN 2.0, JBPM3 und XPDL	BPMN 2.0
Testbarkeit	Der Bonita Konnektor, Groovy	JUnit
Deployment	automatisch	*.*xml*-Files und *.*form*-Files werden als den *.*zip*-File zusammenkomprimiert und durch Activiti Probe in der Process Engine hochgeladen.
Verfügbarkeit	Java API	Java API
Integrierbarkeit	Verbindung der Webservice durch den Konnektor.	Verbindung der Webservice durch die Definition der XML-Elemente
Ausführung Verwaltung	Prozessmodelle werden durch die Bonita Execution Engine ausgeführt und durch Bonita User Experience verwaltet.	Prozessmodelle werden durch die Activiti Engine ausgeführt und durch Activiti Explorer verwaltet

Tabelle 5.1: Ergebnis für den Vergleich der Proeess Engines

für einfache Projekte und Anfänger geeigneter. Für erfahrenere Java-Entwickler ist Activiti eine gute Wahl, weil Activiti viele Java-API anbieten und die Aktivitäten einfach durch Java-Programme aufgerufen und bearbeitet werden können. In der Tabelle 5.1 wird eine Zusammenfassung für diese zwei Process Engines dargestellt.

Für die ISAG ist die Bonita Open Solution in Summe besser geeignet, weil die größtenteils bereits vorhandenen Benutzeroberflächen mit relativ geringem Aufwand eingebunden werden können. Auch sind die Programmierung der einzelnen Funktionen sowie die Kommunikation der Server miteinander und mit der Datenbank bei der Wartung -und Weiterentwicklung dieser Software durch andere Personen einfacher nachzuvollziehen.

6 Definition und Implementierung der Programmtests

In diesem Kapitel werden Methoden zum Testen der Benutzeroberflächen-Funktionen vorgestellt.

6.1 Ziel des Testens

Der Programmtest ist für die Entwicklung korrekt funktionierender Software und hat im Allgemeinen die folgenden Ziele:

- *Testing is the Process of executing a program with the intent of finding errors* [MBTS04, S.6].
- *Testing is the process of establishing confidence that a program or system does what it is supposed* [Het88, S.4].

Zusammenfassend kann man sagen: das Ziel des Testens ist, dass durch ein mehrfaches, also wiederholtes oder auch unterschiedliches Ausführen des Programms Fehler in der Oberflächenprogrammierung oder in der Datenverwaltung aufgedeckt werden.

6.2 Testprotokoll

Für den hier beabsichtigten Programmtest wird zunächst eine Testplanung erstellt, anschließend werden die Testmethode und die Testauswertung tabellarisch dargestellt. In dem IEEE-Standard 829 [IEE08] wird die Testplanung vollständig beschrieben. Die Testvorgehensweise, die Zeitplanung der Testphase und der Testinhalt werden in der Planung festgelegt. Das Testprotokoll wird in dem Anhang A.2 gegeben.

Die Gesamtzeit des Programmtests beträgt mitunter bis zu 60 Stunden. Die Oberfläche wird durch den Webbrowser Firefox dargestellt, weil Firebox ein Web-Entwicklungs-Tool Firebug bietet, durch das Fehler im Quellcode einer Webseite angezeigt werden können. SQL-Befehle werden in MySQL durchgeführt. Dabei wird überprüft, ob die Daten richtig in die Datenbank eingefügt, aktualisiert oder ausgelöscht werden. Der Java-Code wird im Eclipse-Debugger überprüft. Da es derzeit kein besonders gutes Werkzeug zum Testen des Code von JavaScript und jQuery gibt, muss man den Wert jedes Parameters der Funktion durch die Funktion `alert()` im Webbrowser anzeigen und auf Korrektheit prüfen. Weil Ein-und Ausgaben der Oberfläche getestet werden, wird in diesem Test die Vorgehensweise

„Black-Box-Test“ verwendet. So soll z. B. bei Fehlereingaben das Pop-Up-Fenster angezeigt werden.

6.3 Black-Box-Test

Um die Funktionalität der Benutzeroberfläche umfassend und zuverlässig zu testen, müssen folgende Punkte betrachtet werden:

- Die Durchführung eines vollständigen Testens des Programms muss gewährleistet sein.
- Ein Tester muss verfügbar sein, der die entsprechende Geisteshaltung besitzt [Mye01, S.3].

In dem ersten Fall werden alle Funktionen des Programms überprüft, in dem zweiten Fall wird die Benutzeroberfläche der Webbrowser der verschiedenen Betriebssysteme getestet. Dabei werden die Programme durch „Black-Box-Test“ getestet.

Bei diesem Test wird die innere Funktionsweise des Programms nicht betrachtet, sondern nur die Funktionen der Benutzeroberfläche getestet. Dazu trägt der Benutzer Daten in die Oberfläche ein und nach dem Klick auf unterschiedliche werden die erzeugten Ergebnisse an die Oberfläche dargestellt. Die die korrespondierende Fehlerinformation soll in dem Fenster angezeigt werden und die leeren Textfelder nach dem Klick auf den Button mit gelber Farbe markiert werden.

In dem Black-Box-Test sollten die folgenden Testfälle [LL07, S.471] durchgeführt werden, diese Testfälle werden für Testen der Benutzeroberfläche benötigt.

- alle Funktionen des Programms aktivieren, bzw. die Funktionsüberdeckung.
- alle möglichen Eingaben bearbeiten, bzw. die Eingabeüberdeckung.
- alle möglichen Ausgaben erzeugen, bzw. die Ausgabeüberdeckung.
- die spezifizierten Mengengrenzen ausschöpfen, bzw. die Überdeckung der Grenzfälle.

6.3.1 Testfälle für die Funktionsüberdeckung

In diesem Testfall wird jede Funktion des Buttons einmal ausgeführt. Damit wird überprüft, ob die Funktionen den gewünschten Anforderungen entsprechen, also die gewünschte Aktion auslösen. In der Benutzeroberfläche gibt es folgende zu testende Funktionen:

F_0: Das Eingabeformular des Tagesberichts wird in der Seite *Tagesbericht.jsp* geöffnet.

F_1: Die neue Seite *search_frame_neu.jsp* wird geöffnet, diese Seite enthält zwei Teile. Die Auswahlseite des Tagesberichtes wird in dem ersten Teil *search_neu.jsp* dargestellt, die Auswahlliste wird in dem zweiten Teil *search_table_neu.jsp* dargestellt.

F_2: Die neue Seite *search_frame_frei.jsp* wird geöffnet, diese Seite enthält ebenfalls zwei Teile. Die Auswahlseite des Tagesberichts wird in dem ersten Teil mit *search_frei.jsp* dargestellt, die Auswahlliste im zweiten Teil wird durch *search_table_frei.jsp* dargestellt.

F_3: Die neue Seite *search_frame_exp.jsp* wird geöffnet, diese Seite enthält ebenfalls zwei Teile. Die Auswahlseite des Tagesberichts wird im ersten Teil mit *search_exp.jsp* dargestellt, die Auswahlliste in dem zweiten Teil wird mit *search_table_exp.jsp* dargestellt.

F_4: Die Auswahlliste, die die Identifikation des Tagesberichtes, die Auftragsnummer, das Datum, den Einsatzort, die Ansprechpartner und den Status des Tagesberichtes enthält, wird angezeigt.

F_5: Daten eines Tagesberichtes werden in der Seite *tagesbericht_kor.jsp* aufgelistet.

F_6: Daten eines Tagesberichtes werden in die Datenbank geschrieben, der Status des Tagesberichts wird in der Datenbank mit „neu“ gekennzeichnet.

F_7: Daten eines Tagesberichtes werden in die Datenbank geschrieben, der Status des Tagesberichts wird in der Datenbank mit „gen“ gekennzeichnet.

F_8: Daten eines Tagesberichtes werden in die Datenbank geschrieben, der Status des Tagesberichts wird in der Datenbank mit „exp“ gekennzeichnet.

F_9: In einen Tagesbericht wird eine neue Zeile eingefügt.

F_{10}: Die Daten eines Tagesberichtes werden von der Datenbank gelöscht.

F_{11}: Aus der Seite *Tagesbericht_kor.jsp* wird ein PDF mit den Daten dieser Seite generiert und anschließend das PDF geöffnet.

F_{12}:: Daten werden von der Datenbank nach Excel exportiert und anschließend wird Excel geöffnet.

F_{13}:: Die Seite wird geschlossen.

Die Tabelle 6.1 zeigt diesen Satz von Testfällen.

In diesem Testfall wird überprüft, ob die Eingaben und die Ausgaben korrekt sind, ob der Button durch Drücken der Entertaste oder einen Mausklick betätigt wird und welche Ergebnisse von der Benutzeroberfläche zurückgegeben werden, ob z. B. ein Meldefenster angezeigt wird oder die Seite gewechselt oder geschlossen wird.

Die Funktion mancher Seiten (also Formulare) ist es, weitere Formulare abzuschicken. Dazu werden HTML<input> Tags vom Typ „*text*“ in ein solches Formular geschrieben. Auf diese Weise kann eine HTML<input> Tag mit dem Typ „*text*“ in einem Formular so definiert werden, dass in diesem Formular ein bestimmter Button betätigt wird, wenn die „Enter“-Taste drückt wird. In diesem Fall würde dann das mit diesem Button korrespondierende Formular abgeschickt. Wenn das Formular nicht fertig ausgeführt, aber der Benutzer dennoch die Entertaste drückt, wird das Formular unvollständig abgeschickt und die Daten werden gegebenenfalls nicht vollständig in die Datenbank geschrieben. Um dies zu verhindern, wird die Funktion der Entertaste in manchen Seiten durch jQuery abgeschirmt. Der Button (Abbrechen) wird auf vielen Seiten dargestellt. In manchen Seiten wird der Button über die

Funktion	Name(Button)	Eingaben		Ausgaben		
		Maus	Tastatur	Meldung	Seite wechseln	Seite schließen
F_0	neuen TB erstellen	Ja	Ja	Nein	Ja	-
F_1	gespeicherten TB korrigieren	Ja	Ja	Nein	Ja	-
F_2	freigegebenen TB korrigieren	Ja	Ja	Nein	Ja	-
F_3	freigegebenen TB exportieren	Ja	Ja	Nein	Ja	-
F_4	Auswählen	Ja	Ja	Ja	Nein	Nein
F_5	Öffnen	Ja	Ja	Nein	Ja	-
F_6	Speichern	Ja	Nein	Ja	Nein	-
F_7	Speichern && Prüfen	Ja	Nein	Ja	-	Ja
F_8	Exportieren && Prüfen	Ja	Nein	Ja	-	Ja
F_9	+	Ja	Nein	Nein	Nein	-
F_{10}	Löschen	Ja	Nein	Ja	-	Ja
F_{11}	PDF erstellen	Ja	Nein	Nein	Ja	-
F_{12}	Excel öffnen	Ja	Nein	Nein	Ja	-
F_{13}	Abbrechen	Ja	Ja,Nein	Nein	-	Ja

Tabelle 6.1: Testfall der Funktionsüberdeckung

Entertaste und den Mausklick betätigt, in anderen Seiten kann er nur mit dem Mausklick betätigt werden. Das „-“ Zeichen in der Tabelle 6.1 bedeutet, dass die Ausgabewerte in diesem Fall nicht betrachtet werden.

