Bachelorarbeit

Eine leichtgewichtige Schnittstelle zwischen digitalen Formularen und heterogenen Datenbeständen

Mark Müller
m_mueller09@informatik.uni-kl.de

7. Februar 2013

Technische Universität Kaiserslautern
Fachbereich Informatik
AG Softwaretechnik

Betreuung
Prof. Dr. Arnd Poetzsch-Heffter
Dipl.-Inf. Patrick Michel

Zusammenfassung

Formulare spielen in vielen Wirtschaftsbetrieben eine wichtige Rolle. Mit der informationstechnischen Unterstützung von Arbeitsabläufen, welche auf dem Umgang mit Papierformularen basieren, könnte in diesen Betrieben ein höheres Maß an Produktivität erreicht werden. Zwei Kernprobleme bei der Entwicklung eines solchen informationstechnischen Systems liegen erstens in der Festlegung einer geeigneten digitalen Repräsentation der Formulare, sowie zweitens in der Angabe einer leichtgewichtigen Schnittstelle zwischen den digitalen Formularen und heterogenen Datenbeständen. Für die digitale Repräsentation von Formularen gibt es bereits vielversprechende Lösungen. Daher ist es das Ziel dieser Bachelorarbeit ein informationstechnisches System vorzuschlagen, welches eine leichtgewichtige Schnittstelle zwischen digitalen Formularen und heterogenen Datenbeständen realisiert.

Dazu wird, abgeleitet von einem konkreten Beispiel, beschrieben, welche Klasse von Arbeitsabläufen vorausgesetzt wird. Anschließend wird der Aufbau eines entsprechenden informationstechnischen Systems beschrieben und eine prototypische Realisierung vorgestellt. Auf Grundlage der Realisierung und des motivierenden Arbeitsablaufes wird dann eine Bewertung der vorgeschlagenen Lösung vorgenommen.

Abstract

Forms play an important role in many organization's workflows. Using IT systems to support paper based workflows could increase productivity in these organizations. There are two core problems to solve while developing such systems: First, defining a well suited digital representation of forms. Second, finding a lightweight interface between digital forms and heterogeneous data resources. There already are convenient solutions for the first problem. Hence, aim of this bachelor's thesis is to propose an IT system which realises a lightweight interface between digital forms and heterogeneous data resources.

Therefore the class of supported workflows, derived from a concrete example, is described. A presentation of the architecture of such an IT system and a description of a prototypical realisation follow afterwards. Based on the realisation and the motivating workflow the proposed solution is evaluated.

Inhaltsverzeichnis

1 Einführung

Formulare spielen in vielen Wirtschaftsbetrieben eine wichtige Rolle. Formulare werden verfasst, geändert, ausgegeben, ausgefüllt, geprüft, verarbeitet, weitergereicht, abgezeichnet und abgelegt. Wenn dies in großer Anzahl geschieht, wird in signifikantem Maße Arbeitszeit mit der Bearbeitung von Formularen verbracht. Daher bietet es sich an, zu untersuchen, inwiefern sich Arbeitsabläufe, welche auf dem Umgang mit Formularen basieren, sinnvoll informationstechnisch unterstützen lassen.

Oftmals werden solche Arbeitsabläufe bereits teilweise digital abgewickelt. So werden Formulare typischerweise digital erstellt, ausgedruckt, dann in Papierform ausgefüllt und schließlich die gewonnenen Daten wiederum in ein informationstechnisches System übertragen. Dabei haben Formulare in Papierform Nachteile gegenüber einer digitalen Repräsentation: Beispielsweise beim Lesen von Handschriften, dem relativ langwierigen Transport und der manuellen Konsistenzprüfung. Zudem stellt jede Umwandlung eines Formulars zwischen Papierform und digitaler Repräsentation innerhalb des Arbeitsablaufes einen unerwünschten Mehraufwand dar. Eine stärkere Digitalisierung solcher Abläufe wäre also in vielen Fällen gewinnbringend.

Obwohl in vielen Betrieben die notwendige technische Infrastruktur für eine weitere Digitalisierung bereits existiert, fehlt es oft an der softwaretechnischen Unterstützung. Vermutlich ist die Ursache darin zu suchen, dass für eine maximale softwaretechnische Unterstützung individuelle Lösungen erfordert werden und damit zu aufwändig und teuer ist. Um den Aufwand einer Neueinführung zu reduzieren sollten darum die Grundlagen solcher informationstechnischen Systeme untersucht werden.

Den ersten wichtigen Bestandteil eines solchen informationstechnischen Systems stellt eine geeignete Möglichkeit zur digitalen Repräsentation der Formulare dar. Diese Repräsentation sollte einerseits für den Benutzer innerhalb des Arbeitsablaufes, vergleichbar mit Papierformularen, intuitiv zu nutzen sein, andererseits von ihrem technischen Aufbau her auch für die weitere automatische Verarbeitung geeignet sein. Zudem sollte sie sich auch parallel zu Papierformularen einsetzen lassen, da eine vollständige Digitalisierung des Arbeitsablaufes oft nicht erwünscht ist. Für eine solche digitale Repräsentation der Formulare gibt es bereits vielversprechende Lösungen wie die an der Technischen Universität Kaiserslautern entwickelten Flexoforms. Aufgrund ihrer in 1.1 beschriebenen Eigenschaften bieten diese eine geeignete Grundlage für den Einsatz innerhalb eines entsprechenden informationstechnischen Systems.

Bei der Arbeit mit Formularen werden typischerweise jedoch nicht nur die innerhalb des Formulars angegebenen Daten benötigt. Vielmehr fließen oft

auch Zweitinformationen aus anderen Datenquellen ein. Diese Zweitinformationen können vom Benutzer innerhalb des Arbeitsablaufes per Hand abgerufen werden, oder aus anderen informationstechnischen Systemen stammen. Liegen die Datenbestände bereits digital vor, so sollten diese auch automatisch verarbeitet werden können. Da davon auszugehen ist, dass die Datenbestände technisch sehr unterschiedlich aufgebaut sind, werden diese hier als heterogene Datenbestände bezeichnet. Typisch wären beispielsweise relationale Datenbanken oder XML-basierte Technologien. Um eine möglichst hohe Automatisierung des Arbeitsablaufes gewährleisten zu können, ist also eine enge Zusammenarbeit des zu entwickelnden informationstechnischen Systems mit diesen heterogenen Datenbeständen erforderlich.

Daher ist die Schnittstelle zwischen den digitalen Formularen und den heterogenen Datenbeständen neben der Repräsentation der Formulare die zweite wichtige Komponente des Systems. Auf der einen Seite können über diese Schnittstelle die Daten aus einem digitalen Formular manipuliert werden, auf der anderen Seite die Daten aus einem bestimmten Datenbestand abgerufen werden. Die Kombination aus digitalen Formularen und einer geeigneten Schnittstelle bildet die Grundlage für die Implementierung eines entsprechenden informationstechnischen Systems.

Das Ziel dieser Bachelorarbeit ist es, eine solche Schnittstelle vorzuschlagen und zu bewerten. Dazu wird in Kapitel 2, abgeleitet von einem konkreten Beispiel, beschrieben, welche Klasse von Arbeitsabläufen vorausgesetzt wird. Darauf aufbauend wird im nächsten Kapitel der Aufbau des informationstechnischen Systems beschrieben und im folgenden Kapitel eine prototypsiche Realisierung vorgestellt. Auf Grundlage der Realisierung und des motivierenden Arbeitsablaufes wird in den letzten Kapiteln dann eine Bewertung vorgenommen und die Ergebnisse der Arbeit zusammengefasst. Auf Basis der beschriebenen Schnittstelle und eines geeigneten Formularsystems können dann für alle beschriebenen Arbeitsabläufe informationstechnische Systeme implementiert werden, um in den Wirtschaftsbetrieben letztendlich ein höheres Maß an Automatisierung und Produktivität zu erreichen.

1.1 Flexoforms

Flexoforms [2] wurden an der Technischen Universität Kaiserslautern mit dem Ziel entwickelt, die gleichen Arbeitsabläufe zu unterstützen, welche auch in dieser Arbeit behandelt werden. Sie stellen eine digitale Repräsentation von Formularen zur Verfügung und werden im Rahmen dieser Arbeit als Grundlage verwendet. In implementierten Systemen könnten jedoch auch verwandte Repräsentationen eingesetzt werden. Zur Vorstellung dieser Technologie folgt hier eine kurze Zusammenfassung der Ergebnisse ihres Entwicklers

Christian Fillibeck.

Flexoforms repräsentieren ein Papierformular mit genau einer XML-Datei. Die Struktur einer solchen XML-Datei ist in Quelltext 1 dargestellt. Das *data*-Element repräsentiert die Daten des Formulars. Es kann beliebige XML-Ausdrücke enthalten. Das Aussehen des Formulars ist im *view*-Element definiert. Das *form*-Element beschreibt, welche Darstellungselemente das Formular enthält, das *stylesheet*-Element, wie diese formatiert werden. Hierfür werden die entsprechenden HTML-Ausdrücke für Formulare und Cascading Stylesheets verwendet. Mit dieser Dreiteilung folgt die Struktur des Flexoforms dem Module-View-Controller Architekturmuster. Zudem gehört zu jedem Formular eine Schema-Beschreibung.

```
<document>
   <data>
         <!-- beliebgie Daten -->
   </data>
   <view>
      <stylesheet>
         <!-- Cascading Style Sheets -->
      </stylesheet>
      <form>
         <!-- HTML forms -->
      </form>
   </view>
</document>
```

Quelltext 1: Struktur eines Flexoforms

Diese Struktur ist gut geeignet zur interaktiven Bearbeitung des Formulars über einen Web-Browser. Abbildung 1 zeigt das Erscheinungsbild eines Flexoforms im Browser. Dem Benutzer kann zum Zeitpunkt des Ausfüllens bereits Rückmeldung über seine Eingaben gegeben und damit unerwünschte Einträge verhindert werden. Zudem ist eine Weboberfläche auch Neulingen bekannt und somit sehr zugänglich. Aufgrund ihrer klaren Struktur sind Flexoforms zudem zur weiteren automatischen Verarbeitung geeignet. Damit bieten Flexoforms sowohl für den Endbenutzer innerhalb des Arbeitsablaufes, als auch für die Entwicklung eines informationstechnischen Systems, geeignete Voraussetzungen.

2 Formularbasierte Arbeitsabläufe

Ziel dieser Arbeit ist es, bestimmte Arbeitsabläufe zu automatisieren. Daher werden in diesem Kapitel zunächst diese Arbeitsabläufe näher charakterisiert und beschrieben, inwiefern diese stärker automatisiert werden können. Dazu

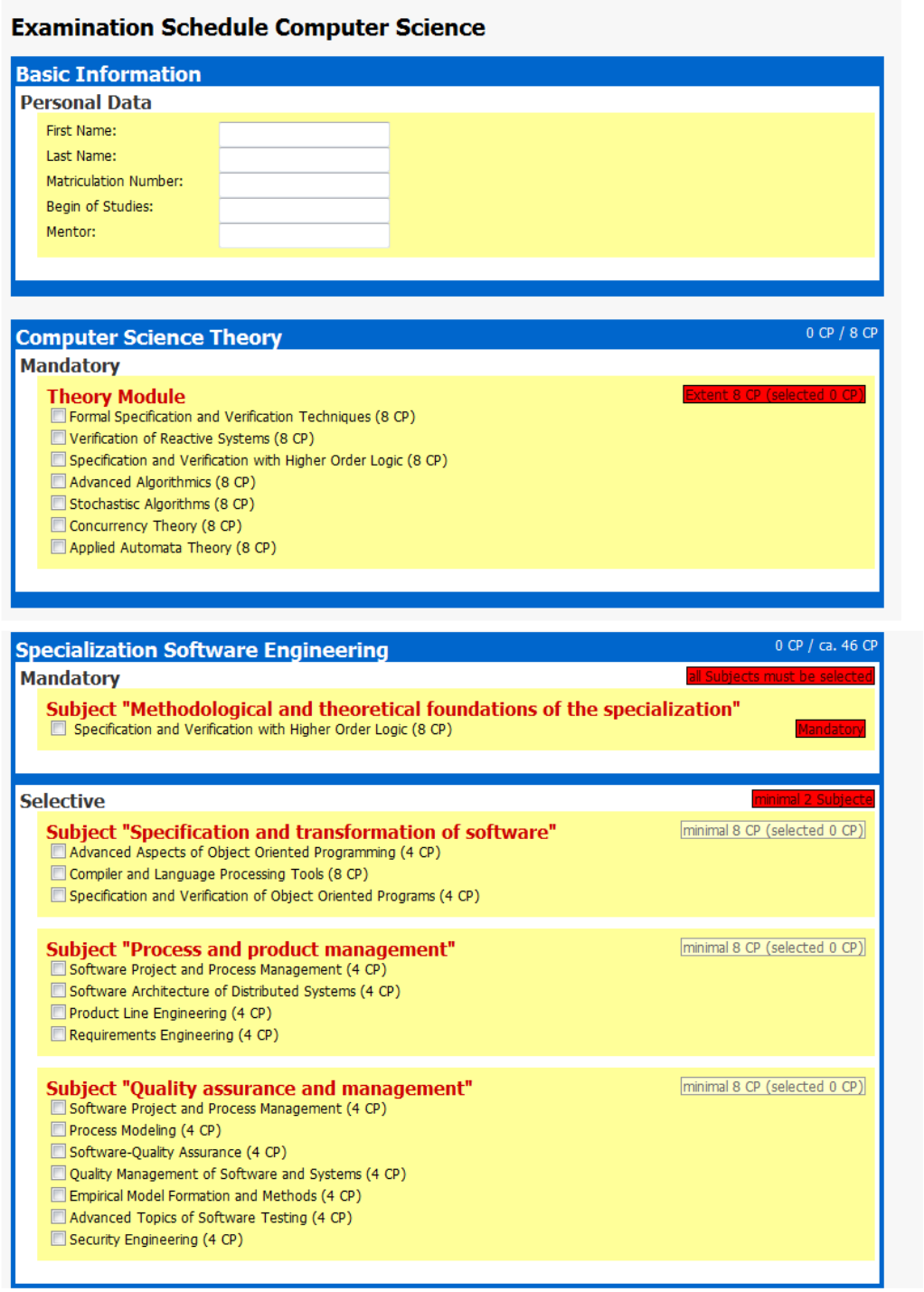

Abbildung 1: Ein Flexoform im Browser

wird zunächst mit einem konkreten Arbeitsablauf eine Motivation gegeben. Dieses Beispiel wird dann auf eine Klasse von Arbeitsabläufen verallgemeinert. Schließlich werden daraus die Anforderungen an ein informationstechnisches System abgeleitet.