6.3.2 Testfälle für die Eingabeüberdeckung

Jede Eingabe wird in einem Testfall ausgeführt. In diesem Testfall wird das Datum, die Auftragsnummer, der Einsatzort, der Ansprechpartner, der Name, die Beginnzeit, die Endzeit, die Pausenzeit und die Arbeitszeit überprüft. Bei diesem Test sollen sogenannte Äquivalenzklassen gebildet werden, die alle äquivalenten Testbedingungen in einer Klasse zusammenfassen [BM11, S.32]. Es werden gültige Äquivalenzklassen und ungültige Äquivalenzklassen unterschieden. In gültigen Klassen werden gültige Eingabewerte definiert, d. h. Eingaben im definierten Wertebereich und im Gegensatz dazu in ungültigen Klassen.

In Testfälle werden gültige Äquivalenzklassen als „g-$Ä_n$“ bezeichnet und die ungültige Äquivalenzklassen als „ug-$Ä_n$“ bezeichnet.

- gültige Äquivalenzklassen:

  g-$\ddot{A}_0$: das Datum, die Auftragsnummer, der Einsatzort und der Ansprechpartner sind nicht leer. Weil diese Daten von der Unterliste der Oberfläche ausgewählt werden, wird die Formel dieser Daten nicht überprüft.

  g-$\ddot{A}_1$: die Beginnzeit, die Endzeit und die Pausenzeit sind 4-stellige oder 3-stellige Zahlen und werden mit dem Doppelpunkt getrennt. Nach dem Doppelpunkt folgt eine 2-stellige Zahl.

  g-$\ddot{A}_2$: $0 \leq$ (Zahl vor dem Doppelpunkt) < 24, weil die Zeitangabe der Stunden für einen Tag von 0 bis 24 sein kann.

  g-$\ddot{A}_3$: $0 \leq$ (Zahl nach dem Doppelpunkt) < 59, weil die Zeitangabe der Minuten für eine Stunde von 0 bis 60 sein kann.

  g-$\ddot{A}_4$: $0 <$ (Endzeit – Beginnzeit – Pausenzeit) ≤ 10 für die Frühschicht und die Spätschicht oder $0 <$ (24 – (Endzeit – Beginnzeit – Pausenzeit)) ≤ 10 für die Nachtschicht. Da die Arbeitszeit in Deutschland gesetzlich auf maximal 10 Stunden pro Tag begrenzt ist.

  g-$\ddot{A}_5$: falls der Name nicht leer ist, darf die Arbeitszeit nicht „NaN“, „0“ oder leer sein.

- ungültige Äquivalenzklassen:

  ug-$\ddot{A}_0$: das Datum, die Auftragsnummer, der Einsatzort oder der Ansprechpartner ist leer.

  ug-$\ddot{A}_1$: die Beginnzeit, die Endzeit und die Pausenzeit sind keine 4-stelligen oder 3-stelligen Zahlen, werden jedoch mit einem Doppelpunkt getrennt, z. B. können sie Buchstaben oder Symbole beinhalten; nach dem Doppelpunkt gibt es keine Zahl; usw.

  ug-$\ddot{A}_2$: die Beginnzeit, die Endzeit und die Pausenzeit sind 4-stellige oder 3-stellige Zahlen und werden durch den Doppelpunkt getrennt, es gibt jedoch nur eine Stelle oder keine Stelle nach dem Doppelpunkt.

  ug-$\ddot{A}_3$: die Beginnzeit, die Endzeit und die Pausenzeit werden nicht durch den Doppelpunkt getrennt.

  ug-$\ddot{A}_4$: die Zahl vor dem Doppelpunkt ist < 0 oder die Zahl nach dem Doppelpunkt ist ≥ 24.

  ug-$\ddot{A}_5$: (Endzeit – Beginnzeit – Pausenzeit) ≤ 0 oder (Endzeit – Beginnzeit – Pausenzeit) > 10 für die Frühschicht und die Spätschicht, (24 – (Endzeit – Beginnzeit – Pausenzeit)) ≤ 0 oder (24 – (Endzeit – Beginnzeit – Pausenzeit)) > 10 für die Nachtschicht.

  ug-$\ddot{A}_6$: der Name ist leer oder nicht leer, aber die Arbeitszeit ist nicht leer; oder der Name ist nicht leer, aber die Arbeitszeit ist „NaN“, „0:00“ oder leer.

Die Tabelle 6.2 zeigt diesen Testfall. Das Datum wird mit „D“ bezeichnet, die Auftragsnummer wird mit „Nr.“ bezeichnet, der Einsatzort wird mit „O“ bezeichnet, der Ansprechpartner wird mit „P“ bezeichnet, der Name wird mit „N“ bezeichnet, die Beginnzeit wird mit „Bz“ bezeichnet, die Endzeit wird mit „Ez“ bezeichnet, die Pausenzeit wird mit „Pz“ bezeichnet und die Arbeitszeit wird mit „Az“ bezeichnet.

Eingabename	Eingabewerte	gültige Äquivalenzklasse	ungültige Äquivalenzklasse	erwartete Ausgabe
D, Nr., O, P	von der Unterliste auswählen	g-$\ddot{A}_0$		der entsprechende Werte von D, Nr., O, P
D, Nr., O, P	leer auswählen		ug-$\ddot{A}_0$	Fehlermeldung
Bz, Ez, Pz, N	Bz: 6:00, Ez: 14:00, Pz: 0:30, N: von der Unterliste auswählen	g-$\ddot{A}_1$, g-$\ddot{A}_2$, g-$\ddot{A}_3$, g-$\ddot{A}_4$, g-$\ddot{A}_5$		Az: 7:30
Bz, Ez, Pz, N	Bz: 22:00, Ez: 6:00, Pz: 0:30, N: von der Unterliste auswählen	g-$\ddot{A}_1$, g-$\ddot{A}_2$, g-$\ddot{A}_3$, g-$\ddot{A}_4$, g-$\ddot{A}_5$		Az: 7:30
Bz, Ez, Pz, N	Bz: 6:5a, Ez: 1$:00, Pz: 0:0, N: von der Unterliste auswählen		ug-$\ddot{A}_1$	Az: NaN, Fehlermeldung
Bz, Ez, Pz, N	Bz: 6:234, Ez: 132:00, Pz: 0:3, N: von der Unterliste auswählen		ug-$\ddot{A}_1$, ug-$\ddot{A}_4$, ug-$\ddot{A}_5$, ug-$\ddot{A}_6$	Az: NaN, Fehlermeldung
Bz, Ez, Pz, N	Bz: 6:234, Ez: 14:99, Pz: 0:30, N: von der Unterliste auswählen		ug-$\ddot{A}_2$, ug-$\ddot{A}_4$, ug-$\ddot{A}_6$	Az: NaN, Fehlermeldung
Bz, Ez, Pz, N	Bz: 62:34, Ez: 70:99, Pz: 0:30, N: von der Unterliste auswählen		ug-$\ddot{A}_4$	Az: NaN, Fehlermeldung
Bz, Ez, Pz, N	Bz: 6:5, Ez: 14:30, Pz: 0:30, N: von der Unterliste auswählen		ug-$\ddot{A}_2$	Az: NaN, Fehlermeldung
Bz, Ez, Pz, N	Bz: 6:30, Ez: 143:, Pz: 0:30, N: von der Unterliste auswählen		ug-$\ddot{A}_2$, ug-$\ddot{A}_4$	Az: NaN, Fehlermeldung
Bz, Ez, Pz, N	Bz: 6,30, Ez: 14,30, Pz: 0,30, N: von der Unterliste auswählen		ug-$\ddot{A}_3$	Az: NaN, Fehlermeldung

nächste Seite

Tabelle 6.2 – vorherige Seite

Eingabename	Eingabewerte	gültige Äquivalenzklasse	ungültige Äquivalenzklasse	erwartete Ausgabe
Bz, Ez, Pz, N	Bz: 18:30, Ez: 6:30, Pz: 1:00, N: von der Unterliste auswählen		$ug\text{-}Ä_5$	Az: 11:00, Fehlermeldung
Bz, Ez, Pz, N	Bz: 6:30, Ez: 14:30, Pz: 1:00, N: leer auswählen		$ug\text{-}Ä_6$	Az: 7:00, Fehlermeldung
Bz, Ez, Pz, N	Bz: 6:30, Ez: 7:30, Pz: 1:00, N: von der Unterliste auswählen		$ug\text{-}Ä_6$	Az: 0:00, Fehlermeldung
Bz, Ez, Pz, N	Bz: 6:30, Ez: leer, Pz: 1:00, N: von der Unterliste auswählen		$ug\text{-}Ä_1$, $ug\text{-}Ä_6$	Az: NaN, Fehlermeldung
Bz, Ez, Pz, N	Bz: 6:30, Ez: 6:30, Pz: 1:00, N: von der Unterliste auswählen		$ug\text{-}Ä_6$	Az: NaN, Fehlermeldung
Bz, Ez, Pz, N	Bz: leer, Ez: leer, Pz: leer, N: von der Unterliste auswählen		$ug\text{-}Ä_6$	Az: leer, Fehlermeldung

Tabelle 6.2: Die ausgewählten Werte für den Testfall der Eingabeüberdeckung

6.3.3 Testfälle für die Ausgabeüberdeckung

Jede mögliche Ausgabe wird mindestens einmal getestet. In diesem Testfall wird anhand der Informationsmeldungen überprüft, ob nach dem Anklicken der unterschiedlichen Buttons die entsprechende Information, beispielsweise eine Fehlermeldung, dargestellt wird. Für diese Ausgabeüberprüfung werden die Buttons „Speichern“, „Speichern&&Prüfen“, „Exportieren&&Prüfen“, „Auswählen“ und „Öffnen“ verwendet. In der Tabelle 6.3 ist der Testfall für die Ausgabeüberdeckung dargestellt.