2.1 Motivation

Im Folgenden wird ein Arbeitsablauf von der Technischen Universität Kaiserslautern beschrieben. Anhand dieses Arbeitsablaufes soll motiviert werden, was an vergleichbaren Abläufen durch stärkere informationstechnische Unterstützung verbessert werden kann. Zudem ist dieser Arbeitsablauf die Grundlage der Bewertung der vorgestellten Technologie.

Jeder Student des Masterstudiengangs Informatik an der Technischen Universität Kaiserslautern muss zu Beginn seines Studiums dem Prüfungsamt der Universität einen gültigen Prüfungsplan vorlegen. Ein solcher Prüfungsplan ist ein Papierformular, welches eine Liste von Modulen enthält. Module repräsentieren die Lehrveranstaltungen der Universität. Welche Module von der Universität angeboten werden, ist im sogenannten Modulhandbuch festgehalten. Ob ein Prüfungsplan gültig ist, hängt davon ab, ob die eingetragene Kombination von Modulen gültig ist. Welche Kombinationen von Modulen wiederum gültig sind, ist in der Fachprüfungsordnung [6] für den Masterstudiengang Informatik festgelegt. Die genauen Bedingungen können dort nachgelesen werden.

Der Arbeitsablauf, einen solchen gültigen Prüfungsplan vorzulegen, gestaltet sich typischerweise wie folgt. Der Student holt zunächst persönlich einen leeren Prüfungsplan beim Prüfungsamt ab, oder lädt ein PDF-Dokument über das Internet herunter und druckt dieses aus. Nun liest der Student die Prüfungsordnung, um aus dem Modulhandbuch eine gültige Kombination von Modulen auszuwählen. Das Modulhandbuch wird vom sogenannten Service Center Informatik (SCI) als HTML-Dokument über das Internet zur Verfügung gestellt. Die Prüfungsordnung kann als PDF-Dokument ebenfalls über das Internet heruntergeladen werden. Die getroffene Auswahl trägt der Student handschriftlich in das Formular ein.

Ein Gültigkeitskriterium in der Prüfungsordnung sieht vor, dass die ausgewählte Kombination der Module von dem Mentor des Studenten genehmigt wird. Der Mentor ist ein Professor des Fachbereiches, welcher dem Studenten zu Beginn des Studiums zugeordnet wurde, um ihn bei der Zusammenstellung des Prüfungsplans zu beraten. Um diese Rücksprache zu treffen, legt der Student seinem Mentor den ausgefüllten Prüfungsplan vor. Der Mentor prüft nun einige Gültigkeitskriterien. Dazu benötigt er neben der Definition der Kriterien als solche, also der Prüfungsordnung, das Modulhandbuch, wel-

ches er über das Internet einsehen kann. Ist der Mentor mit der Kombination von Modulen nicht einverstanden, so wird der Prüfungsplan vom Studenten überarbeitet und dem Mentor erneut zur Prüfung vorgelegt. Ist der Mentor einverstanden, so unterschreibt er den Prüfungsplan.

Den unterschriebenen Prüfungsplan liefert der Student dann beim Prüfungsamt ab. Das Prüfungsamt führt erneut eine Gültigkeitsprüfung durch. Das Prüfungsamt benötigt hierzu ebenfalls das Modulhandbuch. Hinzu kommen zusätzliche Informationen, beispielsweise die bisherigen Prüfungsleistungen des Studenten. Diese Informationen werden aus einem Universitätssystem abgerufen. Falls das Prüfungsamt die Modulkombination aus dem unterschriebenen Prüfungsplan bestätigt, so ist der Arbeitsablauf abgeschlossen und das Prüfungsamt legt die gewonnenen Daten ab. Wird die Modulkombination vom Prüfungsamt nicht bestätigt, so muss der Student nach der gleichen Vorgehensweise einen überarbeiteten Prüfungsplan anfertigen.

Abbildung 2 zeigt eine statische Sicht auf den beschriebenen Arbeitsablauf. Das SCI und das Universitätssystem stellen die zur Verarbeitung der Prüfungspläne notwendigen Daten zur Verfügung. Diese Daten können vom Mentor beziehungsweise dem Prüfungsamt abgerufen werden. Dort findet auch die eigentliche Bearbeitung der Prüfungspläne statt: Leere Prüfungspläne werden zur Verfügung gestellt, Gültigkeitskriterien der Prüfungsordnung anhand der abgerufenen Daten überprüft, Prüfungspläne in Papierform angenommen und abgegeben, Prüfungspläne unterzeichnet, gültige Prüfungspläne abgelegt. Die Aufgabe des Studenten besteht darin, den Prüfungsplan auszufüllen. Zwischen Mentor, Prüfungsamt und Student werden nur Prüfungspläne ausgetauscht.

Abbildung 3 zeigt einen leeren Prüfungsplan. In die Zeilen werden die Module eingetragen. Die leeren Prüfungspläne liegen zwar digital als PDF-Dokument vor, können aber nicht digital weiterverarbeitet werden, sondern müssen zur weiteren Bearbeitung ausgedruckt werden. Überhaupt wird der Arbeitsablauf kaum informationstechnisch unterstützt. Lediglich der Zugriff auf die Datenbestände des SCI und des Universitätssystems erfolgt digital. Statt Prüfungspläne in Papierform auszutauschen, könnten digitale Repräsentationen verwendet werden. Davon würden viele Arbeitsschritte profitieren: Beim Abgeben und Aufnehmen von Formularen müssten lediglich elektronisch Daten ausgetauscht werden, was den langwierigen Transport von Papierformularen ersetzt. Beim Prüfen der Formulare müsste keine Handschrift gelesen werden, Korrekturen könnten vermieden werden. Das Prüfen könnte stärker automatisiert werden und somit bei Mentor und Prüfungsamt Zeit einsparen. Das Ausdrucken von Formularen sowie das Abtippen der Inhalte beim Weiterverarbeiten im Prüfungsamt würden entfallen. Bereits das Ausfüllen durch den Studenten könnte interaktiv erfolgen, sodass typische

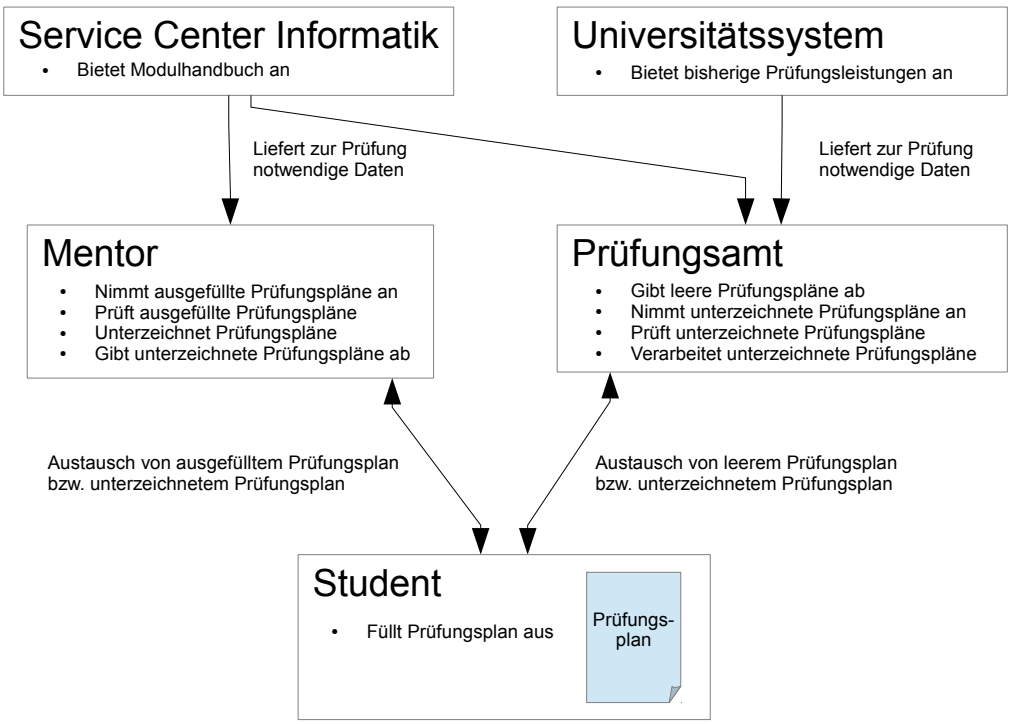

Abbildung 2: Eine statische Sicht auf den Arbeitsablauf zur Vorlage gültiger Studienpläne

TU Kaiserslautern
Prüfungsangelegenheiten

Prüfungsplan Masterstudiengang Informatik

Vor- und Nachname: _____ Studienbeginn:_____

Matrikelnummer:_____ Mentor:_____

Geplante Kurse / Planned Courses

Kursnr.	Kursname	ECTS	Sem.	Datum	Unterschrift Mentor
Informatiktheorie:					
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
Vertiefung:	.....			.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
Nebenfach:	.....			.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....

Abbildung 3: Die erste Seite eines leerern Prüfungsplans

Fehlerquellen frühzeitig und automatisch erkannt würden. Weiterhin wäre das digitale Unterzeichnen von Formularen sicherer. Es könnten verschiedene Formulare verwaltet und interaktiv erstellt werden, sodass eine stärkere Strukturierung möglich wird. Insgesamt gibt es also zahlreiche Argumente, welche für eine weitere Digitalisierung des Arbeitsablaufes sprechen.

2.2 Arbeitsabläufe mit Papierformularen

Nachdem nun ein konkreter Arbeitsablauf analysiert wurde, wird im Folgenden eine Verallgemeinerung auf eine Klasse von typischen formularbasierten Arbeitsabläufen vorgenommen. Dabei werden insbesondere die beteiligten Stellen, deren einzelne Arbeitsschritte und die Kommunikation näher beschrieben. Auf Grundlage dieser Beschreibung werden anschließend die Anforderungen an ein informationstechnisches System abgeleitet.

Abbildung 4 zeigt eine statische Sicht auf die betrachtete Klasse von Arbeitsabläufen. An dem Arbeitsablauf beteiligt sind eine oder mehrere Datenverwaltungsstellen, eine oder mehrere Formularverwaltungsstellen und den Kunden. Die Datenverwaltungsstellen bieten Daten aus unterschiedlichen Anwendungsgebieten an, welche zur Bearbeitung der Formulare notwendig sind. Die Daten liegen typischerweise digital vor. Bei den Formularverwaltungsstellen findet die eigentliche Bearbeitung der Formulare statt. Hier werden zahlreiche elementare Arbeitsschritte ausgeführt. Welche Auswahl der aufgeführten Arbeitsschritte im konkreten Ablauf tatsächlich bearbeitet werden, bleibt dabei offen. Der einzige Arbeitsschritt des Kunden besteht darin, Formulare auszufüllen. Dies beinhaltet auch das Unterzeichnen von Formularen. Es folgt eine genauere Beschreibung der Arbeitsschritte der Formularverwaltungsstellen und des Kunden.

Erstellen von Formularen: An einer Formularverwaltungsstelle werden neue Leerformulare erstellt. Dies erfolgt meistens digital in einem Textverarbeitungsprogramm. Da Formulare aus Elementen wie beispielsweise Textfeldern, Listen oder Kontrollkästchen bestehen, wird hier vorallem eine Anordnung dieser Elemente ausgewählt und mit anwendungsspezifischen Texten versehen. Das Ergebnis dieses Arbeitsschrittes ist typischerweise ein PDF-Dokument oder ein vergleichbares Datenformat auf Basis dessen die Formulare vervielfältigt werden.

Formulare werden jedoch nicht nur von Grund auf neu erstellt, sondern auch bestehende Formulare durch überarbeitete Versionen ersetzt. Dabei ist zu beachten, dass sich veraltete Versionen des Formulars nach wie vor im Umlauf befinden könnten und diese ebenfalls akzeptiert werden sollten.

Abgeben von Formularen: Eine Formularverwaltungsstelle gibt die zuvor bereits erstellten leeren Formulare oder bereits teilweise ausgefüllte

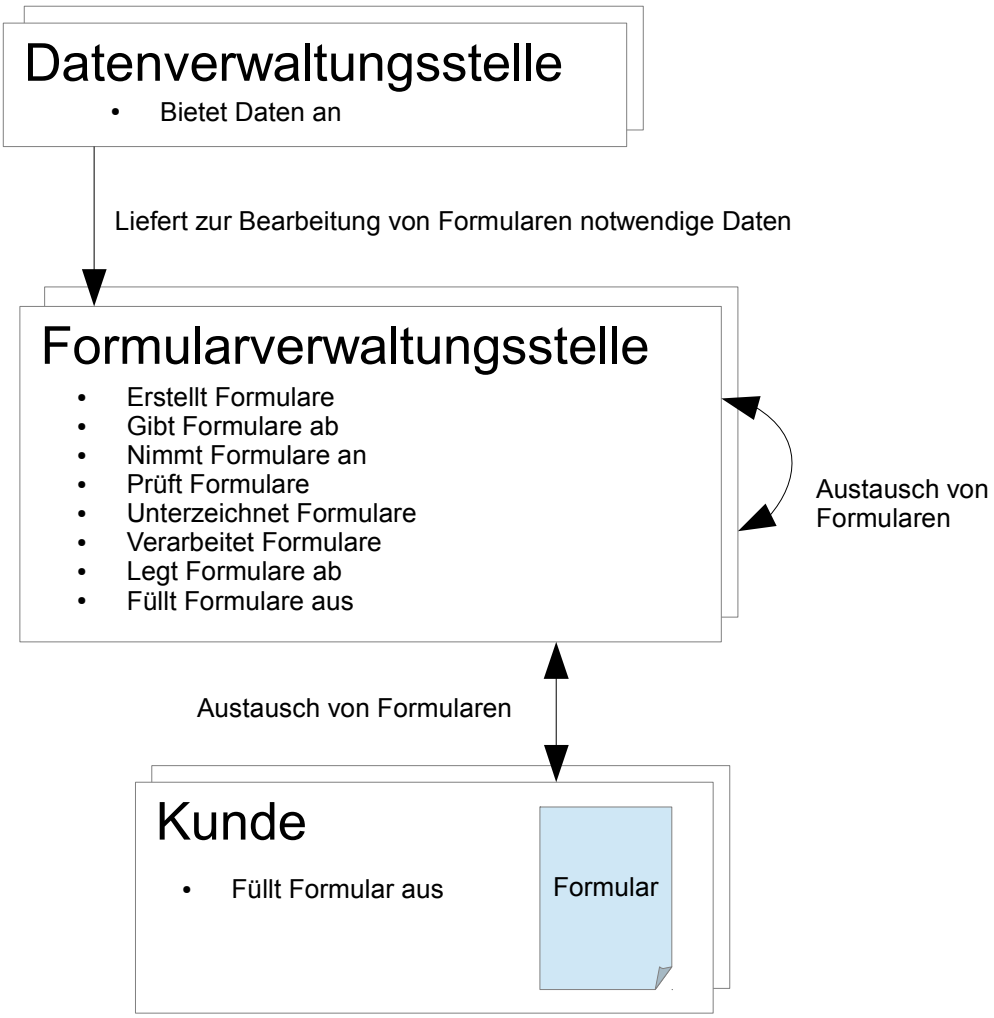

Abbildung 4: Eine statische Sicht auf eine Klasse typischer formularbasierter Arbeitsabläufe

Formulare ab. Abnehmer sind dabei nicht nur Kunden, sondern auch andere Formularverwaltungsstellen. Angeboten werden die leeren Formulare in gedruckter Form oder, wie im motivierenden Beispiel, zum Ausdrucken beim Kunden als digitales Formular. Spätestens zum Ausfüllen liegt das Formular jedoch in Papierform vor.

Hier ist zu beachten, dass verschiedene Formulare möglicherweise auch für unterschiedliche Kundenkreise bestimmt sind. Dann werden Formulare auch zielgerichtet angeboten, beispielsweise werden die digitalen Formulare auf unterschiedlichen Internetseiten bereitgestellt, welche von den Kunden des jeweiligen Kundenkreises auch besucht werden, beziehungsweise die in Papierform gegebenen Formulare werden an unterschiedlichen Orten zu Abholung ausgelegt.

Annehmen von Formularen: An einer Formularverwaltungsstelle werden Formulare nicht nur abgegeben, sondern auch ausgefüllte Formulare entgegengenommen. Als Quelle für Formulare kommen hierbei sowohl Kunden, als auch andere Formularverwaltungsstellen infrage. Die ausgefüllten Formulare werden in Papierform zugestellt, kommen also im Briefkasten an oder werden persönlich abgegeben. Da dem Kunden die Interna der Formularverwaltungsstelle nicht bekannt sind, werden die Formulare nicht an einen Mitarbeiter direkt adressiert sondern innerhalb der Formularverwaltungsstelle an den zuständigen Mitarbeiter übergeben. Danach wird typischerweise eine Prüfung des Formulars vorgenommen.

Prüfen von Formularen: Eine Formularverwaltungsstelle prüft die Gültigkeit von Formularen vor deren weiterer Verarbeitung. Dies umfasst bis zu vier Schritte: Prüfung des Schemas, Prüfung auf Vollständigkeit, Prüfung auf Korrektheit und Prüfung der Unterschrift.

Bei der Prüfung des Schemas wird zunächst festgestellt, ob das Formular überhaupt Teil des Arbeitsablaufes ist, also ob es die richtigen Formularelemente und Texte enthält. Falls nicht, so ist das Formular vermutlich an einen falschen Ort zugestellt worden oder es handelt sich um eine veraltete und nicht mehr akzeptierte Version eines Formulars.

Danach wird eine Prüfung auf Vollständigkeit durchgeführt. Diese Prüfung ist genau dann erfolgreich, wenn alle der notwendigen Formularelemente, deren Existenz ja zuvor geklärt wurde, auch ausgefüllt worden sind. Falls einige der notwendigen Formularelemente nicht ausgefüllt sein sollten, so muss das Formular vor der weiteren Verarbeitung erst vervollständigt werden.

Sind alle notwendigen Daten vorhanden, so wird deren Korrektheit überprüft. Möglicherweise kann diese Prüfung ausschließlich auf den Daten des Formulars stattfinden. Dann wird das Formular vom bearbeitenden Mitarbeiter der Formularverwaltungsstelle aufmerksam durchgelesen und anschließend die Korrektheit beziehungsweise Falschheit festgestellt. Typischerweise

werden zur Korrektheitsprüfung jedoch auch externe Daten benötigt. Diese werden im hier beschriebenen Arbeitsablauf von den Datenverwaltungsstellen zur Verfügung gestellt. Das beutet, dass in diesem Fall der Mitarbeiter weiß, welche Daten er einholen muss und woher er diese bekommen kann, sodass er diese beispielsweise mit einem Telefonat, Aufruf einer Internetseite, oder einem persönlichen Gespräch einholen kann. Scheitert die Korrektheitsprüfung, so kann das Formular nicht weiterverarbeitet werden.

Schließlich kann geprüft werden, ob das Formular unterzeichnet worden ist. In dem hier beschriebenen papierbasierten Arbeitsablauf bedeutet das, dass der Mitarbeiter einen Blick auf das Unterschriften-Feld des Formulars wirft. Die Unterschrift kann sowohl von einem Kunden stammen, als auch von einem Mitarbeiter einer anderen Formularverwaltungsstelle. Handelt es sich um die Unterschrift eines Kunden, so kann nur deren Vorhandensein geprüft werden. Handelt es sich um die Unterschrift einer anderen Formularverwaltungsstelle, so kann diese weiter verifiziert werden, beispielsweise indem dem prüfenden Mitarbeiter das Schriftbild bekannt ist, oder er sich bei dem Verfasser der Unterschrift erkundigt. Dieser Teilschritt ist optional, da nicht jedes Formular über eine Unterschrift verfügen muss.

Unterzeichnen von Formularen: Sowohl von Formularverwaltungsstellen als auch von Kunden können Formulare unterzeichnet werden. Im engeren Sinne handelt es sich dabei nur um einen Teilschritt des Ausfüllens von Formularen. Wegen der besonderen Bedeutung für das Prüfen wird dieser Arbeitsschritt für Formularverwaltungsstellen hier jedoch noch einmal explizit beschrieben.

Das Unterschreiben erfolgt handschriftlich durch einen Mitarbeiter der Formularverwaltungsstelle. Die Unterschrift soll gegenüber anderen Formularverwaltungsstellen belegen, dass das Formular an der ersten Stelle bereits in irgendeiner Art und Weise erfolgreich bearbeitet worden ist. Dies kann beispielsweise das Prüfen bestimmter Eigenschaften betreffen, das Anlegen einer Kopie, oder das Auslesen bestimmter Daten. Spätestens ab dem Zeitpunkt des Unterschreibens darf am Formularinhalt nichts mehr geändert werden. Sollten sich Änderungen ergeben, so muss ein neues Formular angelegt und unterschrieben werden.

Verarbeiten von Formularen: Die Formularverwaltungsstellen verarbeiten Formulare. Das bedeutet, dass die Daten des Formulars zumindest teilweise ausgelesen werden und dann anwendungsspezifisch weiterverarbeitet werden. Das Formular kann anschließend zu anderen Formularverwaltungsstellen oder dem Kunden weitergereicht werden, typischerweise endet mit diesem Arbeitsschritt jedoch der Lebenszyklus des Formulars. Es wird vernichtet oder abgelegt.

Ablegen von Formularen: Das Ablegen von Formularen erfolgt eben-

falls an den Formularverwaltungsstellen. Ablegen bedeutet, dass das Formular zunächst nicht weiter benötigt wird und daher archiviert werden kann. Papierformulare werden typischerweise unter einem bestimmten Schlüsselwert in einem Ordner abgelegt. Sollten die Daten des Formulars noch einmal benötigt werden, so kann das Formular über den entsprechenden Schlüssel wiedergefunden werden.

Ausfüllen von Formularen: Ausgefüllt werden die Formulare sowohl an Formularverwaltungsstellen, als auch beim Kunden. Beim Ausfüllen werden die Formularelemente mit beliebigem Inhalt gefüllt. Die Formulare werden handschriftlich ausgefüllt. Bis zur Unterzeichnung des Formulars können die Inhalte nachträglich beliebig verändert werden.

Damit wären die Arbeitsschritte in der vorgestellten Klasse von papierbasierten Arbeitsabläufen beschrieben. Nun werden innerhalb eines solchen Arbeitsablaufes jedoch nicht nur Arbeitsschritte auf Formularen ausgeführt, sondern auch Informationen zwischen den beteiligten Stellen ausgetauscht. Für diesen Informationsaustausch sind verschiedene Verfahrensweisen vorgesehen. Die Kommunikation zwischen Kunde und einer Formularverwaltungsstelle erfolgt im Wesentlichen durch den Austausch der Papierformulare. Durch das abwechselnde Ausfüllen und Austauschen der Formulare zwischen Kunde und einer Formularverwaltungsstelle werden Daten in beiden Richtungen übertragen. Formulare können auch zwischen verschiedenen Formularverwaltungsstellen umhergereicht werden. Die Kommunikation zwischen Daten- und Formularverwaltungsstellen umfasst lediglich das Bereitstellen der Daten durch die Datenverwaltungsstelle. Darüber, in welcher Form dies geschieht, werden keine Annahmen gemacht. Es kann sich um den Zugriff auf eine Datenbank über ein Computernetzwerk handeln, im Rahmen papierbasierter Arbeitsabläufe sind jedoch auch andere Kommunikationsmittel verbreitet.

Nun sind also die Beteiligten, deren wichtigste Arbeitsschritte und Kommunikationsverfahren vom motivierenden Beispiel auf eine Klasse von Arbeitsabläufen verallgemeinert worden. Mentor und Prüfungsamt sind nun konkrete Ausprägungen von Formularverwaltungsstellen. Sie bearbeiten eine Auswahl der für Formularverwaltungsstellen beschriebenen Arbeitsschritte. Sie kommunizieren über den Austausch von bestimmten Formularen, den Prüfungsplänen, mit dem Kunden, im Konkreten einem Studenten. Neu hinzugekommen ist die Möglichkeit zwischen Formularverwaltungsstellen Formulare auch direkt auszutauschen, also nicht nur über einen Kunden. Das Service Center Informatik und das Universitätssystem sind nun als Datenverwaltungsstellen repräsentiert.

Damit ist die Verallgemeinerung der papierbasierten Arbeitsabläufe ab-

geschlossen. Wie im Motivationsteil bereits aufgezeigt wurde, können solche Arbeitsabläufe von einer stärkeren informationstechnischen Unterstützung profitieren. Um eine Systemarchitektur ableiten zu können, muss zunächst genauer geklärt werden, welche Aufgaben das System innerhalb eines solchen Arbeitsablaufes sinnvollerweise übernehmen sollte.

2.3 Informationstechnische Unterstützung

Im folgenden werden auf Basis der im vorherigen Abschnitt eingeführten allgemeinen Ablaufbeschreibung die Anforderungen an das System abgeleitet.

Eine zentrale Aufgabe des neuen Systems ist natürlich eine starke funktionale Unterstützung der beschriebenen Arbeitsabläufe, sodass möglichst viel Zeit bei der Bearbeitung der Formulare eingespart werden kann. Da sowohl die Formularverwaltungsstellen als auch der Kunde eine zentrale Rolle bei der Bearbeitung der Formulare übernehmen, soll das System von diesen beiden Beteiligten eingesetzt werden können. Damit bilden deren Arbeitsschritte und Kommunikationsverfahren die Grundlage für die Ableitung der funktionalen Anforderungen. Diese Ableitung erfolgt am Ende des Abschnitts.

Neben den funktionalen Anforderungen spielen jedoch auch nicht-funktionale Anforderungen für dieses System eine wichtige Rolle. Die Einführung von Systemen dieser Art ist oft mit einem hohen Aufwand verbunden, da diese an den konkreten Arbeitsablauf angepasst werden müssen und daher individuelle Lösungen erfordert werden. Darum ist es ein wichtiges Ziel bei der Entwicklung des neuen Systems, einen möglichst leichtgewichtigen Ansatz zu finden, welcher die Einführung und Wartung des Systems mit geringem Aufwand ermöglicht. Dies motiviert die Festlegung der nicht-funktionalen Anforderungen, wie sie nachfolgend beschrieben werden.

Ein besonderer Fokus der nicht-funktionalen Anforderungen liegt, wie bei allen informationsverarbeitenden Systemen, auf der Benutzbarkeit. Das System soll von den Mitarbeitern der Formularverwaltungsstellen und dem Kunden mit möglichst wenig technischem Wissen zu benutzen sein. Das beinhaltet beispielsweise, dass alle produktiven Funktionen des Systems durch graphische Benutzeroberflächen zu bedienen sind. Damit sinkt der Einführungsaufwand des Systems und es kann schnell gewinnbringend eingesetzt werden.