Button	Operation	geforderte Ausgabe	benötigte Eingabe
Speichern	anklicken	Fehlermeldung, das leere Feld wird gelb markiert.	Eine der Eingabe Datum, Auftragsnummer, Ansprechpartner oder Einsatzort ist leer.
Speichern	anklicken	Fehlermeldung, Felder werden gelb markiert.	zwei gleiche Namen treten in dem Tagesbericht auf.
Speichern	anklicken	Fehlermeldung	Eingabewerte entsprechen den ungültigen Äquivalenzklassen.
Speichern	anklicken	Meldung: Speichern ist erfolgreich.	Eingabewerte entsprechen den ungültigen Äquivalenzklassen.
Speichern &&Prüfen	anklicken	Fehlermeldung	Eingabewerte entsprechen den ungültigen Äquivalenzklassen.
Speichern &&Prüfen	anklicken	Fehlermeldung, Felder werden rot markiert.	Datensätze existieren in der Datenbank.
Speichern &&Prüfen	anklicken	Fehlermeldung, Felder werden gelb markiert.	zwei gleiche Namen treten im selben Tagesbericht auf.
Speichern &&Prüfen	anklicken	Meldung: Speichern ist erfolgreich.	Datensätze existieren noch nicht in der Datenbank Eingabewerte entsprechen den gültigen Äquivalenzklassen.
Exportieren &&Prüfen	anklicken	Fehlermeldung	Eingabewerte entsprechen den ungültigen Äquivalenzklassen.
Exportieren &&Prüfen	anklicken	Fehlermeldung, Felder werden gelb markiert.	zwei gleiche Namen treten im selben Tagesbericht auf.
Exportieren &&Prüfen	anklicken	Meldung: Speichern ist erfolgreich.	Eingabewerte entsprechen den gültigen Äquivalenzklassen.
Auswählen	anklicken	Fehlermeldung, das leere Feld wird gelb markiert.	Eine der Eingabe Datum, Auftragsnummer, Ansprechpartner oder Einsatzort ist leer.
Auswählen	anklicken	Fehlermeldung, keine Datensätze existieren.	Eine der Eingabe Datum, Auftragsnummer, Ansprechpartner oder Einsatzort ist leer.
Öffnen	anklicken	Fehlermeldung, Datensätze haben die Duplizierung.	In der Tagesbericht-Liste gibt es mehr als einen Datensatz.

Tabelle 6.3: Testfall für die Ausgabeüberdeckung

6.3.4 Test der Grenzfälle

Grenzfälle sind in diesem Oberflächenprogramm Situationen, bei denen nach dem Überschreiten eines bestimmten Wertes etwas in der Oberfläche anderes passiert als vorher. Oft werden diese Werte auch als „sprungfixe“ Werte bezeichnet, da sich die Reaktion auf entsprechende Änderungen ab diesem Wert sprunghaft ändert. Ein passendes Beispiel ist die Zahl der Mitarbeiter-Zeilen im Tagesbericht, in denen die Arbeitszeiten angegeben werden. Diese Tabelle hat den Ausgangswert 6 Zeilen. In der Oberfläche gibt es dort zusätzlich den Button „+“. Wird der Butten „+“ angeklickt, wird die Zahl der Zeilen der Tabelle von 6 auf 7 geändert, in dem eine neue Zeile eingefügt wird. Die aktuelle Zahl der Zeilen der Tabelle ist demzufolge danach 7. Dieser Testfall überprüft, ob der Datensatz der zusätzlichen Zeile nach „Speichern“ korrekt in die Datenbank geschrieben wird sowie ob nach dem Korrigieren des Tagesberichts der Datensatz der zusätzlichen Zeile in der Datenbank aktualisiert wird. Das Korrigieren des Datensatzes bedeutet, dass entweder die Werte dieses Datensatzes verändert oder der gesamte Datensatz gelöscht wird. Darüber hinaus können Eingabefehler in der zusätzlichen Zeile gefunden werden, indem überprüft wird, ob eine entsprechende Fehlermeldung ausgegeben wird. Die Tabelle 6.4 beschreibt diesen Grenzfall der Zeilenerweiterung von 6 auf 7 Zeilen detailliert.

6.4 Testfälle für die Methoden der Java-Klasse

Um die Methoden der Java-Klasse zu testen, wird JUnit verwendet. JUnit ist ein Testing-Framework für Java-Programme, das von Erich Kent Beck und Gamma entwickelt wurde. Mit den Java-Klassen werden die sich wiederholenden Teile beim Schreiben von Tests automatisiert definiert [Bec05, S.8].

Seite	Operation	Daten in der DB speichern	Daten von der DB korrigieren	Fehlermeldung	Eingabefehler
Tagesbericht.jsp Tagesbericht_kor.jsp Tagesbericht_exp.jsp	„+“ anklicken, Eingabewerte entsprechen den gültigen Äquivalenzklassen, „Speichern“ oder „Speicher&&Prüfen“ anklicken	Ja			
Tagesbericht.jsp Tagesbericht_kor.jsp Tagesbericht_exp.jsp	„+“ anklicken „Speichern“ oder „Speicher&&Prüfen“ anklicken	Nein		Ja	Eingabe-werte in ungültigen Äquivalenzklassen.
Tagesbericht.jsp Tagesbericht_kor.jsp Tagesbericht_exp.jsp	„+“ anklicken, „Speicher&&Prüfen“ anklicken	Nein		Ja	gleiche Namen in dem Tagesbericht eingetragen.
Tagesbericht.jsp Tagesbericht_kor.jsp	„+“ anklicken „Speichern“ oder „Speicher&&Prüfen“ anklicken	Nein		Ja	Datensatz hat in der DB bereits existiert.
Tagesbericht.jsp Tagesbericht_kor.jsp Tagesbericht_exp.jsp	„Öffnen“ anklicken Die Tabelle hat 7 Zeilen, korrigieren Werte von der Zeile 7, „Speichern“ oder „Speicher&&Prüfen“ anklicken		Ja		

Tabelle 6.4: Testfall für die Zeilenerweiterung

6.4.1 Vorteile für den JUnit-Test

Für die Verwendung von JUnit-Test sprechen folgende Vorteile:

- Es ist einfach zu benutzen. Ein JAR-Archiv junit.jar kann von der Webseite[1] heruntergeladen und dann *junit.jar* zum CLASSPATH hinzugefügt werden.
- Fehler sind früh erkennbar. Weil jede Methode der Klasse durch JUnit selbständig getestet wird, wenn eine Methode in der Klasse fertig geschrieben ist, kann diese Methode sofort getestet werden. Falls Fehler in der Methode gefunden werden, können diese Fehler auf diese Weise unmittelbar korrigiert werden.
- Testklassen und Klassen werden separat definiert. Beim Test werden die Anwendungscodes nicht geändert, da die Testcodes in der davon unabhängigen Testklasse beschrieben werden.
- Die Tests sind wiederverwendbar. Falls zwei Klassen ähnlich sind, können diese zwei Klassen in einer Testklasse geprüft werden.

6.4.2 Die angewendete Methoden in dem JUnit-Test

Weil Java-Klassen die Datenbankverbindung und die Durchführung der SQL-Befehlen beschreiben, werden assert-Methoden für den Test dieser Klasse verwendet. In der Testklasse wird assertEquals, assertNull und assertNotNull benutzt.

assertEquals: Diese Methode überprüft, ob zwei Objekte gleich sind. Durch diese Methode werden die Java-Klassen getestet, die für die Methoden für die Ausführung des INSERT-Befehles, des SELECT-Befehles und des UPDATE-Befehles benutzt wird. Wenn z. B. ein Datensatz in der Datenbanktabelle geschrieben wird, bekommt dieser Datensatz einen Hauptschlüssel zur Identifikation. Durch die Identifikation kann der geschriebenen Datensatz in der Datenbanktabelle eindeutig selektiert werden. Nach der Ausführung der INSERT-Befehle wird der eingegebene Datensatz mit dem Datensatz verglichen, der mit dem Hauptschlüssel in der Datenbanktabelle identifiziert wurde. Falls diese zwei Datensätze gleich sind, war die Ausführung der INSERT-Befehle erfolgreich.

assertNull: Diese Methode überprüft, ob das Objekt leer ist. Durch diese Methode wird die Java-Klasse *deleteData* getestet, die für die Methoden zur Ausführung des DELETE-Befehles benutzt wird. Die Idee ist ähnlich wie die zur Überprüfung der Ausführung des INSERT-Befehls, nach der Ausführung des DELETE-Befehls wird der Datensatz überprüft. Falls der Datensatz null ist, gibt assertNull den Wert WAHR zurück, anderenfalls den Wert FALSCH.

assertNotNull: Diese Methode überprüft, ob die Datenbankverbindung erfolgreich ist. Falls die Datenbankverbindung unterbrochen oder fehlerhaft ist, gibt assertNotNull den Wert WAHR zurück, anderenfalls den Wert FALSCH.

[1] https://github.com/KentBeck/junit/downloads

Listing 6.1 JUnit-Test für die Ausführung des INSERT-Befehles

```
1. import org.junit.Assert.assertEquals;
2. import org.junit.Test
3. import com.bean.ZeitTB
4. import org.junit.BeforeClass;
5.
6. public class insertDataTest
7. {
8.     @Before class
9.     public static void init()
10.    {
11.            insertData = new InsertData();
12.    }
13.
14.    private void compareZeit(ZeitTB zeit1, ZeitTB, zeit2)
15.    {
16.            assertEquals(zeit1.getBeginn(), zeit2.getBeginn());
17.            assertEquals(zeit1.getEnde(), zeit2.getEnde());
18.            assertEquals(zeit1.getPause(), zeit2.getPause());
19.    }
20.    @Test
21.    public void testInsertZeit()
22.    {
23.            ZeitTB zeit1 = new Zeit(6.00, 14.00, 0.30);
24.            insertData.insert(zeit1);
25.            ZeitTB zeit2 = insertData.getZeitByMaxId();
26.            this.compareZeit(zeit1, zeit2);
27.    }
28 }
```

Listing 6.1 stellt einen Beispielcode für den Test der Ausführung des INSERT-Befehles dar.

In diesem Beispiel wird die Methode `InsertZeit()` von der Klasse „*insertData*“ getestet. In der Datenbanktabelle „*zeitTB*“ stellt die „*Id_Zeit*“ den automatisch inkrementierten Hauptschlüssel dar. Angenommen, ein Datensatz wird in die Datenbanktabelle geschrieben, dann existiert dieser Datensatz nach der Ausführung der Methode `InsertZeit()` in der Tabelle „*zeitTB*“. Weil dieser Datensatz gerade neu in die Datenbanktabelle eingefügt wurde, wird der Identifikation dieses Datensatzes in der Tabelle der in diesem Moment höchste Wert zugeordnet. Durch den Aufruf der Methode `getZeitByMaxID()` kann der Datensatz mit dem höchsten Wert der Identifikation in der Tabelle eindeutig identifiziert werden, was dem zuletzt eingefügten Datensatz entspricht. Wird dann in einem Vergleich festgestellt, dass diese zwei Datensätze gleich sind, wurde die Methode `InsertZeit()` erfolgreich ausgeführt, anderenfalls nicht.

Die Tabelle 6.5 werden Ergebnisse für den JUnit-Test dargestellt. Das Symbol „-“ zeigt an, dass die Methode nicht angewendet wird.

Methode / Klassename	assertEquals	assertNull	assertNotNull
TBDataInsert	Ja	-	-
TBDataDelete	-	Ja	-
TBDataUpdate	Ja	-	-
TBNeuSelect	Ja	-	-
TBGenSelect	Ja	-	-
TBExpSelect	Ja	-	-
ExcelExportDataSelect	Ja	-	-
IdSelect	Ja	-	-
ListDataSelect	Ja	-	-
DbConnection	-	-	Ja

Tabelle 6.5: Ergebnisse für den JUnit-Test

6.5 Weitere Testfälle

Aufgrund von Eigenschaften der verwendeten Funktionen zum Auslesen der Daten aus der jeweiligen Oberfläche werden die eingetragenen Werte beim Auslesen aus der Oberfläche entfernt. Im Falle eines festgestellten Fehlers müssen diese eingetragenen Werte deshalb aktiv wieder in die noch geöffnete Oberfläche eingetragen werden. Bei diesem Test soll die folgende Fragestellung untersuchen.