Hinzu kommt, dass das System den Arbeitsablauf zwar stark digital unterstützen soll, die parallele Benutzung von Papierformularen jedoch weiterhin möglich bleibt. Das bedeutet, dass auch nach der Einführung des Systems jeder Arbeitsschritt auch mit Papierformularen durchgeführt werden kann und es möglich ist, innerhalb des Lebenszyklusses eines Formulars zwischen digitaler und papierbasierter Form zu wechseln. Damit wird ein hohes

Maß an Ausfallsicherheit erreicht, da die Arbeitsschritte bei einem Ausfall des Systems weiterhin in Papierform abgewickelt werden können. Zudem ist eine vollständige Digitalisierung des Arbeitsablaufes in vielen Fällen nicht erwünscht, da dies zu aufwändig wäre oder die technische Infrastruktur nicht vorhanden ist. Beispielsweise sollte dem Kunden auch die Möglichkeit eingeräumt werden, Papierformulare zu verwenden, falls dieser nicht die Möglichkeit hat, in digitaler Form am Arbeitsablauf teilzunehmen. Zudem wird eine schrittweise Umstellung der Arbeitsabläufe ermöglicht und notwendige Änderungen an vorhandenen Arbeitsabläufen werden reduziert, sodass der Einführungsaufwand weiter abgesenkt werden kann.

Weiterhin ist zu berücksichtigen, dass es zu Änderungen am Arbeitsablauf kommen kann. Das System soll sich gegenüber solchen Änderungen möglichst robust verhalten, das heißt, dass der Aufwand bei der Durchführung der nötigen Änderungen am System möglichst gering sein soll. Änderungen betreffen beispielsweise das Einführen neuer Formularverwaltungsstellen, neuer Datenverwaltungsstellen, Änderungen an den Arbeitsschritten wie eine Änderung der Prüfkriterien, technische Änderungen bei den Datenverwaltungsstellen und ähnliches. Eine Bewertung des Aufwandes bei typischen Szenarien wird nach der Beschreibung des Systems vorgenommen.

Trotzdem soll der softwaretechnische Aufwand bei der Einführung möglichst gering sein, damit das System schnell und einfach für konkrete Arbeitsabläufe umgesetzt werden kann. Da bereits zu vermuten ist, dass dies im Widerspruch zur Forderung nach möglichst großer Flexibilität des Systems steht, muss hier ein Kompromiss gefunden werden.

Damit wäre nun die Ableitung der funktionalen Anforderungen begründet und die nicht-funktionalen Anforderungen beschrieben. Es folgt nun die Auflistung der funktionalen Anforderungen sowie daran anschließend eine Zusammenfassung der nicht-funktionalen Anforderungen.

Unterstütztes Erstellen: Da der Arbeitsablauf möglichst vollständig digitalisiert werden soll, wird eine digitale Repräsentation der Formulare benötigt. Dabei soll bereits das Erstellen eines neuen Formulars in der digitalen Repräsentation erfolgen, um zusätzlichen Aufwand bei einer späteren Digitalisierung zu vermeiden. Das Erstellen eines neuen Formulars beinhaltet, wie in der Beschreibung der papierbasierten Arbeitsabläufe bereits beschrieben, das Zusammenstellen von elementaren Formularkomponenten mit anwendungsspezifischen Texten. Dieses Zusammenstellen der neuen Formulare soll über eine intuitive graphische Benutzeroberfläche erfolgen. Damit müssen den Mitarbeiten der Formularverwaltungsstellen beim Erstellen keine technischen Details der digitalen Repräsentation bekannt sein.

Unterstütztes Abgeben: Um eine starke digitale Unterstützung zu erreichen, soll auch der Kunde die digitale Repräsentation der Formulare ver-

wenden. Daher sollen die leeren Formulare dem Kunden auch in digitaler Form angeboten werden. Da der Kunde aber eventuell auf Papierformulare angewiesen ist, soll eine Funktion angeboten werden, mit der er ein digitales Formular ausdrucken kann, um es dann in Papierform auszufüllen. Es soll auch berücksichtigt werden, dass Formularverwaltungsstellen Formulare untereinander austauschen.

Um die digitale Repräsentation der Formulare gezielt anbieten zu können, sollen diese von beliebigen Internetseiten bezogen werden können. Somit können Formulare auf genau den Internetseiten angeboten werden, welche von den entsprechenden Kundenkreisen auch besucht werden.

Unterstütztes Annehmen: Die Formulare werden digital von den Formularverwaltungsstellen entgegengenommen. Für das Einreichen von Formularen durch den Kunden soll es eine Benutzeroberfläche geben. Während der Kunde die Formulare nur an Formularverwaltungsstellen adressiert, sollen die Mitarbeiter anderer Formularverwaltungsstellen die Formulare auch direkt an andere Mitarbeiter schicken können.

Unterstütztes Prüfen: Alle Schritte der Prüfung sollen automatisch ablaufen können. Dies umfasst das Prüfen des Schemas, das Prüfen der Vollständigkeit, das Prüfen von Korrektheitskriterien und das Prüfen einer eventuell vorhandenen Unterschrift. Die zu prüfenden Eigenschaften können über eine Benutzeroberfläche ausgewählt werden. Die notwendigen Daten werden automatisch aus der Datenverwaltungsstelle abgerufen, falls die Daten dort digital vorliegen. Änderungen an den Korrektheitskriterien sollten mit geringem Aufwand möglich sein. Das Aufrufen der Prüfung soll ohne technisches Wissen möglich sein. Unterschriften von anderen Formularverwaltungsstellen soll das System zuverlässig prüfen können.

Unterstütztes Unterzeichnen: Die Formulare können mit einer digitalen Signatur versehen werden. An anderen Formularverwaltungsstellen kann damit im Rahmen der Prüfungsfunktionalität die Echtheit garantiert werden. Zudem garantiert das System, dass signierte Formulare seit der Unterzeichnung nicht mehr geändert wurden. Damit wird gegenüber handschriftlichen Signaturen zusätzliche Sicherheit gewonnen.

Unterstütztes Verarbeiten: Am Ende des Arbeitsablaufes werden die Daten eines Formulars anwendungsspezifisch weiterverarbeitet. Um eine Automatisierung dieser Weiterverarbeitung zu erleichtern, soll das Formularsystem über eine klare Schnittstelle zu anderen Systemen verfügen, über die diese auf die Formulardaten zugreifen können.

Unterstütztes Ablegen: Das System soll digitale Formulare archivieren. Dazu wird eine Benutzeroberfläche angeboten, über welche Formulare unter einem bestimmten Schlüssel abgespeichert werden können. Auch das Abrufen bereits abgelegter Formulare über den entsprechenden Schlüssel soll

über eine Benutzeroberfläche möglich sein.

Unterstütztes Ausfüllen: Der Kunde soll die Formulare über eine graphische Benutzeroberfläche ausfüllen können, damit er nicht mit technischen Details konfrontiert wird. Das Ausfüllen über diese Oberfläche soll zudem interaktiv erfolgen. Falls bestimmte Eigenschaften der Formulardaten verletzt werden, welche unmittelbar aufgrund der Eingaben erkannt werden können, so soll dies angezeigt werden. Damit wird der Kunde direkt auf Fehler aufmerksam gemacht und die Anzahl eingereichter Formulare mit fehlerhaften Inhalten kann reduziert werden. Der Kunde soll aber nicht zu bestimmten Eingaben gezwungen sein, damit der Mitarbeiter in der Formularverwaltungsstelle Ausnahmereglungen berücksichtigen kann.

Hohes Maß an Benutzbarkeit: Formularverwaltungsstellen und Kunden sollen möglichst wenig technisches Wissen für den Umgang mit dem System benötigen. Darum soll es geeignete Benutzeroberflächen geben.

Parallele Benutzung von Papierformularen: Das System soll den Einsatz von digitalen Formularen parallel zum Einsatz von Papierformularen ermöglichen.

Flexibilität gegenüber Änderungen am Arbeitsablauf: Es ist davon auszugehen, dass es zu Änderungen am unterstützten Arbeitsablauf kommt. Dies soll die Systemarchitektur berücksichtigen.

Geringer softwaretechnischer Einführungsaufwand: Es soll ein guter Kompromiss zwischen dem Erfüllen der Anforderungen und dem softwaretechnischen Einführungsaufwand gefunden werden.

Nun sind alle Grundlagen zur Ableitung der Systemarchitektur gegeben. Im nächsten Kapitel wird ein System vorgeschlagen, welches diesen Anforderungen entsprechen soll. Inwiefern die Anforderungen dadurch umgesetzt werden konnten, wird danach diskutiert.

3 Systembeschreibung

Nun wird die Architektur des vorgeschlagenen Systems beschrieben. Zunächst wird ein kurzer Überblick über die technische Ausgangssituation gegeben, dann der Zugriff auf die verschiedenen Daten beschrieben und anschließend die Funktionsweise der bereitgestellten Dienste geklärt. Auf Basis dieser Ergebnisse wird dann in den folgenden Kapiteln eine Realisierung dieser Architektur für das motivierende Beispiel vorgestellt und diskutiert, inwiefern die Anforderungen erfüllt werden konnten.

3.1 Überblick

Da das System mit Daten arbeiten muss, welche sich auf unterschiedlichen Rechnern befinden, handelt es sich um ein verteiltes System. Weil das System lediglich Dienste anbietet und sich ansonsten nur passiv verhält, bietet sich eine Client-Server-Architektur an. Auf dem Server läuft ein Programm, welches die Dienste anbietet mit denen die funktionalen Anforderungen abgedeckt werden. Als Client fungieren die Rechner der Formularverwaltungsstellen und des Kunden. Diesen werden unterschiedliche Dienste angeboten. Von den Clients werden neben dem Aufrufen der Dienste hauptsächlich digitale Repräsentationen der Formulare mit dem Server ausgetauscht. Der Server greift zudem auf verschiedene bereits vorhandene digitale Datenbestände zu. Da diese Datenbestände technisch unterschiedlich aufgebaut sein können, werden diese als heterogene Datenbestände bezeichnet. Abbildung 5 zeigt einen ersten Überblick über diese Client-Server-Architektur.

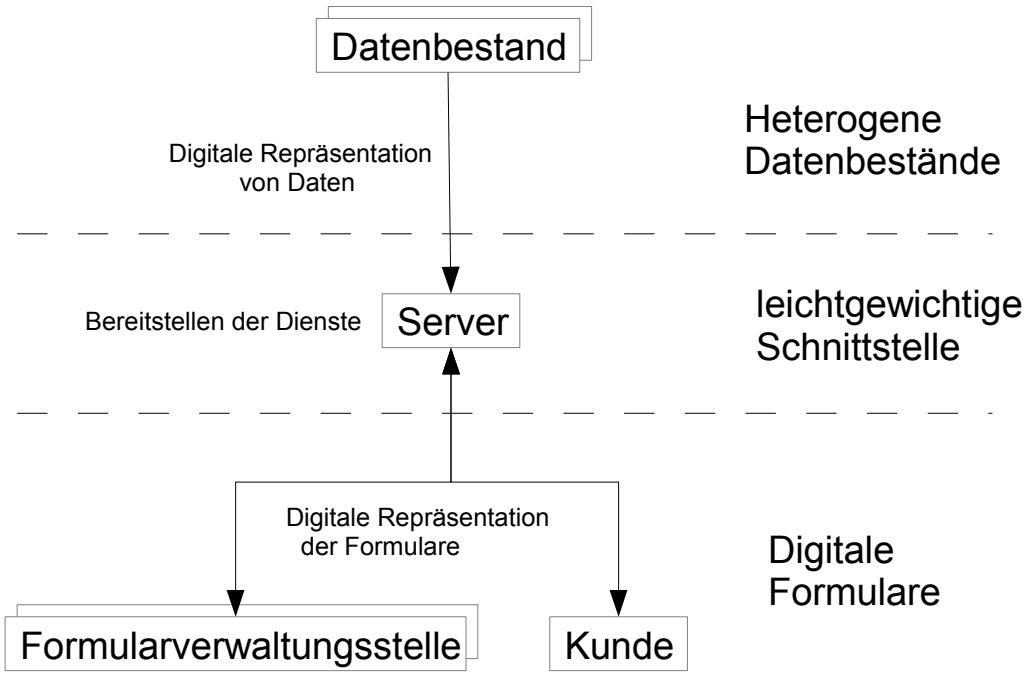

Abbildung 5: Ein Überblick über die Client-Server-Architektur des Systems

Zur weiteren Beschreibung des Systems werden nun nacheinander drei Kernprobleme dieser Architektur genauer betrachtet: Der Zugriff auf die digitalen Formulare, der Zugriff auf die heterogenen Datenbestände und das Bereitstellen der Dienste. Vor dem Hintergrund der Wahl der nicht-funktionalen

Anforderungen bildet eine Lösung dieser Probleme eine leichtgewichtige Schnittstelle zwischen digitalen Formularen und heterogenen Datenbeständen. Es folgt die Diskussion dieser Kernprobleme.

3.2 Zugriff auf digitale Formulare

Um mit den Formularen arbeiten zu können wird ein Verfahren zum Verändern der Formulardaten benötigt. Dies beinhaltet sowohl die Übertragung der Daten zwischen Client und Server, als auch das eigentliche Lesen und Schreiben der Formularinhalte. Um dies klären zu können, muss zunächst eine Formularrepräsentation festgelegt werden.

Zur digitalen Repräsentation der Formulare werden Flexoforms verwendet. Diese stellen ein Formular mit genau einer XML-Datei und dem zugehörigen Schema dar. Sie sind geeignet zur Darstellung über eine Web-Oberfläche und können dort interaktiv bearbeitet werden. Zudem lässt sich ihre Struktur gut automatisch weiterverarbeiten. Eine kurze Begründung dieser Eigenschaften befindet sich am Anfang dieser Arbeit.

Bei der Bearbeitung dieser XML-Dateien über den Server werden diese vollständig übertragen, da für viele Dienste ein Großteil der Formulardaten benötigt wird. Zudem belegen die Formulare nur wenig Speicherplatz und dies ist die softwaretechnisch einfachste Lösung der Übertragung. Für das eigentliche Bearbeiten der XML-Struktur bieten sich verschiedene verbreitete Möglichkeiten an.

Eine Möglichkeit bildet das Document Object Model (DOM) [9]. Beim Zugriff über diesen Standard wird die gesamte Baumstruktur des Dokuments als Objektgraph in den Speicher geladen. Dort kann dann wahlfrei gelesen und geschrieben werden. Bei Änderungen kann anschließend aus dem Speicherabbild wieder ein XML-Dokument erzeugt werden. Diese Vorgehensweise hat einen hohen Speicherbedarf. Vor dem Erzeugen des Speichermodells wird typischerweise anhand des Schemas die Gültigkeit des Dokuments überprüft.

Eine andere verbreitete Möglichkeit beschreibt der Simple API for XML-Standard (SAX) [1]. Hier wird die Baumstruktur des Dokuments sequentiell durchlaufen und dabei Ereignisse ausgelöst. Beispielsweise beim Beginn oder Ende eines neuen Knotens. Basierend auf solchen Ereignissen erfolgt dann die Bearbeitung des Dokuments. Diese Lösung ist speichereffizienter. Zudem kann die Verarbeitung parallel zum Durchlaufen des Dokuments erfolgen. Um dem Geschwindigkeitsvorteil und die Speichereffizienz aber tatsächlich auch zu erreichen, kann aber vor dem Beginn der Bearbeitung keine vollständige Prüfung der Gültigkeit vorgenommen werden.

In der später vorgestellten Realisierung der Architektur wird das Verfahren nach dem Document Object Model-Standard verwendet, da der Speicher-

bedarf der Dokumente gering ist und das wahlfreie Lesen und Ändern der Dokumente die angeforderte Flexibilität ermöglicht. So werden Dokumente übertragen und auf Grundlage ihres Schemas bearbeitet.

3.3 Zugriff auf heterogene Datenbestände

Neben dem Zugriff auf die Formulare wird auch ein Verfahren zum Lesen der Datenbestände benötigt. Im Gegensatz zu den Formularen, welche alle in Form bestimmter XML-Dateien vorliegen, können die Datenbestände softwaretechnisch unterschiedlich realisiert sein. Typische Bespiele wären relationale Datenbanken und XML-Dateien. Wegen diesen unterschiedlichen Techniken liegen die Daten insbesondere in verschiedenen Datenstrukturen vor. Dies erschwert den Zugriff im Vergleich zu den Formularen erheblich. Weiterhin muss berücksichtigt werden, dass die Daten diesmal nicht einfach vollständig zum Server übertragen werden können, um diese dann dort zu verarbeiten, da die Datenbestände potentiell sehr groß sind.

Die Daten müssen also übertragen werden ohne den gesamten Bestand vollständig herunterzuladen. Daher erfolgt die Übertragung nach dem Remote-Access-Muster. Das bedeutet, dass der Server eine Anfrage generiert und an den entfernten Rechner schickt. Dort wird die Anfrage lokal ausgewertet und das Ergebnis rückübermittelt. Beim vollständigen Herunterladen bestand die Anfrage nur aus der Aufforderung, alle Daten zu übertragen. Nun muss die Anfrage eine Beschreibung derjenigen Daten beinhalten, welche tatsächlich benötigt werden. Da die Daten aber in unterschiedlichen Strukturen vorliegen, stellt sich die Frage, wie eine solche Beschreibung der Daten aussehen sollte.

Eine Möglichkeit wäre, die Anfrage in der Server-Anwendung direkt passend für eine bestimmte Datenstruktur zu generieren. Also beispielsweise SQL-Text für relationale Datenbanken. Kommt es nun jedoch zu Änderungen am Datenbestand, so müssen eventuell die Anwendungsprogramme umgeschrieben werden, da sich die Form der Anfragen ändert und diese direkt in den Anwendungsprogrammen generiert werden. Um die Anwendungsprogramme also von den Strukturen der Datenbestände zu entkoppeln, wird eine gemeinsame Datenstruktur für alle Datenbestände definiert. Somit können die Anfragen für eine gemeinsame Datenstruktur formuliert werden und Änderungen an den Datenrepräsentationen beeinflussen die Anwendungsprogramme nicht.

Um die Anfrage für einen konkreten Datenbestand dann aber auch umsetzen zu können, wird eine Abbildung zwischen der speziellen Struktur des Datenbestandes und der gemeinsamen Struktur benötigt. Abbildung 6 zeigt, wie eine solche Abbildung aussehen kann. In diesem Beispiel werden Tabel-

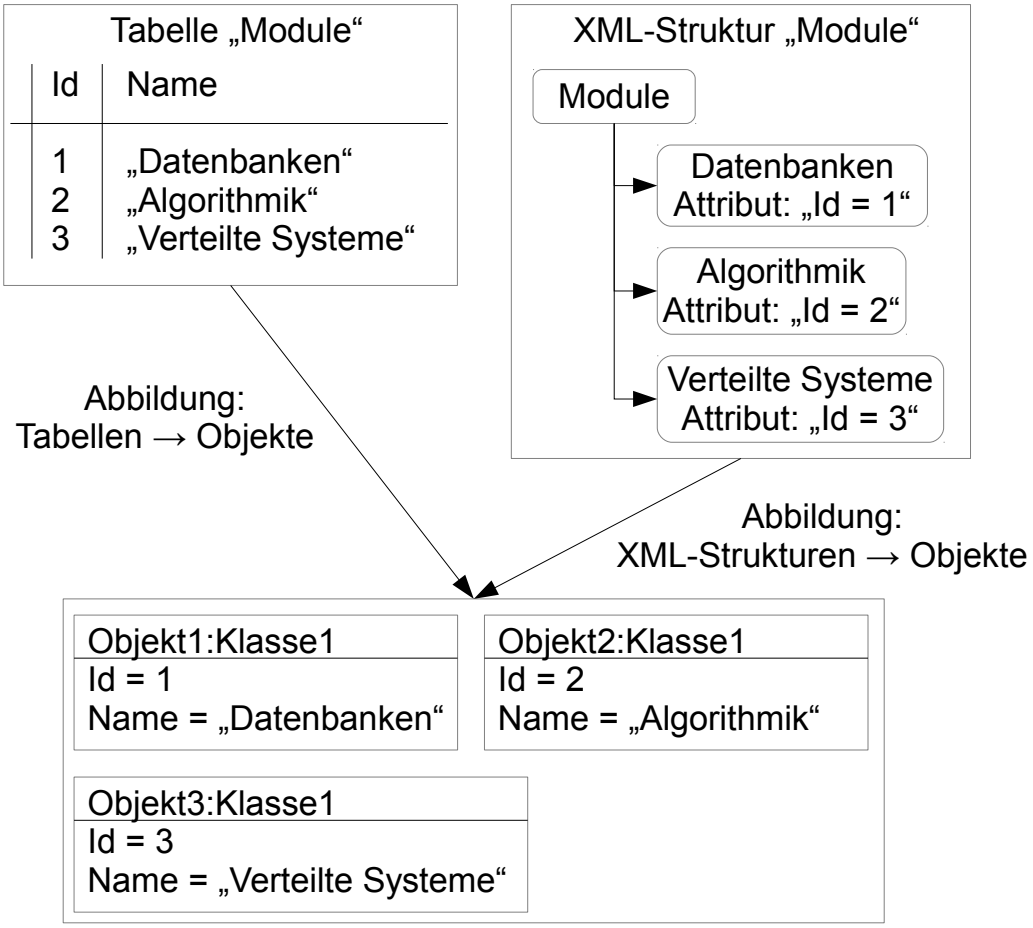

Abbildung 6: Ein Beispiel für die Vereinheitlichung der Datenstrukturen beim Zugriff auf die Datenbestände

len und die Baumstruktur eines XML-Dokuments mit Objekten dargestellt. Objekte wurden ausgewählt, da diese eine einfache Repräsentation in den Anwendungsprogrammen besitzen und somit zur dortigen Weiterverarbeitung geeignet sind. Liegt also eine Anfrage nach Objekten vor, so kann diese auf Tabellen umgesetzt werden, indem jede Tabelle als Klasse und jede Zeile einer Tabelle als Objekt dieser Klasse angesehen wird. Wie XML-Dokumente mit Objekten repräsentiert werden können, beschreibt der bereits bei den Formularen eingesetzte DOM-Standard.

Benötigt werden zum Zugriff auf die Datenbestände im Allgemeinen also drei Softwarekomponenten: Eine zur Realisierung der Kommunikation, eine zur Realisierung der Abbildung zwischen den Datenstrukturen und eine zum Umsetzen der Anfragen auf der gemeinsamen Datenstruktur. Da die Abbildung zwischen den Datenstrukturen ein sehr komplexes Problem sein kann, man denke beispielsweise an weiterführende Konstruktionen relationaler Daten wie Fremdschlüsselbeziehungen, bietet sich hier der Einsatz von vorhandenen Persistenzframeworks an. Da die Einführung solcher Frameworks für gegebene Datenbestände jedoch aufwändig sein kann, wird in der später vorgestellten Realisierung jedoch eine eigene Implementierung verwendet. Die Komplexität wird dadurch gemindert, dass nur die einfachsten Eigenschaften der Datenstrukturen abgebildet werden, also Konstruktionen wie Fremdschlüsselbeziehungen nicht direkt unterstützt werden. Da der Abbildungsalgorithmus aber durch eine klare Schnittstelle gegenüber dem restlichen System gekapselt ist und er für alle Datenbestände der entsprechenden Struktur wiederverwendet werden kann, ist dieser zur späteren Änderung und Erweiterung geeignet. Damit realisieren diese Softwarekomponenten den Zugriff auf die Datenstrukturen.

3.4 Bereitstellen der Dienste

Bisher wurde geklärt, wie das Serverprogramm auf die verschiedenen Datenquellen zugreifen kann. Im Folgenden wird die eigentliche Architektur des Serverprogramms diskutiert und dargestellt, wie es die in den funktionalen Anforderungen bestimmten Aufgaben erfüllt.

3.4.1 Softwaretechnische Architektur

Bevor bestimmte Dienste ausgeführt werden können, müssen diese von einem Benutzer angefordert werden. Wie in den Anforderungen bereits festgelegt wurde, soll diese Interaktion mit dem Benutzer über eine graphische Benutzeroberfläche erfolgen. Darum werden die Dienste über eine Weboberfläche angeboten, also in Form von statischen HTML-Seiten, welche über das

HTTP-Protokoll vom Browser des Benutzers angefragt und dann zugestellt werden. Eine solche Oberfläche sollte auch für die meisten Kunden intuitiv zu benutzen sein. Die zur Realisierung dieser Benutzeroberfläche und der Verarbeitung der Eingabedaten notwendigen Softwaremodule bilden eine erste Schicht der Serverarchitektur.

In einer zweiten Schicht befinden sich die Module, welche den eigentlichen Funktionsumfang der Anwendung zur Verfügung stellen. Beispielsweise ein Modul für das Prüfen von Formularen. Dieses erhält aus der oberen Schicht die notwendigen Benutzereingaben, also ein Speichermodell des Formulars, die zu prüfenden Eigenschaften und gegebenenfalls eine Referenz auf einen externen Datenbestand. Dann wird die Prüfung innerhalb dieses Moduls durchgeführt und das Ergebnis über die Benutzeroberfläche zurückgegeben.

Der Zugriff auf Formulare und externen Datenbestände wird über eine dritte Schicht abgewickelt. Dort befinden sich zwei Module. Das erste Modul nimmt XML-Formulare in Textform entgegen und liefert ein entsprechendes Speichermodell zurück. Die Formulare wurden zuvor innerhalb der ersten Schicht übertragen. Das zweite Modul in der dritten Schicht nimmt die Anfragen an die externen Datenbestände entgegen und liefert das Ergebnis der Anfrage zurück.

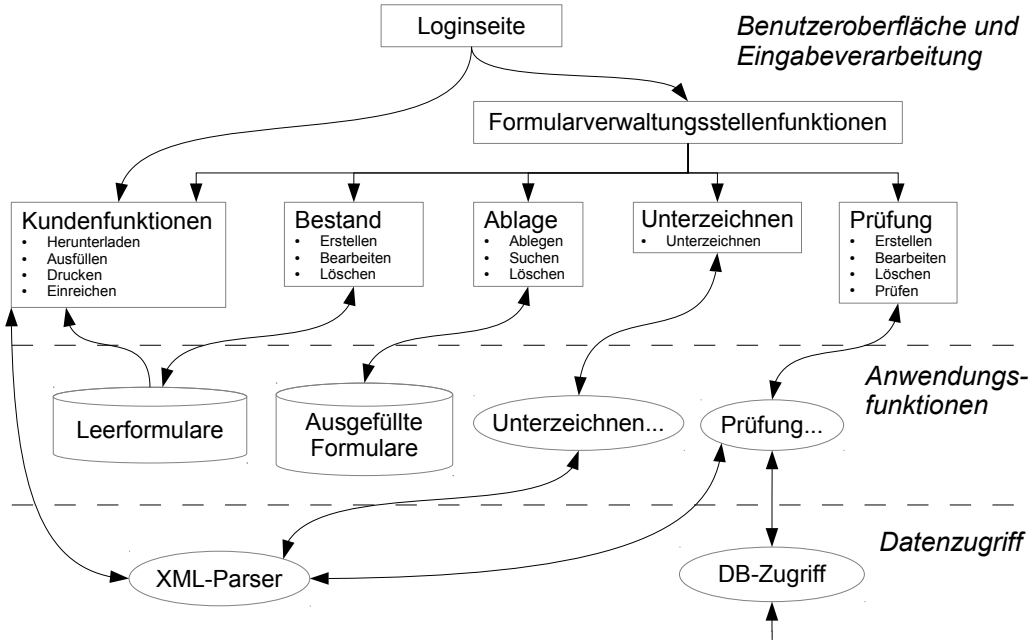

Abbildung 7: Die Software-Architektur des Servers

Abbildung 7 zeigt einen Überblick über die Software-Architektur des Ser-

vers. Innerhalb der ersten Schicht wird über eine Loginseite festgestellt, welcher Funktionsumfang dem Benutzer angeboten wird. Der Funktionsumfang für den Kunden ist stark eingeschränkt, während die Mitarbeiter der Formularverwaltungsstelle alle Funktionen verwenden können. Einige Funktionen, wie beispielsweise die Ablage-Funktionen, bestehen im Wesentlichen aus dem Verwalten eines Datensatzes. Daher sind für diese Funktionen keine eigenen Module in der Anwendungsschicht aufgeführt, sondern nur der entsprechende Datensatz. Die in der Anwendungsschicht angebotenen Funktionen werden in den folgenden Unterkapiteln noch genauer beschrieben und sind daher in dieser Abbildung nicht in der größten Granularität dargestellt. In der untersten Schicht befinden sich die Module für den Zugriff auf die Formulare und die externen Datensätze.