- Ob im Falle von Eingabefehlern nach dem Anklicken des Buttons die eingetragenen Werte zurück in der Oberfläche geschrieben werden.
- Ob, wenn die eingetragenen Datensätze bereits in der Datenbank existieren, nach dem Anklicken des Buttons die eingetragenen Werte zurück in die Oberfläche geschrieben werden.

Sind die Ergebnisse dieser Testfälle positiv, weil es Fehler bei der Eintragung gibt, werden die Werte nach dem Anklicken des entsprechenden Buttons zurück auf die Oberfläche geschrieben.

7 Bewertung der erarbeiteten Ergebnisse

In diesem Kapitel werden die Weiterentwicklung der bisher in der ISAG verwendeten, Dokumenten-gesteuerten Projektabwicklung sowie ein darauf aufbauendes, neues Konzept der Projektsteuerung, nämlich eine Prozess- und Dokumenten-gesteuerte Projektabwicklung hinsichtlich der in der Einleitung beschriebenen Ziele der Diplomarbeit bewertet.

7.1 Problemstellung dieser Arbeit

Die Firma ISAG verwendet bisher dokumentenbasierte Vorgehensweisen für die Projektbearbeitung, bei denen sie hauptsächlich Microsoft Office Programme benutzt. Aufgrund des steigenden Aufwands zur Fehlersuche und -behebung sowie des zunehmenden Auftragsvolumens wurde von der ISAG eine Weiterentwicklung der Prozessabwicklung mit den Schwerpunkten a) Verringerung der unterschiedlichen Fehlermöglichkeiten und b) Reduktion des Fehlerentdeckungsaufwandes gefordert und darüber hinaus aus Effektivitätsgründen eine Reduktion des Arbeitsaufwandes zur Projektabwicklung gewünscht.

Das Ziel dieser Diplomarbeit war es deshalb, die unterschiedlichen Fehlermöglichkeiten zu verringern und den Fehlerentdeckungsaufwand sowie den Arbeitsaufwand zur Projektabwicklung zu reduzieren. Dazu war eine strukturierte und redundanzfreie Ablage der Projektdaten in einer Datenbank zu erstellen, mit der dann effektiv Konsistenzprüfungen der einzugebenden Projektdaten durchgeführt werden können. Passend zu den Anforderungen der ISAG sollten Benutzeroberflächen entwickelt werden, die auf die Projektdaten abgestimmte Eingabe-Datenmasken haben und so insbesondere die Gefahr von Eingabefehlern reduzieren. Zusätzlich sollten in diese Oberflächen Funktionen zur Konsistenzprüfung zu den bereits in der Datenbank befindlichen Projektdaten implementiert werden. Dazu muss selbstverständlich eine Verbindung zwischen den Benutzeroberflächen und der obengenannten Datenbank erstellt werden, sogenannte Konnektoren. damit die Daten in die Datenbank importiert oder aus der Datenbank exportiert werden können. Aufgrund der leichteren Erlernbarkeit sollen die zu entwickelnden Benutzeroberflächen ähnlich wie die bisher verwendete prototypische Excel-Vorlage bzw. der Tagesbericht sein, so dass Mitarbeiter die Funktionen der Benutzeroberflächen intuitiv erkennen können. Eine besondere Schwierigkeit ist es dabei, dass der Tagesbericht viele komplizierte Tabellen enthält. Eine weitere Schwierigkeit besteht darin, dass die verschiedenen Projektdokumente, wie Tagesbericht und Gesamtrechnung, in unterschiedlichen Versionen von Microsoft Excel vorliegen.

7.2 Design und Funktion der Benutzeroberflächen

Die Weiterentwicklung der Dokumenten-gesteuerten Projektabwicklung erforderte zunächst die Erstellung einer zentralen Steuer-Oberfläche, aus der heraus alle erforderlichen Funktionen gestartet und die damit verbundenen Benutzeroberflächen geöffnet werden können. Diese Steuer-Oberfläche, das sogenannte Hauptfenster ist in der Abbildung 7.1 dargestellt. Darin sind die zur Steuerung erforderlichen Buttons „neue TB erstellen“, „gespeicherte TB korrigieren“, „freigegebene TB korrigieren“, „freigegebene TB exportieren“ und „exportierte TB korrigieren“ implementiert. Diese Funktionen sind zur Bearbeitung von Tagesberichten in der ISAG zwingend notwendig.

Abbildung 7.1: Benutzeroberfläche für das Hauptfenster

Die drei Funkionen zum Korrigieren stellen unterschiedlich komplexe Korrekturmöglichkeiten dar. Bei „gespeicherte TB korrigieren“ werden lediglich vorab ohne Konsistenzprüfung gespeicherte, also dem Datenstatus nach „neue“ Tagesberichtdaten korrigiert. Wurden die Daten hingegen schon freigegeben, so wird mit einer Korrektur durch „freigegebene TB korrigieren“ diese Freigabe wieder zurückgenommen, wobei der Status der Daten von „genehmigt“ auf „neu“ zurückgesetzt werden muss. Noch gravierender sind die Änderungen, wenn bereits in die Gesamtrechnung exportierte Daten geändert werden müssen. Um die Konsistenz der Daten in der Gesamtrechnung zur Datenbank zu garantieren, muss die Korrektur von der Benutzeroberfläche aus mit „exportierte TB korrigieren“ gesteuert werden. Hierbei werden die zu korrigierenden Daten nicht nur in der Datenbank korrigiert, sondern beim nächsten Aufruf von „freigegebene TB exportieren“ auch in der Gesamtrechnung korrigiert. Nach dem Anklicken des jeweiligen Buttons wird die entsprechende Benutze-

roberfläche geöffnet, in der die Tagesberichte je nach Aufgabenstellung unterschiedlich bearbeitet werden.

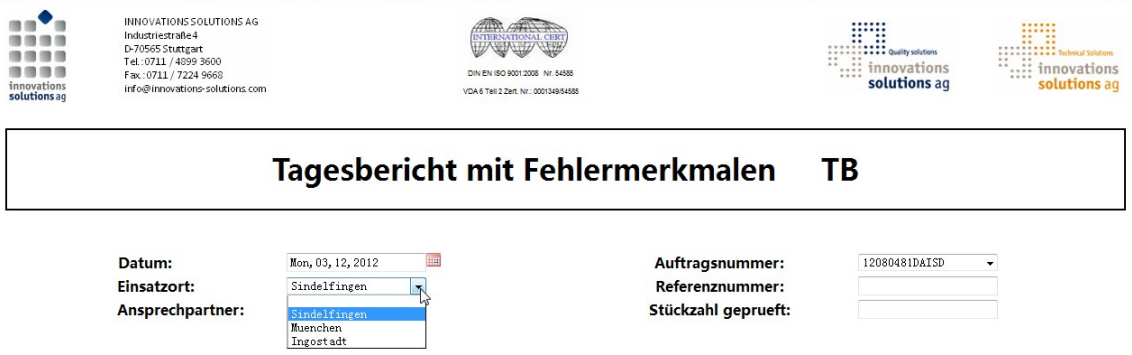

Abbildung 7.2: Der obere Teil von der Benutzeroberfläche des Tagesberichtes

In der Abbildung 7.2 wird ein Teil der Benutzeroberfläche für den Tagesbericht dargestellt. Das Datum wird aus dem Kalender, die Auftragsnummer, der Einsatzort und der Ansprechpartner werden aus der Unterliste ausgewählt. Zu einer Auftragsnummer können verschiedene Einsatzorte und Ansprechpartner gehören.

Wenn also die Auftragsnummer ausgewählt wird, sind damit im Allgemeinen der Einsatzort und der Ansprechpartner nicht identifiziert. Allerdings ist damit die in Frage kommende Menge von Einsatzorten und Ansprechpartnern schon deutlich reduziert. Somit müssen nicht alle Einsatzorte und Ansprechpartner in der Unterliste zur Auswahl bereitgehalten werden, sondern nur die zur Auftragsnummer passenden Einsatzorte und Ansprechpartner.

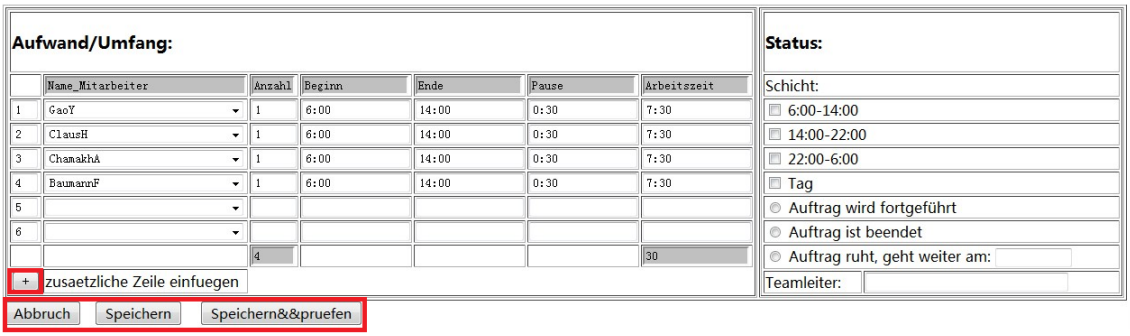

Abbildung 7.3: Der untere Teil der Benutzeroberfläche des Tagesberichts

In der Abbildung 7.3 wird der untere Teil der Benutzeroberfläche des Tagesberichtes angezeigt. Wenn es mehr als 6 Datensätze gibt, muss der Button „+“ angeklickt werden, wodurch eine zusätzliche Zeile nach der Zeile „6“ eingefügt wird. In den bisher verwendeten, in Papier vorliegenden Tagesberichtsformularen gibt es dafür ein Extrablatt. Die Zusammenfassung aller Mitarbeiter-Datensätze auf einer Oberfläche vergrößert jedoch die Übersichtlichkeit und

erlaubt ein einfacheres Datenhandling. Die gesamte Arbeitszeit und die gesamte Anzahl an Mitarbeitern werden durch den Aufruf der Funktion in der letzten Zeile dargestellt. Dies entspricht dem bisher verwendeten Excel-Formular und ist eine geeignete Plausibilitätsprüfung der Eingabedaten. Im Normalfall sind der eingebende Mitarbeiter und der überprüfende Mitarbeiter verschiedene Personen. In diesem Fall müssen die Daten des Tagesberichts nach der Eingabeprüfung durch den Anklicken des Buttons „Speichern“ als nicht-konsistenzgeprüfte Daten in der Datenbank mit dem Status „neu“ gespeichert werden. Dies erlaubt darüber hinaus das Ablegen unvollständiger oder unplausibler Tagesberichtsdaten und erhöht damit die Eingabe-Effektivität deutlich. In dem Fall, dass der eingebende Mitarbeiter und der überprüfende Mitarbeiter dieselbe Person sind, kann der Tagesbericht direkt überprüft und genehmigt werden. Dazu wird der Button „Speichern&&Prüfen“ verwendet, der jedoch nur für Personen mit dieser speziellen Berechtigung funktionieren darf. Wird der Button „Speichern&&Prüfen“ angeklickt, werden die Daten nach der Eingabeprüfung und der Konsistenzprüfung in der Datenbank gespeichert. Wenn der Button „Abbruch“ angeklickt wird, wird diese Benutzeroberfläche geschlossen, ohne dass Daten in die Datenbank geschrieben werden.