Diese Architektur folgt dem Schichtenmuster. Das bedeutet, dass die Softwarekomponenten im Wesentlichen nur mit anderen Komponenten der höheren und niedrigeren Schicht interagieren. Damit weist die Architektur eine geringe Kopplung und klare Beziehungen zu den funktionalen Anforderungen auf. Das Hinzufügen weiterer Anwendungsfunktionen entspräche dem Hinzufügen neuer Module in der Benutzeroberfläche sowie neuen Modulen in der Schicht der Anwendungsfunktionen. Veränderungen im Datenzugriff betreffen hingegen nur die unterste Schicht. So stellt die Architektur die Server-Dienste zur Verfügung und bleibt dabei leicht erweiterbar und änderbar.

3.4.2 Prüfen der Formulare

Als eine zentrale funktionale Anforderung an das System soll die Funktionsweise des Prüfens von Formularen als eine Anwendungsfunktion innerhalb des Servers genauer geklärt werden. Die Eingaben für die Prüfungsfunktionalität bestehen aus dem Speichermodell eines Formulars, einer Definition der zu prüfenden Eigenschaften und gegebenenfalls einer Referenz auf einen externen Datenbestand, gegen welchen geprüft wird. Das Speichermodell des Formulars wird über die Benutzeroberfläche und den XML-Parser erzeugt. Es wird angenommen, dass die Referenz auf den externen Datensatz ebenfalls von der Benutzeroberfläche geliefert wird. Je nach Anwendung könnte diese Referenz auch Teil der Definition der zu prüfenden Eigenschaften sein. Dann wird diese dort ausgelesen. Offen bleibt nur die Frage, in welcher Form die Definition der zu prüfenden Eigenschaften vorliegt.

Da potentiell beliebige Eigenschaften auf den Daten des Formulars und des Datenbestandes zu prüfen sind, wird eine sehr aussagekräftige Sprache benötigt. Daher würde es sich anbieten, die Prüfungen direkt in der Programmiersprache der Anwendung zu schreiben. Dies würde aber bei jeder Änderung an einem Prüfungskriterium ein erneutes Übersetzen von Quell-

texten bedeuteten. Da aber davon auszugehen ist, dass solche Änderungen häufiger auftreten, werden die Prüfungskriterien mittels einer Skriptsprache angegeben, welche zur Laufzeit interpretiert wird. Somit können beliebige Eigenschaften geprüft werden, Änderungen sind trotzdem einfach und schnell umzusetzen.

Die Prüfungsskripte werden also von einem Mitarbeiter mit Kenntnissen in der entsprechenden Skriptsprache geschrieben. Innerhalb des Skriptes wird auf das Speichermodell des Formulars zugegriffen. Zudem werden Anfragen an externe Datensätze generiert und von dem Modul der Datenzugriffsschicht verarbeitet. Auf diesen Daten werden dann vom Skript die notwendigen Eigenschaften geprüft. Schließlich endet die Verarbeitung des Skriptes damit, dass es einen Wahrheitswert für das Ergebnis der Prüfung, sowie eine Meldung für die Benutzeroberfläche an das Hauptprogramm zurückgegeben werden. Falls die Prüfung nicht erfolgreich ist, kann über die Meldung dem Benutzer übermittelt werden, welche Eigenschaft des Formulars die Ursache für das Fehlschlagen darstellt. So können die Prüfungskriterien leistungsstark und trotzdem flexibel realisiert werden.

Nach diesem Muster erfolgt die Prüfung der Vollständigkeit und Korrektheit von Formularen. Der Vorgang des Prüfens von Unterschriften und Einhaltung des Formularschemas ist unveränderlich und wird daher nicht in der Skriptsprache geschrieben. Für das Prüfen der Einhaltung des Schemas gibt es zahlreiche Software-Bibliotheken. Das Verfahren, nach dem Unterschriften geprüft werden, wird im nächsten Abschnitt beschrieben.

3.4.3 Unterzeichnen der Formulare

Als eine zweite Anwendungsfunktion innerhalb der Architektur soll das Unterzeichnen von Formularen geklärt werden. Wie bereits in den Anforderungen beschrieben soll das Unterzeichnen eines Formulars gegenüber anderen Mitarbeitern von Formularverwaltungsstellen belegen, wer das Formular unterzeichnet hat und sicherstellen, dass sich die Formulardaten seitdem nicht mehr geändert haben.

Hierzu kann ein Verfahren verwendet werden, welches in Abbildung 8 dargestellt ist. Beim Unterzeichnen wird zunächst eine Hash-Funktion auf den Datenteil des Formulars angewendet. Der generierte Hash-Wert wird dann mit dem privaten Schlüssel des Unterzeichners verschlüsselt. Der private Schlüssel ist Ausgangswert der Verschlüsselung und nur seinem Besitzer bekannt. Umgekehrt werden kann diese Verschlüsselung mit dem öffentlichen Schlüssel des Besitzers, der auch den anderen Mitarbeitern der Formularverwaltungsstellen bekannt sein muss. Das Ergebnis dieser Verschlüsselung bildet die digitale Signatur. Sie wird zusätzlich zum Datenteil in das Formular

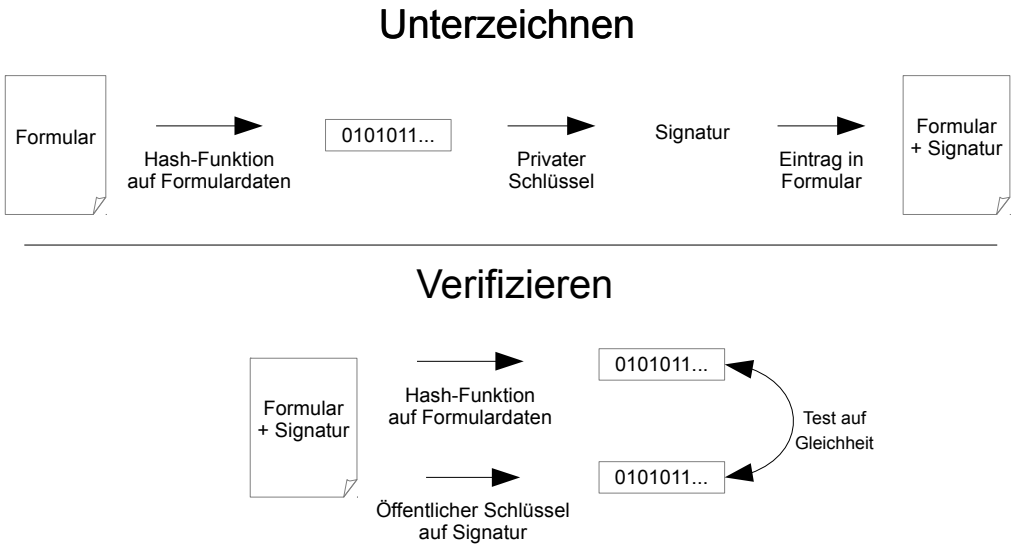

Abbildung 8: Ein Verfahren zur digitalen Signatur von Formularen

eingetragen.

Beim Verifizieren einer Signatur wird erneut der Hash-Wert über den Datenteil des Formulars berechnet. Zudem wird aus der Signatur mit dem öffentlichen Schlüssel des Unterzeichners der ursprüngliche Hash-Wert rekonstruiert. Sind beide Werte gleich, so kann bis auf Kollisionen bei der Hash-Generierung und im Rahmen der Sicherheit des Verschlüsselungsverfahrens garantiert werden, dass die Daten des aktuellen Formulars tatsächlich vom Besitzer des Schlüssels unterzeichnet worden sind.

Bei der vorgeschlagenen Verschlüsselung handelt es sich um ein sogenanntes asymmetrisches Verfahren, da zur Ver- und Entschlüsselung der Signatur verschiedene Schlüssel verwendet werden. Der Vorteil gegenüber symmetrischen Verfahren ist, dass nur der Unterzeichner einen Schlüssel geheim halten muss, zur Durchführung der Verifikation muss kein geheimer Schlüssel verwendet werden. Die asymmetrische Verschlüsselung kann mit dem RSA-Verfahren umgesetzt werden. Dieses hat eine hohe Verbreitung und verfügt damit über ausgereifte Softwarebibliotheken, welche in der Realisierung des Systems angewendet werden können. [4]

Der private und öffentliche Schlüssel jedes Mitarbeiters der Formularverwaltungsstellen kann im System automatisch generiert und verwaltet werden. Das Speichern und Bearbeiten der Schlüssel kann genau so erfolgen, wie die Verwaltung der Benutzerdaten, also des Benutzernamens und des Passworts, welche auch für den Login beim System verwendet werden. Der öffentliche

Schlüssel ist wie der Benutzername auch für andere Benutzer sichtbar, der private Schlüssel - analog zum Benutzerpasswort - dagegen nicht. Wie die Verwaltung der Benutzerdaten, also auch der beiden Schlüsselwerte, abläuft, wird hier nicht näher beschrieben.

3.4.4 Verwalten der Formulare

Neben dem Unterzeichnen und Prüfen der Formulare werden auch noch Dienste zur Verwaltung des Bestandes an Leerformularen, Verwaltung der Ablage und verschiedene Kundenfunktionen zur Bearbeitung der Formulare angeboten. Nachdem bereits zwei typische Anwendungsfunktionen zur Vorführung der Arbeitsweise der vorgeschlagenen Software-Architektur diskutiert wurden, wird hier noch ein Überblick über diese drei weiteren Funktionalitäten gegeben.

Die Kundenfunktionen umfassen das Herunterladen von Instanzen der Leerformulare, das interaktive Ausfüllen von Formularen, das Ausdrucken von Formularen und das Einreichen von Formularen. Beim Herunterladen wählt der Benutzer über die Benutzeroberfläche einen Eintrag aus der Liste der angebotenen Formulare aus. Anschließend wird das entsprechende Formular aus dem Datenbestand der Leerformulare per HTTP zum Benutzer übertragen. Welche Formulare angeboten werden, wird im Rahmen der Funktionalität der Bestandsverwaltung von dem Mitarbeiter einer Formularverwaltungsstelle konfiguriert. Eine mögliche Realisierung des interaktiven Ausfüllens wurde bereits bei der Entwicklung der Flexoforms vorgeschlagen. Zum Ausdrucken von Formularen ist eine Schaltfläche in der Benutzeroberfläche vorgesehen. Diese Schaltfläche öffnet einen Dialog zur Auswahl eines lokal vorliegenden Formulars, welches dann beim Benutzer ausgedruckt wird. Zum Einreichen von Formularen wird vom Benutzer eine Formularverwaltungsstelle und ein Formular ausgewählt. Das Formular wird dann im einfachsten Fall per E-Mail an einen der Formularverwaltungsstelle zugeordneten Mitarbeiter geschickt. Diese Zuordnung zwischen Formularverwaltungsstellen und Mitarbeitern ist Teil der Benutzerverwaltung des Systems.

Zur Organisation der Ablage werden Funktionen zum Ablegen, Suchen und Löschen von Formularen aus dem entsprechenden Datenbestand angeboten. Beim Ablegen wird ein Formular zum Server übertragen und dort dauerhaft abgespeichert. Zudem wird eine Datenbank unterhalten, in welcher zu jedem abgelegten Formular eine Indexstruktur vorliegt, über welche Formulare wieder gefunden werden können. Dies geschieht bei der Suchenfunktionalität. Wurden die Formulare aus dem Datenbestand über die Suche ausgewählt, so können Kopien davon erneut heruntergeladen werden, oder sie werden aus der Ablage wieder entfernt. Änderungen an abgelegten For-

mularen sind nicht möglich.

Die Verwaltung des Bestandes beinhaltet das Erstellen und Löschen von Formularen. Das Erstellen ermöglicht dem Benutzer das Zusammenstellen neuer Leerformulare aus den typischen Formularkomponenten sowie das Eintragen der benötigten Texte. Die softwaretechnische Realisierung ähnelt der des interaktiven Ausfüllens. Da zum Erstellen der Formulare jedoch im Falle der Flexoforms nicht nur die Anordnung der Komponenten, sondern zusätzlich auch CSS- und XPath-Ausdrücke sowie eine Schema-Beschreibung benötigt wird, kann die Benutzeroberfläche das Generieren der XML-Datei für das Formular lediglich unterstützen. Eine manuelle Bearbeitung der XML-Datei und ihres Schemas ist nach wie vor notwendig. Nach dem Erstellen wird das Leerformular mit seinem Schema abgespeichert und kann von anderen Benutzern über die entsprechende Funktionalität heruntergeladen werden. Beim Löschen von Formularen werden diese aus dem Datenbestand ersatzlos entfernt. Diese Formulare sollten dann von den Formularverwaltungsstellen nicht mehr akzeptiert werden.

4 Realisierung

Um die beschriebene Architektur zu bewerten, wurde eine prototypische Realisierung in der Programmiersprache Java entwickelt. Die Realisierung umfasst den Zugriff auf die Daten, die Benutzeroberfläche und als Beispiel für eine der Anwendungsfunktionen das Prüfen von Formularen mit Hilfe einer Skriptsprache. Dabei werden in diesem Kapitel die Aspekte beschrieben, welche sich nicht auf den konkreten Anwendungsfall beziehen, sondern nur auf die allgemeine Klasse von Arbeitsabläufen. Der konkrete Anwendungsfall wird im nächsten Kapitel zur abschließenden Bewertung des Systems diskutiert.