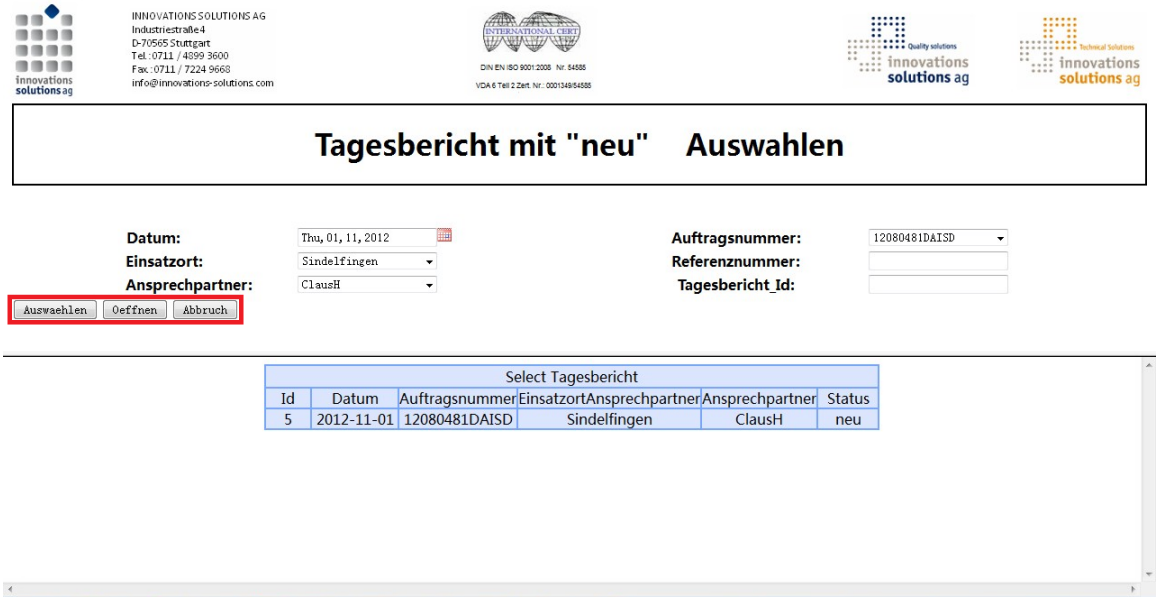

Abbildung 7.4: Der untere Teil der Benutzeroberfläche des Tagesberichts

Die Abbildung 7.4 zeigt, wie ein Datensatz eines in der Datenbank gespeicherten Tagesberichtes ausgewählt werden kann. Wenn der Button „Auswählen“ nach der Eingabe der Identifikationsmerkmale angeklickt wird, werden die dazu passenden Datensätze der Tagesberichte in der unteren Tabelle dargestellt. Solange noch mehrere Datensätze angezeigt werden, müssen weitere Identifikationsmerkmale eingegeben werden. Wenn schließlich nur noch ein Datensatz in der Tabelle angezeigt wird, kann der Button „Öffnen“ angeklickt werden, woraufhin die Daten dieses Tagesberichts auf der entsprechenden Benutzeroberfläche dargestellt werden, die dazu auf dem Bildschirm erscheint.

Zur Erstellung der Benutzeroberflächen wurde JSP verwendet, weil HTML eine Webprogrammiersprache ist. Im Gegensatz zu einer HTML Seite, die statische Inhalte enthält, können dynamische Inhalte bzw. Berechnungsfunktionen mit statischen Inhalten gemischt werden [Ber07, S.3]. Dies ist erforderlich, weil viele Tabellen des Tagesberichtes Berechnungsfunktionen enthalten, die auch auf den entsprechenden Benutzeroberflächen vorhanden sein sollen. Darüber hinaus kann JSP von allen Webbrowsern aufgerufen werden. Somit sind die hier erstellten Benutzeroberflächen auf allen Betriebssystemen lauffähig, die Webbrowser unterstützen, also nicht nur PCs und Laptops sondern auch Tablet-PCs und Smartphones beliebiger Hersteller. Die Benutzeroberflächen sind also für alle Mitarbeiter, auf fast beliebiger Hardware und mit einer VPN-Verbindung sogar überall verfügbar. Eine VPN-Verbindung ist ein Virtuelles Privates Netzwerk, mit dem zwei Computer verschlüsselt über das Internet. Außerdem können viele Java-APIs in der JSP Seite aufgerufen werden, womit Daten aus der Datenbank direkt von den Benutzeroberflächen aus abgerufen werden können. Die Gesamtheit dieser Maßnahmen bietet bereits ein relativ großes Potential zur Effizienzsteigerung in der ISAG und erfüllt damit die Anforderungen der ISAG bereits recht gut.

7.3 Prozess- und Dokumenten-gesteuerte Projektabwicklung

Um die Projektabwicklung besser steuern und dennoch die Flexibilität der Dokumentengesteuerten Vorgehensweise erhalten zu können, wurde das neue Konzept der Prozess- und Dokumenten-gesteuerten Projektabwicklung entwickelt. Dabei werden die Geschäftsprozesse durch die Process Engine gesteuert, indem Mitarbeiter über innerhalb eines Prozesses auszuführende Aufgaben informiert werden. So wird von der Process Engine überwacht, dass der vorab definierte Projektablaufplan eingehalten wird. Die Process Engine definiert dabei, welche Prozessschritte als nächstes durchzuführen sind und welche Voraussetzungen dafür gegeben sein müssen. Zu Effizienz-Analysezwecken kann auch protokolliert werden, wie lange die Abarbeitung jedes Prozessschrittes gedauert hat [FR10, S.6-8]. Mit diesen Informationen können die Projektprozesse gezielt optimiert werden. Die Geschäftsprozesse werden in dieser Arbeit mit BPMN 2.0 modelliert und durch BOS ausgeführt. Jeder Prozessschritt wird dabei als ein Arbeitspaket betrachtet. So ein Arbeitspaket bezeichnet zum Beispiel, wie ein Tagesbericht bearbeitet wird, d. h. ob ein Tagesbericht in diesem Prozessschritt erstellt, korrigiert oder exportiert werden soll.

Die Prozessorientierung berücksichtigt die Interaktion zwischen den Geschäftsprozessen und Ein- und Ausgaben dieser Geschäftsprozesse. In der Projektabwicklung soll eine Interaktivität zwischen den Geschäftsprozessen und dem Prozessbeteiligten hergestellt werden, d. h. eine Oberfläche soll in der Aktivität definiert werden. Beispielsweise werden durch den <iFrame> Tag Benutzeroberflächen in den entsprechenden Aktivitäten eingebettet. Damit können die Prozessbeteiligten ihre Aufgaben erledigen. Das hier entwickelte Konzept der Prozess- und Dokumenten-gesteuerten Projektabwicklung wurde zunächst pilothaft implementiert und anschließend hinsichtlich der Anwendbarkeit in der Projektabwicklung analysiert. Die noch verbliebene Anforderung der ISAG, nämlich den für die Projektabwicklung erforderlichen Arbeitsaufwand zu reduzieren, wird damit also ebenfalls erfüllt.

7.4 Aufwand für Programmierung und Bedienung

Um die Wirtschaftlichkeit einer Verbesserungsmaßnahme abschätzen zu können, müssen mehrere unterschiedliche Aufwände betrachtet werden. Neben dem reinen Entwicklungs- und Programmieraufwand sind auch noch die Aufwände beim Einlernen in das neue Programm, der Bedienungsaufwand und der Wartungs- bzw. der Weiterentwicklungsaufwand zu berücksichtigen. Je strukturierter und in sich abgeschlossener die einzelnen Prozessschritte sind, umso weniger Vorbildung müssen die betreffenden Mitarbeiter haben, was für eine Firma letztlich niedriger qualifizierte Mitarbeiter und somit günstigere Stundenlöhne bedeutet. All diese Aufwände übersichtlich in ein Diagramm zu bringen, ist jedoch recht schwierig.

In der Norm ISO/IEC 9126 [ISO01] wird die Evaluation der Softwarequalität standardisiert. In diesem Standard wird lediglich die Bedienbarkeit als Kriterium definiert. Der Bedienaufwand bezeichnet hier den Aufwand für Anwendung der verschiedenen Konzepte. Die Abbildung 7.5 zeigt, wie hoch der Aufwand für die Dokumenten-gesteuerte Projektabwicklung, die Prozess-gesteuerte Projektabwicklung und die neu entwickelte Kombination der beiden Methoden ist.

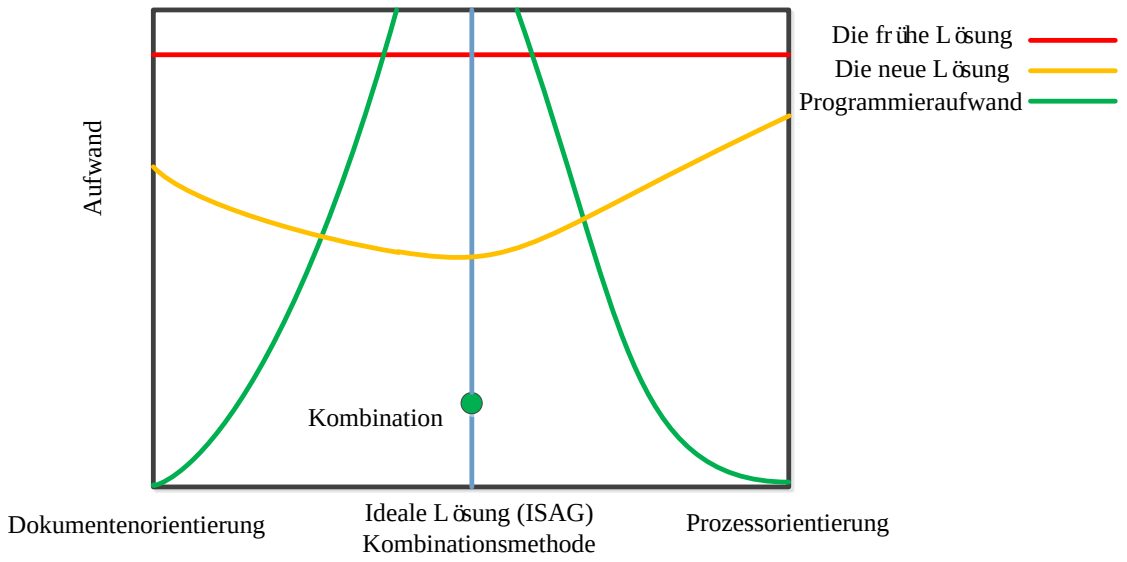

Abbildung 7.5: Bedienaufwand und Programmierungsaufwand für Vorgehensweisen der Projektabwicklung [Bau12]

Die blaue, vertikale Achse bezeichnet den Bedienaufwand für die ideale Lösung bzw. die Kombinationsmethode. Die geschätzten Aufwände für die dokumentierte und die prozessorientierte Vorgehensweise werden durch die linke und die rechte Achse dargestellt. Die rote Kurve kennzeichnet den bisherigen Bedienaufwand bei der Projektabwicklung in der ISAG.

Sie stellt damit eine obere Grenze dar, die von neuen Methoden auf jeden Fall unterschritten werden sollte, damit die neuen Lösungen überhaupt eine Verbesserung der Wirtschaftlichkeit darstellen. Die gelbe bzw. orangefarbene Kurve repräsentiert den Bedienaufwand, der für die ideale Lösung natürlich ein deutliches Minimum hat. Der Bedienaufwand für reine Prozessorientierung steigt wieder auf ein ähnliches Niveau an, wie die reine Dokumentenorientierung. Das liegt daran, dass es einen sehr aufwändigen Projektablaufplan erfordert, um Eingangsdaten, die weder zeitlich gemeinsam noch geordnet sequenziell vorliegen sondern in zufälliger Reihenfolge eintreffen können und auch so abgearbeitet werden sollten, überhaupt in einer Prozess-gesteuerter Projektabwicklung zu bearbeiten. Die grüne Linie stellt die Summe aus Programmieraufwand und zu erwartenden Wartungs- und Weiterentwicklungsaufwand dar. Der linke und der rechte Ast der grünen Kurve gehen bei der Annäherung an die blaue Linie in Richtung sehr hoher Aufwände, wie es bereits in der Arbeit von Baumann [Bau12] gezeigt wurde. Der Programmieraufwand für die Kombination von Dokumenten-gesteuerter und Prozess-gesteuerter Projektabwicklung wird durch den grünen Punkt auf der blauen Linie repräsentiert. Zwar liegt dieser Punkt etwas höher als die bei reiner Dokumenten-gesteuerter bzw. Prozess-gesteuerter Projektabwicklung, da der Bedienaufwand dort jedoch ein Minimum hat, ist der Gesamtaufwand dennoch niedriger.

7.5 Herausforderung

Bei der Implementierung der Benutzeroberflächen und der Auswahl der Process Engine werden einige Herausforderungen aufgetroffen.

- die unbeständige Datenbankstruktur

  Die Entwicklung der Benutzeroberflächen ist abhängig von der angegebenen Datenbankstruktur. Aufgrund der neuen Anforderungen seitens der Firma ISAG wird die Datenbankstruktur immer wider geändert. Wenn die Datenbankstruktur nicht bestätigt wird, entsteht ein Konflikt in der Implementierung der Benutzeroberflächen. Ein Konflikt bezeichnet verschiedene Meinungen [WE08, S.20], z. B. durch eine Funktion der Benutzeroberfläche die Daten von der Datenbanktabelle auswählt, wobei für den Falls, dass ein Attribut in dieser Datenbanktabelle geändert wird, diese Funktion neu geschrieben wird, womit die Zeitkosten und der Programmaufwand sehr hoch und viele Arbeiten wiederholt werden müssen.

- fehlende Testdaten

  Weil die Datenbankstruktur neu hergestellt wird, enthalten die Datenbanktabellen keine Daten, um Funktionen der Benutzeroberflächen zu überprüfen. Es werden jedoch viele Testdaten benötigt. Wegen der fehlenden Testdaten werden manche Funktionen nur unzureichend getestet, mit der Folge, dass in der Anwendung für die Firma ein Risiko entsteht unerwartete Handlungen oder negative Ereignisse [Gau04, S.3], z. B. Verlust von Daten des Tagesberichtes, falsche Berechnung von Arbeitszeiten und damit Lohn usw.

- die stetige aktualisierte Process Engine

  Die Process Engine wird über einen Zeitraum bei den Entwicklern aktualisiert. Nach der Aktualisierung werden viele Funkionen der Process Engine verbessert, wenn Geschäftsprozesse in die neue Process Engine eingesetzt werden, sollen Process Engine und Apache Tomcat neu konfiguriert werden. Definierte globale Variablen und Konnektoren müssen überprüft werden, damit Geschäftsprozesse im neuen Process Engine laufen können. Die Wartungskosten werden erhöht.

7.6 Erweiterung

In den Benutzeroberflächen sollen einige Funktionen verbessert werden, z. B. werden Arbeitszeiten in der Datenbanktabelle als „BigDecimal“ definiert. Wenn die Arbeit die Frühschicht oder Spätschicht ist, ist die Arbeitszeit gleich wie die Endzeit - die Beginnzeit - die Pausenzeit. Wenn die Arbeit die Nachtschicht ist, ist die Arbeitszeit gleich wie 24 – (Endzeit - die Beginnzeit - die Pausenzeit). Aber wenn die Arbeit Spätschicht ist, wird die Arbeitszeit mit dem Datum undeutlich identifiziert. Um dieses Problem zu erledigen, sollen Arbeitszeiten werden in der Datenbanktabelle als „Datetime“ definiert werden, Arbeitszeiten werden so deutlich dargestellt. Buttons werden in den Benutzeroberflächen kontrolliert, d. h. zwei Tagesberichte können nicht gleichzeitig bearbeitet werden. Wenn Button „neu TB erstellen“ in den Hauptfenster (Abb. 7.1) angeklickt wird, sollen andere Buttons nicht geschaltet werden. Wenn die Funktion „neu TB erstellen“ geschaffen wird, werden andere Buttons wieder aktiviert. Um die Buttonkontrolle durch Programme zu realisieren, soll die Eigenschaft „*disabled*“ in dem Button erstellt werden. Falls „*disabled*“ gleich wahr ist, wird der Button nicht aktiviert. Außerdem soll eine Anmeldungsbenutzeroberfläche erstellt werden, durch die Zugriffsrechte für die Buttonkontrolle vergeben werden können.

8 Zusammenfassung und Ausblick

In diesem letzten Kapitel werden die wesentlichen Ergebnisse und Erkenntnisse zusammengefasst und ein Ausblick gegeben, in dem die weiteren Entwicklungs-Ansätze und Möglichkeiten skizziert werden, die sich aus dieser Arbeit ableiten lassen. Diese Arbeit in dem praxisbezogenen Forschungsgebiet der Projektabwicklung wurde in Zusammenarbeit mit der Firma Innovations-Solutions AG (ISAG), also einem Kooperationspartner aus der Industrie, durchgeführt. Ausgangspunkt war die Anforderung der ISAG, ihre bisherige Dokumenten-gesteuerte Projektabwicklung, die mit Hilfe von Excel-Dokumenten durchgeführt wird, durch eine weiterentwickelte Prozessabwicklung mit folgenden Schwerpunkten zu ersetzen: Verringerung der unterschiedlichen Fehlermöglichkeiten, Reduktion des Fehlerentdeckungsaufwandes und Reduktion des Arbeitsaufwandes zur Projektabwicklung sowohl quantitativ als auch qualitativ.

Zusammenfassung

Die Einleitung in das Thema dieser Arbeit, die Darstellung der Motivation und der Ziele dieser Arbeit sowie eine kurze Angabe der Inhalte der nachfolgenden Kapitel erfolgt in Kapitel 1. Insbesondere wird auf die zwei grundlegend unterschiedlichen Vorgehensweisen für die Projektabwicklung eingegangen: die Dokumenten-gesteuerte und die Prozess-gesteuerte Projektabwicklung. Aus den Anforderungen der ISAG und der dort hauptsächlich stattfindenden Art von Projekten werden die obengenannten Ziele dieser Diplomarbeit abgeleitet.

Im Kapitel 2 wird auf die für diese Arbeit notwendigen Grundlagen eingegangen. Detailliert werden die Geschäfts- und Projektprozesse, die Workflow Management Systeme inklusive der darin enthaltenen Process Engine, sowie Prozessmodelle und Notationen beschrieben. Aufgrund der Anforderungen der ISAG scheinen Datenbank-basierte Webanwendungen ein geeignetes Werkzeug zu sein, weshalb kurz auf ihre Grundlagen eingegangen wird.

Die unterschiedlichen Konzepte für Benutzeroberflächen werden in Kapitel 3 analysiert und bewertet. Aus den Anforderungen an zukünftige Projektabwicklungen werden Anforderungen an die zu erstellenden Benutzeroberflächen sowie die zu beachtenden Randbedingungen abgeleitet. Dann werden die unterschiedlichen Benutzeroberflächen-Konzepte, also auf Basis von Microsoft Excel mit VBA-Makros, auf Basis von Microsoft Excel und Java-Programmen sowie auf Basis von Webanwendungen einander gegenüber gestellt. Die Notwendigkeit der Verwendung einer Datenbank zur Vermeidung von Fehlern durch redundante Daten ergibt sich bei allen drei Konzepten. Weil die Bearbeitung der Tagesberichte in der Firma ISAG durch Microsoft Office Excel realisiert wird, war es naheliegend, die Bearbeitung der

Tagesberichte in den Excel-Dateien zu belassen. Das Konzept auf Basis von Microsoft Excel mit VBA-Makros scheidet jedoch wegen der aufwändigen Implementierung im Office-Paket und dem großem Aktualisierungs- und Wartungsaufwand aus. Der Vergleich der Konzepte auf Basis von Microsoft Excel und Java-Programmen sowie auf Basis von Webanwendungen erforderte jedoch tiefere Analysen.

Das Benutzeroberflächen-Konzept auf Basis von Microsoft Excel und Java-Programmen zum Speichern und Abrufen der Daten in der Datenbank scheint aus den oben genannten Gründen zunächst geeignet zu sein. Die beispielhafte Entwicklung dieses Konzepts zeigt jedoch, dass sich Konsistenzprüfungen nicht oder nur sehr schwer in dieses Konzept implementieren lassen. Weiterhin ist dieses Konzept relativ unflexibel im Hinblick auf Weiterentwicklungen. Deutlich geeigneter erscheint deshalb das Benutzeroberflächen-Konzept auf Basis von Webanwendungen. Dieses Konzept ermöglicht auf sehr einfache Weise die Realisierung von detaillierten und umfassenden Konsistenzprüfungen sowie die unkomplizierte Fehlerbehandlung, wenn eine Inkonsistenz festgestellt wurde. Der Export der in der Datenbank vorhandenen Daten in die bisher zur Rechnungserstellung verwendeten Excel-Dateien oder PDF-Dateien ist ebenfalls unkompliziert möglich. Dies ermöglicht es, das Konzept auf Basis von Webanwendungen in die bisherige Arbeitsumgebung zu integrieren und so die Projektabwicklung der ISAG schrittweise umzustellen. Der Vergleich der verschiedenen Benutzeroberflächen-Konzepte zeigt, dass die Webanwendungen die mit Abstand größten Vorteile bei nur geringen Nachteilen haben.