4.1 Objektrelationale Abbildung

Als typischer Anwendungsfall wurde in der Realisierung der Zugriff auf eine relationale Datenbank als Datenbestand realisiert. Hierzu wurde ein Adapterprogramm entwickelt, welches unabhängig von der Serveranwendung auf einem beliebigen Rechner der Datenverwaltungsstelle eingesetzt werden kann. Es ermöglicht das Entgegennehmen von Anfragen über das entsprechende Netzwerk, die Übersetzung der Anfrage für relationale Datenbestände und das Auswerten dieser Anfrage.

In der Architektur wurde gefordert, die Anfragen auf allen Datenbeständen für eine gemeinsame Datenstruktur zu stellen. Als eine solche gemein-

```
public class DataClass {
   public String className;
   public String[] fieldNames;
   public DataType[] fieldTypes;
}

public enum DataType {
   BOOLEAN, INT, FLOAT, STRING;
}

public class DataObject {
   public String className;
   public String[] fieldValues;
}

public interface DataInterface {
   public DataClass[] getClasses();
   public DataClass[] getClasses(String... classNames);
   public String[] getClassNames();
   public DataObject[] getObjects(String className);
}
```

Quelltext 2: Die gemeinsame Datenstruktur der Datenbestände

same Datenstruktur wurde in dieser Realisierung eine Menge von Objekten ausgewählt. Quelltext 2 zeigt die Implementierung dieser Datenstruktur und ihrer Schnittstelle. Mit Hilfe der Klasse *DBClass* werden Objekte mit gemeinsamen Eigenschaften zusammengefasst. Zur Identifikation der *DBClass*-Objekte wird ein eindeutiger Name verwendet. Die anderen Attribute definieren die Eigenschaften. Als Typen der Eigenschaften sind analog zu den entsprechenden Java-Typen *BOOLEAN*, *INT*, *FLOAT* und *STRING* vorgesehen. Zudem haben die Eigenschaften einen innerhalb ihrer Klasse eindeutigen Namen. Mit der Java-Klasse *DBObject* werden die eigentlichen Objekte repräsentiert. Sie referenzieren ihre Klasse über deren Namen. Zudem besitzen sie die mit ihrer Klasse spezifizierten Eigenschaften. Der Zugriff auf die Objekte erfolgt über die *DBInterface*-Schnittstelle. Sie ermöglicht das Suchen nach bestimmten Klassen sowie das Übertragen der Objekte einer bestimmten Klasse. Sollten Datenbestände mit einer sehr großen Anzahl von Objekten einer Klasse unterstützt werden, so könnten weitere Methoden zum Zugriff auf Objekte mit bestimmten Eigenschaften hinzugefügt werden, damit nicht alle Objekte übertragen werden müssen.

Das Adapterprogramm beantwortet also solche Anfragen auf Objekten. Die Abbildung zum relationalen Datenbestand erfolgt wie im Architekturkapitel beschrieben: Klassen entsprechen Tabellen, die Eigenschaften den Spal-

ten und jedes Objekt einer Zeile. Welche Tabellen und Spalten des Datenbestandes über den Adapter bereitgestellt werden, wird in einer Konfigurationsdatei beschrieben. Dort kann zudem eine Umbenennung vorgenommen werden, also beispielsweise eine Tabelle unter anderem Namen als Klasse exportiert werden. Ein Beispiel für eine solche Konfigurationsdatei befindet sich im nächsten Kapitel. Weiterführende Aspekte von objektrelationalen Abbildungen, wie beispielsweise Objektreferenzen über Fremdschlüsselbeziehungen, wurden hier nicht implementiert. Profitiert die Anwendung von einer mächtigeren Abbildung, so lohnt sich statt einer eigenen Implementierung der Einsatz von Persistenzframeworks wie im objektrelationalen Fall beispielsweise der Java Persistence API.

Die gesamte Abwicklung einer Anfrage spielt sich dann folgendermaßen ab. Ein Anwendungsprogramm auf dem Server benötigt Objekte eines bestimmten Datenstandes. Dann wird zunächst über eine Abfrage der im Datenbestand vorhandenen Klassen geprüft, ob der Datenbestand überhaupt über Objekte mit den geforderten Eigenschaften verfügt, also ob diese den geforderten Namen und die geforderte Typisierung haben. Ist dies nicht der Fall, so kann die Anfrage nicht durchgeführt werden und es findet eine Fehlerbehandlung statt. Sind die Objekte vorhanden, so werden diese über die *getObjects*-Methode angefragt. Dieser entfernte Methodenaufruf findet in der Realisierung als Remote Method Invocation statt. Das Serverprogramm wird blockiert, während im Adapterprogramm über die Konfigurationsdatei mittels der Zuordnung zwischen Klassennamen und Tabellennamen sowie Spaltennamen und Eigenschaftennamen eine Abfrage an den relationalen Datenbestand gestellt wird. Ist die Konfiguration fehlerhaft, so muss erneut eine Fehlerbehandlung durchgeführt werden. Die Abfrage an den relationalen Datenbestand erfolgt mittels SQL-Ausdrücken über den sogenannten Java Database Connectivity-Treiber [3], welcher für alle gängigen relationalen Datenbanksysteme vorliegt. Mit dem Ergebnis der SQL-Anfragen werden entsprechend der Konfiguration die Objekte erstellt, serialisiert, übertragen und die Ausführung in der Serveranwendung fortgesetzt.

Mit diesem Adapterprogramm kann auf die Daten beliebiger relationaler Datenbestände, welche über einen Java Database Connectivity-Treiber verfügen, zugegriffen werden. Die Konfiguration des Programms für einen bestimmten Datenbestand umfasst die Konfiguration des Treibers und der objektrelationalen Abbildung. Eine Diskussion der Vor- und Nachteile dieses Verfahrens wird im nächsten Kapitel anhand des Anwendungsbeispiels vorgenommen.

4.2 Benutzeroberfläche

Im Rahmen der prototypischen Implementierung wurde eine Benutzeroberfläche realisiert. Die Interaktion mit dem Benutzer findet wie in der Architektur beschrieben über eine Web-Oberfläche statt, also im wesentlichen mittels des Übertragens von HTML-Seiten über das HTTP-Protokoll. Zur Realisierung dieser Oberfläche wurde das Apache Wicket-Framework verwendet. [7]

Dieses Framework verknüpft die Darstellung der Webseiten über HTML-Dateien mit der Verarbeitungslogik durch Java-Programme. So ist jeder Webseite eine HTML-Datei zugeordnet, welche die statischen Elemente der Seite repräsentiert. Hinzu kommt eine Java-Klasse, welche die Verarbeitung der Eingaben und die Generierung der Inhalte übernimmt. Eine unmittelbare Interaktion mit dem Benutzer kann darüber hinaus beispielsweise über JavaScript mit Ajax erfolgen. Das eingesetzte Framework erlaubt dabei die Verwendung von Ajax ohne JavaScript.

Somit ist das Framework zur Realisierung von Weboberflächen mit einem hohen Grad an Interaktion geeignet. Dies begünstigt die Realisierung des interaktiven Ausfüllens und Erstellens von Formularen, welche im Rahmen dieses Prototyps nicht durchgeführt wurde. Die klare Trennung von Darstellung und Kontrollfluss erleichtert zudem das Wiederverwenden und Erweitern von innerhalb des Frameworks erstellten Komponenten.

Das Serverprogramm interagiert also über HTML-Seiten und Java-Klassen mit dem Benutzer. Der Benutzer fordert eine Seite an, mit Hilfe der zugeordneten Java-Klasse wird diese dynamisch erstellt und übertragen. Der Benutzer wählt die gewünschte Anwendungsfunktion aus, sodass diese wie im Java-Programm definiert aufgerufen wird. Mit dem Ergebnis wird erneut eine HTML-Seite generiert und dem Benutzer das Ergebnis somit graphisch dargestellt. Als Beispiel der Arbeitsweise einer solchen aufgerufenen Anwendungsfunktion wurde das Prüfen von Formularen realisiert.

4.3 Einbindung der Skriptsprache

Der Prototyp unterstützt das Prüfen von Formularen. Wie bereits beschrieben wurde geschieht dies, indem ein Skript mit Schnittstellen zu dem Modell eines Formulars und einem oder mehreren Datenbeständen ausgeführt wird. Der Interpreter des Skripts gibt dann das Prüfungsergebnis an das Hauptprogramm zurück.

Als Skriptsprache wurde in der Realisierung Jython [8] verwendet. Jython ist ein in Java implementierter Python-Interpreter, welcher Java-Bytecode generiert und in der JVM ausführt. Damit kann beliebiger Java-Bytecode in das Skript eingebunden werden. Jython verbindet die gute Lesbarkeit und

Flexibilität von Python-Programmen mit der hohen Verfügbarkeit an ausgereiften Java-Bibliotheken.

Vor der Ausführung eines Prüfungsskriptes wird zunächst ein DOM-Modell des Formulars generiert. Auf dieses Modell kann im Skript dann mit der gleichen Bibliothek zugegriffen werden, mit welcher es auch erstellt wurde. Dazu wird beim Initialisieren des Jython-Interpreters der Wurzelknoten des Modells an das Skript übergeben. Dann wird das Skript ausgeführt. Die Rückgabe liegt zum Ende der Ausführung als Wert einer bestimmten Variable vor und wird vom Interpreter zur Verfügung gestellt. Zum Zugriff auf Datenbestände aus dem Skript heraus können weitere Java-Bibliotheken eingebunden werden. Zur weiteren Untergliederung der Prüfungskriterien kann innerhalb eines Prüfungsskriptes auch ein anderes Prüfungsskript oder sogar eine Java-Bibliothek importiert werden. Quelltext 3 zeigt den Aufruf des Interpreters. Ein Beispielskript für konkrete Formulare und Datenbestände befindet sich im nächsten Kapitel.

So realisiert der Prototyp das Prüfen von Eigenschaften auf den Formularen unter Verwendung von Daten aus externen Datenbeständen. Anhand der durchstichsartigen Umsetzung dieser Anwendungsfunktion, welche auf alle Schnittstellen zugreift, sollte die Arbeitsweise der zuvor beschriebenen Software-Architektur verdeutlicht werden. Für die Umsetzung weiterer funktionaler Anforderungen wäre der Prototyp entsprechend zu erweitern. Es bleibt zu bewerten, ob die Anforderungen erfüllt werden konnten. Dies erfolgt im nächsten Kapitel.

```
public static boolean check(Node root, String check) {
        PythonInterpreter python = new PythonInterpreter();
        python.exec("from org.w3c.dom import Node");
        python.set("root", root);
        python.exec(check);
        PyObject resultValue = python.get("resultValue");
        return resultValue.asString().equals("true");
}
```

Quelltext 3: Aufruf des Jython-Interpreters mit Übergabe von Argumenten und Rückgabewert

5 Fallstudie

Anhand des am Anfang dieser Arbeit vorgestellten motivierenden Beispiels soll nun geklärt werden, inwiefern der dort beschriebene Arbeitsablauf von einer Realisierung des Systems profitieren würde, also ob das System die gestellten Anforderungen erfüllen kann. Dies betrifft insbesondere die nicht-

funktionalen Anforderungen und damit den Einführungs- und Wartungsaufwand. Daher werden diese Aspekte im folgenden genauer beschrieben.

5.1 Entwicklung und Einführung des Systems

Zunächst müssen für alle digitalen Datenbestände passende Adapter implementiert und konfiguriert werden. Der prototypisch entwickelte Adapter lässt sich für jede relationale Datenbank einsetzen, für welche ein JDBC-Treiber existiert und die Daten entsprechend der implementierten objektrelationalen Abbildung vorliegen, also jedes Objekt einer Zeile entspricht. Dann müsste der Adapter lediglich passend konfiguriert werden. Quelltext 4 zeigt eine entsprechende Konfiguration für ein hypothetisches Modulhandbuch. Im motivierenden Arbeitsablauf gab es jedoch noch einen zweiten Datenbestand. Handelt es sich dabei ebenfalls um einen relationalen Datenbestand, so kann der gleiche Adapter möglicherweise mit einer anderen Konfiguration erneut eingesetzt werden. Handelt es sich um eine andere Technologie, so muss nach dem gleichen Prinzip ein anderer Adapter programmiert werden. Zumindest zur Konfiguration der Adapter ist eine genaue Kenntnis des Datenbankstruktur notwendig, sodass diese in Kooperation mit der Datenverwaltungsstelle erfolgen muss.

```
EXPORT TABLE "module" AS CLASS "modulhandbuch";
EXPORT COLUMN "id" AS FIELD "nummer" OF CLASS "modulhandbuch"
 AND TYPE STRING;
EXPORT COLUMN "sws" AS FIELD "semesterwochenstunden" OF CLASS
 "modulhandbuch" AND TYPE STRING;
EXPORT COLUMN "cp" AS FIELD "leistungspunkte" OF CLASS
 "modulhandbuch" AND TYPE INT;
EXPORT COLUMN "level" AS FIELD "level" OF CLASS "modulhandbuch"
 AND TYPE INT;
EXPORT COLUMN "desc" AS FIELD "beschreibung" OF CLASS
 "modulhandbuch" AND TYPE STRING;
EXPORT COLUMN "resp" AS FIELD "verantwortlicher" OF CLASS
 "modulhandbuch" AND TYPE STRING;
```

Quelltext 4: Konfiguration der objektrelationalen Abbildung des prototypisch realisierten Adapters

Neben den Adaptern muss die Serveranwendung programmiert werden. Die Benutzeroberfläche müsste gegenüber dem Prototypen stark erweitert werden, da dieser nur einen kleinen Teil der in den funktionalen Anforderungen geforderten Dienste anbietet. Es müssten also zusätzliche HTML-Dateien und damit assoziierte Java-Klassen geschrieben werden. Für Seiten mit besonderen Interaktionsmöglichkeiten würden Ajax-basierte Lösungen hinzugefügt werden, beispielsweise beim Ausfüllen und Erstellen von Formularen.