Im Kapitel 4 werden die Benutzeroberflächen als Webanwendungen implementiert. Hierbei stellte sich heraus, dass sich das in [Bau12] angegebene Datenbankschema nicht schlüssig mit den erforderlichen Benutzeroberflächen verbinden lässt. Deshalb wurde ein auf ein Minimum reduziertes und klar strukturiertes, jedoch gleichzeitig flexibles und für alle in der ISAG relevanten, zukünftigen Anforderungen erweiterbares Datenbankschema entwickelt. In dieses Datenbankschema lassen sich mit entsprechenden Erweiterungen selbstverständlich alle Daten integrieren, die in [Bau12] angegeben sind. Die Konnektoren zwischen den Benutzeroberflächen und der Datenbank zum Lesen und Schreiben der Microsoft Office-Dateien durch Apache POI, zum Import der Daten von Microsoft Excel 2003 in die Datenbank und dem Export von Daten von der Datenbank zurück nach Microsoft Excel 2003 wurden neu erstellt und detailliert erläutert. Auch auf die Funktionsweise der neu entwickelten Benutzeroberflächen für die Eingabe, Korrektur, Überprüfung und den Export der Daten der Tagesberichte wird eingegangen. Hierbei wird insbesondere die Programmierung der Benutzer-Interaktionen im Zusammenhang mit Datumsangaben und mit Konsistenzfehlern beschrieben, da diese Funktionen die Benutzerfreundlichkeit wesentlich mitbestimmen und helfen, die Fehlermöglichkeiten bei der Verwendung dieses neuen Konzepts der Projektabwicklung zu reduzieren.

Das Kapitel 5 beschreibt ein neu entwickeltes Konzept für eine gesamtheitliche Prozess-Steuerung der Projektabwicklung, die jedoch in einzelnen Projektabschnitten eine Dokumenten-Steuerung zulässt. Die Diskussion der prinzipiellen Eigenschaften der Process Engines zeigt, dass die in der ISAG benötigte Projektabwicklung mit einer reinen Prozess-Steuerung nicht mit vertretbarem Aufwand realisierbar ist. Um die Vorteile der

Prozess-Steuerung mit denen der Dokumenten-Steuerung zu verbinden, wird das Konzept einer gestuften Anwendung von Process Engine und Servlet Engine entwickelt. Die Kommunikationswege zwischen den Benutzern, den Engines und der Datenbank werden beschrieben. Daraus werden Anforderungen an die Process Engine abgeleitet, deren Erfüllungsgrade bei der anschließend beschriebenen Auswahl der geeignetsten Process Engine detailliert analysiert, bewertet und miteinander verglichen werden. Die Anforderungen bzw. Bewertungskriterien sind dabei die Verfügbarkeit, die Einbettbarkeit, die Steuerbarkeit, die Integrierbarkeit, die Testbarkeit, die Komplexität und die Benutzbarkeit. Der Vergleich zeigt schließlich, dass die beiden Process Engines Bonita Open Solution (BOS) und Activiti einfach mit der Open-Source-Software Apache Tomcat (Servlet Engine) konfigurierbar sind. Die Ausführung -und Verwaltung der Geschäftsprozesse sind bei beiden ausgezeichnet, lediglich die Steuerbarkeit und damit die Einbindung bereits vorhandener Webformulare ist bei BOS besser. Für erfahrenere Java-Entwickler, die das Programm längere Zeit betreuen bzw. weiterentwickeln und spezielle Java-Funktionen einsetzen wollen, ist Process Engine Activiti eine gute Wahl. Für die ISAG ist die Bonita Open Solution in Summe die bessere Wahl, weil sich die Programmierung im Wesentlichen auf die Einbindung größtenteils bereits vorhandener Benutzeroberflächen beschränkt. Darüber hinaus ist die Programmierung bei der Wartung und Weiterentwicklung dieser Software durch andere Personen einfacher nachzuvollziehen.

Die Definition und Implementierung der Programmtests wird in Kapitel 6 dargestellt. In dem neu entwickelten Konzept der gestuften Anwendung von Process Engine und Servlet Engine zur Steuerung der Kommunikation zwischen den Benutzern und der Datenbank werden mehrere unterschiedliche Programmiersprachen und Programmsysteme verwendet. Dies erschwert das Testen des erstellten Programmkodes erheblich. Deshalb wird in dieser Arbeit ein Testkonzept entwickelt, das optimal auf die unterschiedlichen Eigenschaften der verwendeten Werkzeuge abgestimmt ist. Der Test der SQL-Befehle wird direkt in MySQL[1] durchgeführt. Dabei wird überprüft, ob die Daten richtig in die Datenbank eingefügt, aktualisiert oder ausgelöscht werden. Die Benutzeroberflächen werden in dem Webbrowser Firefox getestet, weil Firefox das Web-Entwicklungs-Tool Firebug enthält, durch das Fehler im Quellcode einer Webseite angezeigt werden können. Der teilweise eingebettete Java-Code wird im Eclipse-Debugger überprüft. Um die für die Konnektoren verwendeten Methoden der Java-Klasse zu testen, wird JUnit verwendet. Da es derzeit kein geeignetes Werkzeug zum Testen des Code von Javascript und jQuery gibt, muss man den Wert jedes Parameters der Funktion durch die Funktion *alert*() im Webbrowser anzeigen und auf Korrektheit überprüfen. Zum Testen der Ein-und Ausgaben der Oberfläche wird die Vorgehensweise „Black-Box-Test“ verwendet. Der Testaufwand wird jedoch reduziert, indem sogenannte Äquivalenz-Klassen gebildet werden, die alle äquivalenten Testbedingungen in einer Klasse zusammenfassen.

In Kapitel 7 wird auf die Bewertung der erarbeiteten Ergebnisse eingegangen und der beim praktischen Einsatz zu erwartende Nutzen abgeschätzt. Die Ergebnisse werden dabei aus der Sicht eines Anwenders beschrieben, weil nur dadurch der direkte Mehrwert

[1] http://www.mysql.de/

für den Kooperationspartner, also die ISAG, sichtbar wird. Als Basis für diese Bewertung wird zunächst einmal die der Arbeit zugrunde liegende Problemstellung dargestellt. Danach werden das Design und die Funktionen der erstellten Benutzeroberflächen erläutert. Insbesondere wird dargelegt, weshalb und wozu die unterschiedlichen Funktionen zur Tagesbericht-Bearbeitung zwingend gebraucht werden. Anschließend wird auf die zahlreichen Vorteile der Benutzeroberflächen mit JSP in Bezug auf die Anforderungen der ISAG eingegangen. Die Lauffähigkeit auf allen Betriebssystemen, die Webbrowser unterstützen, war zwar keine Anforderung der ISAG, wird aber sicherlich in Zukunft einen nicht zu unterschätzenden Zusatznutzen darstellen. Weiterhin wird auf das neue Konzept der Prozess- und Dokumenten-gesteuerten Projektabwicklung eingegangen, das in dieser Arbeit pilothaft implementiert wurde. Bei diesem Konzept wird ein Stück der Kontrolle der Process Engine an die Projektdatenbank abgegeben, jedoch gewinnt man die wertvolle Möglichkeit hinzu, einige Projektaufgaben Dokumenten-gesteuert durchführen zu können. Zudem kann man die für die Dokumenten-Steuerung entwickelten Benutzeroberflächen, die Datenbankstruktur und die Konsistenzprüfungen direkt aus der Process Engine aufrufen. Es ist sogar möglich, dass einige Mitarbeiter in ihrem Arbeitsgebiet rein Dokumenten-gesteuert arbeiten, während andere diese Arbeiten über die Process Engine steuern. Anschließend wird abgeschätzt, wie hoch der Aufwand für die Programmierung und die Bedienung bei der Dokumenten-gesteuerten Projektabwicklung, bei der Prozess-gesteuerten Projektabwicklung und bei der neu entwickelten Kombination der beiden Methoden ist. Das Aufwandsdiagramm veranschaulicht, dass die neu entwickelte Kombination der Methoden den Programmieraufwand auf ein realistisches Maß begrenzt und so das bisher bekannte, immer steilere Ansteigen der Programmieraufwandskurven vermeidet.

Das Ziel dieser Diplomarbeit, also die unterschiedlichen Fehlermöglichkeiten zu verringern und den Fehlerentdeckungsaufwand sowie den Arbeitsaufwand zur Projektabwicklung zu reduzieren, wurde damit vollumfänglich erreicht. Darüber hinaus wurde mit dem Konzept der Prozess- und Dokumenten-gesteuerten Projektabwicklung ein Weg zur strukturierten Projektabwicklung aufgezeigt, der weiteres Effizienzpotential bietet.

Ausblick

Die in Kapitel 4 beschriebene Dokumenten-gesteuerte Projektabwicklung bildet derzeit im Wesentlichen die Bearbeitung der in schriftlichen Tagesberichten dokumentierten Mitarbeiter-Arbeitszeiten ab. Die Ablage von Scans der den Eingabedaten zugrunde liegenden Dokumente sowie die Dokumentation der Person, die die Daten eingegeben hat, bietet zusätzliches Potential zur Effizienzsteigerung der Fehlersuche. Die ebenfalls auf den Tagesberichten dokumentierten Fehlerberichte können in der aktuellen Datenbankstruktur noch nicht abgebildet werden. Dafür sind Erweiterungen dieser Struktur erforderlich und auch möglich. Weiterhin enden die Oberflächen-gesteuerten Prozesse derzeit beim Export in existierenden Excel-Dokumenten, wie z. B. in der Gesamtrechnung. Eine weiterführende Steuerung durch Benutzeroberflächen erfordert zum einen eine genaue Analyse des bisher angewandten, nachgeschalteten Rechnungslegungs-Prozesses und zum anderen die Programmierung dazu passender Oberflächen und Funktionen sowie möglicherweise auch Erweiterungen der

Datenbankstruktur. Die pilothafte Implementierung des neu entwickelten Konzepts der Prozess- und Dokumenten-gesteuerten Projektabwicklung hat die Anwendbarkeit in der Projektabwicklung bestätigt. Für den praktischen Einsatz ist jedoch ein vollständiger, an dieses neue Konzept angepasster Projektablaufplan zu erstellen und entsprechend zu modellieren. Die Einbindung der zeitlich sicherlich vorlaufend erstellten Dokumenten-gesteuerten Projektabwicklung einschließlich der Rechnungsstellung kann dann mit relativ geringem Aufwand in die kombinierte Projektabwicklung eingebunden werden. Weiteres Potential bietet die Verfügbarkeit der Benutzeroberflächen auf vielen elektronischen Geräten sowie die Verbindungsmöglichkeit mit der Projektsteuerung und der Datenbank über VPN[2]. Auf diese Weise könnten die Mitarbeiter in Zukunft ihre Arbeitszeiten auf der jeweils gerade verfügbaren Hardware elektronisch eingeben und an die Datenbank übermitteln. Dadurch könnte das manuelle Übertragen der Daten vom schriftlichen Tagesbericht entfallen. Die Genehmigung und Freigabe der entsprechenden Daten ist jedoch immer erforderlich. Einen weiteren Zusatznutzen kann es auch durch entsprechenden Zugriff der Kunden auf die für sie wichtigen Mitarbeiter-Arbeitszeiten geben, weil die Kunden dann täglich prüfen können, ob die eingetragenen Arbeitszeiten von bei ihnen tätigen Mitarbeitern für sie auch plausibel sind. Dies kann aufwändige Reklamationsbearbeitungen bereits im Vorfeld vermeiden. Diese Arbeit legt einen soliden Grundstein für die automatisierte Projektabwicklung und öffnet ein weites Feld für Weiterentwicklungen in diesem Thema.