Zudem müssten einige zusätzliche Module in der Anwendungsschicht der Serveranwendung ergänzt werden, da der Prototyp lediglich das Prüfen von Formularen unterstützt. Einige dieser weiteren Anwendungen, wie beispielsweise das Signieren von Formularen, das Verwalten diverser Formularbestände und die Benutzerverwaltung, wurden in der Systembeschreibung erläutert aber nicht realisiert. Die Benutzerverwaltung und damit auch die Verwaltung der digitalen Signaturen müsste zudem zunächst konfiguriert werden.

```
modules = moduleHandbook.getModules()

documentIds = []
for i in range(root.getChildNodes().getLength()):
  data = root.getChildNodes().item(i)
  for j in range(data.getChildNodes().getLength()):
    if data.getChildNodes().item(j).getNodeName() == "module":
      documentIds.append(data.getChildNodes().item(j)...)

moduleHandbookIds = []
for key in modules.keySet():
  moduleHandbookIds.append(key)

if set(documentIds) <= set(moduleHandbookIds):
  resultValue = "true"
else:
  resultValue = "false"
```

Quelltext 5: Prüfung auf Existenz von Modul-IDs in Jython

Neben der Entwicklung der Adapter- und Serverprogramme müssen noch weitere Software-Artefakte erstellt werden. Für jedes Papierformular muss ein entsprechendes digitales Formular realisiert werden. Je nach Funktionsumfang des Serverprogramms erfolgt dies teilweise automatisiert, da das unterstützte Erstellen von Formularen eine funktionale Anforderung an das System darstellt. Neben den Formularen müssen aber noch die Prüfungskriterien programmiert werden. Quelltext 5 zeigt ein Jython-Skript, welches zunächst alle Modul-IDs aus dem gegebenen Formular ausliest, danach alle Modul-IDs aus dem Modulhandbuch ausliest und schließlich prüft, ob die erste Menge von Modul-IDs in der zweiten enthalten ist. Damit wird also geprüft, ob alle in dem Formular eingetragenen Module auch tatsächlich im Modulhandbuch aufgeführt sind. Das Skript greift nicht direkt auf die Schnittstelle des Modulhandbuchs zu, da diese Schnittstelle sehr allgemein gehalten und damit schlecht zu lesen und umständlich zu handhaben ist. Daher wurde in der Serveranwendung eine Klasse definiert, welche den Zugriff auf das Modulhandbuch kapselt und eine anwendungsspezifische und damit gut lesbare

Schnittstelle hat. Zudem kann neben der Rückgabe des Prüfungsergebnisses über den gleichen Mechanismus eine Meldung zurückgegeben werden um den Benutzer über Details der Prüfung zu informieren.

Nach der Entwicklung und Einführung des Systems gestaltet sich der Arbeitsablauf im vollständig digitalisierten Fall dann wie folgt. Der Student lädt den Prüfungsplan herunter. Über die interaktive Benutzeroberfläche des Systems kann er den Prüfungsplan ausfüllen und erhält dabei gegebenenfalls bereits eine automatische Rückmeldung über seine Eingaben. Nach dem Ausfüllen reicht er den Prüfungsplan über das System bei seinem Mentor ein oder schickt es seinem Mentor als Anhang einer E-Mail. Der Mentor loggt sich dann im System ein und erhält den erweiterten Funktionsumfang. Er führt einige Prüfungsskripte aus. Sind alle Prüfungen erfolgreich, so kann er den Prüfungsplan digital signieren. Scheitert eine Prüfung, so erhält er eine Rückmeldung über die Ursache des Scheiterns. Nach einer manuellen Sichtung des Prüfungsplans über die Funktionalität des Ausfüllens entscheidet er, ob er den Prüfungsplan trotzdem signiert oder er ihn unsigniert an den Studenten zurückschickt. Früher oder später erhält der Student jedenfalls einen signierten Prüfungsplan. Diesen reicht er wiederum über das System oder eine E-Mail beim Prüfungsamt ein. Die Mitarbeiter des Prüfungsamtes haben andere Rechte im System und können daher analog zum Mentor weitere Prüfungen ausführen, dies betrifft insbesondere die Prüfung der Unterschrift des Mentors. Schließlich wird der Prüfungsplan wieder zurückgegeben, mit der Ablage-Funktionalität archiviert oder von einem System des Prüfungsamtes weiterverarbeitet. Das Weiterverarbeiten kann wie das hier beschriebene System die Daten des Formulars über ein DOM-Modell auslesen und verarbeiten. So gestaltet sich die digitalisierte Version des Arbeitsablaufes.

5.2 Wartung des Systems

Da die Wartung des Systems potentiell einen großen Aufwand verursachen kann, sollen hier einige realistische Änderungsszenarien und ihre Auswirkungen auf das beschriebene System diskutiert werden.

Es ist anzunehmen, dass es in Zukunft zu Änderungen am externen Datenbestand kommen wird, also im konkreten Fall dem Modulhandbuch. Es könnte sein, dass bei Beibehaltung der Struktur des Modulhandbuch neue Module hinzugefügt werden, Werte von Eigenschaften der Module geändert werden oder Module gelöscht werden. Dies entspricht dem Hinzufügen, Ändern und Löschen von Objekten aus Sicht der gemeinsamen Datenstruktur. Dies könnte sich höchstens auf einige Prüfungskriterien auswirken, es sollte jedoch kein Neuübersetzen von Quelltexten erforderlich sein. Das gleiche gilt, falls sich die Eigenschaften der Module, also die Klassen der Objekte, ändern.

Es könnte jedoch auch zu Änderungen an der Struktur des Modulhandbuches kommen. Beispielsweise durch das Verteilen von Modulen verschiedener Blöcke über mehrere Tabellen mit den gleichen Spalten. Dies würde Änderungen an der objektrelationalen Abbildung erfordern, da sich nun Objekte der gleichen Klasse in verschiedenen Tabellen befinden können. Erlaubt man dies, so werden keine Änderungen am Serverprogramm notwendig. Manche Änderungen an der Struktur des Modulhandbuchs würden lediglich ein Anpassen der Konfiguration des Adapters erfordern. Änderungen am Serverprogramm sollten nicht benötigt werden.

Denkbar wäre auch ein vollständiger Technologiewechsel. Gemeint ist damit, dass sich nicht nur die Repräsentation der Module mit Tabellen ändert, sondern eine andere Repräsentation gewählt wird, beispielsweise durch ein XML-Dokument. Nun würde ein neuer Adapter benötigt werden. Ist ein Adapter für XML bereits vorhanden, so kann dieser eventuell mit geänderter Konfiguration wiederverwendet werden oder mit geringen Änderungen am Quelltext verallgemeinert werden. Das Serverprogramm sollte dann nicht betroffen sein, solange die gleichen Informationen nur in geänderter Form vorliegen.

Werden neue Datenbestände an das System angeschlossen, so wird ein entsprechender Adapter benötigt. Möglicherweise kann ein Adapter wiederverwendet werden. Zudem müssen die mit dem neuen Datenbestand hinzugekommenen Anwendungsfunktionen im Serverprogramm implementiert werden.

Häufig wird es zudem zu Änderungen an den Prüfkriterien und den Prüfungsplänen kommen. Die Prüfkriterien wurden mit Hilfe einer Skriptsprache implementiert, damit bei Änderungen keine Quelltexte neu übersetzt werden müssen und die Lesbarkeit möglichst hoch ist, sodass Änderungen unproblematisch sind. Das Ändern an Formularen ist als Anwendungsfunktion im System vorgesehen. Denkbar ist beispielsweise, dass veraltete Formulare in der Verwaltung des Formularbestands entsprechend markiert werden und somit nicht mehr von Studenten heruntergeladen werden können, aber Prüfungsskripte, welche diesen Plänen zugeordnet sind, nach wie vor verfügbar bleiben.

Kommen neue Anwendungsfunktionen hinzu, so müssen diese im Serverprogramm implementiert werden. Die Schichtenarchitektur soll dies möglichst erleichtern, da die Kopplung gering ist und eine klare Beziehung zwischen funktionalen Anforderungen und Softwaremodulen besteht. So ist das System für das nachträgliche Hinzufügen von beliebigen funktionalen Anforderungen geeignet. Nicht-funktionale Anforderungen hingegen betreffen jedes Softwaremodul und können daher nur mit großem Aufwand nachträglich geändert werden.

5.3 Bewertung der nicht-funktionalen Anforderungen

Nachdem nun die zur Einführung und Wartung des Systems notwendigen Schritte beschrieben wurden, kann eine Bewertung der Umsetzung der nicht-funktionalen Anforderungen vorgenommen werden. Als nicht-funktionale Anforderungen wurden zu Beginn ein hohes Maß an Benutzbarkeit, die parallele Benutzung zu Papierformularen, große Flexibilität gegenüber Änderungen am Arbeitsablauf sowie ein geringer softwaretechnischer Einführungsaufwand identifiziert.

Ein hohes Maß an Benutzbarkeit wird durch die Weboberfläche gewährleistet. Der Benutzer ist im Umgang mit einer solchen Oberfläche bereits vertraut und wird durch das interaktive Ausfüllen der Formulare unterstützt. Problematisch in dieser Hinsicht ist allerdings das Erstellen der Formulare, da dort XML-Text geschrieben werden muss. Zudem erfordert das Bearbeiten von Prüfungskriterien durch die Mitarbeiter der Formularverwaltungsstellen Kenntnisse der entsprechenden Programmiersprache.

Die parallele Benutzung von Papierformularen sollte weitestgehend gewährleistet sein. Der Wechsel zwischen den Repräsentationen erfordert das Erstellen eines neuen digitalen Formulars über das interaktive Ausfüllen beziehungsweise das Ausdrucken eines digitalen Formulars. Signierte Formulare erfordern jedoch ein erneutes Unterzeichnen.

Die Flexibilität gegenüber Änderungen am Arbeitsablauf wurde zuvor bereits diskutiert. Die Schnittstelle zu den Datenbeständen und der Ablauf des Prüfens wurden unter besonderer Beachtung dieser Anforderung entworfen.

Der softwaretechnische Einführungsaufwand wird dadurch reduziert, dass die Architektur des Systems die spätere Ergänzung von zusätzlichen Anwendungsfunktionen begünstigt. Dadurch kann das System schrittweise eingeführt werden. Zudem können die Adapterprogramme für verschiedene Datenbestände eventuell wiederverwendet werden. Um den Einführungsaufwand weiter zu senken wurde eine eigene Implementierung der objektrelationalen Abbildung einem vorhandenen Framework vorgezogen. Damit wird eine hohe Flexibilität erreicht und der Funktionsumfang kann den Anforderungen entsprechend gewählt werden.

6 Ergebnis

Ziel diese Arbeit war es, eine leichtgewichtige Schnittstelle zwischen digitalen Formularen und heterogenen Datenbeständen vorzuschlagen und zu bewerten. Dazu wurde zunächst die Klasse der durch diese Schnittstelle zu unterstützenden Arbeitsabläufe beschrieben. Daraus wurde abgeleitet, welche Ar-

beitsschritte innerhalb eines solchen Ablaufes von einer Digitalisierung profitieren könnten und damit die Anforderungen an ein informationstechnisches System abgeleitet, welches die gesuchte Schnittstelle realisiert. Es wurden technische Lösungen für den Zugriff auf die digitalen Formulare, den Zugriff auf die heterogenen Datenbestände, die Architektur des Gesamtsystems, die Benutzeroberfläche des Systems, das Prüfen von Formulareigenschaften, das Signieren von Formularen und das Hinzufügen weiterer Anwendungsfunktionen vorgeschlagen. Um entscheiden zu können, inwiefern diese Lösungen den gestellten Anforderungen entsprechen konnten, wurde ein Prototyp entwickelt, welcher einen Durchstich durch das Gesamtsystem realisiert. Anhand dieses Prototyps wurde dann die Bewertung bezüglich der Anforderungen vorgenommen. Es zeigte sich, dass das System über eine leistungsfähige Benutzeroberfläche verfügt, es sich weitestgehend parallel zu Papierformularen einsetzen lässt und die Architektur Änderungen am Arbeitsablauf sowie einen geringen softwaretechnischen Einführungsaufwand berücksichtigt.

Einige an das System gestellte funktionalen Anforderungen wurden nicht genauer beschrieben. Dies betrifft verschiedene Kundenfunktionen wie das Einreichen von Formularen, die Verwaltung des Formularbestandes, die Ablagefunktionalität und die Verwaltung von Benutzerrechten und -eigenschaften innerhalb des Systems, also die Verwaltung der Benutzernamen, Passwörtern und Signaturschlüssel. Dies wäre im Rahmen der vorgeschlagenen Architektur noch umzusetzen. Das dann vorliegende Gesamtsystem würde in diesem Sinne eine leichtgewichtige Schnittstelle zwischen digitalen Formularen und heterogenen Datenbeständen bilden und damit letztendlich zur stärkeren Automatisierung von formularbasierten Arbeitsabläufen beitragen um damit höhere Produktivität in entsprechenden Wirtschaftsbetrieben zu erreichen.

Literatur

[1] David Brownell David Megginson. *SAX.* http://sourceforge.net/projects/sax.

[2] Christian Fillibeck. *Digital Structured Forms Supporting Flexible Workflows.* Software Technology Group, Department of Computer Science, University of Kaiserslautern, 2012.

[3] Oracle. *Java Database Connectivity.* http://www.oracle.com/technetwork/java/javase/jdbc/index.html.

[4] L.Adleman R.L. Rivest, A. Shamir. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.*

[5] Ian Sommerville. *Software Engineering.* Addison-Wesley, 2010.

[6] Technische Universität Kaiserslautern. *Fachprüfungsordnung für die Masterstudiengänge „Informatik/Computer Science", „Angewandte Informatik/Applied Computer Science" und „European Master on Software Engineering" an der Technischen Universität Kaiserslautern.* 2011.

[7] The Apache Software Foundation. *Apache Wicket.* http://wicket.apache.org.

[8] The Jython Project. *Jython.* http://www.jython.org.

[9] World Wide Web Consortium. *DOM4.* http://www.w3.org/TR/dom, 2012.

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Ausführungen, die anderen Schriften entnommen wurden, habe ich kenntlich gemacht. Die Arbeit war nicht Bestandteil einer anderen Prüfungsleistung.

Ort, Datum	*Unterschrift*