[2]http://www.vpn.de/

A Anhang

A.1 Anhang 1

Konfiguration der BOS PE und Apache Tomcat Server:

1. Das BOS-Tomcat Paket wird von der Webseite geladen und entpackt.
2. In dem Ordner bin gibt es das Batch-File setenv.bat, durch Notepad++ wird dieses File geöffnet und entsprechend der folgenden Liste geändert. Die Pfade CATALINA_HOME und JAVA_HOME müssen der Liste A.1 hinzugefügt werden.

Listing A.1 Konfiguration des Batch-Files

```
1 @echo on
2 rem Sets some variables
3 set CATALINA_HOME="C:\BOS-5.8-Tomcat-6.0.35"
4 set TOMCAT_HOME="C:\BOS-5.8-Tomcat-6.0.35"
5 set BONITA_HOME="-DBONITA_HOME=%CATALINA_HOME%\bonita"
6 set SECURITY_OPTS="-Djava.security.auth.login.config=%CATALINA_HOME%\
    external\security\jaas-standard.cfg"
7 set CATALINA_OPTS=%CATALINA_OPTS% %SECURITY_OPTS%
8 %BONITA_HOME% -Dfile.encoding=UTF-8 -Xshare:auto -Xms512m -Xmx1024m -XX:MaxPermSize=256m
    -XX:+HeapDumpOnOutOfMemoryError
```

3. Die zwei Umgebungsvariablen CATALINA_HOME und JAVA_HOME müssen definiert werden. CATALINA_HOME ist der Pfad des entpackten BOS-Tomcat Pakets, JAVA_HOME ist der Pfad der JDK.

Konfiguration der Activiti und Apache Tomcat Server:

1. Activiti 5.6 wird in einer manuell festzulegenden Partition dekomprimiert.
2. Die Umgebungsvariablen ANT_HOME und JAVA_HOME werden erstellt. ANT_HOME ist der Pfad des entpackten Pakets Ant. JAVA_HOME ist der Pfad der JDK.
3. Das Cmd-Fenster wird geöffnet, durch DOS-Befehle wird das Verzeichnis activiti5.6setup geöffnet, dann wird der Befehl *ant.demo.start* ausgeführt. Die Tomcat 6.0.32 wird automatisch von der Webseite heruntergeladen und mit der Activiti konfiguriert. Unter dem Ordner Activiti 5.6 gibt es einen neuen Ordner *apps*, in den Tomcat installiert wird.
4. Die Umgebungsvariable CATALINA_HOME wird definiert. CATALINA_HOME ist der Pfad des Tomcat, das in dem Ordner *apps* installiert ist.

A.2 Anhang 2

Funktion	Vorgehensweise	Werkzeug	Ergebnisse
neuen TB erstellen	JUnit, Black-Box-Test	Eclipse, MySQL, Firefox, Firebug	positiv
gespeicherte TB korrigieren	JUnit, Black-Box-Test	Eclipse MySQL Firefox Firebug	positiv
freigegebene TB korrigieren	JUnit, Black-Box-Test	Eclipse MySQL Firefox Firebug	positiv
freigegebene TB exportieren	JUnit, Black-Box-Test	Eclipse MySQL Firefox Firebug	positiv

Tabelle A.1: Testplanung für den Programmtest

A.3 Anhang 3

Software	Version
Activiti	5.6 und 5.11
Apache Ant	1.8.3
Apache POI	3.8
Apache Tomcat	6.0.36
BOS-Tomcat	5.6.2-6.0.33
iText	5.3.2
JDK	1.6.0_37
jQuery	1.8.1
jQuery-UI	1.7.3
MySql	5.2

Tabelle A.2: Verwendete Software und Bibliotheken

Literaturverzeichnis

[Act] Activiti 5.11 User Guide. http://activiti.org/userguide/. (Zitiert auf den Seiten 8 und 88)

[All05] Thomas. Allweyer. *Geschäftsprozessmanagement. Strategie, Entwurf, Implementierung, Controlling*. Verlag Moritz Diesterweg, Herdecke, 2005. (Zitiert auf Seite 19)

[All09] Thomas Allweyer. *BPMN 2.0 - Business Process Model and Notation: Einführung in den Standard für die Geschäftsprozessmodellierung*. Books on Demand, 2009. (Zitiert auf Seite 22)

[Bau08] Günther Bauer. *Architekturen für Web-Anwendungen: Eine praxisbezogene Konstruktions-Systematik*. Vieweg+Teubner Verlag, 2008. (Zitiert auf Seite 23)

[Bau12] Felix Baumann. Automatisierung der Projektabwicklung in Unternehmen. Diplomarbeit, Universität Stuttgart, Institut für Rechnergestützte Ingenieursystem, 2012. (Zitiert auf den Seiten 8, 16, 47, 69, 70, 110, 111 und 114)

[Bec05] Kent Beck. *JUnit kurz & gut*. O'Reillys Taschenbibliothek. O'Reilly Vlg. GmbH & Company, 2005. (Zitiert auf Seite 99)

[Ber07] Hans Bergsten. *JavaServer Pages*. Oreilly Series. O'Reilly Media, 2007. (Zitiert auf Seite 109)

[BM11] Graham Bath and Judy McKay. *Praxiswissen Softwaretest - Test Analyst Und Technical Test Analyst: Aus- Und Weiterbildung Zum Certified Tester - Advanced Level Nach ISTQB-Standard*. Dpunkt Verlag, 2011. (Zitiert auf Seite 94)

[BOS] Add An IFrame Field. http://www.bonitasoft.com/resources/documentation/bos-56/form-field-design/use-widgets-define-form-data-fields-and-form-buttons/add-iframe-field. (Zitiert auf Seite 74)

[BOS11] Bonita Open Solution Version 5.6 Connectors Reference Guide. 2011. (Zitiert auf Seite 76)

[BRJ06] Grady Booch, James Rumbaugh, and Ivar Jacobson. *Das UML-Benutzerhandbuch*. Addison-Wesley, München [u.a.], 2006. (Zitiert auf Seite 54)

[Fot10] Egmont Foth. *Exzellente Geschäftsprozesse mit SAP: Praxis des Einsatzes in Unternehmensgruppen*. Springer, 2010. (Zitiert auf Seite 22)

[FR10] Jakob Freund and Bernd Rücker. *Praxishandbuch BPMN 2.0*. Carl Hanser Verlag GmbH & CO. KG, 2010. (Zitiert auf den Seiten 21 und 109)

[Gad12] Andreas Gadatsch. *Grundkurs Geschäftsprozess-Management*. Vieweg, 7., erw. u. überarb. edition, 2012. (Zitiert auf den Seiten 19 und 20)

[Gau04] Markus Gaulke. Risikomanagement in it-projekten. volume 23 of *LNI*. GI, 2004. (Zitiert auf Seite 111)

[Het88] Bill Hetzel. *The complete guide to software testing*. QED Information Sciences, Inc., Wellesley, MA, USA, 2nd edition, 1988. (Zitiert auf Seite 91)

[HV07] Peter A. Henning and Holger Vogelsang, editors. *Handbuch Programmiersprachen*. Carl Hanser Verlag, München, 2007. (Zitiert auf Seite 29)

[IEE08] IEEE. IEEE Standard for Software and System Test Documentation. *IEEE Std 829-2008*, 2008. (Zitiert auf Seite 91)

[ISO01] ISO/IEC. *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC, 2001. (Zitiert auf Seite 110)

[LL07] Jochen Ludewig and Horst Lichter. *Software Engineering - Grundlagen, Menschen, Prozesse, Techniken*. dpunkt.verlag, 2007. (Zitiert auf Seite 92)

[Low10] Bruno Lowagie. *iText in Action: Second Edition*. Manning Publications, 2 edition, October 2010. (Zitiert auf Seite 63)

[LR00] Frank Leymann and Dieter Roller. *Production workflow concepts and techniques*. Prentice Hall PTR, 2000. (Zitiert auf Seite 21)

[LS] Frank Leymann and David Schumm. Process Engine. http://wirtschaftslexikon.gabler.de/Definition/process-engine.html. (Zitiert auf Seite 21)

[MBTS04] G.J. Myers, T. Badgett, T.M. Thomas, and C. Sandler. *The Art of Software Testing*. Business Data Processing: a Wiley Series. John Wiley & Sons, 2004. (Zitiert auf Seite 91)

[Mye01] Glenford J. Myers. *Methodisches Testen von Programmen*. Reihe Datenverarbeitung. Oldenbourg, München [u.a.], 7. aufl., unveränd. nachdr. der 3. aufl edition, 2001. (Zitiert auf Seite 92)

[NS02] Rainer Nägele and Peter Schreiner. Bewertung von Werkzeugen für das Management von Geschäftsprozess. pages in: Zeitschrift Führung + Organisation 71, 2002. (Zitiert auf den Seiten 7 und 20)

[OMG11] Object Management Group, Business Process Model and Notation. http://www.omg.org/spec/BPMN/2.0/PDF/, 2011. (Zitiert auf Seite 22)

[POI] Apache POI - the Java API for Microsoft Documents. http://poi.apache.org/. (Zitiert auf Seite 50)

[Rü12] Bernd Rücker. Ein Überblick über aktuelle Tools und Strömungen Prozesse in Bewegung. 01.2012. (Zitiert auf Seite 70)

[Rad12] Tijs Rademakers. *Activiti in Action : Executable business processes in BPMN 2.0*. Manning Publications, Shelter Island, NY, first edition, 2012. (Zitiert auf den Seiten 8, 81, 87 und 88)

[Sch73] Harald Jürgen Schröder. *Projekt-Management*. Gabler, Wiesbaden, 1973. (Zitiert auf Seite 13)

[Sil09] Bruce Silver. *Bpmn Method and Style: A Levels-Based Methodology for Bpm Process Modeling and Improvement Using Bpmn 2.0*. Cody-Cassidy Press, 2009. (Zitiert auf Seite 22)

[Sou10] Charles Souillard. *Custom Application Development with Bonita Execution Engine*. 2010. (Zitiert auf Seite 79)

[SS08] Hermann J. Schmelzer and Wolfgang Sesselmann. *Geschäftsprozessmanagement in der Praxis*. Hanser, München, 6., vollst. überarb. und erw. aufl. edition, 2008. (Zitiert auf Seite 19)

[SZ05] Manfred Schulte-Zurhausen. *Organisation*. Vahlen, München, 4, überarb. aufl edition, 2005. (Zitiert auf Seite 19)

[VBA] Erläuterung des Begriffs Visual Basic for Applications. http://www.it-visions.de/glossar/alle/226/lexikon.aspx. (Zitiert auf Seite 28)

[WE08] S.M. Weh and C. Enaux. *Konfliktmanagement: Konflikte kompetent erkennen und lösen*. Kienbaum bei Haufe. Haufe Lexware GmbH, 2008. (Zitiert auf Seite 111)

Alle URLs wurden zuletzt am 15. 01. 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschift