Evaluation eines Java Web Frameworks zur Ablösung bestehender Java Swing Applikationen

Diplomarbeit

Student	Roman Würsch
Auftraggeber	Bernhard Mäder - Zürcher Kantonalbank
Betreuer	Beat Seeliger
Experte	Marco Schaad

Studiengang Informatik

Hochschule für Technik Zürich

März 2011 bis Juni 2011

Zusammenfassung

Es werden Methoden zur Analyse von Java Swing Applikationen und zur Evaluation von Java Web Frameworks ausgearbeitet werden.

Mit der Methode zur Analyse werden bestehende Java Swing Applikationen der Zürcher Kantonalbank analysiert. Der Fokus liegt darauf in den Applikationen die gemeinsamen Muster, genutzter Swingkomponenten, zu erkennen und zu kategorisieren. Java Web Frameworks, welche sich am Markt etabliert haben, werden durch einen Evaluationsprozess auf deren Einsatz in der Zürcher Kantonalbank geprüft. Zusätzlich wird geprüft, ob mit den jeweiligen Frameworks die genutzten Swingkomponenten äquivalent umgesetzt werden können. Mit einer Analyse der IT Infrastruktur der Zürcher Kantonalbank, wird geprüft, ob eine Integration in die bestehende IT Infrastruktur möglich ist.

Durch die Umsetzung eines Prototypen soll gezeigt werden, dass das evaluierte Java Web Framework den geforderten Funktionalitäten entspricht.

Mit den gewonnenen Erkenntnissen, wird eine Empfehlung, für den Einsatz eines Java Web Frameworks zur Ablösung bestehender Java Swing Applikationen ausgesprochen.

Inhaltsverzeichnis

1. Einführung

Das Internet hat sich in den letzten Jahren zu einem strategisch wichtigen Vertriebskanal entwickelt. Viele Unternehmen haben das erkannt und setzen im Betriebsalltag die Möglichkeiten ein, die das Internet als Vertriebskanal bietet. In der Finanzindustrie wurde dieser Schritt mit dem Online-Banking gegen Ende der Neunzigerjahre gemacht. Dabei wurden einige Dienstleistungen mit einer Web Applikation über das Internet nutzbar gemacht, die sonst beim klassischen Bankschalter bezogen werden konnten. Dass eine Onlinestrategie funktioniert und die Nachfrage dafür besteht, hat sich in den letzten Jahren in der Finanzindustrie gezeigt.

Die Informatik spielt in der Finanzindustrie im Allgemeinen eine wichtige Rolle. Viele Finanzinstitute, hier in der Schweiz, zählen heute zu den grössten Arbeitgebern in der Informatik, siehe [Ges]. Insgesammt gibt es in der Schweiz acht Unternehmen, welche selber nicht aus der Informatik Branche kommen, die mehr als 500 Informatikfachleute beschäftigen. Die Hälfte davon sind Finanzinstitute, namentlich sind das Credit Suisse, UBS, Zürich Versicherung und die Post. In der Zürcher Kantonalbank (ZKB) ist die Informatik ebenfalls stark vertreten.

Durch die zunehmende Komplexität im Finanzgeschäft, reicht manchmal erwerbliche Softwarelösungen nicht vollends aus. Um den Vorsprung zur Konkurenz auszubauen, werden vielmals Computerprogramme in der Finanzindustrie selber entwickelt, welche die Automatisierung ihrer Prozesse ermöglicht oder mit denen Arbeiten in ihrem Kerngeschäft verrichtet werden können.

Aus einer eigens entwickelten Lösung heraus kann sich ein neues Geschäft für das Finanzinstitut eröffnen. Eine solche Software könnte beispielsweise externen Vermögensverwaltern zur Verfügung gestellt werden. Dabei gibt es verschiedene Vertriebskanäle für Softwareprodukte. Das Internet ist einer davon. Er hat sich mit dem Onlinebanking bewährt und könnte sich somit auch für andere bestehende Lösungen eignen.

In der ZKB existieren viele solcher Lösungen bereits. Diese sind aber nicht für eine Online-Strategie entwickelt worden, sondern für den internen Einsatz. Einige dieser Programme wurden als Desktop Applikationen entwickelt, das entspricht den Anforderungen für den

internen Einsatz. Um den Vertrieb zentral verwalten zu können, macht das Umrüsten bestehender Desktop Applikationen auf eine Online-Lösung Sinn.

1.1. Motivation

Die Zürcher Kantonalbank setzt bei der Entwicklung von Inhouse Softwarelösungen auf Java Swing Applikationen und auf Java Web Applikationen. Die Java Web Applikationen basieren auf dem Web Framework Apache Struts 1.3.10 mit einer von der ZKB erstellten Erweiterung namens ZKB Internet Plattform (ZIP). Da eine Lösung als Java Web Applikation zentral verwaltet werden kann, gibt es erhebliche Einsparungen im Bereich Testing, Deployment und Patching. Somit sollen in Zukunft Java Swing Applikationen auf Java Web Applikationen umgerüstet werden.

Das Framework Apache Struts 1.3.10 ist mittlerweile in die Jahre gekommen. Es basiert auf dem Prinzip von "full site reload". Wann immer eine Aktion von einem User in der Applikation ausgeführt wird, sei es das Ansteuern eines Links oder das Absenden eines Formulars, wird die Webseite, welche die Applikation repräsentiert, vollständig neu geladen. Heutzutage gibt es Frameworks, welche einem ermöglichen eine Web Applikation zu entwickeln, die sich bei der Bedienung wie eine klassische Desktop Applikation anfühlt. Dabei werden nur Bereiche neu geladen, wo sich die zu darstellenden Informationen geändert haben. Dies wird durch die Technik von Asynchronous JavaScript and XML (Ajax)[1] realisiert.

Bei Ajax können Daten einer Web Applikation, ohne die komplette Webseite neu zu laden, verändert werden. Dies erlaubt es Web Applikationen, auf Benutzeraktionen schneller zu reagieren, da vermieden wird, dass statische Daten, die sich unter Umständen nicht verändert haben, immer wieder neu übertragen werden. Das beinhaltet nicht nur die sichtbaren Informationen, sondern auch die Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) und Java Script Ressourcen, die für die Darstellung notwendig sind.

Apache Struts 1.3.10 unterstützt Ajax nicht direkt, das kann über Erweiterungen ermöglicht werden. Es gibt Java Web Frameworks, welche diese Technik "out of the box" mitbringen. Eine Evaluation soll zeigen, ob eines dieser Java Web Frameworks besser für eine Anwendung geeignet ist.

1.2. Zielsetzung

Gegenstand der vorliegenden Diplomarbeit ist die Analyse, welche Java Web Frameworks für die Ablösung von bestehenden Java Swing Applikationen in Frage kommen. Es sollen

[1] XML ist nicht zwingend, es kann auch mit JSON oder anderen Technologien gelöst werden.

anhand eines strukturierten Vorgehens eine Menge von bestehenden Java Swing Applikationen der Zürcher Kantonalbank auf deren Funktionalität der Benutzeroberfläche eruiert werden. Aufgrund der Anforderungen der IT-Architektur der Zürcher Kantonalbank sollen Java Web Frameworks auf die Validität der erhobenen Funktionalität verifiziert werden. Die Java Web Frameworks, welchen diesen Ansprüchen genügen, sollen auf einen möglichen Einsatz in der Informatik Infrastruktur der Züricher Kantonalbank untersucht werden. Durch die Implementierung eines Prototypen soll gezeigt werden, dass die Umsetzung möglich ist. Schlussendlich soll eine Empfehlung für ein Java Web Framework ausgesprochen werden, für den Fall, dass eine bestehende Java Swing Applikation in eine Web Applikation umgebaut werden soll. Ebenso sollen die gewonnen Erkenntnisse, für zukünftige Revisionen im Bereich der Java Web Frameworks der IT-Architektur der Zürcher Kantonalbank, als Grundlage dienen können.

1.3. Struktur

Die Arbeit wurde in sechs Teile gegliedert. Diese Teile bauen aufeinander auf und setzten meistens die Erkenntnisse der vorhergehenden Bestandteile voraus.

Einführung Im Kapitel 1 Einführung wird die Motivation für die Diplomarbeit erläutert, ebenso wird auf die Zielsetzung eingegangen. Die Gliederung des Berichts wird in der Struktur aufgeschlüsselt. Im Ablauf wird gezeigt wie die Diplomarbeit durchgeführt wurde. Zudem befindet sich hier auch eine Danksagung.

Basisinformationen In den Kapiteln 2 Java Swing Applikationen und 3 Rich Internet Applikationen werden Hintergrundinformationen zu diesen Themen vermittelt. Dabei werden hauptsächlich Begriffe aus diesen Bereichen erklärt und mit Hilfe von Abbildungen dargestellt. Die Informationen welche erläutert werden, dienen als Grundlage für die weiteren Kapitel und auch für das Verständnis des Themas.

Methoden In den Kapiteln 4 Methoden zur Analyse von Java Swing Applikationen und 5 Methoden zur Entscheidungsfindung bei einer Evaluation werden Methoden erarbeitet. Diese Methoden sollen für die Durchführung, der in der Aufgabenstellung gestellten Aufgaben, verwendet werden.

Aufgaben In den Kapiteln 6 bis 13 werden die erwarteten Ergebnisse entsprechend der Aufgabenstellung erarbeitet. Das beinhaltet die Analyse der Infrastruktur der Zürcher Kantonalbank und die Analyse der Java Swing Applikationen. Die Evaluation der Java Web Frameworks mit sinnvollen Rahmenbedingungen in der Form von Soll- und KO-Kriterien. Mit den erhaltenen Resultaten, wird die Integration in die ZKB Infrastruktur und die Implementierung der gefundenen Swingkomponenten geprüft. Als

Abschluss werden im Kapitel Proof of Concept - Umsetzung durch einen Prototypen die erarbeiteten Ergebnisse bestätigt.

Erkenntnisse Die Kapitel 14 und 15 widerspiegeln die Erkenntnisse, welche im Laufe der Diplomarbeit gewonnen wurden. Dabei wird eine Empfehlung für ein Java Web Framework für die Züricher Kantonalbank ausgesprochen. Mit einer Reflexion sollen dargelegte theoretische Grundlagen, die gewählten Methoden sowie die Auswertung der Ergebnisse untersucht werden. Zudem wird ein Ausblick gewährt.

Anhang Im Anhang sind Informationen zu finden, die entweder mit den Rahmenbedingungen der Diplomarbeit zu tun haben, wie das Personalienblatt, die Aufgabenstellung oder die Protokolle zu den offiziellen Terminen. Oder es handelt sich um Zusatzinformationen, welche vom Umfang her zu kostspielig gewesen sind, dass sie in der Arbeit Platz gefunden hätten, darunter fallen die Grundsätze der ZKB IT-Architektur und die 18 Anforderungen an Web Frameworks nach AgileLearn. Ein spezieller Anhang beinhaltet die Projektadministration, in der die ganze Planung und Nachvollziehbarkeit der getätigten Aufgaben beschrieben wird. Zudem befinden sich hier auch sämtliche Verzeichnisse, insbesondere das Literaturverzeichnis.

1.4. Ablauf

Der Ablauf der Kernaufgaben der Diplomarbeit wurde so strukturiert, dass die aufeinander folgenden Teile auf die jeweils vorher erarbeiteten Resultate aufbauen. Bilder sagen manchmal mehr als Worte, deshalb wurde der Ablauf in Form eines Flussdiagramms in der Abbildung 1.1 dargestellt.

Zusätzlich zu den dargestellten Aufgaben der Diplomarbeit kommen noch die Planung, Meetings, das Verfassen der Dokumentation und der Präsentationen, und vieles mehr. Diese Aspekte der Diplomarbeit sind in der Abbildung 1.1 nicht ersichtlich.

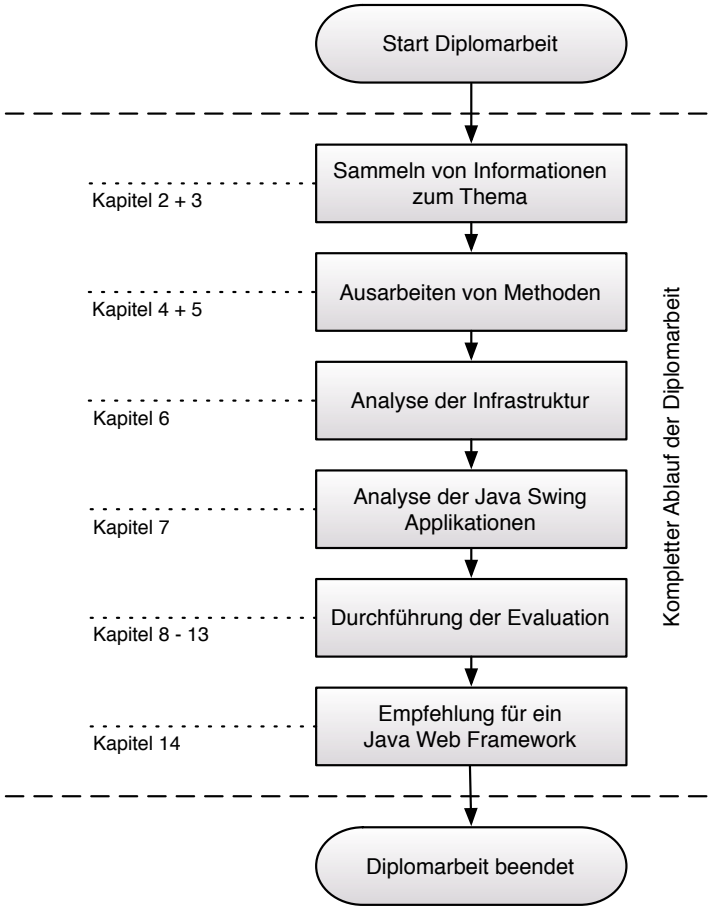

Abbildung 1.1.: Strukturierte Durchführung der Diplomarbeit

2. Java Swing Applikationen

Swing kommt aus dem Hause Oracle, ehemals Sun Microsystems, und ist ein Bestandteil der Java Foundation Classes (JFC). Seit der Java Version 1.2 ist Swing Bestandteil der Java Runtime Environment (JRE). Swing wurde in den letzten Jahren immer weiter ausgebaut und ist somit der Standard für die Entwicklung von Desktop- und Applet-Applikationen in Java.

2.1. Grundlagen

Unter Swing versteht man eine Reihe von leichtgewichtigen Komponenten zur Programmierung von grafischen Oberflächen. Mit leichtgewichtigen Komponenten meint man, dass alle Komponenten zu 100% in Java geschrieben sind. Sie sind somit plattformübergreifend einsetzbar und sehen überall gleich aus. Die Swing Komponenten sind im JRE unter dem Packet *javax.swing* eingeordnet, zusätzlich gibt es das Packet *javax.accessibility*, welches zur Abstraktion des User Interfaces dient.

2.1.1. Komponenten

Gemäss [Sch10], kennt Swing drei Hierarchien von Komponenten: Top-Level, Intermediate und Atomic Components.

Die Top-Level Components sind $javax.swing.JFrame$, zur Darstellung einer vollwertigen Fenster-Applikation, $javax.swing.JDialog$, für Dialogfenster und $javax.swing.JApplet$, zur Entwicklung von Java-Applets mit Swing.

Intermediate Components bieten vielfältige Möglichkeiten an, um andere Intermediate Components zu unterteilen oder zu gruppieren. Zusätzlichen können sie auch eine beliebige Anzahl von Atomic Components enthalten. Gängige Vertreter sind $javax.swing.JPanel$, eine Komponente zur Gruppierung anderer Komponenten, oder $javax.swing.JSplitPane$, eine Komponente zur Unterteilung einer Bereiches in zwei Teile.

Unter Atomic Components versteht man einzelne Bausteine wie ein $javax.swing.JButton$, ein einfacher Knopf, oder ein $javax.swing.JTextField$, ein einfaches Textfeld.

2.1.2. Layouts

Viele Komponenten in Swing haben ein Layout, vor allem die Intermediate Components. Das Layout regelt die Anordnung der Komponenten und das Verhalten, falls sich die Grösse einer Komponente ändert. Swing definiert ein paar eigene Layouts, man kann aber auch die bestehenden Layouts aus dem Paket *java.awt*, wie zum Beispiel *java.awt.GridBagLayout* verwenden. Swing bietet auch die Möglichkeit eigene Layouts zu definieren, was zum Beispiel JGoodies mit dem Freeware Projekt *JGoodiesForms* gemacht hat, siehe [Len10].

2.1.3. Eventbasierte Kommunikation der Komponenten

Das Programmiermodell mit Java Swing ist Eventbasiert. Dabei können Events definiert werden, zum Beispiel ein Mausklick auf einen Knopf, bei welchem eine Aktion ausgeführt werden soll. Die Aktion könnte das Speichern eines Dokuments sein. Die eventbasierte Kommunikation zwischen den Swing Komponenten ist nach dem Observer Design Pattern implementiert, siehe [Wik11k].

2.1.4. Hilfsmittel

Die Klasse *javax.swing.SwingUtilities* bietet eine vielzahl von Hilfsfunktionen, welche bei der Entwicklung von Swing Applikationen verwendet werden können. Zusätzlich bietet Swing eine handvoll Klassen, welche einem das Leben als Programmierer erleichtern. Ein paar nennenswerte sind: *javax.swing.UIManager*, um das aktuelle Look & Feel zu managen, *javax.swing.BorderFactory*, für das zeichnen von Rahmen um eine Komponente, oder auch *java.swing.SwingWorker*, um asynchrone Tasks zu verarbeiten.

2.2. Pluggable Look & Feel

Das Erscheinungsbild und das Verhalten der Swing Komponenten, auch genannt pluggable Look & Feel, kann für alle Swing Komponenten separat definiert werden. Es lässt dem Programmiere die Möglichkeit offen, alle Komponenten individuell zu gestalten. Durch die Delegation an ein separates Objekt, kann das Look & Feel zur Laufzeit ausgetauscht werden.

2.3. Multithreading

Swing ist zum grössten Teil nicht thread-save. Das heisst, auf ein Java Swing Objekt sollte nur mit einem Thread zugegriffen werden. Für den Zugriff und die Instanziirung von Java Swing Objekten steht der Event Dispatch Thread (EDT) zur Verfügung, welcher zusätzlich auch alle Events, welche von Java Swing Komponenten generiert wurden, abarbeitet.

2.4. Asynchrone Tasks

Damit der EDT bei länger dauernden Tasks nicht blockert wird, stellt Swing das Konzept vom SwingWorker zur Verfügung, siehe [Wik10c]. Darüber kann der EDT einen zusätzlichen Worker Tread starten. Durch das Abgeben von langandauernden Arbeiten an zusätzliche Threads, wird der EDT nicht blockiert und kann sich so um das abarbeiten von Actions kümmern. Somit ist auch gewährleistet, dass die grafische Benutzeroberfläche (GUI) dadurch nicht blockiert wird.

3. Rich Internet Applikationen

Rich Internet Application (RIA) ist kein Standard, sondern ein synonym für Applikationen, welche eine "reichhaltige" Benutzeroberfläche bieten und eine Verbindung mit dem Internet haben. Siehe [Wik11m] und [All02] S. 2.

3.1. Grundlagen

Der Begriff RIA ist mit der Entwicklung des Internets entstanden und wird heute oft verwendet. Für viele Leute ist RIA ein synonym für Webanwendungen, welche mit Ajax realisiert werden. Ajax bietet die Möglichkeit eine Benutzeroberfläche zu entwickeln, so wie man sich das von klassischen Desktopanwendungen gewöhnt ist. Die Grenze zwischen der klassischen Webanwendung und einer Desktopapplikation scheint damit zu verschwinden. Genauer betrachtet steht eine RIA im Technologiespektrum aber zwischen dem Rich Client und dem Thin Client, siehe Abbildung 3.1.

Rich Client steht für kompilierte Desktopapplikationen und Thin Client für Webapplikationen welche im Webbrowser laufen, siehe [LW] S. 4f. Da die Grenze zwischen RIA und Thin Client nicht klar definiert ist, werden diese beiden Begriffe zum Teil als Synonym verwendet.

3.2. Klassifikation

Es wird eine Klassifikation aller RIA in die Klassen - Pluginorientiert, Clientorientiert und Browserorientiert - gemacht.

3.2.1. Pluginorientiert

Plugin orientierte RIA sind Applikationen, welche in einem Browser Plugin laufen. Dabei sind Adobe Flash, Java Applet und Microsoft Silverlight die drei grössten Vertreter in dieser Sparte, siehe [Wik11n] und [OWL11].

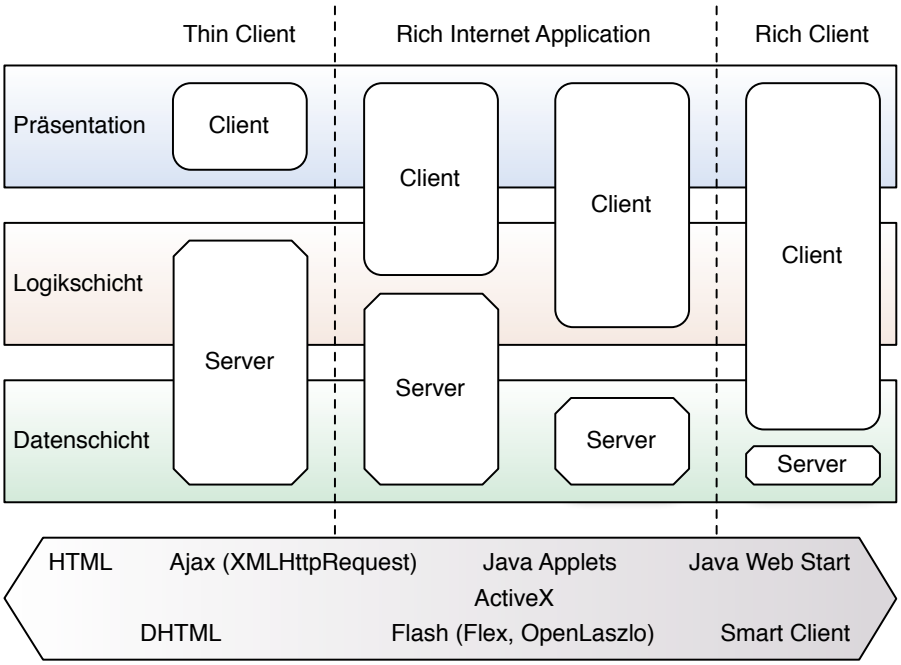

Abbildung 3.1.: Web Anwendungen (nach [Sch07] S. 6f. und [LW] S. 5)

3.2.2. Clientorientiert

Unter Clientorientiert versteht man lediglich, dass alle Desktopanwendungen, welche entweder mit den Internet kommunizieren oder zumindest über dieses ausgeliefert werden, zu dieser Kategorie gehören. Dies trifft nur zu, in der Annahme, dass Desktopanwendungen eine "reichhaltige" Bedienoberfläche anbieten, siehe [Wik11m]. Eine Java Swing Applikation könnte also auch als Client orientierte RIA klassifiziert werden.

3.2.3. Browserorientiert

Browser orientierte RIA kommen aus der Evolution der Internettechnologie heraus. Da sich die Grundkonzepte der Internettechnologie in den letzten Jahren stark verändert haben, müssen wir zwischen klassischen Webanwendungen und Webanwendungen mit Ajax unterscheiden.

Klassische Webanwendung

Um die Jahrtausendwende wurden klassische Webanwendung nach dem Prinzip von "full site reload" aufgebaut. Der Benutzer konnte mit einer Interaktion einen Statuswechsel von einer Seite zur Nächsten auslösen, zum Beispiel durch das anklicken eines Links oder mit dem Versenden eines HTML-Formulars, siehe Abbildung 3.2. Der zeitliche Ablauf sieht mit einem Unified Modeling Language (UML) Sequenzdiagramm wie folgt aus, siehe Abbildung 3.3. Dabei wurde immer der gesamte Seiteninhalt neu geladen, was zu langen Wartezeiten beim Laden der Seite und zu einem ungewohnten Anwendergefühl, im Vergleich zu Desktopanwengungen, geführt hat. Zudem wurden immer alle Daten vom Server an den Browser übermittelt, welche für die Darstellung der Webanwendung von Nöten waren, siehe [DC05] S. 44ff.

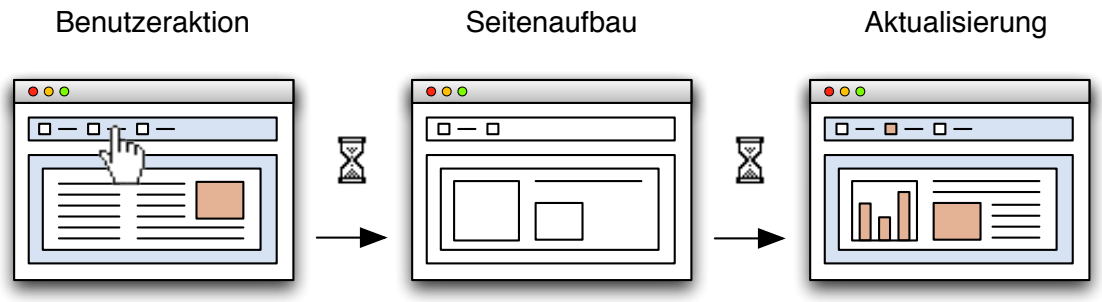

Abbildung 3.2.: Klassische Webanwendung aus der Usersicht (nach [Sch07] S. 10)

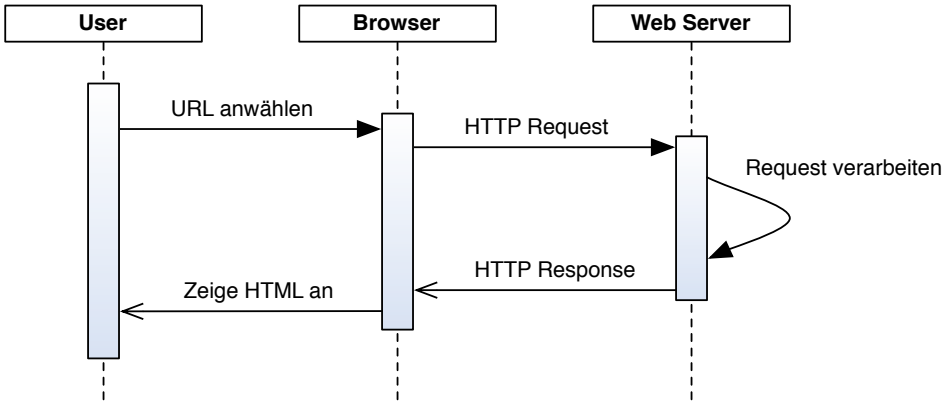

Abbildung 3.3.: HTTP Request als UML Sequenzdiagramm (nach [Mug06] S. 10)

Webanwendung mit Ajax

Mit dem Konzept von Ajax, bei dem der Browser asynchron Daten vom Server nachladen kann, wurde die Möglichkeit geschaffen, Anwendungen zu entwickeln, welche sich in der Bedienung wie Desktopanwendungen anfühlen, siehe Abbildung 3.4. Der zeitliche Ablauf einer Userinteraktion wird mit einem UML Sequenzdiagramm in der Abbildung 3.5 dargestellt. Das Prinzip funktioniert dadurch, dass eine zusätzliche Schicht zwischen dem Browser und Server eingerichtet wird. Diese Schicht, sie wird hier Ajax-Engine genannt, übernimmt die Kontrolle über die Datenkommunikation zum Server. Die Ajax-Engine bietet die Möglichkeit asynchron zur Clientinteraktion Daten vom Server anzufordern und bei Erhalt dynamisch in die bestehende Seite einzuflechten. Das Ergebnis ist, dass der Browser vom Server entkoppelt wird, wobei der Benutzer die Seite weiterhin verwenden kann. Der Server kann nun getätigte Interaktionen im Hintergrund verarbeiten.

Abbildung 3.4.: Webanwendung mit Ajax aus der Usersicht (nach [Sch07] S.12)

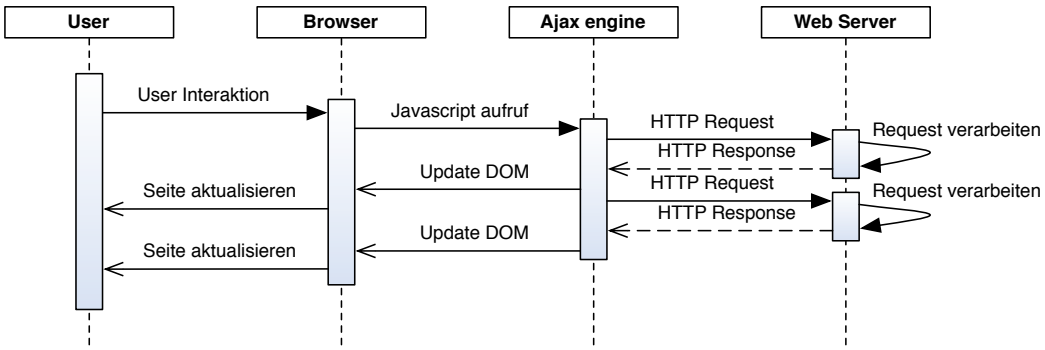

Abbildung 3.5.: Ajax Request als UML Sequenzdiagramm

Security

Nach [Wik11m] gelten Browser orientierte RIA, welche auf Webstandards[1] basieren, als relativ sicher. Mögliche Sicherheitslöcher stellen vor allem der jeweils verwendete Browser, in der die Applikation dargestellt wird, und Attacken nach dem Prinzip von Social Engineering[2].

Suchmaschinenoptimierung

Suchmaschinenoptimierung dient dazu im Ranking einer Suchmaschine besser abzuschliessen. Dies ist vor allem bei Unternehmen von Bedeutung, welche das Internet als Vertriebskanal sehen. Bei statischen Webseiten ist das Indexieren ein Prozess der gut funktioniert. Bei einer RIA, welche über Ajax Daten zur Laufzeit einer Webapplikation asynchron nachlädt, wird das für eine Suchmaschine ein schwierigeres Unterfangen, die Daten Sinnvoll abzugreifen.

3.3. Vorgaben aus der IT-Architektur der Zürcher Kantonalbank

In der Zürcher Kantonalbank gibt es klare Vorschriften, welche Arten von RIA eingesetzt werden dürfen. Diese Vorschriften werden im Handbuch der IT-Architektur zusammengefasst, siehe [uS10] S. 140ff.

3.3.1. Restriktion

Aus den Vorschriften der IT-Architektur, geht hervor, dass keine neuen Applikationen als Plugin orientierte RIAs entwickelt werden, siehe [uS10] S. 143f. Zudem wird festgelegt, dass Client orientierte RIAs für neue Applikationen nicht in Frage kommen, falls die Business-Logik im Client liegt, siehe [uS10] S. 141f.

3.3.2. Mögliche Implementierung

Als mögliche Implementierung werden von der IT-Architektur der ZKB zwei Gruppen genannt. Die Gruppe der Thin Clients und die Gruppe der Ultra Thin Clients. Die Gruppe der Thin Clients ist gemäss dem IT-Architektur Handbuch eine Java Desktop Applikation, welche nur eine Präsentations-Logik enthält. Die Gruppe der Ultra Thin Clients entspricht

[1] Webstandards werden durch das Gremium W3C definiert, siehe http://www.w3.org/
[2] Bei Social Engineering geht es darum, durch zwischenmenschliche Beeinflussung, unberechtigt an Daten oder Dinge zu gelangen, siehe [Wik11p]

den Browser orientierten RIA, siehe Abbildung 3.6.

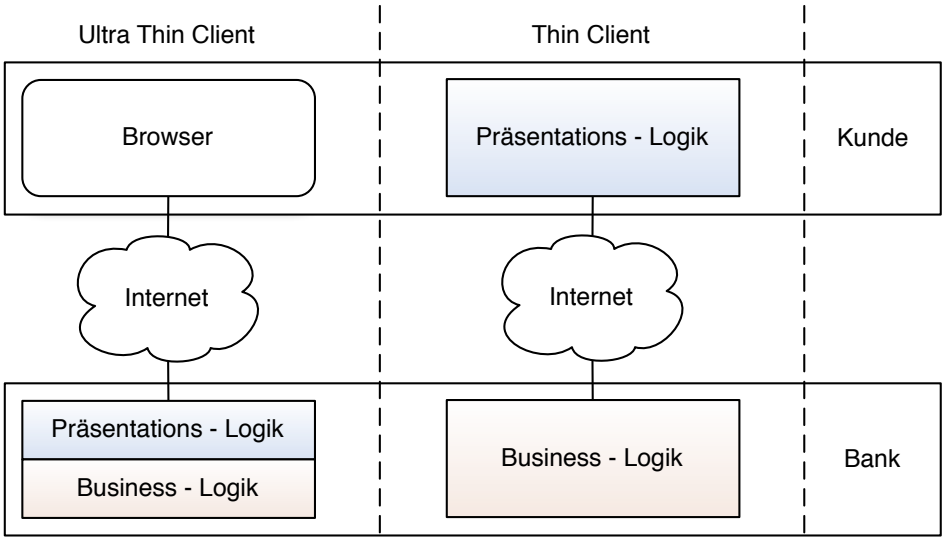

Abbildung 3.6.: Möglicher Aufbau von Web Applikationen (nach [uS10] S. 141)

4. Methoden zur Analyse von Java Swing Applikationen

Dieses Kapitel zeigt Methoden zur Analyse von Java Swing Applikationen. Dabei wird auf statische und dynamische Programmanalyse und das Problem bei der Verwendung von Bibliotheken eingegangen. Es existieren wahrscheinlich noch weitere Ansätze, welche hier nicht behandelt werden.

4.1. Grundlagen

Zur Analyse von bestehender Software gibt es verschiedene Ansätze. Aus dem Bereich der Qualitätssicherung kommt die Technik der statischen oder dynamischen Programmanalyse, siehe [Pep05] S. 5. Die dynamische Programmanalyse wird darin unterteilt in White-Box- und Black-Box-Tests, siehe Abbildung 4.1. In meiner Arbeit werde diese Methoden zur Analyse von Elementen der grafischen Benutzeroberfläche einsetzten.

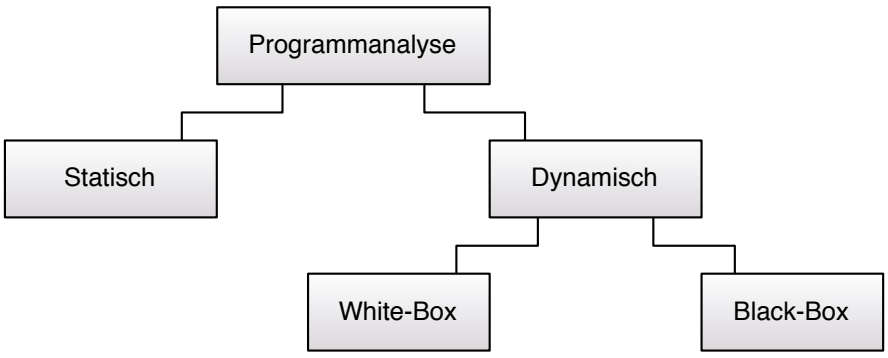

Abbildung 4.1.: Überblick über konventionelle Testmethoden (nach [Pep05] S. 5)

4.1.1. Statische Programmanalyse

Die statische Programmanalyse basiert auf der Analyse des Sourcecodes. Das bedingt, dass der Sourcecode zugänglich ist. Als Softwareentwickler, kennt man das Verfahren eventuell aus der integrierten Entwicklungsumgebung (IDE). Es gibt IDEs, wie zum Beispiel IntelliJ, siehe [s.r], die das Verfahren der statischen Programmanalyse nutzen. Wenn man Code schreibt, dann analysieren die IDEs den geschriebenen Code auf dessen syntaktische, semantische und lexikalische Informationen, siehe [Wal01] S. 216f.

Im Artikel *GUI Analysen und Bibliotheken*, siehe [Wit08] befasst sich der Autor damit, was man bei GUI-Lastiger Software durch eine statische Programmanalyse in Erfahrung bringen kann. Nach Aussage des Autors kann man bei deren Vorgeschlagenen Techniken folgendes erreichen:

> "*Im Rahmen dieser Analyse wird erkannt, welche Teile des Programms zur GUI gehören, welche Widgets das Programm enthält, welche Attribute, mit ihren Werten, diese Widgets besitzen, wie diese Widgets mit Events untereinander verbunden sind und wie sie in den einzelnen Fenstern der GUI strukturiert sind.*" [1]

4.1.2. Dynamische Programmanalyse

Bei der dynamischen Programmanalyse wird das zu analysierende Programm ausgeführt. Die dynamische Programmanalyse wird in zwei Unterarten aufgeteilt, das Black-Box Verfahren und das White-Box Verfahren.

Black-Box Verfahren

Der Analyst hat keinerlei Informationen über das Innenleben des zu analysierenden Programms. Die Analyse basiert auf der intuitiven Bedienung der grafischen Oberfläche, durch den Analysten.

White-Box Verfahren

Der Analyst hat explizite Kenntnisse über das Innenleben des zu analysierenden Programms, zudem steht ihm der Sourcecode zur Verfügung. Mit der Hilfe des Sourcecodes können sinnvolle Schlüsse im Bezug auf die Analyse getroffen werden, da Spezialfälle, welche nicht offensichtlich sind, analysiert werden können. Als Beispiel dafür, möchte ich eine kurze Codesequenz zeigen, siehe Listing 4.1 Zeile 4. Dabei wird auf einen dreifachen Mausklick abgefragt.

[1] [Wit08] Zusammenfassung - Seite 1

Da ein solches Verhalten nicht intuitiv ist, würde das ohne Sourcecode wahrscheinlich nicht gefunden werden.:

```
1  component.addMouseListener(new MouseAdapter() {
2    public void mouseClicked(MouseEvent evt) {
3      if (evt.getClickCount() == 3) {
4        System.out.println("triple-click");
5      }
6    }
7  });
```

Listing 4.1: Spezialfall - dreifacher Mausklick

4.1.3. Probleme bei der Verwendung von Bibliotheken

Gemäss [Wit08] S. 1f. besteht ein Problem bei der Verwendung von Bibliotheken. Die Grundproblematik besteht darin, dass eine Bibliothek eventuell ohne Zugriff auf deren Sourcecode verwendet wird. Damit wird das Verfahren der statischen Programmanalyse und der dynamischen White-Box Programmanalyse unmöglich.

Falls eine statische Programmanalyse angestrebt wird, und der Sourcecode der verwendeten Bibliotheken vorhanden ist, kann sich das negativ auf die Präzision der Analyse auswirken. Denn es ist üblich, dass Bibliotheken meist um ein Vielfaches grösser sind als das zu analysierende Programm. Der Aufwand wird durch die grössere Menge an Sourcecode zudem erhöht.

4.2. Wahl des Verfahrens

Die Wahl des Verfahrens der Analyse soll wie in der Abbildung 4.2 durchgeführt werden. Leider wurde während der Durchführung der Diplomarbeit kein sinnvoll einsetzbares Werkzeug zur statischen Programmanalyse für Java Swing Applikationen gefunden. Somit wurde im Falle von vorhandenem Quellcode nur das dynamische White-Box Verfahren angewendet.

4.3. Durchführung der Analyse

Die Durchführung der Analyse soll wie in der Abbildung 4.3 gezeigt durchgeführt werden.

Als erstes werden die Applikationen, welche analysiert werden sollen ausgewählt. Es handelt sich dabei um Java Swing Applikationen.

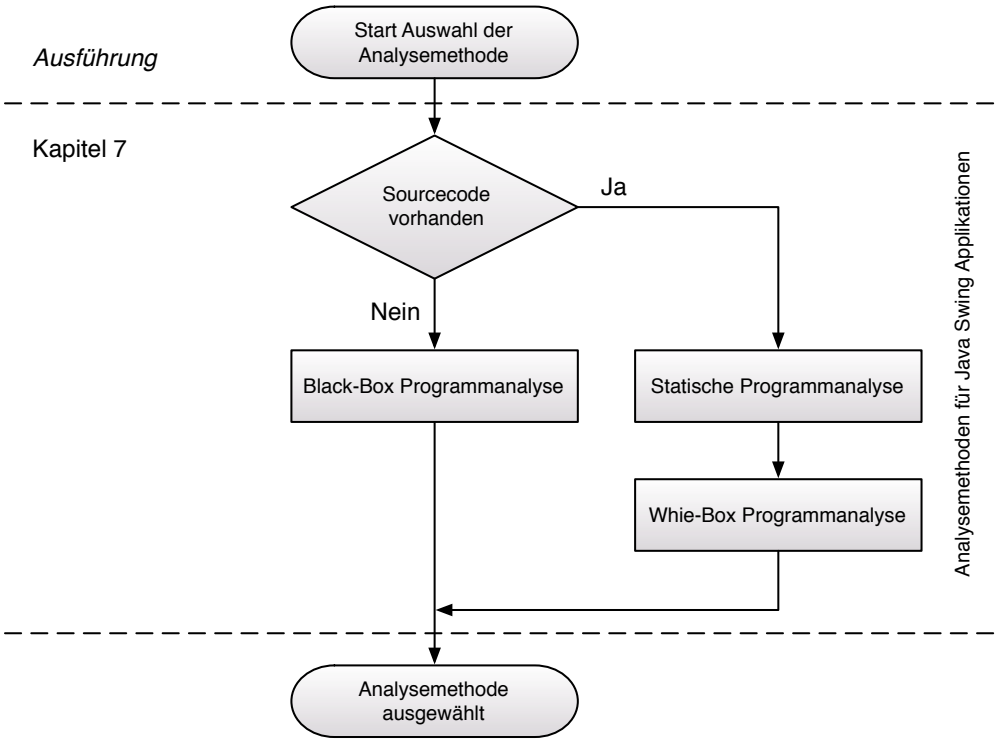

Abbildung 4.2.: Wahl des Verfahrens der Programmanalyse

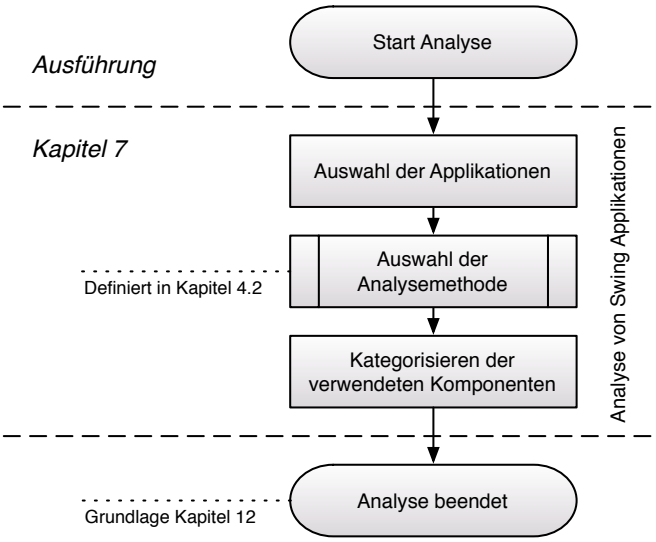

Abbildung 4.3.: Ablauf der Analyse der Java Swing Applikationen

Als nächstes soll die Wahl der Programmanalyse von Java Swing Applikationen, wie in der Abbildung 4.2 erläutert, gemacht werden. Gemäss des gewählten Verfahrens sollen alle Komponenten, die Java Swing zur Verfügung stellt, in der Applikation gesucht werden. Die Komponenten können in der Dokumentation von Oracle[2] zu Java Swing gefunden werden, siehe [Ora11]. Zudem sollen alle Komponenten, welche durch externe Bibliotheken zur Verfügung gestellt werden, gesucht werden. Die Suche findet durch einen visuellen Vergleich der Komponenten statt, wenn möglich kann auch der Sourcecode in die Suche mit einbezogen werden. Um das ganze für den Leser nachvollziehbar zu machen, sollen die gefundenen Komponenten mit Hilfe von Screenshots gezeigt werden, wenn die Applikation für die Öffentlichkeit bestimmt ist. Falls es sich bei der Analyse um eine Applikation für den internen Gebrauch handelt, werden keine Screenshots gezeigt. Es könnten Rückschlüsse auf die Business-Logik gemacht werden, was nicht im Sinne der ZKB ist.

Als Resultat soll eine Auflistung aller gefundenen Komponenten erstellt werden. Die Auflistung wird in folgende Gruppen unterteilt:

- Top-Level-Komponenten
- Intermediate-Komponenten
- Atomic-Komponenten
- Spezielle Komponenten

Als Ergänzung zu den Komponenten, sollen auch erkannte GUI Paradigmen aufgelistet werden. Dabei gibt es Design-Patterns, siehe [Wik11c], die verwendet werden. Zusätzlich können durch das Zusammenspiel einzelner Swing Komponenten auch “neue” Komponenten entstehen. Leider sind die meisten Design-Pattern und “neuen” Komponenten nicht mehr mit Hilfe von Screenshots nachvollziehbar. Das jeweilige Verhalten soll deshalb in ein paar Sätzen beschrieben werden.

Als Resultat soll eine Auflistung erstellt werden. Die Auflistung wird in die folgenden Gruppen unterteilt:

- Design-Patterns
- neue Komponenten

[2] Oracle ist die Firma, welche Java entwickelt.

5. Methoden zur Entscheidungsfindung bei einer Evaluation

Dieses Kapitel zeigt Methoden zur Entscheidungsfindung bei einer Evaluation auf. Dabei werden die Methoden der gewichteten Nutzwertanalyse und des Analytic Hierarchy Process (AHP) gezeigt. Es existieren wahrscheinlich noch weitere Ansätze, welche hier nicht behandelt werden.

5.1. Grundlagen

Es gibt verschiedene Methoden wie man bei der Auswahl einer Softwarelösung vorgehen kann. Um eine möglichst objektive Betrachtung zu gewährleisten, wurde die Methode der gewichteten Nutzwertanalyse (NWA) gewählt, siehe [Wik11j]. Diese Methode stammt aus dem Bereich der quantitativen Analysemethoden der Entscheidungstheorie. Um eine möglichst präzise objektive Gewichtung der einzelnen Faktoren zu erhalten wurde dafür die Methode AHP gewählt, siehe [Wik11a]. Diese Methode stammt aus dem Bereich der präskriptiven Entscheidungstheorie. Der AHP wurde von Thomas L. Saaty in den 70er Jahren des 20. Jahrhunderts entwickelt und im Buch "The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation", siehe [Saa80], veröffentlicht.

5.2. Gewichtete Nutzwertanalyse

Bei der gewichteten NWA wird eine Menge von Alternativen $\{A1, A2, A3, \ldots\}$, auf deren Nutzen, miteinander verglichen.

Als Rahmenbedingung für die Durchführung der Nutzwertanalyse werden m KO-Kriterien $\{KO_1, KO_2, KO_3, \ldots, KO_m\}$ bestimmt. Ein KO-Kriterium wird definiert durch die Erfüllung einer Bedingung. Wenn die Bedingung zutrifft, führt das zum KO[1] einer darauf geprüften Alternative.

[1] Der Begriff KO steht für Knock-Out und stammt aus der Boxwelt. Im falle einer Nutzwertanalyse bedeutet es, dass die Alternative aus der Nutzwertanalyse ausgeschlossen wird.

Für den Vergleich der Alternativen werden n vergleichbare Soll-Kriterien, auch Anforderungen genannt, definiert $\{Soll_1, Soll_2, Soll_3, \ldots, Soll_n\}$. Jedes Soll-Kriterium wird durch einen Gewichtungsfaktor g_i versehen, was die Präferenz des Soll-Kriteriums widerspiegelt.

Die Gewichte g_i werden so gewählt, dass ihre Summe 1.0 (100%) ergibt, siehe Formel 5.1.

Für jede zu evaluierende Alternative soll nun geprüft werden, ob eines der KO-Kriterien erfüllt ist, was zu einem Ausschluss der Alternative führen würde, siehe Abbildung 5.1.

Falls das nicht für alle Alternativen der Fall ist, wird ein Vergleich der im Rennen bleibenden Alternativen über die definierten Soll-Kriterien geführt. Dabei werden für jede Alternative die einzelnen Soll-Kriterien mit einem Erfüllungsgrad e_i bewertet. Die Skala der Erfüllungsgrade ist in der Tabelle 5.1 ersichtlich.

Der Nutzwert einer Alternative ergibt sich durch die Formel 5.2.

Erfüllungsgrad	**Skala**
nicht erfüllt	0
schlecht	1, 2
mittel	3 - 5
gut	6 - 8
sehr gut	9

Tabelle 5.1.: Skala der Erfüllungsgrade

Diejenige Alternative mit dem grössten Nutzwert entspricht am meisten den Anforderungen. Wie gut eine Alternative nun den gestellten Anforderungen entspricht, kann anhand ihres berechneten Nutzwertes in der Tabelle 5.1 abgelesen werden.

$$Gewicht := \sum_{i=1}^{n} g_i = 1.0 \tag{5.1}$$

$$Nutzwert := \sum_{i=1}^{n} e_i \cdot g_i \tag{5.2}$$

5.3. Anschauliches Beispiel einer gewichteten Nutzwertanalyse

Als anschauliches Beispiel sollen zwei Alternativen - Auto $A1$ und Auto $A2$ - miteinander auf die Soll-Kriterien - Leistung, Aussehen und Alltagstauglichkeit - verglichen werden. Es wurden keine KO-Kriterien definiert. In der Tabelle 5.2 ist ersichtlich, dass das Auto $A1$ dem Auto $A2$ gegenüber bevorzugt werden soll, da der Nutzwert von $A1$ (5.0) grösser ist als der Nutzwert von $A2$ (4.7).

Kriterien	**Gewichtung** g	e_{A1}	**Wertigkeit** $A1$	e_{A2}	**Wertigkeit** $A2$
Leistung	0.3	7	2.1	9	2.7
Aussehen	0.2	2	0.4	5	1.0
Alltagstauglichkeit	0.5	5	2.5	2	1.0
Ergebnis	1.0		5.0		4.7

Tabelle 5.2.: Beispiel einer Nutzwertanalyse

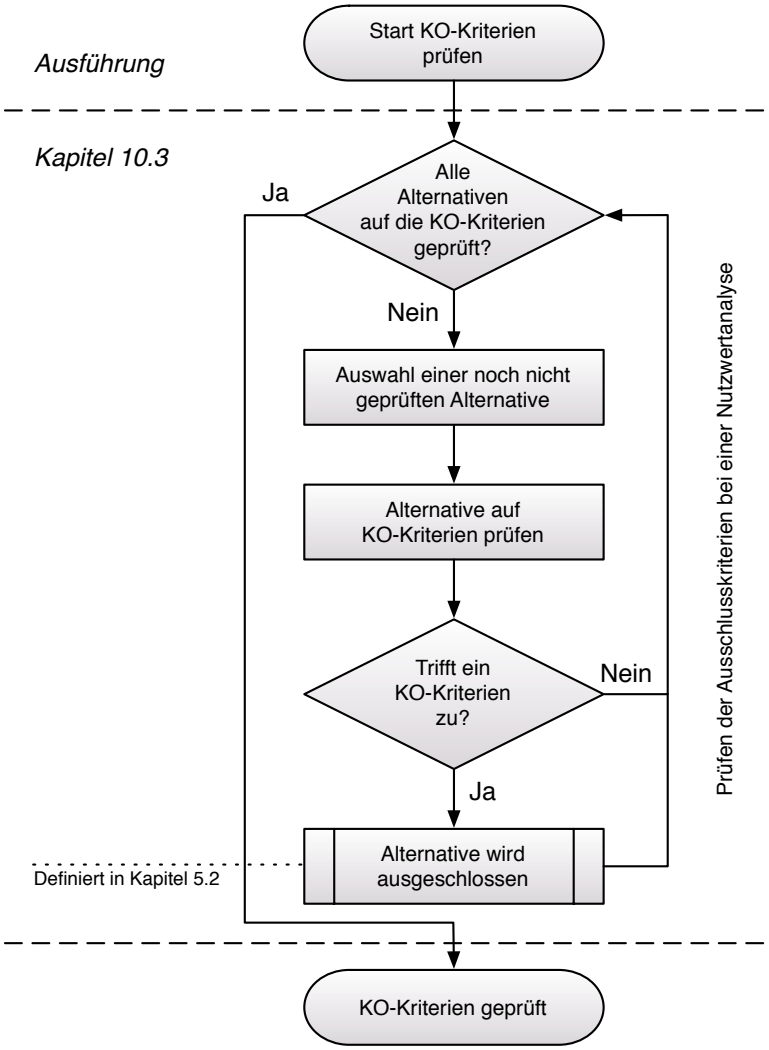

Abbildung 5.1.: Die Alternativen sollen gegen KO-Kriterien geprüft werden.

5.4. Analytic Hierarchy Process

Der Analytic Hierarchy Process stamt aus der Feder eines Mathematikers. Aus diesem Grund ist das Verfahren auch einiges Anspruchsvoller als eine gewichtete Nutzwertanalyse. Ich gehe hier nicht auf die ganzen Details ein, da es den Rahmen der Diplomarbeit übersteigen würde. Trotzdem soll eine grober Überblick über den AHP gegeben werden.

Der AHP besteht aus drei Phasen, siehe [Wik11a]:

1. Sammeln der Daten
2. Daten vergleichen und gewichten
3. Daten verarbeiten

5.4.1. Sammeln der Daten

In der ersten Phase sollen alle Daten, die für eine Entscheidungsfindung erheblich sind, gesammelt werden.

- Zuerst soll eine konkrete Frage formuliert werden, für welche die beste Antwort gesucht wird.
- Danach sollen zu der gestellten Frage alle Kriterien gesucht werden, welche die Lösung beeinflussen können.
- Als letztes sollen alle Alternativen gesucht werden, welche als mögliche Lösung infrage kommen.

5.4.2. Daten vergleichen und gewichten

In der zweiten Phase folgt nun die Gegenüberstellung, Vergleich und Bewertung aller Kriterien beziehungsweise Alternativen in zwei Unterschritten.

- Jedes Kriterium wird jedem anderen gegenübergestellt und darauf verglichen, was eine grössere Bedeutung in der gestellten Frage hat. Die Skala geht von 1 bis 9, siehe Tabelle 5.3.
- Für jedes Kriterium wird jede mögliche Alternativen mit jeder anderen gegenübergestellt und auf ihre Eignung hin untersuchen, welche Alternative am besten zur Erfüllung des jeweiligen Kriteriums passt. Die Skala geht von 1 bis 9, siehe Tabelle 5.3.

Bedeutung	Skala
gleiche Bedeutung	1
leicht grössere Bedeutung	2 - 3
viel grössere Bedeutung	4 - 6
erheblich grössere Bedeutung	7 - 8
absolut dominierend	9

Tabelle 5.3.: Skala der Vergleichsgrade

5.4.3. Daten verarbeiten

Mit einem mathematischen Modell, kann der AHP nun eine präzise Gewichtung aller Kriterien errechnen. Mit der Gewichtung der Kriterien und dem Vergleich der Alternativen, kann der AHP nun berechnen, welches die beste Lösung (Alternative) für die gestellt Frage ist. Aus dem mathematischen Modell heraus, kann nun ein Inkonsistenzfaktor errechnet werden, der eine Aussage macht, wie logisch die Bewertungen zueinander sind.

Diese Berechnungen werden meistens mit der Unterstützung einer Software gemacht. Als Beispiel gibt es JAHP 2.1[2], dies ist ein Java Programm mit dem der gesamte Prozess des AHP abgebildet und berechnet werden kann. Das Programm wird unter den Bedingungen der GNU General Public License vertrieben.

5.5. Kombination beider Methoden

Diese beiden Methoden können auch kombiniert eingesetzt werden, siehe [Buc05], da der AHP relativ komplex in der Umsetzung ist. Als Kombination kann die Nutzwertanalyse als Methode verwendet werden, und der AHP wird für die präzise Berechnung der Gewichtung einzelner Entscheidungskriterien verwendet. Aus der Kombination entsteht somit eine neue Methode, welche durch die Verständlichkeit der NWA und der objektiven Gewichtung des AHP eine plausible Evaluation ermöglicht.

[2]Für Hintergrundinformationen zum JAHP 2.1, siehe [Mor03]

5.6. Verbesserung der Gewichtung

Damit die Gewichtung noch präziser bestimmt werden kann, gibt es verschiedene Ansätze. Zwei Ansätze möchte ich hier erläutern:

- Die Sensitivitätsanalyse, siehe [Wik11o].

  Dabei wird ein einzelner Werte in der Gewichtung gezielt verändert, und das Resultat wird danach geprüft. Bei der Methode des AHP könnte zum Beispiel der Inkonsistenzfaktor geprüft werden, wenn dieser sich Positiv verändert, dann zeigt das eine Verbesserung der Gewichtung. Das Vorgehen wird iterativ, bis zur Unterschreitung einer definierten Schwelle des Inkonsistenzfaktors, fortgeführt.

- Statistische Methoden, wie der Mittelwert oder der Median, siehe [Wik11h] und [Wik11i].

  Dabei wird der Prozess der Gewichtung von mehr als einer Person durchgeführt. Das Ergebnis wird aufgrund der statistischen Methoden normalisiert, was zu einem objektiveren Resultat führen wird.

Diese Ansätze zur Verbesserung der Gewichtung werden im Rahmen der Diplomarbeit nicht durchgeführt, sollen aber bei einer weiteren Anwendung nicht ausgeschlossen werden.

5.7. Integration in die IT-Infrastruktur

Damit ein Java Web Framework in der IT-Infrastruktur betrieben werden kann, muss es deren Ansprüchen genügen. Es soll in einer Analysephase die IT-Infrastruktur untersucht werden. Es soll analysiert werden, welche Version des JRE zur Verfügung gestellt wird, zusätzlich sollen die eingesetzten Serverkomponenten veranschaulicht werden. Ein Augenmerk soll auch auf eingesetzte Protokolle für die Kommunikation, sowie auf die vorgegebene logische und physische Trennung von Schichten in der Serverinfrastruktur gelegt werden.

Wenn die Auswahl der zu evaluierenden Java Web Frameworks getroffen wurde, soll aufgrund der analysierten IT-Infrastruktur geprüft werden, ob ein Betrieb möglich ist. Falls der Betrieb nicht möglich wäre, würde das Java Web Framework aus der Evaluation ausgeschlossen werden. Der genaue Ablauf ist als Flussdiagramm in der Abbildung 5.2 ersichtlich.

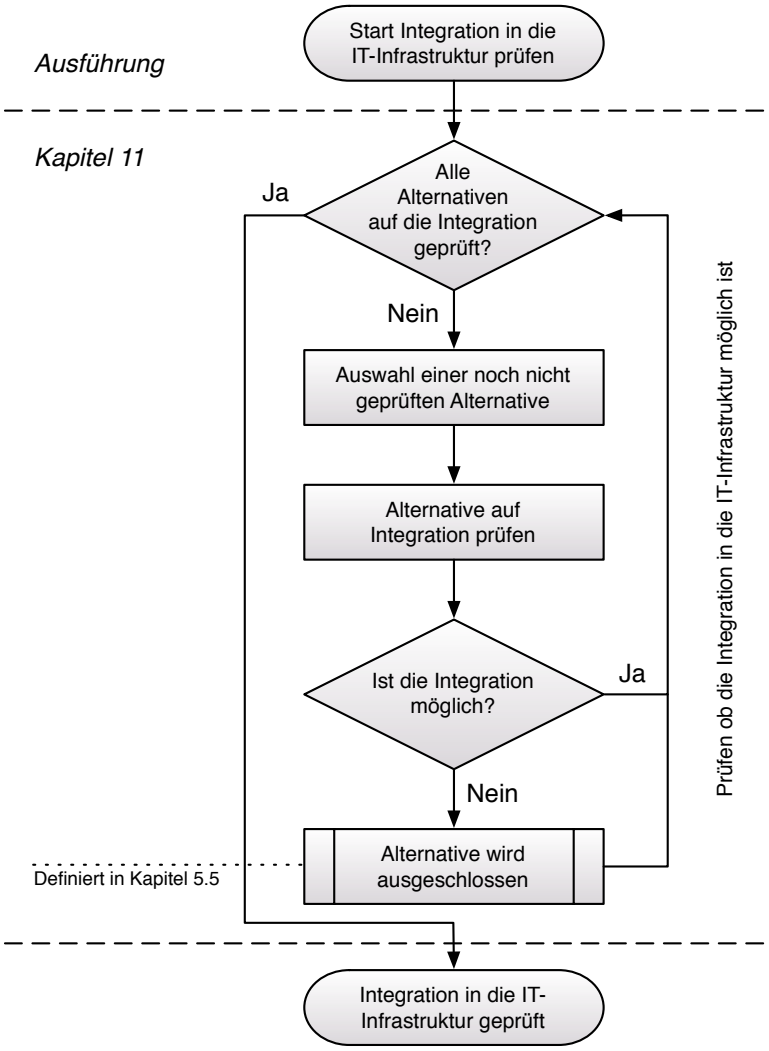

Abbildung 5.2.: Es soll geprüft werden, ob die Integration in die IT-Infrastruktur möglich ist

5.8. Abdeckung der GUI-Komponenten und Paradigmen

Da die Evaluation gemacht wird, um bestehende Java Swing Applikationen abzulösen, soll auch ein Fokus auf das GUI gelegt werden. Im Kapitel 4 (Methoden zur Analyse von Java Swing Applikationen, S. 15ff) wird beschrieben, wie bestehende Java Swing Applikationen analysiert werden können. Das daraus resultierende Ergebnis soll als Grundlage für die Prüfung der Abdeckung der GUI-Komponenten und Paradigmen dienen.

Die Prüfung der Abdeckung zielt darauf ab, herauszufinden, ob eine Implementierung mit

dem zu evaluierenden Java Web Framework, sinnvoll machbar ist. Dabei werden alle erkannten GUI-Komponenten und GUI-Paradigmen der analysierten Java Swing Applikationen genommen und mit der Dokumentation des zu evaluierenden Java Web Frameworks verglichen. Im diesem Vergleich soll festgestellt werden, ob eine Komponente oder ein Paradigma äquivalent umsetzbar ist. Der Vergleich findet für jede Komponente und jedes Paradigma statt und soll schlussendlich in der Form einer prozentualen Abdeckung dokumentiert werden.

Die jeweiligen Prozentangaben berechnen sich wie in den beiden Formeln 5.3 und 5.4 dargestellt.

$$Abdeckung\ GUI\ Komponenten = \frac{gefundene\ GUI\ Komponenten}{mögliche\ GUI\ Komponenten} * 100 \tag{5.3}$$

$$Abdeckung\ GUI\ Paradigmen = \frac{gefundene\ GUI\ Paradigmen}{mögliche\ GUI\ Paradigmen} * 100 \tag{5.4}$$

Als Voraussetzung, das sich das Java Web Framework für eine mögliche Ablösung eignet, soll eine Mindestabdeckung von 80% aller Komponenten und Paradigmen erreicht werden. Diese Schwelle ist verhandelbar und wurde in diesem Fall aufgrund des Paretoprinzips[3] gewählt. Der Gedanke dahinter ist, dass in einem möglichen Projekt zur Ablösung von einer Java Swing Applikation mit dem zu evaluierenden Java Web Framework, in 20% der Zeit die meisten Views äquivalent implementiert werden können. Für die eventuell fehlenden GUI-Komponenten oder GUI-Paradigmen sollte dann genügend Zeit zur Verfügung stehen, um einen Workaround in der Implementierung auszuarbeiten.

Ob eine Schwelle von 80% genügend ist, sei dahingestellt, dies sollte in einer weiteren Anwendung dieser Methode neu geprüft werden. In der Anwendung im Rahmen dieser Diplomarbeit wird ein Java Web Framework, welches eine Abdeckung von weniger als 80% hat, aus der Evaluation ausgeschlossen. Dies wird in der Abbildung 5.3 als Flussdiagramm veranschaulicht.

Natürlich wird auf eine möglichst hohe Gesamtabdeckung abgezielt. Wenn zwei Java Web Frameworks den selben Nutzwert ausweisen, soll das Framework mit der höheren Gesamtabdeckung der GUI-Komponenten und Paradigmen präferiert werden. Die Gesamtabdeckung berechnet sich wird wie in der Formel 5.5 dargestellt.

$$Gesamtabdeckung = \frac{gefundene\ Komponenten\ \&\ Paradigmen}{mögliche\ Komponenten\ \&\ Paradigmen} * 100 \tag{5.5}$$

[3] Das Paretoprinzip, auch 80-zu-20-Regel, besagt, dass 80% der Ergebnisse in 20% der Gesamtzeit eines Projekts erreicht werden. Die verbleibenden 20% der Ergebnisse verursachen die meiste Arbeit, siehe [Wik11l].

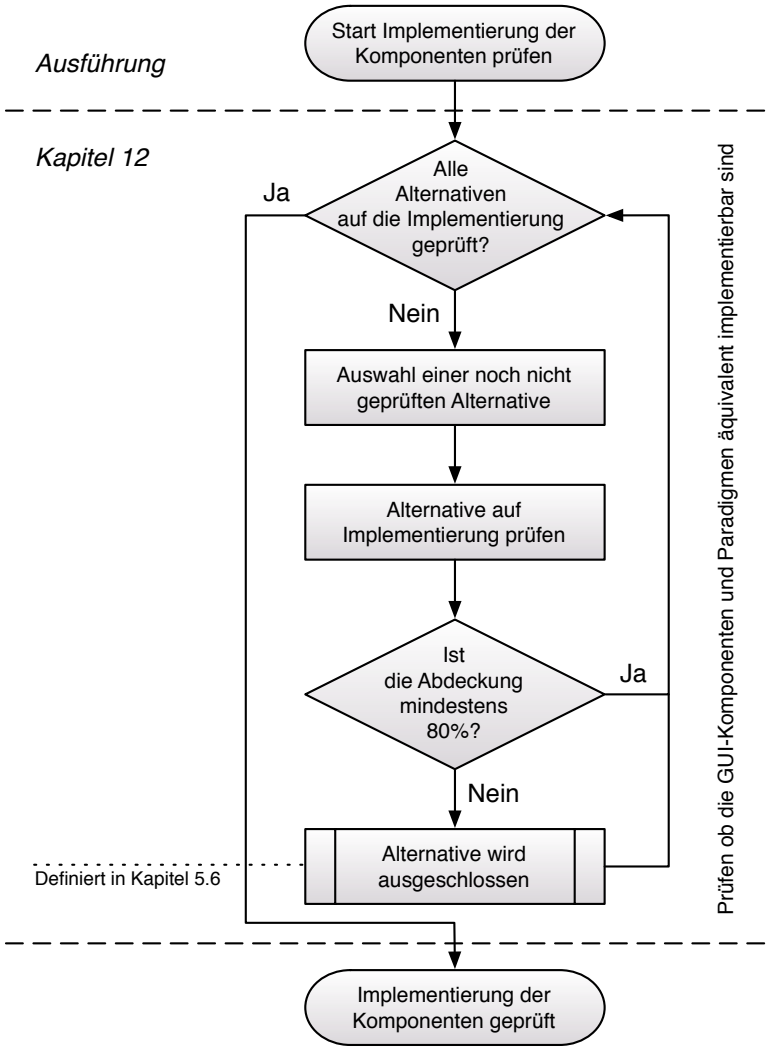

Abbildung 5.3.: Es soll geprüft werden, ob die GUI-Komponenten und Paradigmen implementiert werden können

5.9. Proof of Concept

Für diejenige Alternative, für welche der höchste Nutzwert errechnet wurde, die in die IT-Infrastruktur der ZKB integriert werden kann und die eine Mindestabdeckung von 80% der gefundenen GUI-Komponenten und Paradigmen aufweist, soll in einem Proof of Concept anhand eines Prototypen gezeigt werden, dass sich die Evaluationsmethode bewährt hat.

Die Definition der Anforderungen an den Prototypen werden mit der Technik der User Stories gemacht, siehe [Wel99b].

5.9.1. Was sind User Stories

User Stories beschreiben Anforderungen an eine Software in einer für Jedermann verständlichen Sprache. Es sollen keine technischen Details genannt, sondern viel eher Eigenschaften erläutert werden, die jeder versteht, siehe [Wel99b]. Oft werden User Stories als Aktionen in einem GUI verfasst. Aus den User Stories kann man jeweils Akzeptanztests ableiten, siehe [Wel99a].

Eine User Story sollte nicht mehr als drei Sätze haben. Das kommt daher, dass eine User Story nie zu komplex sein darf. Wenn man mehr als drei Sätze für die Beschreibung einer User Story braucht, sollte diese in mehrere kleinere User Stories aufgeteilt werden, welche genug simpel sind, um wiederum in drei Sätzen beschrieben zu werden.

Jede User Story wird mit einer eindeutigen Nummer versehen: US-{User Story Nummer}

5.9.2. Priorisierung der User Stories

Es soll eine Priorisierung der definierten User Stories gemacht werden. Die Prioritäten werden in *hoch*, *mittel* und *tief* eingeteilt. Es sollen nur die "hoch" priorisierten User Stories umgesetzt werden.

5.9.3. Akzeptanztests zur Prüfung

Aus den "hoch" priorisierten User Stories können nun Akzeptanztests abgeleitet werden. Eine User Story ist dann fertig und erfolgreich, wenn sie technisch umgesetzt wurde und die dazu definierten Akzeptanztests abgenommen wurden. Jeder Akzeptanztest wird mit einer eindeutigen Nummer versehen: T-{User Story Nummer}.{Test Nummer}

Der Proof of Concept ist dann erfolgreich, wenn alle Akzeptanztest erfolgreich abgenommen wurden.

5.10. Ablauf einer Evaluation

In der Abbildung 5.4 ist der komplette Ablauf einer Evaluation mit der kombinierten Methode aus Nutzwertanalyse und AHP ersichtlich.

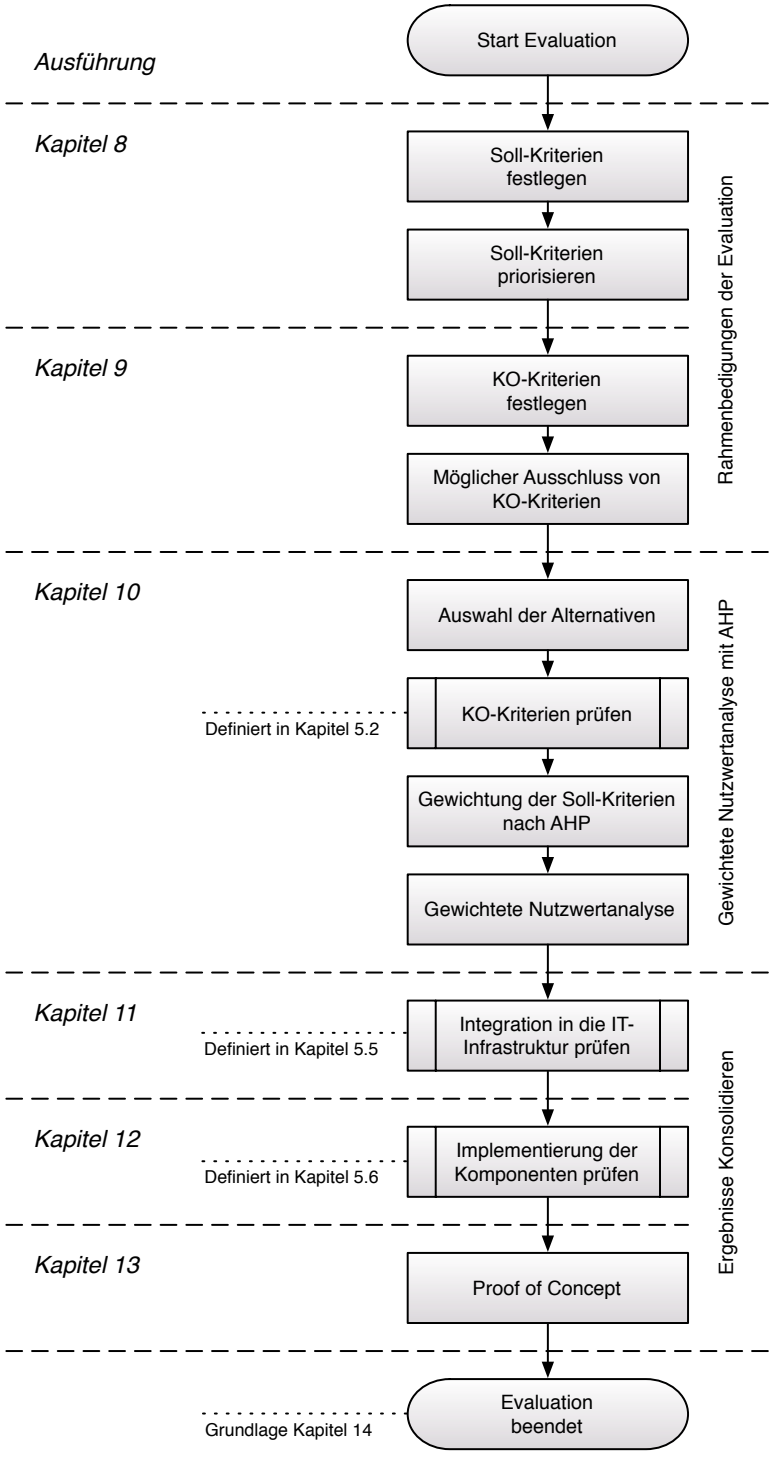

Abbildung 5.4.: Ablauf einer Evaluation mit der kombinierten Methode aus Nutzwertanalyse und AHP

6. Analyse der Infrastruktur der Zürcher Kantonalbank

Im diesem Kapitel, soll auf die IT-Infrastruktur der ZKB im Bereich der Web Applikationen eingegangen werden. Diese Informationen werden später dazu verwendet, um zu Prüfen, ob ein Java Web Framework den gestellten Ansprüchen genügt. Die Informationen, welche hier beschrieben werden, stammen aus dem Handbuch der ZKB IT-Architektur, siehe [uS10]. In der Abbildung 6.1 ist die Konfiguration ersichtlich, wie sie heute in vielen Applikationen der ZKB verwendet wird.

6.1. Java Runtime Environment

In praktisch allen Segmenten der IT-Infrastruktur wird das JRE von Oracle[1] mindestens in der Version 1.5 zur Verfügung gestellt. An einigen Stellen, zum Beispiel bei den Clients ist auch die Version 1.6 verfügbar. Diese Informationen beziehen sich auf den aktuellen Stand und können sich jederzeit ändern. Grundsätzlich wird nicht davon ausgegangen, dass ein Rückschritt in der Version vorgenommen wird, sondern das man gemäss der Evolution auf eine neuere Version wechselt.

6.2. Load-Balancing und Transport

Für das Load-Balancing und den Transport wird der Apache HTTP Server 2 verwendet. Er gilt als anerkannter Standard und ist für alle von der ZKB verwendeten Plattformen verfügbar, siehe [uS10] S. 146.

Der Transport über Hypertext Transfer Protocol (HTTP) und Hypertext Transfer Protocol Secure (HTTPS) gehört zu den Standardfunktionen des Apache HTTP Servers 2. Zudem können statische Daten direkt über den Apache HTTP Server 2 zur Verfügung gestellt werden, ohne das die Präsentations-Logik oder die Business-Logik damit belastet werden.

[1] ehemals Sun Microsystems

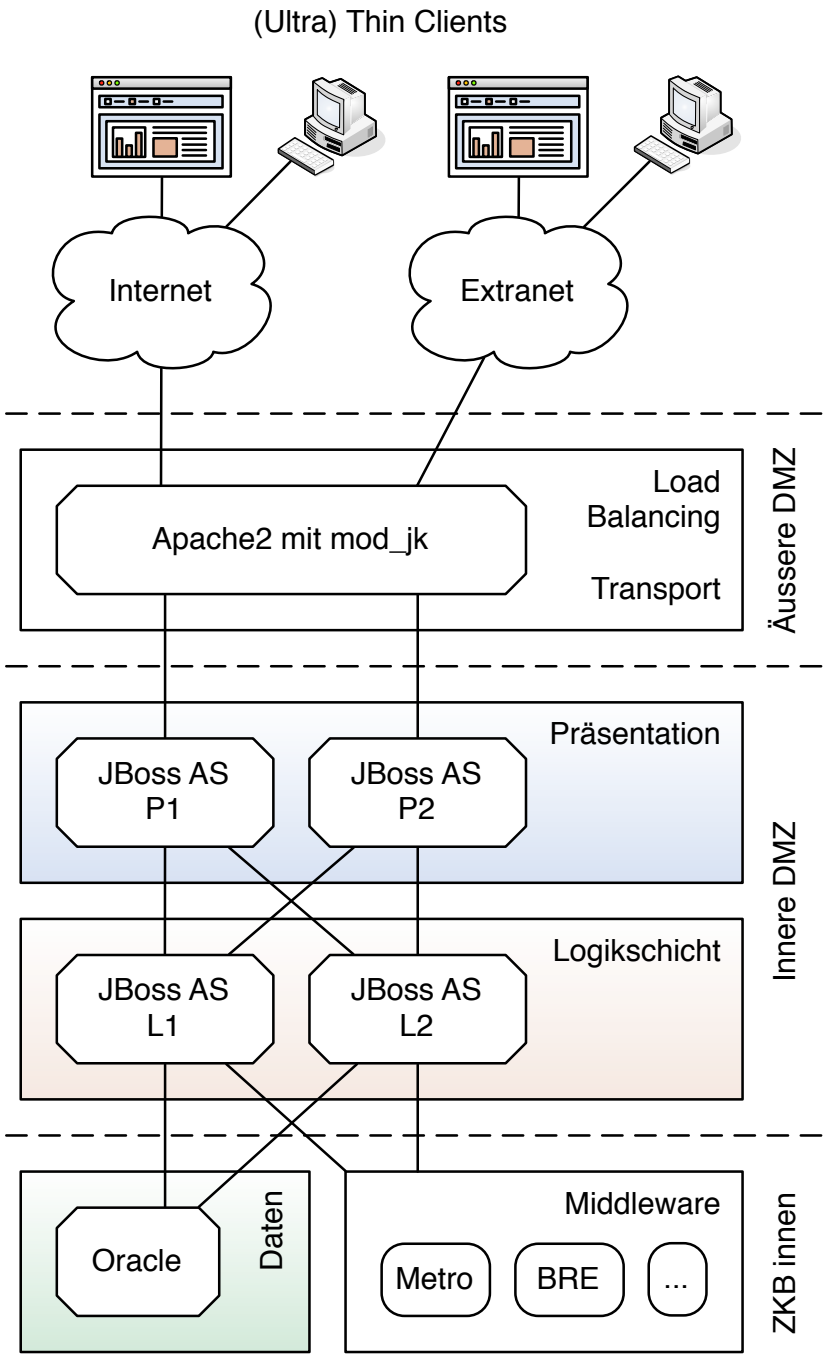

Abbildung 6.1.: Architektur von Java Web Applikationen (nach [uS10] S. 145)

Die Verbindung zur Präsentations-Logik und das Load-Balancing werden mit dem Modul[2] *mod_jk*[3] gemacht.

6.3. Präsentations-Logik

Für die Präsentations-Logik wird der Servlet/JSP Container JBoss Web[4] verwendet, der ein Bestandteil des JBoss AS[5] ist und auf der Open Source Software Apache Tomcat[6] basiert. Der JBoss Web ist auch für die Verwaltung der User Sessions zuständig.

Für die Präsentations-Logik wird eine komplette Instanz eines JBoss AS verwendet. Für die Skalierung der Web Applikation können weitere JBoss AS Instanzen dazu geschalten werden. In der Standardkonfiguration sind es zwei.

Für die Kommunikation zwischen der Präsentations-Logik und der Business-Logik werden Webservices auf der Basis von Simple Object Access Protocol (SOAP), Remote Method Invocation (RMI) oder Common Object Request Broker Architecture (CORBA) verwendet. Es gibt auch eine ZKB spezifische Lösung namens ZIP.net.

Das für die Präsentations-Logik und die Business-Logik eigene JBoss AS Instanzen verwendet werden, entspricht nicht ganz den Vorgaben, wie sie im Handbuch der IT-Architektur beschrieben sind, es wird aber in der Praxis oft so umgesetzt.

6.4. Business-Logik

Für die Business-Logik wird auch eine komplette Instanz eines JBoss AS verwendet. Für die Skalierung dieser Komponente können weitere JBoss AS Instanzen dazu geschalten werden. In der Standardkonfiguration sind es zwei.

Die Datenschicht wird über die Datenbankschnittstelle Java Database Connectivity (JDBC) angebunden. Der JBoss AS bietet für die meisten Relational Database Management System (RDBMS) alles, was dafür benötigt wird. Es gibt auch die Konstellation, bei der ein Object Relational Mapping (ORM), wie zum Beispiel Hibernate, verwendet wird.

Die Kommunikation zwischen der Applikation und der ZKB spezifischen Middleware findet ebenfalls hier in der Business-Logik statt. Für die synchrone Kommunikation steht der

[2] Der Apache HTTP Server 2 hat einen modularen Aufbau, womit er durch Module erweitert werden kann.
[3] Für Informationen zum Apache Tomcat Connector, siehe [Fou10a]
[4] JBoss Web basiert auf Apache Tomcat 6.0 und implementiert die Standards Servlet 2.5 und JavaServer Pages 2.1, siehe [RHM08]
[5] JBoss AS steht für JBoss Application server, siehe [RHM11]
[6] Für mehr Informationen zu Apache Tomcat, siehe [Fou11]

Business Request Exchange (BRE) zur Verfügung. Für die asynchrone Kommunikation gibt es Metro, welches eine Message oriented Middleware (MOM) ist.

6.5. Datenschicht

Die Datenschicht wird meistens mit einem RDBMS abgebildet. Normalerweise wird die Enterpriselösung von Oracle eingesetzt. Je nach Grösse der Applikation kann es sein, dass mehrere verschiedene oder mehrere Instanzen des selben RDBMS zum Einsatz kommen.

7. Analyse der Java Swing Applikationen

Dieses Kapitel behandelt die Analyse von Java Swing Applikationen. Die Analyse betrifft die Swing Komponenten, und Komponenten die in einen Zusammenhang mit Swing Komponenten stehen. Es werden die gezeigten Methoden zur Analyse von Java Swing Applikationen aus dem Kapitel 4 (Methoden zur Analyse von Java Swing Applikationen, S. 15ff) angewendet. Als Ergebnis wird eine Kategorisierung der verwendeten Swing Komponenten aufgelistet. Der Ablauf wird in der Abbildung 4.3 dargestellt.

7.1. Auswahl der Java Swing Applikationen

Es sollen drei Applikationen, welche von der Zürcher Kantonalbank entwickelt wurden, analysiert werden. Die Applikationen sollen mit Java Swing entwickelt worden sein. Die Applikationen werden in der Tabelle 7.1 aufgelistet.

Applikation	Version	Sourcecode vorhanden	GUI enthält sensible Informationen
Strukti Live	1.2	Ja	Nein
Strukti Online	2.10.0	Ja	Ja
Hedo Tool	1.0.710	Ja	Ja

Tabelle 7.1.: Zu analysierende Java Swing Applikationen

7.2. Begründung

Strukti Live Wurde gewählt, weil diese Applikation von der ZKB veröffentlicht wurde. Somit werden bei der Analayse keine sensiblen Informationen preisgegeben.

Strukti Online Wurde gewählt, weil eine mögliche Migration zur Web Applikation schon einmal in der ZKB zur Diskussion stand.

Hedo Tool Wurde gewählt, da die Applikation über ein hochstehendes GUI verfügt.

7.3. Strukti Live

Strukti Live ist ein Lerntool der Zürcher Kantonalbank für strukturierte Produkte. Die Applikation kann auf dem Internetauftritt [1] der ZKB heruntergeladen werden. Es wurde die aktuelle Version 1.2 für die Analyse verwendet.

In der Tabelle 7.2 sind alle verwendeten Bibliotheken, welche eine Interaktion mit dem GUI haben, ersichtlich.

Bibliothek	Funktion	Version	Lizenz	Quellcode vorhanden
core-renderer.jar	XHtml und CSS Renderer für Swing	R8	LGPL	Ja
jfreechart-1.0.10.jar	Charting Library für Java	1.0.10	LGPL	Ja

Tabelle 7.2.: Verwendete Bibliotheken von Strukti Live 1.2

7.3.1. Gefundene Komponenten

Top-Level-Komponenten

- *javax.swing.JDialog*, siehe Abbildung 7.2
- *javax.swing.JFrame*, siehe Abbildung 7.1

Intermediate-Komponenten

- *javax.swing.JPanel*, siehe Abbildung 7.1
- *javax.swing.JRootPane*, siehe Abbildung 7.1
- *javax.swing.JScrollPane*, siehe Abbildung 7.1

[1] Unter http://www.zkb.ch/struktilive sind alle nötigen Informationen zu Strukti Live enthalten.

- *javax.swing.JTabbedPane*, siehe Abbildung 7.3

Atomic-Komponenten

- *javax.swing.JButton*, siehe Abbildung 7.1
- *javax.swing.JCheckBox*, siehe Abbildung 7.1
- *javax.swing.JLabel*, siehe Abbildung 7.1
- *javax.swing.JRadioButton*, siehe Abbildung 7.2
- *javax.swing.JSlider*, siehe Abbildung 7.2
- *javax.swing.JTextField*, siehe Abbildung 7.2
- *javax.swing.JToolTip*, siehe Abbildung 7.3

Spezielle Komponenten

- *javax.swing.JLabel* als externer Link, siehe Abbildung 7.3
- *org.jfree.chart.JFreeChart*, siehe Abbildung 7.2
- *org.xhtmlrenderer.simple.XHTMLPanel*, siehe Abbildung 7.3

7.3.2. Gefundene Design-Patterns

Observer Die Menüführung wurde durch das Observer-Pattern[2] implementiert. Dabei wird in einem Modell durch das Drücken eines *javax.swing.JButton* der jeweilige Status der Applikation angepasst. Die entsprechen *javax.swing.JPanel*, welche auf Änderungen des Modells hören, können sich dann jeweils neu zeichnen.

Ebenfalls mit dem Observer-Pattern realisiert wurde die Aktualisierung von Komponenten welche zusammen hängen. Als Beispiel dient ein Verbund der Komponenten *javax.swing.JTextField*, *javax.swing.JSlider* und für die grafische Darstellung *org.jfree.chart.JFreeChart*. Dabei werden bei einer Änderung des Wert bei dem *JSlider* oder dem *JTextfeld* die Werte aller anderen Komponenten entsprechend aktualisiert.

[2] Siehe [Amr08] S. 2ff

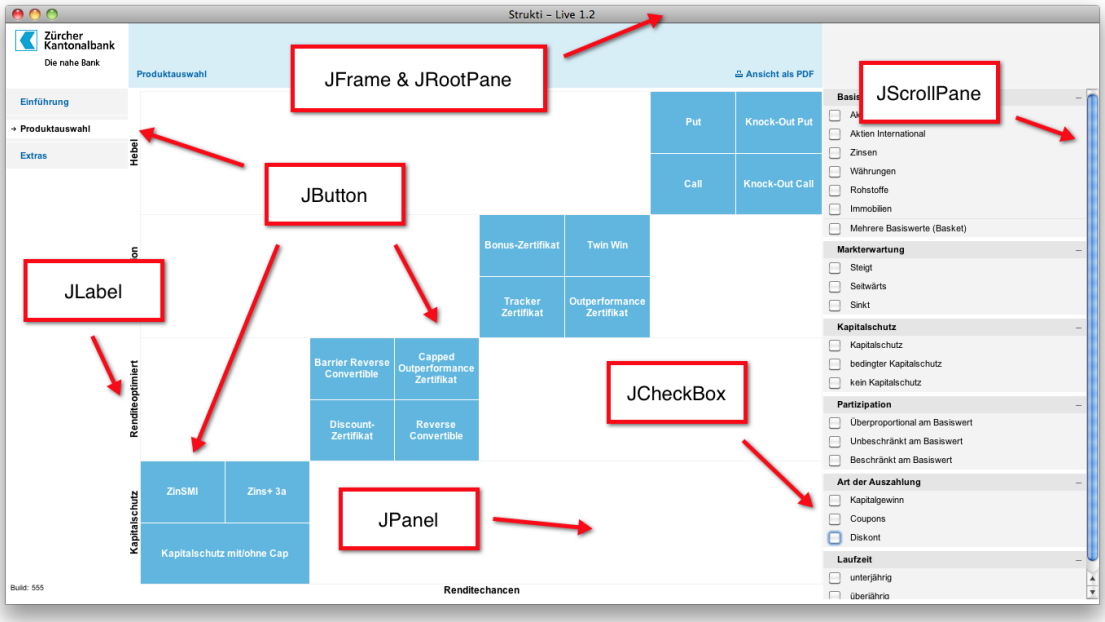

Abbildung 7.1.: Strukti Live 1.2 - Screenshot I

7.3.3. Gefundene "neue" Komponenten

Button-Matrix *JButtons* werden in einer Matrix angeordnet, um sie entsprechend ihrer Funktion zu ordnen. Hinter jedem *JButton* steckt ein Finanzprodukt, das bei einem Klick angezeigt werden kann. Die *JButtons* können entsprechend ihrer Renditechancen (x-Achse) und deren Kapitalschutzes (y-Achse) angeordnet werden.

Akkordeon Ein Verbund von *javax.swing.JPanel* Komponenten, bei welchem ein *JPanel* als Titelleiste funktioniert. In der Titelleiste kann über ein Minus- oder Plussymbol ein weiterer *JPanel* ein- oder ausgeklappt werden. Zudem gibt es in der Titelleiste ein *JButton*, wo ein Dialog mit Hintergrundinformationen geöffnet werden kann.

7.4. Strukti Online

Strukti Online ist ein Emissionstool der ZKB für strukturierte Produkte. Die Applikation wird aktuell nur für den internen Gebrauch entwickelt und basiert auf Java Swing. Es wurde die aktuelle Version 2.10.0 für die Analyse verwendet.

Da es sich um eine interne Applikation handelt, werden keine Screenshots der Applikation gezeigt, da anhand dessen eventuell Rückschlüsse auf die Business-Logik gemacht werden könnte.

Abbildung 7.2.: Strukti Live 1.2 - Screenshot II

In der Tabelle 7.3 sind alle verwendeten Bibliotheken, welche eine Interaktion mit dem GUI haben, ersichtlich, und ob deren Sourcecode zugänglich ist.

7.4.1. Gefundene Komponenten

Top-Level-Komponenten

- $javax.swing.JDialog$
- $javax.swing.JFrame$

Intermediate-Komponenten

- $javax.swing.JLayerdPane$
- $javax.swing.JPanel$
- $javax.swing.JRootPane$
- $javax.swing.JScrollPane$
- $javax.swing.JTabbedPane$

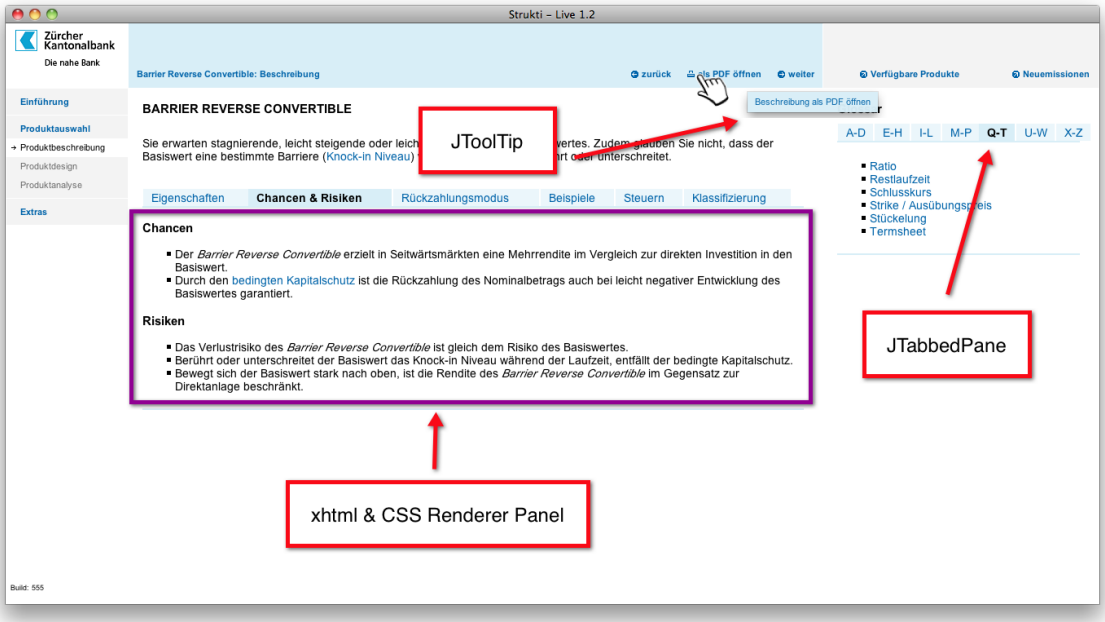

Abbildung 7.3.: Strukti Live 1.2 - Screenshot III

Atomic-Komponenten

- $javax.swing.JButton$
- $javax.swing.JCheckbox$
- $javax.swing.JComboBox$
- $javax.swing.JFileChooser$
- $javax.swing.JLabel$
- $javax.swing.JRadioButton$
- $javax.swing.JPasswordField$
- $javax.swing.JSeparator$
- $javax.swing.JSlider$
- $javax.swing.JSpinner$
- $javax.swing.JTable$
- $javax.swing.JTextArea$
- $javax.swing.JTextField$

Bibliothek	Funktion	Version	Lizenz	Quellcode vorhanden
core-renderer.jar	XHtml und CSS Renderer für Swing	R8	LGPL	Ja
jfreechart-1.0.10.jar	Charting Library für Java	1.0.10	LGPL	Ja
forms.jar	Layout System aus der JGoodies Palette	1.2.1	BSD[1]	Ja
swingx-1.0.jar	Swing Komponenten Erweiterung	1.0	LGPL	Ja

Tabelle 7.3.: Verwendete Bibliotheken von Strukti Online 2.10.0

[1] Es handelt sich dabei um die "BSD open source license"

- $javax.swing.JTextPane$
- $javax.swing.JToolTip$

Spezielle Komponenten

- $javax.swing.JLabel$ als externer Link
- $org.jfree.chart.JFreeChart$
- $org.xhtmlrenderer.simple.XHTMLPanel$
- $org.jdesktop.swingx.JXBusyLabel$
- $org.jdesktop.swingx.JXDatePicker$

7.4.2. Gefundene Design-Patterns

MVC Das Zusammenspiel von Model, View und Kontroller wurde strikte nach dem MVC Design-Pattern[3] implementiert.

Observer Einfache Aktualisierungen zwischen Komponenten wurden strikte nach dem Observer Design-Pattern[4] implementiert.

[3] Siehe [Amr08] S. 15ff.
[4] Siehe [Amr08] S. 2ff.

Tabellen-Filter Durch die Verwendung von *JRadioButton*, *JComboBox*, *JXDatePicker* und *JSpinner* wurden, bei den meisten Ansicht von Tabellen, Filter gebaut, damit die Daten auf das Wesentliche reduziert werden können. Datei wurde für jede relevante Spalten der Tabelle, entsprechend deren Datentyp, eine dieser Komponenten gewählt.

7.4.3. Gefundene "neue" Komponenten

Button-Matrix *JButtons* werden in einer Matrix angeordnet, um sie entsprechend ihrer Funktion zu ordnen. Hinter jedem *JButton* steckt ein spezifisches vorbewertetes Finanzprodukt, das bei einem Mouseklick angezeigt werden kann. Die *JButtons* können entsprechend ihrer Laufzeit (x-Achse) und der Underlyings[5] des hinterlegten Finanzprodukts (y-Achse) angeordnet werden. Die Matrix, kann wie eine Tabelle nach Laufzeiten sortiert werden.

Akkordeon Ein Verbund von *javax.swing.JPanel* Komponenten, bei welchem ein *JPanel* als Titelleiste funktioniert. In der Titelleiste kann über ein Minus- oder Plussymbol ein weiterer *JPanel* ein- oder ausgeklappt werden.

Zeit-Panel Durch die Kombination einer *org.jdesktop.swingx.JXDatePicker* und einer *javax.swing.JSpinner* Komponente kann ein Datum mit Uhrzeit ausgewählt werden.

7.5. Hedo Tool

Hedo Tool ist eine Applikation zur Gewichtung verschiedener Rating-Modelle für Wohneigentum. Die Applikation wird aktuell nur zum internen Gebrauch entwickelt und basiert auf Java Swing. Es wird die aktuelle Version 1.0.710 für die Analyse verwendet.

Da es sich um eine interne Applikation handelt, werden keine Screenshots der Applikation gezeigt, da anhand dessen eventuell Rückschlüsse auf die Business-Logik gemacht werden könnte.

In der Tabelle 7.4 sind alle verwendeten Bibliotheken, welche eine Interaktion mit dem GUI haben, ersichtlich, und ob deren Sourcecode zugänglich ist.

[5] Auch Basiswerte genannt, siehe [Wik10a]

Bibliothek	Funktion	Version	Lizenz	Quellcode vorhanden
binding-2.0.6.jar	Swing Data Binding Framework aus der JGoodies Palette	2.0.6	BSD[1]	Ja
forms.jar	Layout System aus der JGoodies Palette	1.2.1	BSD[1]	Ja
swingx-1.0.jar	Swing Komponenten Erweiterung	1.0	LGPL	Ja
validation-2.0.1.jar	Validiert und präsentiert Validationsresultate	2.0.1	BSD[1]	Ja

Tabelle 7.4.: Verwendete Bibliotheken von Hedo Tool 1.0.710

[1] Es handelt sich dabei um die "BSD open source license"

7.5.1. Gefundene Komponenten

Top-Level-Komponenten

- $javax.swing.JDialog$
- $javax.swing.JFrame$

Intermediate-Komponenten

- $javax.swing.JLayerdPane$
- $javax.swing.JPanel$
- $javax.swing.JRootPane$
- $javax.swing.JScrollPane$

Atomic-Komponenten

- $javax.swing.JButton$
- $javax.swing.JCheckbox$
- $javax.swing.JComboBox$
- $javax.swing.JFileChooser$

- $javax.swing.JFormattedTextField$
- $javax.swing.JLabel$
- $javax.swing.JMenuBar$
- $javax.swing.JMenu$
- $javax.swing.JMenuItem$
- $javax.swing.JProgressBar$
- $javax.swing.JSeparator$
- $javax.swing.JSpinner$
- $javax.swing.JTable$
- $javax.swing.JTextField$
- $javax.swing.JTextPane$
- $javax.swing.JToolTip$

Spezielle Komponenten

- $javax.swing.JLabel$ als externer Link
- $org.jdesktop.swingx.JXBusyLabel$

7.5.2. Gefundene Design-Patterns

MVC Das Zusammenspiel von Model, View und Kontroller wurde strikte nach dem MVC Design-Pattern[6] implementiert.

Observer Einfache Aktualisierungen zwischen Komponenten wurden strikte nach dem Observer Design-Pattern[7] implementiert.

Worker Asynchrone Jobs, die viel Rechenzeit in Anspruch nehmen, wurden mit $javax.swing.SwingWorker$ gelöst.

7.5.3. Gefundene "neue" Komponenten

Es wurden keine "neuen" Komponenten identifiziert.

[6] Siehe [Amr08] S. 15ff.
[7] Siehe [Amr08] S. 2ff.

7.6. Liste der Komponenten

Die Auflistung aller gefunden Komponenten soll die vereinigte Menge aller gefundenen Komponenten widerspiegeln. Die Auflistung soll als Grundlage für die Analyse, ob eine Implementierung der erkannten Swingkomponenten möglich ist, dienen. Folgende 32 Komponenten wurden in den Applikationen gefunden:

7.6.1. Top-Level-Komponenten

- $javax.swing.JDialog$
- $javax.swing.JFrame$

7.6.2. Intermediate-Komponenten

- $javax.swing.JLayerdPane$
- $javax.swing.JPanel$
- $javax.swing.JRootPane$
- $javax.swing.JScrollPane$
- $javax.swing.JTabbedPane$

7.6.3. Atomic-Komponenten

- $javax.swing.JButton$
- $javax.swing.JCheckbox$
- $javax.swing.JComboBox$
- $javax.swing.JFileChooser$
- $javax.swing.JFormattedTextField$
- $javax.swing.JLabel$
- $javax.swing.JMenuBar$
- $javax.swing.JMenu$
- $javax.swing.JMenuItem$
- $javax.swing.JPasswordField$

- $javax.swing.JProgressBar$
- $javax.swing.JRadioButton$
- $javax.swing.JSeparator$
- $javax.swing.JSlider$
- $javax.swing.JSpinner$
- $javax.swing.JTable$
- $javax.swing.JTextArea$
- $javax.swing.JTextField$
- $javax.swing.JTextPane$
- $javax.swing.JToolTip$

7.6.4. Spezielle Komponenten

- $javax.swing.JLabel$ als externer Link
- $org.jfree.chart.JFreeChart$
- $org.xhtmlrenderer.simple.XHTMLPanel$
- $org.jdesktop.swingx.JXBusyLabel$
- $org.jdesktop.swingx.JXDatePicker$

7.7. Liste der GUI Paradigmen

Die Auflistung aller gefundenen GUI Paradigmen soll die vereinigte Menge aller gefundenen GUI Paradigmen widerspiegeln. Die Auflistung soll als Grundlage für die Analyse, ob eine Implementierung der erkannten GUI Paradigmen möglich ist, dienen. Folgende sieben GUI Paradigmen wurden in den Applikationen gefunden:

7.7.1. Design-Patterns

- MVC
- Observer
- Tabellen-Filter
- Worker

7.7.2. “Neue” Komponenten

- Button-Matrix
- Akkordeon
- Zeit-Panel

8. Soll-Kriterien für die Evaluation

Die Eignung, der zu evaluierenden Java Web Frameworks, wird durch eine Menge von Soll-Kriterien bestimmt. Die Soll-Kriterien bestimmen die Anforderungen, welche erfüllt werden sollen. Die Anforderungen stammen von einer Projektgruppe der Hochschule für Technik und Wirtschaft Berlin. Die Soll-Kriterien werden für den Einsatz in der ZKB priorisiert.

Soll-Kriterien sind Vorgaben, die möglichst weitgehend Erfüllt werden sollen. Wenn ein solches Kriterium nicht erfüllt werden kann, schliesst das die Alternative nicht aus. Jedem Soll-Kriterium wird für die Identifikation eine eindeutige ID vergeben. Die ID setzt sich folgendermassen zusammen: {Soll}-{Laufnummer}.

8.1. Anforderungen an Web Frameworks nach AgileLearn

Eine Projektgruppe namens AgileLearn von der Hochschule für Technik und Wirtschaft Berlin befasst sich mit dem Thema Web Frameworks. Die Projektgruppe hat sich die Frage gestellt:

> "*Welche Anforderungen müssen bei der Wahl eines Webframeworks berücksichtigt werden?*" [1]

Aus dieser Fragestellung heraus, wurden 18 Anforderungen an Web Frameworks ausgearbeitet, welche öffentlich in einem Google-Doc[2] ersichtlich sind.

Folgende Anforderungen stammen aus dem Dokument "*18 Anforderungen an Webframeworks - OpenDoc*", siehe [ap11], und werden im Anhang E 18 Anforderungen an Web Frameworks nach AgileLearn (auf Seite 110ff) zusammengefasst und mit einer ID versehen.

[1] Zitat von Raoul Jaeckel, siehe [ap11]
[2] Ein Service von Google um Dokumente öffentlich zu bearbeiten.

8.2. Wichtige Aspekte im Software-Lebenszyklus für die Zürcher Kantonalbank

Die wichtigen Aspekte wurden in Absprache mit dem Auftraggeber ausgearbeitet. Es handelt sich dabei nur um sehr grundlegende Bedürfnisse der ZKB:

Sicherheit Nur eine Bank, die als sicher gilt, hat das Vertrauen der Kunden und somit Erfolg in ihrem Geschäft.

Kosten Die laufenden Kosten sollen so tief wie möglich gehalten werden. Gemäss Abbildung 8.1 liegen die Hauptkosten im Software-Lebenszyklus bei der Maintenance-Phase. Somit können die Kosten für requirements Engineering, Design, Programmierung und Integration vernachlässigt werden.

Ressourcen Damit Software entwickelt und gewartet werden kann, wird die Verfügbarkeit von qualifizierten Ressourcen benötigt.

Image Um sich von der Konkurrenz abzuheben wird ein Image aufgebaut das sich zeitgemäss, flexibel und innovativ präsentieren soll, siehe [SV99] S. 34ff.

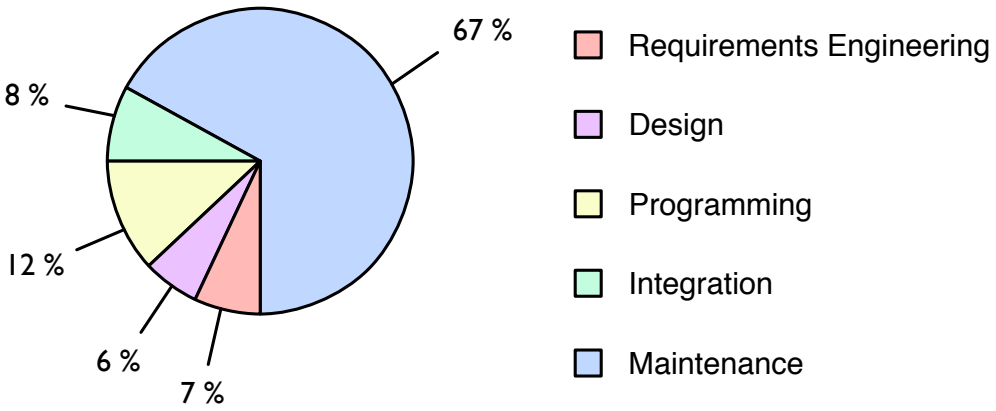

Abbildung 8.1.: Ungefähre relative Kosten der Phasen des Software-Lebenszyklus (nach [Sch99] S. 11 und [OB])

8.3. Priorisierung der Soll-Kriterien

Da 18 Soll-Kriterien definiert wurden und diese in jeder Alternative untersucht werden müssten, würde das den Rahmen der Diplomarbeit sprengen. Deshalb sollen die Soll-Kriterien in eine Hierarchie entsprechend ihrer Wichtigkeit für die ZKB aufgeteilt werden. Es sollen nur die "wichtigen" Soll-Kriterien in die Evaluation mit einbezogen werden. Die Hierarchie wird deshalb in drei Gruppen aufgeteilt:

- Wichtig
- Nice to have
- Unwichtig

Es soll eine Konsolidierung statt finden, zwischen den 18 Soll-Kriterien und den wichtigen Aspekten im Software-Lebenszyklus für die ZKB. Die Konsolidierung wird in der Tabelle 8.1 gezeigt. Die Soll-Kriterien mit zwei oder mehr + werden als "wichtig", mit einem + als "nice to have" und ohne + als "nicht wichtig" priorisiert. Die Gründe für die Priorisierung soll anschliessend erläutert werden.

8.3.1. Wichtig

Diese Soll-Kriterien dienen als Basis für den Kriterienkatalog, welcher in der gewichteten Nutzwertanalyse verwendet wird.

Es handelt sich dabei hauptsächlich um grundlegende Anforderungen, die erfüllt werden müssen, damit Webapplikationen zeitgemäss entwickelt und später gewartet werden können.

Soll-01 - Zugriffskontrolle Auf die Sicherheit von Applikationen in einer Bank wird grossen Wert gelegt. Mit einer genügenden Authentifizierung, Autorisation und Rollenverwalung, kann eine Applikation sicher implementiert werden.

Soll-03 - Modulare Architektur Um während der Maintenance-Phase des Software Lebenszyklus notwendige Anpassungen an einer Applikation machen zu können, ist eine modulare Architektur von grossem Vorteil. Zudem ist das auch während der Entwicklungs-Phase hilfreich.

Soll-06 - Testing Um während der Maintenance-Phase des Software-Lebenszyklus notwendige Anpassungen an einer Applikation machen zu können, ist ein gutes Testing von grossem Vorteil. Das Testing hat auch einen grossen Einfluss auf das Image, denn wer

möchte schon fehlerhafte Software verwenden. Durch eine gute Testbasis, kann auch die Sicherheit einer Applikation gesteigert werden.

Soll-12 - Dokumentation Eine saubere und verständliche Dokumentation eines Web Application Frameworks ist das A und O, um während allen Software-Lebenszyklen effizient arbeiten zu können. Ein Web Application Framework mit einer tiefgreifenden Dokumentation hat bestimmt höhere Chancen sich längerfristig am Markt durchzusetzen.

Soll-13 - Community Damit die notwendigen Ressourcen, für die Umsetzung und Wartung eines Software-Projekts gefunden werden, wird eine entsprechend grosse Community benötigt. Wenn die Community genügend gross ist, kann man davon ausgehen, dass auch in der Zukunft noch genügend Ressourcen in diesem Bereich erreichbar sind.

Soll-18 - AJAX-Unterstützung In der Zeit von Google Mail, Facebook und Twitter wird die Verwendung von Ajax unterstützten Web Applikationen zur Gewohnheit. Um mit einer Web Applikation ein zeitgemässes, flexibles und innovatives Image vermitteln zu können, sollten diese mit Ajax-Unterstützung gebaut werden.

8.3.2. Nice to have

Diese Soll-Kriterien fliessen nicht in den Evaluationsprozess mit ein. Das Vorhandensein würde in der Entwicklungsphase aber sicher einen Erleichterung mit sich bringen.

Soll-02 - Form-Validierung Eine saubere Validierung der Formularfelder gehört zu den notwendigen Werkzeugen, um eine sichere Web Applikation zu entwickeln. Dennoch kann durch gutes Testing und eine sichere Zugriffskontrolle der selbe Effekt erzielt werden.

Soll-04 - Schnittstellen und Webservices Durch die breite Unterstützung von Schnittstellen und Webservice Technologien in der Java Welt, muss ein Web Framework dies nicht schon von Haus aus mitbringen. Fall es doch vorhanden ist, um so besser.

Soll-14 - IDE-Unterstützung Eine IDE-Unterstützung ist zu präferieren, soll aber kein Hindernis für den Einsatz eines guten Web Frameworks sein.

Soll-17 - Lernkurve für EntwicklerInnen Durch die Unterstützung einer soliden Community und einer guten Dokumentation kann die Lernkurve beträchtlich beeinflusst werden.

8.3.3. Unwichtig

Bei diesen Soll-Kriterien handelt es sich meistens um sehr spezifische Anforderungen an ein Web Framework. Da die Evaluation auf Frameworks aus dem Java Umfeld eingeschränkt ist, werden gewisse Anforderungen dadurch schon abgedeckt und sind somit zu vernachlässigen. Durch die Involvierung der ZKB gibt es zudem Anforderungen die hinfällig sind.

Soll-05 - MVC-Entwurfsmuster Das Model View Controller (MVC) Konzept ist sicher gut, sollte aber nicht eine Anforderung an ein Web Framework sein, da es ebenbürtige Alternativen gibt, eine nennenswerte währe Model View Presenter (MVP).

Soll-07 - Internationalisierung und Lokalisierung Die ZKB hat ihr Geschäftsfeld im Kanton Zürich. Somit ist eine Internationalisierung und Lokalisierung nicht wirklich nötig. Da die ZKB im Auftrag des Kantons handelt, gehe ich davon aus, dass sich das nicht so schnell ändern wird.

Soll-08 - Object Relational Mapping (ORM) In der Java Welt gibt es viele ORM Lösungen, welche sich in den Jahren durchgesetzt haben, ein paar nennenswerte sind Hibernate, TopLink und EclipseLink. Da es Sache des Applikationsservers ist, welches ORM verwendet wird, stellt dies keine Anforderung an ein Java Web Framework.

Soll-09 - Scaffolding / Rapid Prototyping Scaffolding und Rapid Prototyping kann in der Programmier-Phase des Software-Lebenszyklus die Arbeit erleichtern. Es gibt sich daraus aber kein Benefit, wenn das in der Maintenance-Phase zur Verfügung steht.

Soll-10 - Caching Caching kann bei Webseiten mit viel statischen Content die Datenmenge, die übertragen wird, enorm reduzieren. Bei Business-Applikationen, die über eine Rollenverwaltung verfügen, sind die Daten die übertragen werden, meistens sehr individuell. Somit besteht keine grosse Nachfrage nach Caching

Soll-11 - View-Engine Da Java eine Objekt-Orientierte Programmiersprache ist, wird das schon von den sprachlichen Begebenheiten her unterstützt.

Soll-15 - Kosten für Entwicklungswerkzeuge Die Kosten der Entwicklungswerkzeuge haben keinen Einfluss auf die Maintenance-Phase des Software-Lebenszyklus. Zudem gibt es genügend IDEs für Java, die Open Source sind. Ein Paar nennenswerte sind Eclipse, NetBeans und IntelliJ.

Soll-16 - Eignung für agile Entwicklung Die Ansätze von Scrum und extreme Programming werden in der ZKB je länger je mehr angewendet. Dennoch hat das keinen Einfluss auf die Maintenance-Phase des Software-Lebenszyklus.

Anforderung	(Si)	(Ko)	(Re)	(Im)
Soll-01 - Zugriffskontrolle	++			
Soll-02 - Form-Validierung	+			
Soll-03 - Modulare Architektur		++		
Soll-04 - Schnittstellen und Webservices		+		
Soll-05 - MVC-Entwurfsmuster				
Soll-06 - Testing	+	++		++
Soll-07 - Internationalisierung und Lokalisierung				
Soll-08 - Object Relational Mapping (ORM)				
Soll-09 - Scaffolding / Rapid Prototyping				
Soll-10 - Caching				
Soll-11 - View-Engine				
Soll-12 - Dokumentation		++		
Soll-13 - Community			++	
Soll-14 - IDE-Unterstuetzung		+		
Soll-15 - Kosten fuer Entwicklungswerkzeuge				
Soll-16 - Eignung fuer agile Entwicklung				
Soll-17 - Lernkurve fuer EntwicklerInnen			+	
Soll-18 - AJAX-Unterstuetzung				++

Tabelle 8.1.: Wie wichtig sind die Soll-Kriterien für die ZKB.

(Si) Sicherheit
(Ko) Kosten in der Maintenance-Phase des Software-Lebenszyklus
(Re) Ressourcen
(Im) Image
++ hat grossen Einfluss
+ hat Einfluss

9. KO-Kriterien für die Evaluation

Anhand einer Menge von KO-Kriterien wird die Auswahl der Alternativen eingeschränkt. Die KO-Kriterien sind aus dem *Handbuch der IT-Architektur*, siehe [uS10], der ZKB entnommen. Die Namensgebung in dem Handbuch unterscheidet sich leicht, es wird nicht von KO-Kriterien, sondern von Grundsätzen gesprochen.

KO-Kriterien sind Vorgaben, welche zwingend erfüllt sein müssen. Falls ein Kriterium nicht erfüllt werden kann, fällt die Entscheidung auf diese Alternative negativ aus. Jedem KO-Kriterium wird für die Identifikation eine eindeutige ID vergeben. Die ID setzt sich folgendermassen zusammen: {KO}-{Laufnummer}.

9.1. Grundsätze aus der IT-Architektur der Zürcher Kantonalbank

Ein Grundsatz wird im Handbuch wer IT-Architektur wie folgt definiert:

> "*Es sind Grundsätze definiert, nach denen sich die Baupläne der IT-Systeme zu richten haben. Die Grundsätze sind ein Regelwerk mit Weisungscharakter.*" [1]

Dabei gibt es eine Hintertür:

> "*Grundsätze sind verbindliche Vorgaben (Konventionen), von denen nur in begründeten Ausnahmen abgewichen werden kann.*" [2]

Das Dokument wurde analysiert und alle Grundsätze, welche für diese Diplomarbeit relevanten sind, werden im Anhang F Grundsätze der ZKB IT-Architektur (auf Seite 115ff) aufgelistet und mit einer ID versehen. Insgesamt wurden 44 Grundsätze erkannt, welche einen Einfluss auf die Evaluation haben könnten.

[1] [uS10] Kapitel 1.3 - *Was ist die IT-Architektur der ZKB*, Seite 11
[2] [uS10] Kapitel 1.8 - *Leseanleitung*, Seite 14

9.2. KO-Kriterien die im Rahmen der Diplomarbeit nicht beachtet werden sollen

Damit die Durchführung der Evaluation einen Sinn ergibt, sollen einige KO-Kriterien nicht beachtet werden. In Absprache mit dem Auftraggeber betrifft das folgende KO-Kriterien, die im Rahmen der Diplomarbeit nicht berücksichtigt werden:

- KO-09 - Für Java-Applikationen (Internet, Extranet und Intranet) wird das ZIP-Framework eingesetzt.
- KO-16 - Die Internet-Applikationen funktionieren auch eingeschränkt, ohne dass die Skript-Funktion im Browser aktiviert ist.
- KO-20 - Für Ultra Thin Clients bzw. Browser-basierende Applikationen muss das aktuelle, Struts-basierende HTML-Client-Framework der ZKB Internet Plattform verwendet werden.

10. Evaluation der Java Web Frameworks

Dieses Kapitel behandelt die Evaluation der Java Web Frameworks. Es wird die kombinierte Methode, aus gewichteter Nutzwertanalyse und AHP, aus dem Kapitel 5 Methoden zur Entscheidungsfindung bei einer Evaluation (auf Seite 20ff) angewendet. Mit der Definition von sinnvollen Rahmenbedingungen, soll die Evaluation durchgeführt und das Resultate in Form einer Rangliste vorgelegt werden.

10.1. Rahmenbedingungen

Über Soll-Kriterien und KO-Kriterien werden die Rahmenbedingungen für die Evaluierung festgelegt. Die Soll-Kriterien widerspiegeln die Anforderungen, welche an ein Java Web Framework gestellt werden. Die KO-Kriterien dienen zum Ausschluss, nicht geeigneter Frameworks.

10.2. Auswahl der Java Web Frameworks

Es sollen vier Java Web Frameworks für die Evaluation ausgewählt werden. Ein Java Web Framework, das in Frage kommt, wird Alternative genannt. Jede Alternative wird mit einer ID in der Form {A}-{Laufnummer} versehen.

10.2.1. Begründung

Es soll für jede Alternative der Grund der Wahl erläutert werden. In Absprache[1] mit dem Projektbetreuer soll aus den verschiednen Gruppen von Java Web Frameworks jeweils eines gewählt werden. Es wurden folgende Typen definiert:

- RIA - Frameworks
- MVC - Frameworks

[1] Siehe Anhang G Kick-off Protokoll S. 120 und H Design-review Protokoll S. 122

- Java Script/HTML/CSS - Cross-Compiler Frameworks
- In der ZKB bereits eingesetzte Frameworks

A-1 - ULC, Canoo RIA Suite

ULC, Canoo RIA Suite zählt zu der Gruppe der RIA Frameworks. Gemäss Kick-off Protokoll Beschluss[2] soll dieses Framework als Alternative gewählt werden. Zudem wird die ULC, Canoo RIA Suite bereits von den Schweizer Banken Credit Suisse und UBS eingesetzt.

A-2 - Struts 1.3.10 mit ZIP-Framework

Struts 1.3.10 mit ZIP-Framework zählt zu der Gruppe der MVC - Frameworks. Es wird in der ZKB bereits eingesetzt. Das Framework wird innerhalb der ZKB für die Umsetzung der Online Bank verwendet und gilt als Standard für neue Projekte, siehe [uS10] S. 140. Damit das Ergebnis der Evaluation Begründet werden kann, sollte ein Vergleich mit diesem Framework existieren.

A-3 - Vaadin 6.6.0

Vaadin zählt zu der Gruppe der Java Script/HTML/CSS - Cross-Compiler Frameworks. Es baut auf Google Web Toolkit (GWT) auf, das in den letzten Jahren an Aufmerksamkeit gewonnen hat und auch im Finanzsektor eingesetzt wird, siehe [Inc10]. Vaadin wird auch in der Finanzbranche der Schweiz eingesetzte, so zählt zum Beispiel, die Betreiberin des Finanzmarks Schweiz, die SIX Group, zu dessen Anwender, siehe [Ltd11a].

A-4 - Apache Wicket 1.4.17

Apache Wicket gehört zu der Gruppe die in der ZKB bereits eingesetzt wird. Die Wahl ist auf Apache Wicket gefallen, da es sich schon länger am Markt etabliert hat und somit auf eine breite Community stützt. Die Einsatzfähigkeit innerhalb der Bank hat es auch schon bewiesen.

[2] Siehe Anhang G Kick-off Protokoll S. 120

10.3. Prüfen der zu beachtenden KO-Kriterien

Es soll eine Prüfung aller Java Web Frameworks statt finden, ob ein solches gegen definierte KO-Kriterien verstösst. Falls das der Fall ist, wird das Framework von der Evaluation ausgeschlossen. Falls das nicht zutrifft, wird das Framework in die Evaluation mit einbezogen, siehe Abbildung 5.4.

In Absprache mit dem Fachbetreuer der ZKB gibt es gewisse KO-Kriterien, die im Rahmen der Diplomarbeit ausser Kraft treten, und somit bei dieser Prüfung nicht beachtet werden sollen, siehe Abschnitt 9.2 KO-Kriterien die im Rahmen der Diplomarbeit nicht beachtet werden sollen (auf Seite 56). Das Resultat wird in der Tabelle 10.1 dargestellt.

Java Web Framework	Verstoss gg KO-Kriterium	Ausschluss
A-1 - ULC, Canoo RIA Suite	KO-18 KO-19	Ja
A-2 - Struts 1.3.10 mit ZIP-Framework	-	Nein
A-3 - Vaadin 6.6.0	-	Nein
A-4 - Apache Wicket 1.4.17	-	Nein

Tabelle 10.1.: Verstösse gegen KO-Kriterien

10.3.1. A-1 - ULC, Canoo RIA Suite - KO-Kriterien gefunden

Leider wurde während der Analyse festgestellt, dass die ULC, Canoo RIA Suite nur in einer Java Virtual Machine[3] lauffähig ist. Das wird über die Technik von Java-Web-Start[4] oder mit der Hilfe von einem Java-Applet[5] vollbracht. Das ist ersichtlich im ULC Architektur Guide, siehe [AG] S. 18. Damit verstösst diese Alternative gegen die KO-Kriterien KO-18 und KO-19. Aufgrund des Vorgehens aus der Abbildung 5.4 wird diese Alternative aus der Evaluation ausgeschlossen.

[3] Die Java Virtual Machine ist der Teil der JRE, der für die Ausführung des Java-Bytecodes verantwortlich ist, siehe [Wik11f].

[4] Java-Web-Start ist eine Technik von Oracle (damals entwickelt von Sun Microsystems), die es ermöglicht, Java-Anwendungen über das Internet mit nur einem Klick zu starten. Im Unterschied zu Java-Applets benötigen Java-Web-Start-Anwendungen keinen Browser, um ablaufen zu können, siehe [Wik10b].

[5] Ein Java-Applet ist ein Computerprogramm, das in der Programmiersprache Java verfasst wurde und normalerweise in einem Webbrowser ausgeführt wird, siehe [Wik11d].

Der Client könnte auch als standalone Client implementiert oder in einen bestehenden Client integriert werden. Das nützt nichts, da diese Situation bereits mit den Java Swing Clients besteht, und abgelöst werden soll.

Falls sich die Grundsätze der IT-Architektur der ZKB in der Zukunft dahingehend ändern, dass Java Web Start oder Java Applets verwendet werden dürften, stellt die ULC, Canoo RIA Suite ein gute Alternative zur Ablösung bestehender Swing Applikationen. Die Eignung müsste zu diesem Zeitpunkt neu geprüft werden.

10.4. Gewichtete Nutzwertanalyse mit dem Analytic Hirarchy Process

Es sollen die Soll-Kriterien, auf welche die Frameworks verglichen werden, bestimmt werden. Danach soll nach der Methode des AHP die Gewichtung der Soll-Kriterien vorgenommen und, zur Berechnung der Nutzwertanalyse, verwendet werden. Für jede mögliche Alternative soll der Erfüllungsgrad der Soll-Kriterien bestimmt und, zur Berechnung der Nutzwertanalyse, verwendet werden. Das Resultat der Nutzwertanalyse soll dargestellt werden.

10.4.1. Bestimmen der Kriterien

Die Kriterien, welche in die Evaluation mit einbezogen werden, und somit für jede Alternative untersucht werden soll, sind die als "wichtigen" priorisierten Soll-Kriterien aus dem Kapitel 8 Soll-Kriterien für die Evaluation (auf Seite 49ff):

- Soll-01 - Zugriffskontrolle
- Soll-03 - Modulare Architektur
- Soll-06 - Testing
- Soll-12 - Dokumentation
- Soll-13 - Community
- Soll-18 - AJAX-Unterstützung

10.4.2. Bestimmen der Gewichte mit dem Analytic Hirarchy Process

Die Bestimmung der Gewichte wird über die Methode des AHP gemacht. Für den Vergleich der Soll-Kriterien wird die Frage gestellt:

> *"Wie wichtig ist das Soll-Kriterium für die ZKB?".*

Es werden nur alle "wichtigen" Soll-Kriterien miteinander verglichen. Die Vergleichsmatrix und die daraus resultierende Gewichtung ist in der Tabelle 10.2 und der Abbildung 10.1 ersichtlich. Für die Werte in der Vergleichsmatrix wird die Skala der Vergleichsgrad, siehe Tabelle 5.3, verwendet. Die gewählten Werte wurden vom Studenten, im Sinne der ZKB, vergeben. Um eine objektivere Vergabe der Punkte zu erhalten, müssten die erläuterten Ansätze[6] aus dem Abschnitt 5.6 Verbesserung der Gewichtung (auf Seite 26) angewendet werden, auf deren Einsatz im Rahmen der Diplomarbeit verzichtet wurde.

Um die Gewichte anhand der erstellten Vergleichsmatrix zu berechnen, wurde das Java Programm JAHP 2.1 verwendet.

Der Inkonsistenzfaktor beträgt 0.04754 und ist somit sehr klein. Die getroffenen Annahmen der Gewichtung sollten deshalb aussagekräftig sein.

Vergleich	Soll-01	Soll-03	Soll-06	Soll-12	Soll-13	Soll-18	**Gewichtung**
Soll-01	1	5	3	5	5	3	34.81 %
Soll-03	$\frac{1}{5}$	1	$\frac{1}{3}$	1	1	$\frac{1}{5}$	5.91 %
Soll-06	$\frac{1}{3}$	3	1	3	3	1	17.93 %
Soll-12	$\frac{1}{5}$	1	$\frac{1}{3}$	1	$\frac{1}{3}$	$\frac{1}{5}$	4.85 %
Soll-13	$\frac{1}{5}$	1	$\frac{1}{3}$	3	1	$\frac{1}{5}$	9.07 %
Soll-18	$\frac{1}{3}$	5	1	5	5	1	27.43 %

Tabelle 10.2.: Vergleichsmatrix der Soll-Kriterien nach der Methode des AHP.

10.4.3. Bestimmen des Erfüllungsgrades

Für die Werte zur Bestimmung der Erfüllungsgrade wird die Skala der Erfüllungsgrade, siehe Tabelle 5.1, verwendet. Die Begründung der gegebenen Punkt wird für jeden Punkt erläutert.

[6] Es gibt sicher noch weitere Ansätze die nicht aufgeführt wurden, welche die Qualität der Gewichtung verbessern würden.

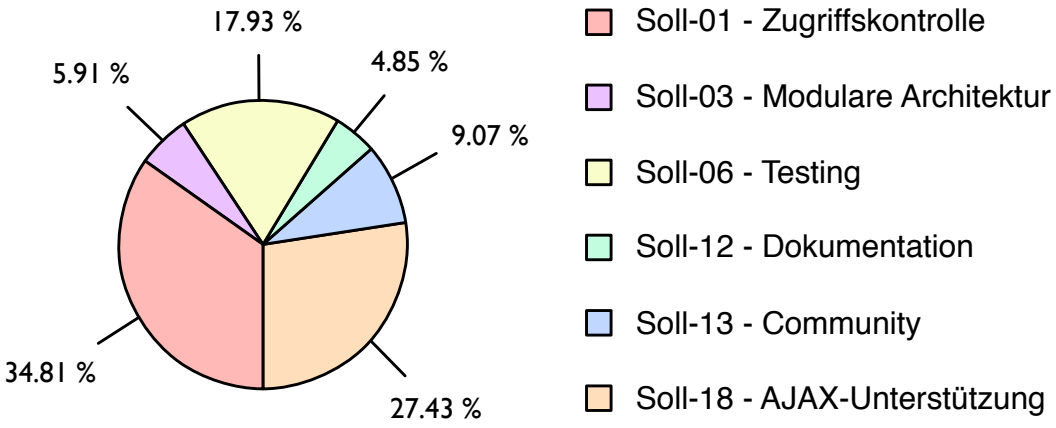

Abbildung 10.1.: Gewichtung der Soll-Kriterien nach der Methode des AHP

10.4.4. A-1 - ULC, Canoo RIA Suite

ULC, Canoo Ria Suite wurde aufgrund der KO-Kriterien KO-18 und KO-19 aus der Evaluation ausgeschlossen.

10.4.5. A-2 - Struts 1.3.10 mit ZIP-Framework

Für Informationen zum Framework Struts 1.3.10 in Kombination mit dem ZIP Framework habe ich Marco Spörri interviewt. Er arbeitet seit längerer Zeit damit. Struts 1.3.10 ist ein Open Source Framework, das von der Apache Software Foundation gepflegt wird. Der ZIP Teil ist eine Erweiterung, welche in der ZKB entwickelt wurde. Marco Spörri gehört zu den Personen, welche aktuell daran Erweiterungen anbringt.

Das Ergebnis der Nutzwertanalyse ist in der Tabelle 10.3 ersichtlich. Der Nutzwert liegt bei 3.9829 und ist somit durchschnittlich, das entspricht dem Erfüllungsgrad *mittel*.

Soll-01 - Zugriffskontrolle - 6 Punkte

Sowohl Struts, wie auch ZIP bietet von Grund auf kein Rollenkonzept an.

Die Authentifizierung wird ganz normal über eine Session-ID gelöst.

Die Autorisierung für alle Business relevanten Requests wird vom Businesslayer jedes mal geprüft.

Kriterium	Gewichtung g	Erfüllungsgrad e	Wertigkeit $_{A-2}$
Soll-01	34.81 %	6	2.0886
Soll-03	5.91 %	5	0.2955
Soll-06	17.93 %	5	0.8965
Soll-12	4.85 %	7	0.3395
Soll-13	9.07 %	4	0.3628
Soll-18	27.43 %	0	0.0
Ergebnis	100.00 %		3.9829

Tabelle 10.3.: Nutzwertanalyse der Alternative A-2 - Struts 1.3.10 mit ZIP-Framework

Cross-Site Scripting wird über Escaping verhindert. Zudem werden alle Formulardaten validiert und verifiziert, somit wird SQL-Injection unterbunden.

Ein Mechanismus des ZIP Frameworks lässt alle Namen der HTML-Formular Felder obfuskieren. Damit wird erreicht, dass aufgrund der Daten, welche vom Browser an den Server gesendet werden, nicht direkt auf die Businessfunktion zurückgeschlossen werden kann. Das Obfuskieren der Formularfelder Namen passiert bei jedem Aufbau einer HTML Seite von neuem. Das Mapping zwischen den generierten Namen und den jeweiligen Businessfunktionen wird auf dem Server gehalten.

Die gängigsten Sicherheitsvorkehrungen können getroffen werden, leider bietet das Framework kein Rollenkonzept an, somit gibt es sechs Punkte

Soll-03 - Modulare Architektur - 5 Punkte

Es steht kein Add-On oder Plugin Mechanismus zur Verfügung. Da JavaServer Pages (JSP)[7] für die View verwendet wird, kann durch die Erweiterung von Tag-Bibliotheken[8] ein modularer Ansatz verfolgt werden. Neu entwickelte Tag's werden im ZIP-Framework integriert und so für zukünftige Verwendung zentral verwaltet.

JSP biete mit dem $< jsp : include >$ Tag die Möglichkeit eines modularen Ansatzes für die Softwareentwicklung. Jedoch ist die Trennung zwischen Model, View und Controller bei

[7] JavaServer Pages, abgekürzt JSP, ist eine von Sun Microsystems entwickelte Web-Programmiersprache. Sie erlaubt, Java-Code und spezielle JSP-Aktionen in HTML- oder XML-Seiten einzubetten, siehe [Wik11g].
[8] Tag-Bibliotheken sind ein Bestandteil der JSP-Spezifikation. Mit ihrer Hilfe ist es möglich, JSP-Seiten zu entwickeln, die nur noch wenig bis gar keinen Java-Code beinhalten, siehe [Wik10d]

JSP-Files meistens nicht sehr klar, was nicht gerade im Sinne einer modularen Architektur ist.

Wie überall in der Javawelt kann durch die Einbindung von JAR-Dateien[9] die Funktionalität des Frameworks erweitert werden.

Durch den modularen Ansatz mit Tag-Bibliotheken gibt es fünf Punkte.

Soll-06 - Testing - 5 Punkte

Struts 1.3.10 biete grundsätzlich die Unterstützt für Unit Tests. Die Notwendigen Klassen befinden sich im Package *org.apache.struts.mock*.

Für das Testen von Views gibt es keine Komponenten, deshalb muss auf Tools von Drittanbieter, wie zum Beispiel Selenium, zurückgegriffen werden.

Durch die Unterstützung von Unit Tests gibts 5 Punkte.

Soll-12 - Dokumentation - 7 Punkte

Das Projekt Struts 1.3.10 bietet Dokumentation in der Form von JavaDoc, ein FAQ und HowTo. Zudem werden Links zu FAQ's und HowTo's von Drittparteien aufgelistet. Es gibt ein User Guide in dem alle Funktionen beschrieben werden.

Auf Amazon gibt es eine Fülle von Büchern welche sich direkt mit Struts oder indirekt mit JSP befassen.

Für das ZIP-Framework existiert die Dokumentation der API, via JavaDoc. Es existiert ein Tutorial Projekt, in welchem die möglichen Funktionen implementiert sind. Zusätzlich gibt es eine Dokumentation die in Text verfasst wurde und es existiert eine Präsentationsreihe, auf der Basis von PowerPoint, die zu Schulungszecken erstellt wurde.

Für das Framework existiert insgesamt eine solide Dokumentation somit gibt es sieben Punkte.

[9] Ein Java Archive (umgangssprachlich aufgrund der Dateiendung auch JAR-Datei genannt) ist eine ZIP-Datei. JARs werden vor allem zur Verteilung von Java-Klassenbibliotheken und -Programmen eingesetzt, siehe [Wik11e]

Soll-13 - Community - 4 Punkte

Apache Struts 1.x wurde im Jahre 2000 ins Leben gerufen und hat somit schon einige Jahre auf dem Buckel. Die aktuell stabile Version ist 1.3.10. Mit dem Java Web Framework Struts 2 wurde die Nachfolgerin ins Leben gerufen, welche aktuell in der Version 2.2.3 verfügbar ist.

Laut ohloh[10] gibt es 89 User, die Struts 1.x verwenden. Laut ohloh gibt es aktuell nur noch eine Person, welche das Framework aktiv weiterentwickelt. Das liegt wohl daran, dass die Version 1.3.10 entweder sehr stabil ist, oder das die meisten Entwickler auf die neuste Version von Apache Struts 2.2.3 umgestiegen sind.

Für Struts 1.x gibt es keine DZone Refcard.

Bei DZone findet man 9136 Posts, welche sich mit Struts 1 befassen. Dies wurde mit dem Suchbegriff "Struts 1" eruiert.

Das ZIP-Framework wird von ca. 30 Personen eingesetzt und aktuell gibt es vier Personen, welche Bugs beheben und Features einbauen. Das Framework wurde von der ZKB für eigene Zwecke entwickelt und wird auch nur innerhalb der ZKB verwendet.

Struts 1.x ist wohl schon länger überholt und durch Struts 2.x abgelöst worden. Das Zip-Framework wird aktuell noch weiterentwickelt und auch eingesetzt. Durch die veraltete Grundlage mit Struts 1.x gibt es nur vier Punkte.

Soll-18 - AJAX-Unterstützung - 0 Punkte

Apache Struts 1.3.10 und auch das ZIP-Framework bieten keine Ajax Unterstützung, somit gibt es null Punkte.

10.4.6. A-3 - Vaadin 6.6.0

Das Ergebnis der Nutzwertanalyse ist in der Tabelle 10.4 ersichtlich. Der Nutzwert liegt bei 7.0168 und ist somit relativ hoch und entspricht dem Erfüllungsgrad *gut*.

[10] ohloh ist eine Community Plattform für Softwareentwickler, siehe http://www.ohloh.net/

Kriterium	Gewichtung g	Erfüllungsgrad e	Wertigkeit $_{A-3}$
Soll-01	34.81 %	7	2.4367
Soll-03	5.91 %	5	0.2955
Soll-06	17.93 %	9	1.6137
Soll-12	4.85 %	8	0.3880
Soll-13	9.07 %	4	0.3628
Soll-18	27.43 %	7	1.9201
Ergebnis	100.00 %		7.0168

Tabelle 10.4.: Nutzwertanalyse der Alternative A-3 - Vaadin 6.6.0

Soll-01 - Zugriffskontrolle - 7 Punkte

Vaadin biete von sich aus keine Unterstützung für Authentifizierung und Autorisierung an. Es gibt aber ein alternatives Projekt, das sich vaadin-appfoundation[11] nennt und im offiziellen Vaadin add-on Repository verfügbar ist. Dieses Projekt beinhaltet ein Autorisierungsmodul, mit dem ein Rollenkonzept implementiert werden kann.

Zum Thema Sicherheit behauptet Vaadin von sich, derzeit eines der sichersten Web Frameworks zu sein.

> *"The server-side architecture of Vaadin is immune to the usual risks affecting many other RIA frameworks. Your business logic and UI state are kept securely on the server, and never sent to the browser where it could be analyzed by a potential attacker. Vaadin doesn't even trust the components or the RPC model of GWT, which is used to render the UI.*
>
> *In addition, we have implemented several other security mechanisms in Vaadin to prevent eg. man-in-the-middle and cross-site scripting attacks."* [12]

Zwei Punkte Abzug, da keine Rollenverwaltung von Grund auf vorhanden ist.

Soll-03 - Modulare Architektur - 5 Punkte

Vaadin bietet ein eigenes Add-on Repository an, unter welchem Erweiterungen zum Framework bezogen werden können. Das deutet auf eine modulare Architektur hin.

[11] Projektinformationen sind hier zu finden: http://code.google.com/p/vaadin-appfoundation/
[12] Vaadin Ltd., 2011, siehe [Ltd11b]

Es können eigene GUI Komponenten erstellt und auch einfach wiederverwendet werden. Dafür verwendet Vaadin den modularen Aufbau von GWT.

Wie modular die Architektur wirklich ist, konnte ich nach meinem Research nicht genau abschätzen, deshalb gibt es nur fünf Punkte.

Soll-06 - Testing - 9 Punkte

Für das Framework steht eine komplette Test Suite namens TestBench zur Verfügung. Mit TestBench können UI Regressionstest aufgezeichnet und automatisiert durchgeführt werden. TestBench ist eine kostenpflichtige Extension zu Vaadin. Das ganze basiert auf Selenium, einem Open Source Projekt zur Abwicklung von Tests im Web-Umfeld, und JUnit, einem Open Source Projekt zur Abbildung von Unit Tests.

Da ein Test Framework existiert, das auf aktuellen Techniken basiert, gibt es die maximale Punktzahl.

Soll-12 - Dokumentation - 8 Punkte

Die Entwickler haben sogleich ein komplettes Buch zum Framework mit vielen hilfreichen Beispielen geschrieben, siehe [Gro10]. Das Buch ist in HTML oder als Portable Document Format (PDF) frei verfügbar und kostenlos. Es enthält alle wichtigen Informationen, welche zur Entwicklung nötig sind. Auf Amazone[13] gibt es nur ein weiteres Buch, das sich diesem Framework gewidmet hat, das ist aber nicht weiter schlimm, da das Framework relativ neu ist.

Die Dokumentation wird auf der Webseite mit einer Menge von Tutorials abgerundet. Es existieren auch die üblichen Dinge, wie das JavaDoc zum Application Programming Interface (API), die Hinweise auf die Lizenz, Frequently Asked Questions (FAQ) und vieles mehr. Das alles wirk sehr sauber, schön und vollständig.

Leider existiert die Dokumentation nur auf Englisch, deshalb gibt es einen Punkt Abzug.

Soll-13 - Community - 4 Punkte

Das Framework Vaadin ist unter seinem Namen noch nicht sehr lange bekannt, früher wurde es unter dem Namen IT Mill entwickelt. Das kann zu einem leicht falschen Resultat führen, wenn man die Community nach Vaadin untersucht.

[13] Amazon ist ein online Buchladen, siehe http://www.amazon.de

Laut ohloh[14] gibt es 183 User, die Vaadin verwenden und 62 Contributer. Die meisten Commits werden von den Mitarbeitern der Firma Vaadin Ltd gemacht, welche das Framework erfunden hat.

Bei DZone[15] findet man 75 Posts, welche sich direkt mit Vaadin befassen. Dies wurde mit dem Suchbegriff "Vaadin" eruiert.

Für Vaadin gibt es ein DZone Refcard[16]. Wenn für ein Framework ein Refcard existiert, sagt das aus, dass es von der Community verwendet wird.

Die Firma Vaadin Ltd bietet mit Vaadin Pro die Möglichkeit von bug-fix priority, direkten Support durch das Vaadin Entwicklungsteam und Zugriff auf eine Vaadin Knowledge-Base, siehe [Ltd11c]. Zudem können gegen ein Entgeld Fachkräfte für einen Tag oder drei Wochen (ein Sprint im Sinne von Scrum) verpflichtet werden. Es gibt auch die Möglichkeit einen Proof of Concept durch das Vaadin Entwicklungsteam erstellen zu lassen.

Der Support durch die Firma Vaadin Ltd sieht sehr gut aus, die Community für Vaadin ist aber gerade erst dabei sich zu bilden und noch nicht sehr stark, somit sind es nur 4 Punkte.

Soll-18 - AJAX-Unterstützung - 7 Punkte

Ajax wird massiv unterstützt. Alle User Interface (UI) Komponenten und deren Aktionen, sowie auch das ganze Layout Management basieren auf diesem Prinzip. Das Programmiermodell sieht vor, dass der Entwickler die komplette Funktionalität in Java entwickeln kann, sodass das Framework alle dafür notwendigen Javascripts, HTML und CSS generiert. Zudem kann eine zusätzliche Javascript Library wie JQuery oder Prototype in die Applikation integriert werden.

Leider können Vaadin Applikationen nicht auf die Verwendung von Java Script verzichten. Wenn Java Script im Browser nicht unterstützt oder deaktiviert ist, können Vaadin Applikationen nicht ausgeführt werden. Durch das fehlen einer Fallback-Lösung gibt es zwei Punkte Abzug.

[14] ohloh ist eine Community Plattform für Softwareentwickler, siehe http://www.ohloh.net/

[15] Eine Community Plattform für Softwareentwickler, welche unter dem Namen "Javalobby - The heart of the Java developer community" einen speziellen Bereich für Java Entwickler hat, siehe http://www.dzone.com/

[16] Sogenannte Cheat Sheets, die meistens von den Entwicklern eines Frameworks selbst verfasst werden. Sie stehen gratis unter der Uniform Resource Locator (URL) http://refcardz.dzone.com/ zur Verfügung.

10.4.7. A-4 - Apache Wicket 1.4.17

Um an wertvolle Informationen über Apache Wicket zu kommen, habe ich Stefan Pudig interviewt. Er hatte schon ein Projekt in Apache Wicket implementiert, und kennt sich im allgemeinen mit der Materie aus. Zudem wurde das Buch "Wicket - Komponentenbasierte Webanwendungen im Java", siehe [RF10] als Referenz verwendet.

Das Ergebnis der Nutzwertanalyse ist in der Tabelle 10.5 ersichtlich. Der Nutzwert liegt bei 6.4282 und sagt aus, dass das Framework dem Erfüllungsgrad *gut* entspricht.

Kriterium	**Gewichtung** g	**Erfüllungsgrad** e	**Wertigkeit** $_{A-4}$
Soll-01	34.81 %	7	2.4367
Soll-03	5.91 %	5	0.2955
Soll-06	17.93 %	6	1.0758
Soll-12	4.85 %	7	0.3395
Soll-13	9.07 %	7	0.6349
Soll-18	27.43 %	6	1.6458
Ergebnis	100.00 %		6.4282

Tabelle 10.5.: Nutzwertanalyse der Alternative A-4 - Apache Wicket 1.4.17

Soll-01 - Zugriffskontrolle - 7 Punkte

Es existieren folgende Securityfeatures, welche von Grund auf dabei sind:

- Wenn über HTTPS kommuniziert wird, wird ein Session Hijacking unterbunden, sofern kein Zugriff auf die Server Komponente, in der Apache Wicket läuft, besteht.
- Cross-Site Scripting wird anhand einer Escaping Funktion[17] verhindert.
- SQL-Injection wird ebenfalls anhand eines Escapings unterbunden. Zudem wird einem geraten in Java mit *PreparedStatements* zu arbeiten, damit übergebene Parameter nicht mehr interpretiert werden, siehe [RF10] S. 299.
- Authentifizierung und Autorisierung Mechanismen werden vom Framework nur grundlegend unterstützt.

[17] Hierunter versteht man ein Maskierten der JavaScript-Funktionszeichen, sodass diese nicht ausgeführt, sondern angezeigt werden, siehe [RF10] S. 299

- Ein Rollenkonzept wird von Grund auf nicht angeboten, dafür gibt es ein zusätzliches Framework namens “Wicket Auth Roles”.

Zwei Punkte Abzug, da keine Rollenverwaltung von Grund auf vorhanden ist.

Soll-03 - Modulare Architektur - 5 Punkte

Bei diesem Punkt stütze ich mich auf die Aussagen von Stefan Pudig.

Ein offizielles Add-on oder Plugin Repository existiert nicht für Apache Wicket. Dafür können zusätzliche Features in der Form von Bibliotheken manuell zum Projekt hinzugefügt werden. Zudem wird eine Maven Integration gewährleistet.

Die Erweiterung von Komponenten des grafischen User Interfaces (UI) wird sehr gut unterstützt. Durch die Fähigkeit des Vererbens können Komponenten einfach abgeleitet und verändert werden. Diese können dann auch einfach wiederverwendet werden.

Wie modular das Framework selber ist, konnte Stefan Pudig mir nicht nennen, er meint aber, dass die meisten Business Applikationen mit den bestehenden Komponenten abgebildet werden können.

Durch das fehlen eines Add-on Repositories gibt es nur fünf Punkte.

Soll-06 - Testing - 6 Punkte

Wicket bietet eine API an, mit welcher Unit Tests durchgeführt werden können. Es können damit alle gängigen funktionalitäten des Frameworks getestet werden, siehe [RF10] S. 255ff. Dafür muss die Applikation nicht deployed werden, sondern die Applikation wird direkt instanziiert.

Leider wird kein Möglichkeit geboten, Regressionstests abzubilden. Dafür muss man auf auf bestehende Lösungen, wie Selenium[18] zurückgreifen.

Da von Grund auf keine Regressionstests möglich sind gibt es nur sechs Punkte

[18] Siehe http://www.seleniumhq.org/ oder http://code.google.com/p/webdriver

Soll-12 - Dokumentation - 7 Punkte

Auf Amazon werden fünf Bücher zum Apache Wicket Framework angeboten. Das sind genug Referenzen, um sich in die Thematik tief einzuarbeiten.

Die Projektseite[19] ist schlicht und übersichtlich, wie man das von Apache Projekten gewohnt ist. Es ist das JavaDoc zum API verfügbar, es gibt eine Handvoll Beispiele, und eine Übersicht zu den unterstützten Komponenten, zudem gibt es eine komplette Dokumentation zum Thema Wicket. Ein paar Links, zu Blogs, die sich mit der Materie befassen, werden auch propagiert.

Eine unumgängliche Erweiterung der Dokumentation ist das offizielle Wiki[20]. Hier werden alle notwendigen Informationen und Tips und Tricks zusammengezogen.

Alles in allem ist der Umfang gut, die erwähnten Bereiche sind aber nur in Englisch verfügbar. Zudem wirkt die Dokumentation eher wie zusammengewürfelt. Deshalb gibt es zwei Punkte Abzug.

Soll-13 - Community - 7 Punkte

Wicket existiert seit dem Jahr 2004, siehe [Wik11b].

Laut ohloh gibt es 119 User, die Apache Wicket verwenden und 42 Contributer.

Bei DZone findet man 546 Posts, welche sich direkt, oder eventuell auch indirekt, mit Apache Wicket befassen.

Für Apache Wicket gibt es eine DZone Refcard.

Durch das sich Apache Wicket nun schon über sieben Jahre im Java Web Framework Zirkus halten konnte und es immer noch eine aktive Community gibt, werden sieben Punkte erteilt.

Soll-18 - AJAX-Unterstützung - 6 Punkte

Wicket bietet eine AJAX Unterstützung an, bei der die Funktionalität komplett in Java entwickelt werden kann. Die dafür notwendigen Javascripts, welche die Funktionalität bieten, werden durch das Framework generiert.

[19] Apache Wicket Projektseite, siehe http://wicket.apache.org/
[20] Siehe https://cwiki.apache.org/WICKET/

Leider werden nur eine begrenzte Anzahl von Komponenten angeboten, welche via Ajax an Funktionalität gewinnen. Es handelt sich dabei um bekannte Anwendungsfälle, wie: Formvalidierung, Paging navigation, Panel lazy loading oder TextField autocomplete. Es gibt noch ein paar weitere Szenarien, die ich hier nicht erwähnt habe. Zudem kann eine zusätzliche Javascript Library wie JQuery oder Prototype in die Applikation integriert werden, um damit das GUI aufzuwerten.

Wicket bietet eine Fallback Strategie für gewisse Komponenten. Damit wird die Funktionalität auch bei fehlender Javascript Unterstützung gewährleistet.

Durch die begrenzte Unterstützung von Komponenten gibt es nur sechs Punkte

10.5. Resultat

Die Evaluation ist knapp ausgegangen, zumindest auf dem vordersten zwei Plätzen. Da hat "A-3 - Vaadin 6.6.0" gegenüber "A-4 - Apache Wicket 1.4.17" den leicht höheren Nutzwert erreicht.

Das Java Web Framework "A-2 - Struts 1.3.10 mit ZIP-Framework", welches aktuell von der IT-Architektur als der Standard vorgeschrieben wird, hat nur mittelmässig abgeschnitten.

Das Framework "A-1 - ULC, Canoo RIA Suite" ist leider aufgrund der KO-Kriterien aus der Evaluation ausgeschlossen worden.

Das Endergebnis wird in der Tabelle 10.6 in Form einer Rangliste zusammenfassend dargestellt.

Rang	Java Web Framework	Nutzwert
1	A-3 - Vaadin 6.6.0	7.0168
2	A-4 - Apache Wicket 1.4.17	6.4282
3	A-2 - Struts 1.3.10 mit ZIP-Framework	3.9829
-	A-1 - ULC, Canoo RIA Suite	—

Tabelle 10.6.: Ergebnis der Evaluation als Rangliste.

11. Integration in die ZKB IT-Infrastruktur

In diesem Kapitel soll behandelt werden, ob eine Integration der jeweiligen Java Web Frameworks in die Infrastruktur der ZKB möglich ist. Dabei soll geprüft werden, ob die notwenigen APIs, auf welchen die Frameworks aufbauen, durch die zur Verfügung gestellten Applikationsserver, abgedeckt sind. Als Grundlage werden die Informationen aus dem Kapitel 6 (Analyse der Infrastruktur der Zürcher Kantonalbank, S. 32ff) verwendet.

Es wird auf die Prüfung der Integration vom Framework "A-1 - ULC, Canoo RIA Suite" verzichtet, da es aufgrund der Evaluation als mögliche Alternative entfällt.

11.1. A-2 - Struts 1.3.10 mit ZIP-Framework

Das Framework "A-2 - Struts 1.3.10 mit ZIP-Framework" kann in die bestehende Infrastruktur der ZKB integriert werden. Für den Betrieb einer Struts Applikation ist ein Servlet Container notwendig, dieser wird in der Infrastruktur mit dem JBoss Web, welcher auf Apache Tomcat 6.0 basiert, zur Verfügung gestellt. Die genauen Anforderungen an die Konfiguration können aus der Dokumentation entnommen werden, siehe [Fou08a].

Das Framework ist zudem in der ZKB schon im Einsatz. Eine Applikation, welche damit entwickelt wurde, ist das Online-Banking, auch genannt "OnBa".

11.2. A-3 - Vaadin 6.6.0

Das Framework "A-3 - Vaadin 6.6.0" kann in die bestehende Infrastruktur der ZKB integriert werden. Der Betrieb setzt ebenfalls ein Servlet Container voraus. In der Dokumentation ist zu entnehmen, dass es sich dabei mindestens um einen Apache Tomcat 6.0 handelt, siehe [Gro10] S. 11. Als Laufzeitumgebung wird mindestens Java 1.5 vorausgesetzt, was ebenfalls verfügbar ist.

Das Framework wurde in der ZKB noch nie eingesetzt. Der Einsatz müsste mit den aktuellen Bestimmungen aus der IT Architektur über einen Ausnahmegenehmigung geregelt werden.

11.3. A-4 - Apache Wicket 1.4.17

Das Framework "A-4 - Apache Wicket 1.4.17" kann in die bestehende IT Infrastruktur der ZKB integriert werden. Für den Betrieb einer Struts Applikation ist auch nur ein Servlet Container notwendig, siehe [Fou].

Das Framework ist ebenfalls bereits in der ZKB im Einsatz. Der Einsatz wurde durch eine Ausnahmegenehmigung geregelt. Eine Applikation, welche damit entwickelt wurde, ist das Bond-Rating-Tool, auch genannt "BoReTo".

11.4. Resultat

Alle drei evaluierten Java Web Frameworks entsprechen den Begebenheiten aus der ZKB IT-Infrastruktur und sind somit für den Einsatz geeignet. Das Resultat wird in der Tabelle 11.1 dargestellt.

Java Web Framework	Integration möglich
A-1 - ULC, Canoo RIA Suite	Prüfung entfällt
A-2 - Struts 1.3.10 mit ZIP-Framework	Ja
A-3 - Vaadin 6.6.0	Ja
A-4 - Apache Wicket 1.4.17	Ja

Tabelle 11.1.: Ob eine Integration in die IT-Infrastruktur der ZKB möglich ist

12. Implementierung der gefundenen Swingkomponenten

In diesem Kapitel soll behandelt werden, ob eine Implementierung, der im Kapitel 7 (Analyse der Java Swing Applikationen, S. 36ff) gefundenen Java Swing Komponenten, äquivalent möglich ist. Dabei sollen die Dokumentationen und Tutorials, welche für die Frameworks vorhanden sind, als Grundlage dienen. Die Prüfung soll durch einen visuellen Vergleich statt finden, ob die 32 gefundenen Komponenten und die sieben gefundenen GUI Paradigmen abgedeckt sind. Das Resultat soll in Form einer Tabelle mit dem jeweiligen Abdeckungsgrad dargestellt werden.

Wie im Kapitel 5 (Methoden zur Entscheidungsfindung bei einer Evaluation, S. 20ff) beschrieben, soll eine Mindestabdeckung von 80% erzielt werden. Wenn das nicht erreicht wird, bedeutet das den Ausschluss des Frameworks aus der Evaluation.

Es wird auf die Prüfung der Komponenten vom Framework "A-1 - ULC, Canoo RIA Suite" verzichtet, da es aufgrund der geprüften KO-Kriterien als mögliche Alternative entfällt.

12.1. A-2 - Struts 1.3.10 mit ZIP-Framework

In der Dokumentation zu Apache Struts 1.3.10 befindet sich eine Übersicht der unterstützten Komponenten, siehe [Fou08c] und [Fou08b]. Es wird schnell klar, dass es sich dabei um die Standard HTML Komponenten handelt, welche über eine Tag-Bibliothek abstrahiert wurden. Folgende Komponenten werden unterstützt:

- Struts 1.3.10 kennt keinen Tag um ein Dialogfenster zu öffnen. Dies kann bewerkstelligt werden, indem auf JavaScript zurückgegriffen wird. Das wird so von Struts 1.3.10 so nicht direkt unterstützt. Somit fehlt bei den Top-Level Komponenten der Dialog
- Es werden alle Intermediate-Komponenten bis auf den $JTabbedPane$ unterstützt. Der $JScrollPane$ wird über das "FrameTag" gelöst. $JPanel$, $JLayerdPane$ und $JRootPane$ werden mit dem "HtmlTag" abgedeckt.
- Für folgende Atomic-Komponenten fehlt die Unterstützung:

  - $JFormattedTextField$
  - $JMenuBar$
  - $JMenu$
  - $JMenuItem$
  - $JProgressBar$
  - $JSlider$
  - $JSpinner$
  - $JToolTip$ (wird nur bedingt auf z.B. Images unterstützt)
- Für folgende speziellen Komponenten fehlt die Unterstützung:
  - $JFreeChart$
  - $JXBusyLabel$
  - $JXDatePicker$
- Die "Neuen" Komponenten können grundsätzlich alle implementiert werden.

Folgende Design-Patterns werden unterstützt:

- Das MVC-Pattern wird unterstützt
- Das Observer-Pattern wird nicht unterstützt
- Tabellen-Filter werden unterstützt
- Duch die fehlende Ajax Unterstützung ist das Worker-Pattern nicht abgedeckt.

Es ist zu erwähnen, das mit der fehlenden Ajax Unterstützung das GUI Verhalten einer Java Swing Applikation nicht nachgebaut werden kann. Alle Views funktionieren nach dem klassischen "Click und Reload" Prinzip.

Das ergibt eine Abdeckung der GUI-Komponenten von 59.38%. Die Abdeckung der GUI-Paradigmen ist ein wenig höher mit 71.43%.

12.2. A-3 - Vaadin 6.6.0

In der Dokumentation zu Vaadin gibt es eine komplette Übersicht aller unterstützten Komponenten, siehe [Ltd11d]. Diese können online mit einem Browser betrachtet und auf deren Usability ausprobiert werden. Folgende Komponenten werden unterstützt:

- Es werden alle Top-Level Komponenten unterstützt
- Es werden alle Intermediate-Komponenten unterstützt
- Es werden alle Atomic-Komponenten bis auf das $JFormattedTextField$ unterstützt.
- Es werden alle speziellen Komponenten bis auf $JFreeChart$ unterstützt. Dafür gibt es ein offizielles Add-on mit dem Namen "JFreeChart wrapper for Vaadin", welches die Unterstützung von $JFreeChart$ in den Formaten Portable Network Graphics (PNG) und Scalable Vector Graphics (SVG) sichert.
- Die "Neuen" Komponenten können alle implementiert werden. Das Akkordeon und der Zeit-Panel werden schon von Haus aus unterstützt.

Folgende Design-Patterns werden unterstützt:

- Das MVC-Pattern wird unterstützt
- Das Observer-Pattern wird unterstützt
- Tabellen-Filter werden unterstützt
- Duch die Unterstützung von Ajax ist das Worker-Pattern abgedeckt.

Es werden alle Komponenten bis auf das $JFormattedTextField$ und $JFreeChart$[1] unterstützt, das ist eine Abdeckung der GUI-Komponenten von 93.75%. Es werden alle GUI-Paradigmen unterstützt, das ist eine Abdeckung der GUI-Paradigmen von 100.00%.

12.3. A-4 - Apache Wicket 1.4.17

Es existiert eine Komponentenübersicht für das Framework Apache Wicket 1.4.17, siehe [Fou10b]. Die Komponenten basieren auf dem HTML Standard. Es gibt ein Projekt namens Wicketstuff, welches zusätzliche Komponenten für das Framework zur Verfügung stellt. Diese Komponenten wurden ebenfalls mit einbezogen, da es sozusagen zu Apache Wicket dazugehört. Folgende Komponenten werden unterstützt:

[1] Es ist zu bedenken, dass für das $JFreeChart$ ein Add-on existiert, diese Komponente wurde nicht in die Abdeckung der GUI-Komponenten mit eingerechnet.

- Es werden alle Top-Level Komponenten unterstützt
- Es werden alle Intermediate-Komponenten unterstützt
- Für folgende Atomic-Komponenten fehlt die Unterstützung:
  - $JFormattedTextField$
  - $JSlider$ (würde mit dem Projekt wiquery funktionieren)
- Es werden alle speziellen Komponenten unterstützt.
- Die "Neuen" Komponenten können alle implementiert werden. Das Akkordeon wird von Wicketstuff schon unterstützt.

Folgende Design-Patterns werden unterstützt:

- Das MVC-Pattern wird unterstützt
- Das Observer-Pattern wird unterstützt
- Tabellen-Filter werden unterstützt
- Duch die Unterstützung von Ajax ist das Worker-Pattern abgedeckt.

Es werden alle Komponenten bis auf das $JFormattedTextField$ und $JSlider$ unterstützt, das ist eine Abdeckung der GUI-Komponenten von 93.75%. Es werden alle GUI-Paradigmen unterstützt, das ist eine Abdeckung der GUI-Paradigmen von 100.00%.

12.4. Resultat

In der Tabelle 12.1 wird die Abdeckung der unterstützten GUI-Komponenten und Paradigmen dargestellt. Es werden nur die Alternativen mit einer Abdeckung von mindestens 80% im weiteren Verlauf der Evaluation berücksichtigt. In der Tabelle 12.2 ist für den Zweifelsfall, dass die Frameworks den selben Nutzwert ausweisen, die Gesamtabdeckung berechnet. Zudem ist ersichtlich, ob eine Implementierung der GUI-Komponenten und GUI-Paradigmen möglich ist, was bedeutet, dass mindestens 80% abgedeckt sind.

Java Web Framework	GUI-Komponenten	GUI-Paradigmen
A1 - ULC, Canoo RIA Suite	Prüfung entfällt	Prüfung entfällt
A2 - Struts 1.3.10 mit ZIP-Framework	59.38%	71.43%
A3 - Vaadin 6.6.0	93.75%	100.00%
A4 - Apache Wicket 1.4.17	93.75%	100.00%

Tabelle 12.1.: Abdeckungsgrad der gefundenen GUI-Komponenten und Paradigmen

Java Web Framework	Gesamtabdeckung	Implementierung möglich
A1 - ULC, Canoo RIA Suite	Prüfung entfällt	Prüfung entfällt
A2 - Struts 1.3.10 mit ZIP-Framework	61.54%	Nein
A3 - Vaadin 6.6.0	94.87%	Ja
A4 - Apache Wicket 1.4.17	94.87%	Ja

Tabelle 12.2.: Gesamtabdeckung der Komponenten und ob die Implementierung möglich ist

13. Proof of Concept - Umsetzung durch einen Prototypen

In diesem Kapitel sollen die Anforderungen und Akzeptanztests für einen Prototypen definiert werden. Als Grundlage sollen die Komponenten und Verhaltensmuster der Applikation Strukti Live 1.2 dienen, da die Applikation öffentlich verfügbar ist.

Wie im Kapitel 5 (Methoden zur Entscheidungsfindung bei einer Evaluation, S. 20ff) beschrieben, gilt der Proof of Concept als erfolgreich, wenn alle Akzeptanztests erfolgreich abgenommen wurden. Die Abnahme der Akzeptanztests erfolgt durch eine visuelle Prüfung.

13.1. Anforderungen mit User Stories

US-1 Es soll das Fenster in sechs Teile (Panels) aufgeteilt werden. Die Aufteilung soll in drei Spalten und zwei Zeilen aufgeteilt werden, wobei die erste Zeile eine Höhe von 150px[1] und die zweite Zeile eine höhe von 606px haben soll.

US-2 Es soll die erste und die dritte Spalte eine Breite von 220px und die zweite und mittlere Spalte eine Breite von 552px haben.

US-3 Es soll der Panel in der ersten Zeile und der ersten Spalte immer ein Logo der Applikation dargestellt werden. Wenn mit der linken Maustaste auf das Bild "geklickt" wird, soll die Startansicht geladen werden.

US-4 Es soll der Panel in der ersten Zeile und der zweiten Spalte der Hintergrund hellblau eingefärbt werden. In der linken unteren Ecke des Panels soll der Titel der aktuellen Ansicht angezeigt werden.

US-5 Es soll der Panel in der zweiten Zeile und der ersten Spalte als Navigationspanel dienen. Die verschiedenen Ansichten sollen über den Navigationspanel angesteuert werden können.

[1] px ist die Abkürzung für Pixel, zu Deutsch Bildpunkt. Ein Pixel bezeichnet die kleinste Einheit einer digitalen Rastergrafik.

US-6 Der Navigationspanel soll eine zweistufige Navigation anbieten. Ansichten, welche Gruppiert werden können, sollen in der zweiten Stufe untergebracht werden. Navigationen, welche sich in der zweiten Stufe befinden, sollen unterhalb der übergeordneten Navigationen eingeruckt dargestellt werden.

US-7 Die aktuell angesteuerte Ansicht soll im Navigationspanel andersfarbig dargestellt werden.

US-8 Im Navigationspanel sollen die verschiednen Navigationspunkte auf der ersten Stufe über ein Separator visuell voneinander getrennt werden.

US-9 Wenn ein Navigationspunkt der ersten Stufe im Navigationspanel "angeklickt" wird, sollen die darunter liegenden zweitstufigen Navigationspunkt sichtbar werden.

US-10 Es sollen zwei Navigationen auf der ersten Stufen angezeigt werden. Die Navigationen sind "Einführung" und "Produktauswahl".

US-11 Es sollen zwei Navigationen auf der zweiten Stufen unterhalb der Navigation "Produktauswahl" angezeigt werden. Die Navigationen sind "Produktbeschreibung" und "Produktdesign".

US-12 Wenn der Navigationspunkt "Einführung" ausgewählt wird, soll in der zweiten Reihe und der zweiten Spalte ein Einführungstext zum Proof of Concept dargestellt werden.

US-13 Wenn der Navigationspunkt "Produktauswahl" ausgewählt wird, soll in der zweiten Reihe und der zweiten Spalte eine Buttonmatrix mit den Produkten "Put Option", "Soft Runner" und "Protein" dargestellt werden. Die Navigationen "Produktbeschreibung" und "Produktdesign" sollen angezeigt werden und deaktiviert sein.

US-14 Wenn der Navigationspunkt "Produktauswahl" ausgewählt wird, sollen in der zweiten Zeile und der dritten Spalte Filteroptionen für die Produkte in der Form von CheckBoxen dargestellt werden.

US-15 Wenn eine Filteroption gewählt wird, sollen die Produkte entsprechend des Filters aktiv oder inaktiv gesetzt werden.

US-16 Wenn ein Produkt ausgewählt wird, dann soll in der zweiten Reihe und der zweiten Spalte die Produktbeschreibung angezeigt werden. Die Produktbeschreibung besteht aus einem Fliesstext und einem TabbedPanel welcher sechs Tabs hat: "Eigenschaften", "Chancen & Risiken", "Rückzahlungsmodus", "Beispiele", "Steuern" und "Klassifizierung". Die Navigationspunkte "Produktbeschreibung" und "Produktdesign" sollen nun aktiv gesetzt werden.

US-17 Wenn ein Tab ausgewählt wird, sollen die entsprechenden Informationen zum Produkt dargestellt werden.

US-18 Wenn der Navigationspunkt "Produktdesign" ausgewählt wird, soll das Auszahlungsprofil eines möglichen Produkts dargestellt werden. Die notwendigen Eckdaten sollen in der zweiten Zeile und der dritten Spalte in der Form von Slidern dargestellt werden.

US-19 Wenn die Werte in den Slidern geändert werden, dann soll sich das Auszahlungsprofil entsprechend anpassen.

13.2. Priorisierung der User Stories

Es wurden alle 19 User Stories als *hoch* priorisiert.

13.3. Akzeptanztests

Für alle Akzeptanztests gilt als Voraussetzung, dass der Prototyp auf einem Applikationsserver läuft. Die Akzeptanztest werden mit den Webbrowsern Chrome, Safari und Firefox durchgeführt, um sicherzustellen dass der Prototyp unabhängig vom Webbrowser implementiert wurde.

Firefox wird in der ZKB offiziell als Browser unterstützt, deshalb wurde er als Testkandidat mit einbezogen. Chrome und Safari sind die beiden Browsern welche in diesem Jahr am meisten Marktanteile dazugewonnen haben, siehe [BS11]. Aus diesem Grund wurden sie als Testkandidaten gewählt.

T-1.1 Die vorgegebenen Masseinheiten der Zeilen sollen mit einem Screenshot geprüft werden.

T-2.1 Die vorgegebenen Masseinheiten der Spalten sollen mit einem Screenshot geprüft werden.

T-3.1 Es soll geprüft werden, ob das Logo in der ersten Zeile und der ersten Spalte vorhanden ist.

T-3.2 Mit einem Klick auf das Logo soll die Startansicht geladen werden.

T-4.1 Es soll geprüft werden, ob der Panel in der ersten Zeile und der zweiten Spalte einen hellblau eingefärbten Hintergrund hat.

T-4.2 Durch das navigieren über verschiedene Menüpunkte, soll der Titel in der ersten Zeile und der zweiten Spalte, entsprechend der aktuellen Ansicht, angepasst werden.

T-5.1 Es soll geprüft werden, ob die Navigation in der zweiten Zeile und der ersten Spalte ersichtlich ist.

T-5.2 Durch das navigieren über verschiedene Menüpunkte, soll sich entsprechend die Ansicht ändern.

T-6.1 Es soll geprüft werden, ob gruppierte Navigationen unterhalb, der von ihnen übergeordnenten Navigation, eingeruckt dargestellt werden.

T-7.1 Es soll geprüft werden, ob die Farbe sich bei einer angesteuerten Navigation ändert.

T-8.1 Es soll geprüft werden, ob die Navigationspunkt auf der ersten Stufe über ein Separator voneinander getrennt sind.

T-8.2 Wenn ein Navigationspunkt der ersten Stufe angewählt wird, unter der sich Navigationspunkte auf der zweiten Stufe befinden, soll überprüft werden, dass kein Separator zwischen der ersten und der zweiten Stufe existiert.

T-9.1 Wenn ein Navigationspunkt der ersten Stufe angewählt wird, unter der sich Navigationspunkte auf der zweiten Stufe befinden, soll überprüft werden, ob diese angezeigt werden.

T-10.1 Die beiden Navigationspunkte “Einführung” und “Produktauswahl” sollen auf der ersten Stufe angezeigt werden.

T-11.1 Wenn der Navigationspunkt “Produktauswahl” angewählt wurde, sollen die beiden Navigationspunkt “Produktbeschreibung” und “Produktdesign” auf der zweiten Stufe angezeigt werden.

T-12.1 Wenn der Navigationspunkt “Einführung” angewählt wurde, soll in der zweiten Zeile und der zweiten Spalte ein Einführungstext zum Proof of Concept dargestellt werden.

T-13.1 Wenn der Navigationspunkt “Einführung” angewählt wurde, soll in der zweiten Reihe und der zweiten Spalte eine Buttonmatrix mit den möglichen Produkten angezeigt werden.

T-13.2 Die Navigationspunkte “Produktbeschreibung” und “Produktdesign” sollen deaktiviert sein.

T-14.1 Es soll geprüft werden, ob die Filteroptionen in der zweiten Reihe und der dritten Spalte dargestellt werden, wenn der Navigationspunkt “Produktauswahl” angewählt wurde.

T-14.2 Die Filteroptionen sollen in der Form von CheckBoxen angezeigt werden.

T-15.1 Es soll eine Filteroption ausgewählt werden, dabei sollen sich die Produkte, welche durch den Filter ausgeschlossen werden, deaktiviert werden.

T-15.2 Es soll eine Filteroption, welche ausgewählt war, wieder abgewählt werden. Dabei sollen sich die Produkte, welche durch den Filter ausgeschlossen waren, wieder aktiviert werden.

T-16.1 Es soll ein Produkt gewählt werden. Wenn das Produkt ausgewählt wurde, soll in der zweiten Reihe und der zweiten Spalte die Produktbeschreibung angezeigt werden.

T-16.2 Die Produktbeschreibung soll aus einem Fliesstext und einem TabbedPanel bestehen.

T-16.3 Wenn die Produktbeschreibung angezeigt wird, sollen sechs Tabs angezeigt werden: "Eigenschaften", "Chancen & Risiken", "Rückzahlungsmodus", "Beispiele", "Steuern" und "Klassifizierung".

T-16.4 Wenn ein Produkt ausgewählt wurde, sollen die Navigationspunkte "Produktbeschreibung" und "Produktdesign" aktiv gesetzt werden.

T-17.1 Wenn das Produkt "Put Option" ausgewählt wird, sollen die Informationen in den Tabs angezeigt werden.

T-18.1 Wenn das Produkt "Put Option" ausgewählt wird, und danach der Navigationspunkt "Produktdesign" angewählt wird, soll das Auszahlungsprofil einer Put Option angezeigt werden.

T-18.2 Wenn der Navigationspunkt "Produktdesign" angewählt wird, soll in der zweiten Zeile und der dritten Spalte der Strike und die Volatilität mit einem Slider angepasst werden können.

T-19.1 Wenn die Werte der Slider geändert werden, soll das Auszahlungsprofil entsprechend angepasst werden.

13.4. Testabdeckung

Die Testabdeckung ist in der Tabelle 13.1 ersichtlich und beträgt 100%. Somit gilt der Proof of Concept, gemäss dem Kapitel 5 (Methoden zur Entscheidungsfindung bei einer Evaluation, S. 20ff), als erfolgreich.

13.5. Resultat

In einem Proof of Concept konnte gezeigt werden, dass Vaadin den Anforderungen an ein Java Web Framework entspricht.

Browser	Version	Testfälle durchgeführt	Erfolgreich
Chrome	12.0.742.91	30	100%
Safari	5.0.5 (6533.21.1)	30	100%
Firefox	4.0.1	30	100%

Tabelle 13.1.: Testabdeckung der Testfälle für den Proof of Concept

Die Art und Weise wie man in Vaadin programmiert, entspricht ziemlich genau dem Programmierstil von Java Swing. Es werden äquivalente Typen von Komponenten und Layoutmanagern angeboten, die Kommunikation zwischen den Komponenten wird über das Observer Design Pattern implementiert. Zudem biete Vaadin eigene Themes an, welche beliebig erweitert werden können, ähnlich dem pluggable Look & Feel von Java Swing. Leider kommt man dabei nicht ganz ohne CSS aus, was den Erwartungen entsprochen hätte.

In den Abbildungen 13.1, 13.2 und 13.3 ist das Resultat in Form vom Screenshots ersichtlich. Es werden dabei drei verschiedene Ansichten mit erkannten Swingkomponenten gezeigt.

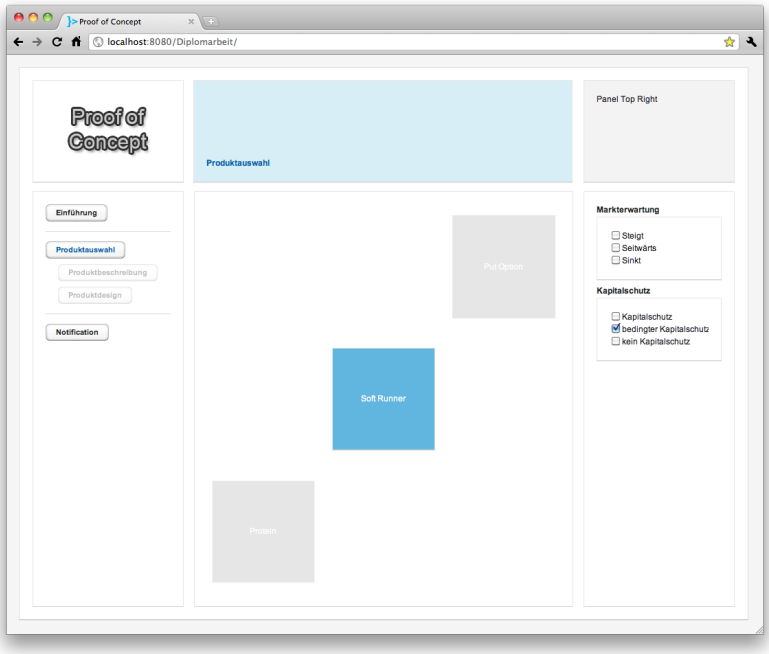

Abbildung 13.1.: Ansicht der Produkt in der Form einer Button Matrix.

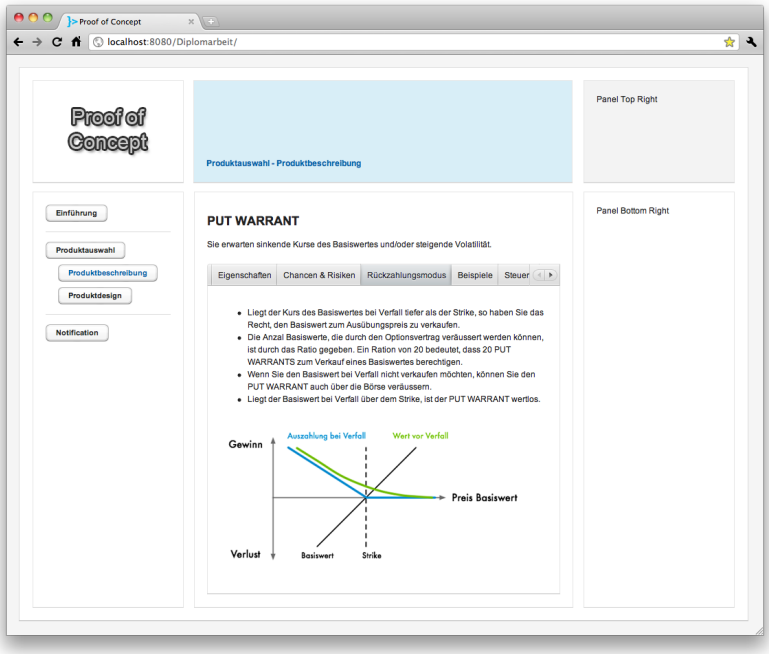

Abbildung 13.2.: Produktbeschreibung einer Put Option mit einem Tabbed Panel.

Der Sourcecode des Proof of Concept ist bei GitHub[2] öffentlich zugänglich unter der URL `https://github.com/sushicutta/Diplomarbeit`. Das Projekt ist für die IDE Eclipse[3] aufgesetzt. Eclipse ist unter der URL `http://www.eclipse.org/downloads/` kostenlos erhältlich. Für den Betrieb wird das Eclipse GlassFish[4] Plug-in benötigt, welches sich um das Deployment der Applikation und den Betrieb des Applikationensserver kümmert. Das Plug-in ist unter der URL `http://glassfishplugins.java.net/` kostenlos erhältlich.

[2]GitHub ist ein webbasierter Hosting-Dienst für Software-Entwicklungsprojekte.
[3]Eclipse ist ein quelloffenes Programmierwerkzeug zur Entwicklung von Software verschiedenster Art.
[4]GlassFish ist ein Open-Source-Projekt eines Java EE Servers von Oracle

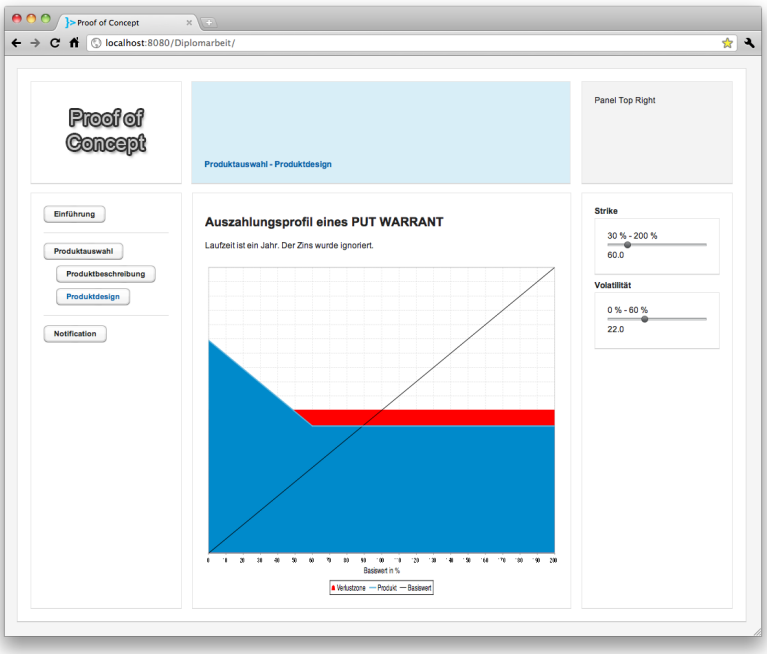

Abbildung 13.3.: Ansicht eines Auszahlungsprofils einer Put Option mit einem JFreeChart.

14. Empfehlung für ein Java Web Framework

Mit dem Abschluss der Evaluation wird nun eine Empfehlung für ein Java Web Framework ausgesprochen. Diese Empfehlung stützt sich auf die Resultate der Evaluation.

Die Empfehlung soll nicht verbindlich sein. Gegebenenfalls soll die Evaluation von der Zürcher Kantonalbank selber nochmals durchgeführt werden, da die Rahmenbedingungen der Evaluation im Rahmen der Diplomarbeit relativ eng gesteckt waren. So wurden nur vier Java Web Frameworks betrachtet und auch nur für eines davon ein Proof of Concept erstellt.

> Vaadin 6.6.0 wird als Java Web Framework zur Ablösung bestehender Java Swing Applikationen empfohlen.

Die Gründe dafür sind in der durchgeführten Evaluation ersichtlich. Das Framework entsprach keiner der definierten KO-Kriterien. Vaadin kam mit dem höchsten Nutzwert aus der gewichteten Nutzwertanalyse. Durch die niedrigen Anforderungen an die Infrastruktur, eignet sich das Framework für den Betrieb in der IT-Infrastruktur der Zürcher Kantonalbank. Mit einer Gesamtabdeckung von 94.87%, der gefundenen Java Swingkomponenten, belegt es ebenfalls den ersten Platz und ist mit Apache Wicket auf einer Augenhöhe. Zu guter Letzt wurde mit dem Proof of Concept gezeigt, das Applikationen im ähnlichen Stil wie mit Java Swing Applikationen entwickelt werden können.

15. Reflexion

In diesem Kapitel wird ein Fazit über die angewandten Methoden und Techniken gezogen. Anschliessend wird ein Rückblick auf die Durchführung der Diplomarbeit gemacht. In einem Ausblick möchte ich auf einen möglichen Einsatz dieser Evaluation eingehen. Zum Schluss bedanke ich mich mit der Danksagung bei den Personen, welche mich unterstützt haben.

15.1. Fazit

Es gibt einige Punkte welche ich bezüglich der gewählten Methoden, und der daraus resultierenden Ergebnisse, ansprechen möchte. Diese Punkte offenbarten sich während der Durchführung der Diplomarbeit. Sie werden hier in der Form von offnen Fragen gestellt, welche ich versuche zu beantworten.

Wieso die gewichtete Nutzwertanalyse und der Analytic Hierarchy Process?

Die gewichtete Nutzwertanalyse ist eine prädestinierte Methode zur Entscheidungsfindung bei einer Evaluation. Sie ist einfach verständlich und umsetzbar. Natürlich gibt es weitere Methoden, wie zum Beispiel die SWOT-Analyse[1], welche dafür auch verwendet werden könnte.

Der Analytic Hierarchy Process habe ich gewählt, weil er mathematisch begründet ist, was zum Einsatzgebiet eines Ingenieurs passt. Zudem kann diese Methode für die Entscheidungsfindung jeglicher Fragen in allen Lebenslagen dienen, auf was ich in Zukunft sicher wieder zurückgreifen werde.

[1] Die SWOT-Analyse wird im Bereich der Betriebswirtschaft häufig übersetzt mit „Analyse der Stärken, Schwächen, Chancen und Risiken“, siehe [Hö01]. SWOT ist ein Werkzeug des strategischen Managements.

Wieso nur drei Swing Applikationen?

Die Antwort ist relativ simpel. Da es sich bei den untersuchten Swing Applikationen um verschiedene Business Lösungen gehandelt hat, sollten die meisten gängigen GUI Komponenten und Konzepte enthalten sein. Mit der Zunahme der zu untersuchenden Applikationen nimmt die Anzahl neu gefundener Komponenten und Paradigmen wahrscheinlich rapide ab.

Wieso nur vier Java Web Frameworks?

Um eine Evaluation stichhaltiger zu machen, wäre es sinnvoll möglichst viele Java Web Frameworks mit einzubeziehen. Durch den zeitlichen Rahmen der Diplomarbeit war es leider nicht möglich mehr Alternativen zu untersuchen.

Wieso die 18 Anforderungen gemäss AgileLearn als Soll-Kriterien?

Anforderungen an Web Frameworks variieren immer aus der Sicht des Betrachters. Aus diesem Grund habe ich ein Anforderungskatalog gesucht, welcher unabhängig von der aktuellen Situation und vom Auftraggeber existiert. Mit AgileLearn habe ich genau das gefunden was ich gesucht habe. Die Projektgruppe kommt aus der HTW Berlin heraus, was den notwenigen akademischen Hintergrund bietet, und kommuniziert im Stil von der heutigen Internet-Generation, via Blog. Somit waren das, im Bezug auf das Thema Java Web Frameworks und der wissenschaftlichen Arbeit als Diplomarbeit, gleich zwei Fliegen auf eine Klappe.

Wieso Sicherheit, Kosten der Maintenance, Ressourcen und Image als Konsolidierung der Soll-Kriterien?

Die Konsolidierung und Priorisierung der gegebenen Soll-Kriterien im Sinne der Zürcher Kantonalbank ist einer der Schwachpunkte der Diplomarbeit. Die Begründungen, wieso diese Aspekte gewählt wurden, sind nicht unbedingt stichfest. Ebenfalls die Vergabe der Punkte pro Soll-Kriterium und die daraus resultierende Priorisierung.

Um eine, der Realität nähere und präzisere, Konsolidierung und Priorisierung der Soll-Kriterien zu erhalten, hätte eine Umfrage, bei den betreffenden Stellen und Ansprechspersonen innerhalb der Zürcher Kantonalbank, dazu durchgeführt werden müssen. Durch die, von mir, in der Aufgabenstellung verfasste Abgrenzung von jeglichen Umfragen, habe ich auf das verzichtet. Als Folge davon habe ich in Treu und Glauben versucht das selber zu erarbeiten, was mit einem fragwürdigen Resultat endet.

Dennoch darf die Evaluation an diesem Punkt nicht komplett aufgehängt werden. Die Aspekte Sicherheit, Kosten der Maintenance, Ressourcen und das Image gehören sicherlich auch

zu den Kernkompetenzen einer Bank. Die Frage bleibt offen, welche Kriterien es noch gibt, und wie sie von der Zürcher Kantonalbank bevorzugt würden.

Die Evaluation hätte auch anders kommen können - Vergabe der Punkte in der Nutzwertanalyse, für die einzelnen Kriterien über einen Kriterien-Bewertungskatalog.

Ein weiterer Schwachpunkt in der Diplomarbeit ist die Vergabe der Punkte während der Durchführung der gewichteten Nutzwertanalyse. Es wurden sechs Soll-Kriterien ausgearbeitet, welche für jede Alternative bewertet werden soll. Die Punkte wurden nach meinem Ermessen möglichst fair vergeben, jedoch ist die Vergabe für aussenstehende nicht komplett nachvollziehbar.

Eine sinnvolle Methode für eine faire, objektive und nachvollziehbare Bewertung der Soll-Kriterien wäre ein vordefinierter Kriterien-Bewertungskatalog mit messbaren Bewertungskriterien. Für jedes der Soll-Kriterien gäbe es einen eigenen Katalog. Jedes Bewertungskriterium würde eine Anzahl Punkte ergeben, wenn es in der Alternative erfüllt wäre. Es würden in allen Alternativen die selben Kriterien geprüft und bewertet werden, was sicherlich zu einem präziseren und strukturierteren Resultat führen würde.

Das ausarbeiten eines solchen Kataloges hätte sicherlich viel Zeit in Anspruch genommen. Der Faktor Zeit ist während der Diplomarbeit leider begrenzt. Trotzdem hätte ich das auf mich genommen. Leider ist mir die Idee dazu erst nach Abschluss der Evaluation gekommen.

Proof of Concept mit nur einem Java Web Framework.

Nach meinen Erfahrungen werden in der Realität jeweils mindestens zwei Proof of Concepts durchgeführt. Zumindest beim Einkauf von Softwareprodukten, gibt man sich mit dem theoretisch besten nie zufrieden, sondern man holt sich noch wenigstens eine Offerte einer zusätzlich möglichen Alternative ein. Hier ist das leider wieder die Zeit, welche das nicht zugelassen hat. Ich bedaure aber sehr, dass ich den Proof of Concept nicht auch mit Apache Wicket machen konnte, da das Resultat der Evaluation relativ knapp war.

Wieso sind keine Lasttests gemacht worden, respektive die Skalierung ist nicht betrachtet worden.

In der Realität wäre das ein wichtiger Bestandteil bei der Abnahme eines Proof of Concepts. In der Dokumentation eines Web Frameworks wird einem immer viel versprochen. Vertrauen ist gut, Kontrolle ist besser!

Um sinnvolle Lasttest zu machen muss einem das Mengengerüst einer zukünftigen Applikation bekannt sein. Da es im Rahmen der Diplomarbeit nicht um die Evaluation für die Ablösung einer bestimmt Applikation gegangen ist, fehlten diese Informationen. Ich hätte natürlich anstatt Lasttest einfach Performance Messungen durchführen können, dies konnte ich leider mangels Infrastruktur nicht bewerkstelligen.

Wenn die Infrastruktur vorhanden gewesen wäre, hätte man mithilfe von zum Beispiel Selenium[2] den Proof of Concept mit einer grossen Anzahl von Browserinstanzen unter Druck setzten können. Dabei wären sicher interessante und nützliche Resultate zum Vorschein gekommen. Richtig nützlich wären diese Ergebnisse dann erst, wenn zwei Proof of Concepts zur Verfügung gestanden hätten, um einen Vergleich anzustellen.

15.2. Rückblick

Rückblickend auf die Diplomarbeit entspricht die Durchführung ziemlich der grob erstellten Planung im Anhang D (Projektadministration, S. 100ff). Einzig der administrative Aufwand wurde viel zu tief geschätzt und hat mehr Zeit beansprucht als angenommen.

Grosse Stolpersteine sind während der Durchführung keine aufgetaucht. Zudem entsprechen die erarbeiteten Resultate ungefähr den Erwartungen.

Die Diplomarbeit war in meinen Auge ein voller Erfolg, da ich alle Ziele erreicht und viel gelernt habe. Ich bin froh das Studium hinter mir zu lassen und bereit für neue Herausforderungen.

15.3. Ausblick

Im Laufe der Diplomarbeit wurde ich zum Thema Java Web Frameworks, mit Schwerpunkt auf Präsentations-Logik, innerhalb der Zürcher Kantonalbank kontaktiert. Dabei handelt es sich um ein Projekt, welches aktuell in den Startlöchern steht und im Handel und Kapitalmarkt angesiedelt ist. Es ging darum, ob es mögliche Alternativen zu den bereits eingesetzten Java Web Frameworks gibt, welche ich empfehlen könnte. Ich habe dabei auf die vier untersuchten Lösungen hingewiesen und im Nachhinein erfahren, dass Vaadin als ernsthafte Alternative für das Projekt evaluiert wird. Ob die Wahl auf Vaadin fallen wird, steht zu diesem Zeitpunkt noch in den Sternen.

[2] Bei Selenium handelt es sich um ein Testframework für Webanwendungen. Es wurde von einem Programmierteam der Firma ThoughtWorks entwickelt und als Freie Software unter der Apache-2.0-Lizenz veröffentlicht.

Auf alle Fälle ist das letzte Wort in dieser Sache noch nicht gesprochen. Wie lange sich Java als Plattform für Businesslösungen am Markt halten kann, weiss niemand. In der Informatik Welt ist bekanntlich alles ein bisschen schnelllebiger. Dasselbe gilt für die zur Verfügung stehenden Web Frameworks. Ich wette, in fünf Jahren würde ich ganz andere zu evaluierenden Alternativen wählen, weil diese moderner sein werden und mehr dem aktuellen Trend entsprechen würden.

Was bleibt ist das Verfahren, wie eine Evaluation durchgeführt werden kann. In fünf Jahren würde ich es wohl wieder auf die selbe Art und Weise realisieren.

15.4. Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich während meiner Studienzeit unterstützt haben. Der grösste Dank gilt meiner Freundin und unserem gemeinsamen Sohn Linus, die mich jederzeit bedingungslos unterstützt haben. Auch möchte ich mich bei meinen Eltern bedankten, welche mich ebenfalls, im Bezug auf das Studium, gefördert haben.

Ich möchte mich in dieser Form bei Beat Seeliger bedanken, der mich als Betreuer bei meiner Diplomarbeit unterstützte und mir mit seiner hilfsbereiten und unkomplizierten Art und Weise zur Seite gestanden ist.

Ebenfalls bedanken möchte ich mich bei Berhard Mäder von der Zürcher Kantonalbank, der mir die Arbeit ermöglicht hat und dessen Türen für mich während der Durchführung immer offen standen.

Weiterhin bedanke ich mich bei meinem Arbeitgeber und Freund Silvan Spross, der mir genügend Zeit für die Vollendung der Diplomarbeit gewährte und mit der notwendigen Infrastruktur der allink GmbH versorgt hat, damit ich einen ruhigen Platz für die Durchführung der Arbeit hatte.

Silvan Spross und Stefan Laubenberger möchte ich auch, da sie die Arbeit gegengelesen haben.

Bei Bernhard Böhm möchte ich für den Aufwand bedanken, den er betrieben hat, um die Arbeit auf mögliche Fehler zu überprüfen.

Zum Schluss bedanke ich mich auch bei Stefan Pudig und Marco Spörri die mich mit wichtigen Informationen, zu ZKB spezifischen Themen, unterstützt haben.

A. Personalienblatt

Name, Vorname:	**Roman Würsch**
Adresse:	**Murhaldenweg 16**
PLZ, Wohnort:	**8057 Zürich**
Geburtsdatum:	**10.11.1980**
Heimatort:	**Emmetten NW**

Ich bestätige, dass die vorliegende Diplomarbeit, “Evaluation eines Java Web Frameworks zur Ablösung bestehender Java Swing Applikationen”, in allen Teilen selbständig erarbeitet und durchgeführt wurde.

Ort und Datum Unterschrift

B. Rahmenbedingungen

Für das Informatik Diplomstudium an der Hochschule für Technik Zürich (HSZ-T) wird von den Studenten verlangt eine Diplomarbeit eigenständig zu verfassen. Mit der Diplomarbeit wird das Studium zum Informatik Ingenieur FH abgeschlossen.

B.1. Sprache

Der Bericht wird in deutscher Sprache verfasst. Englische Ausdrücke werden im Kontext verwendet, wenn man davon ausgehen kann, dass es gängige Ausdrücke aus dem Gebiet der Informatik sind. Um Schwerfälligkeiten im Text zu vermeiden, wurde nur die männliche Form verwendet. Selbstverständlich sind bei allen Formulierungen beide Geschlechter angesprochen.

B.2. Richtlinien

Folgende Dokumente mit Richtlinien der HSZ-T wurden für die Diplomarbeit berücksichtigt:

- Bestimmungen für die Diplomarbeit [JGG06]
- Ablauf Diplomarbeit [DOS09a]
- Bewertungskriterien für die Bachelor Arbeit [DOS09b]
- Richtlinie Poster zur Bachelorarbeit [fTZ09]

B.3. Bewertungskriterien

Es werden die Bewertungskriterien für die Bachelor Arbeit verwendet[1], siehe [DOS09b].

[1] Gemäss Beschluss am Kick-off Meeting, siehe Kapitel G Kick-off Protokoll auf S. 120

C. Aufgabenstellung

Die Aufgabenstellung besteht aus den vier Teilen: Ausgangslage, Ziel der Arbeit, Aufgabenstellung und Erwartete Resultate. Zusätzlich wurde eine Abgrenzung hinzugefügt, damit der Rahmen der Diplomarbeit gesetzt ist.

C.1. Ausgangslage

Die Zürcher Kantonalbank hat viele Applikationen welche als Client Applikationen in Swing implementiert sind. Dabei ist das Deployment der Clients und die Ausbreitung entsprechender Patches, innerhalb der Zürcher Kantonalbank, mit einem Mehraufwand verbunden. Client Applikationen, die einem Kunden zur Verfügung gestellt werden, sind an eine spezifische Java Version gebunden. Bei der Integration in die IT-Landschaft beim Kunden, kann das zur Verzögerung durch einen erhöhten Testaufwand führen. In der Annahme, dass ein Webbrowser in der Zürcher Kantonalbank und bei deren Kunden eingesetzt wird, macht der Einsatz einer Weblösung Sinn.

C.2. Ziel der Arbeit

Es sollen bestehende Java Swing Applikationen der Zürcher Kantonalbank analysiert werden. In diesen Applikationen sollen die gemeinsamen Muster, genutzter Swingkomponenten, erkannt und kategorisiert werden. Java Web Frameworks, welche sich am Markt etabliert haben, sollen auf einen möglichen Einsatz geprüft werden. Es soll geprüft werden, ob mit den jeweiligen Frameworks die genutzten Swingkomponenten äquivalent umgesetzt werden können. Zudem soll geprüft werden, ob eine Integration in die bestehende IT Infrastruktur der Zürcher Kantonalbank möglich ist.

C.3. Aufgabenstellung

Folgende Aufgaben sollen vom Studierenden während der Diplomarbeit durchgeführt werden:

- Analyse bestehender Java Swing Applikationen der Zürcher Kantonalbank.
- Erkennen und Kategorisieren der verwendeten Swingkomponenten.
- Evaluation von Java Web Frameworks, welche sich am Markt etabliert haben.
- Prüfen, ob eine Integration der evaluierten Java Web Frameworks, welche für eine Umsetzung geeignet sind, in der bestehenden IT Infrastruktur der Zürcher Kantonalbank möglich ist.
- Prüfen, ob eine Implementierung der erkannten Swingkomponenten in den evaluierten Java Web Frameworks möglich ist.
- Proof of concept. Erstellen eines Prototypen mit den evaluierten Java Web Frameworks und den erkannten Swingkomponenten.

C.4. Erwartete Resultate

Der Studierende soll dem Auftraggeber ein Dokument erstellen, das folgendes beinhaltet:

- Ergebnis der Analyse von bestehenden Java Swing Applikationen der Zürcher Kantonalbank.
- Kategorisierung von verwendeten Swingkomponenten.
- Ergebnisse der Evaluation von etablierten Java Web Frameworks.
- Ergebnis der Analyse, ob eine Integration der Java Web Frameworks, in der bestehenden IT Infrastruktur der Zürcher Kantonalbank, möglich ist.
- Ergebnis der Analyse von Java Web Frameworks, ob eine Implementierung, der erkannten Swingkomponenten, möglich ist.
- Proof of concept. Es soll anhand eines Prototypen gezeigt werden, dass die Implementierung möglich ist.
- Eine Empfehlung für ein Java Web Framework.

C.5. Abgrenzung

Folgende Punkte werden formell abgegrenzt:

- Die Analysen beschränken sich auf Recherchen im Internet, Büchern und interne Vorgaben der Zürcher Kantonalbank.
- Umfragen, Erhebungen sowie Feldstudien werden nicht durchgeführt.

Folgende Punkte werden inhaltlich abgegrenzt:

- Die Auswahl, welche Java Swing Applikationen analysiert werden, soll wärend der Arbeit durchgeführt werden.
- Die Auswahl, welche Java Web Frameworks geprüft werden, soll wärend der Arbeit durchgeführt werden.

D. Projektadministration

Im Kapitel Projektadministration wird eine Detailanalyse der Aufgabenstellung durchgeführt. Daraus abgeleitet wird die Planung definiert. Zusätzlich gibt es Informationen über die einzelnen Arbeitsschritte und den Erreichungsgrad der gestellten Aufgaben.

D.1. Detailanalyse der Aufgabenstellung

In der Detailanalyse der Aufgabenstellung werden die definierten Aufgaben der Aufgabenstellung untersucht und deren Resultate ausgearbeitet. Aufgrund dieser Erkenntnisse wird eine Grobplanung festgelegt.

D.1.1. Aufgabe - Au1

Analyse bestehender Java Swing Applikationen der Zürcher Kantonalbank.

Im Detail Es sollen die gängigen Mechanismen von Java Swing Applikationen der Zürcher Kantonalbank untersucht werden. Das soll aus der Sicht des Anwenders passieren. In drei bestehenden Java Swing Applikationen soll die Existenz bekannter GUI Paradigmen untersucht werden.

Resultat - R1 Es soll eine Liste der erkannten Paradigmen vorliegen.

D.1.2. Aufgabe - Au2

Erkennen und Kategorisieren der verwendeten Swingkomponenten.

Im Detail Die drei ausgewählten Java Swing Applikationen werden genauer betrachtet. Es wird das Augenmerk auf die Verwendung von Swingkomponenten gelegt. Diese sollen über die drei Applikationen hinweg konsolidiert und kategorisiert werden.

Resultat - R2 Es soll eine Liste der verwendeten Swingkomponenten vorliegen.

D.1.3. Aufgabe - Au3

Evaluation von Java Web Frameworks, welche sich am Markt etabliert haben.

Im Detail Es soll ein Evaluationsverfahren gewählt werden, bei dem eine möglichst objektive Entscheidung gefällt werden kann. Welche Java Web Frameworks in die Evaluation mit einbezogen werden, soll über Recherchen im Internet und in Büchern geschehen. Es sollen vier Java Web Frameworks gewählt werden, welche anhand der Recherchen für valable Optionen in Frage kommen. Über die Definition von Soll- und KO-Kriterien sollen die Rahmenbedingungen für das Evaluationsverfahren geschaffen werden. Anhand der ausgearbeiteten Evalautionsmethode soll gezeigt werden, wie geeignet die Java Web Frameworks wirklich sind.

Resultat - R3 Es soll eine Rangliste der vier Frameworks, in der Anordnung entsprechend ihrer Eignung, vorliegen.

D.1.4. Aufgabe - Au4

Prüfen, ob eine Integration der evaluierten Java Web Frameworks, welche für eine Umsetzung geeignet sind, in der bestehenden IT Infrastruktur der Zürcher Kantonalbank möglich ist.

Im Detail Gemäss den Vorgaben der IT Infrastruktur der Zürcher Kantonalbank, soll ein mögliche Einsatz der evaluierten Java Web Frameworks geprüft werden. Die meisten Java Web Frameworks haben in ihrer Dokumentation die Anforderungen definiert, welche für einen möglichen Betrieb nötig sind. Aufgrund dieser Anforderungen, und der bestehenden IT Infrastruktur soll ein Vergleich gemacht werden.

Resultat - R4 Es sollen die Java Web Frameworks aufgelistet werden, welche für einen Einsatz in der IT Infrastruktur der Zürcher Kantonalbank in Frage kommen.

D.1.5. Aufgabe - Au5

Prüfen, ob eine Implementierung der erkannten Swingkomponenten in den evaluierten Java Web Frameworks möglich ist.

Im Detail Die gewonnenen Erkenntnisse, aus der Analyse der Java Swing Applikationen, sollen nun mit der Liste, der in Frage kommenden Java Web Frameworks, zusammengeführt werden.

Resultat - R5 Es sollen die Java Web Frameworks aufgelistet werden, welche die notwendigen Swingkomponenten und GUI Paradigmen unterstützen.

D.1.6. Aufgabe - Au6

Proof of concept. Erstellen eines Prototypen mit den evaluierten Java Web Frameworks und den erkannten Swingkomponenten.

Im Detail Das Java Web Framework, welches sich entsprechend der Evaluation am meisten für den Einsatz eignet und den Anforderungen der IT Infrastruktur und der notwendigen Swingkomponenten und GUI Paradigmen genügt, soll sich anhand eines definierten Prototypen bewähren.

Resultat - R6 Ein lauffähiger Prototyp.

Resultat - R7 Es soll eine Empfehlung eines Java Web Frameworks, für den möglichen Einsatz in der Zürcher Kantonalbank, ausgesprochen werden.

D.2. Planung

Die Grobplanung soll den chronologischen Ablauf der Diplomarbeit aufzeigen. Zudem sollen die einzelnen Arbeitspakete aufgeteilt und nach deren Aufwand geschätzt werden. Für eine Konsolidierung in der Reflexion, soll der effektiv geleistete Aufwand aufgezeigt werden.

D.2.1. Grobplanung

In der Abbildung D.1 auf Seite 103 sieht man den chronologischen Ablauf der Diplomarbeit wie er beim Kick-off Meeting vorgestellt wurde.

D.2.2. Aufwandschätzung

Die Aufwandschätzung soll mit realistischen Zeiten auf halbe Stunden genau gemacht werden. Zudem soll die effektiv gebraucht Zeit im Laufe der Diplomarbeit ergänzt werden. Der gesamte Aufwand wird in fünf Gruppen unterteilt und ist in der Tabelle D.1 ersichtlich.

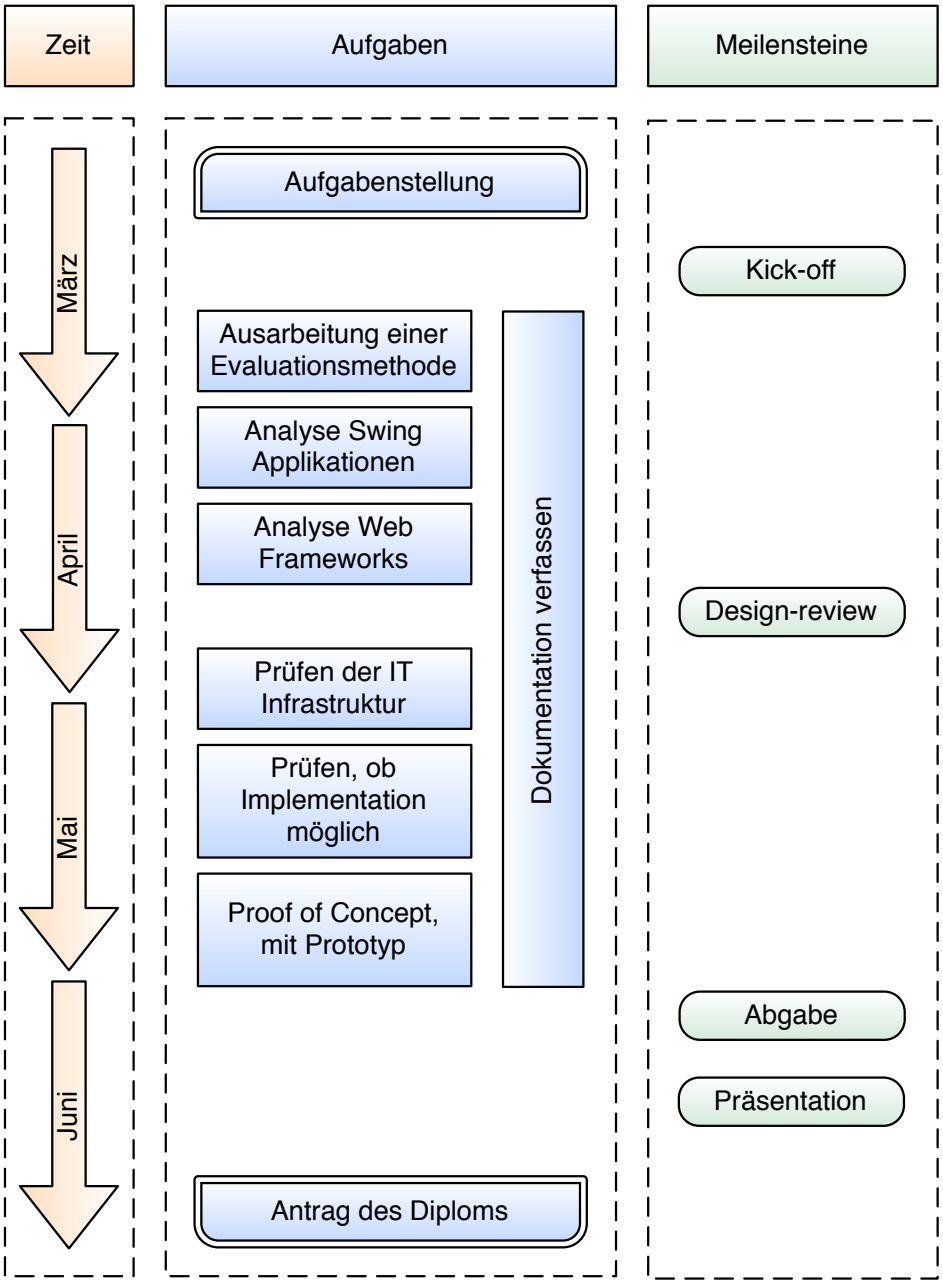

Abbildung D.1.: Chronologischer Ablauf der Diplomarbeit

Arbeitspaket	Geplant	Effektiv
Präsentationen	20.0 h	20.75 h
Gestellte Aufgaben gemäss Aufgabenstellung	144.0 h	137.75 h
Abzugebende Dokumente	94.0 h	87.25 h
Fachbetreuung durch den Dozenten	10.0 h	9.75 h h
Administrative Aufgaben	13.5 h	29.5 h
Total	281.5 h	285.0 h

Tabelle D.1.: Aufwandschätzung der Diplomarbeit

Präsentationen

Gemäss [Des10] Slide 37f. braucht man für eine Stunde Präsentationszeit mindestens eine Vorbereitungszeit von 30 Stunden. Somit kann man das linear herunter brechen auf eine Stunde Vorbereitungszeit pro zwei Minuten Präsentationszeit. Die Planung ist in der Tabelle D.2 ersichtlich.

Arbeitspaket	Geplant	Effektiv
Kick-off Präsentation ca. 5 Minuten	2.5 h	4.5 h
Design-review Präsentation ca. 10 Minuten	5.0 h	3.75 h
Schlusspräsentation ca. 20 Minuten	10.0 h	10.0 h
Prototyp Demo ca. 5 Minuten	2.5 h	2.5 h
Total	20.0 h	20.75 h

Tabelle D.2.: Geplante Zeit für Präsentationen

Gestellte Aufgaben gemäss Aufgabenstellung

Die einzelnen Aufgaben sollen in deren Teilaufgaben unterteilt werden. Die Planung ist in der Tabelle D.4 auf Seite 106 ersichtlich.

Abzugebende Dokumente

Gemäss den Bestimmungen für die Diplomarbeit, siehe [JGG06] S. 3, müssen einige Dokumente erstellt und abgegeben werden. Die Planung ist in der Tabelle D.3 ersichtlich.

Arbeitspaket	Geplant	Effektiv
Bericht in LaTeX aufsetzen	10.0 h	9.5 h
Bericht in LaTeX verfassen	60.0 h	65.75 h
Poster für die Diplomausstellung im A0 Format	16.0 h	12.0 h
Zusammenfassung à zwei A4-Seiten	8.0 h	0.0 h
Total	94.0 h	87.25 h

Tabelle D.3.: Geplante Zeit für Dokumentation

Fachbetreuung durch den Dozenten

Gemäss den Bestimmungen für die Diplomarbeit, siehe [JGG06] S. 2, stehen dem Diplomand zehn Stunden Betreuungszeit durch den Dozenten zur Verfügung. Die Planung ist in der Tabelle D.5 ersichtlich.

Administrative Aufgaben

Unter administrative Aufgaben fallen Tätigkeiten wie die Planung von Terminen, das Druckenlassen der Dokumentation, die Kommunikation mit der Schulleitung und dem Dozenten, usw. Auf Grund meiner Erfahrung mit Projekten, macht das in etwa fünf Prozent des gesamten Aufwandes aus.

D.3. Arbeitsschritte

Alle vorgenommenen Arbeitsschritte werden in einem Wiki für die Nachvollziehbarkeit protokolliert. Das Wiki ist im Internet öffentlich zugänglich unter der URL:

Aufgabe	Arbeitspaket	Geplant	Effektiv
Au1	Research zu Analyseverfahren von Java Swing Applikationen	8.0 h	7.0 h
	Definition des Analyseverfahrens	4.0 h	9.25 h
	Analyse von drei Java Swing Applikationen, à 4 Stunden	12.0 h	18.5 h
Au2	Research zu Java Swing Komponenten	4.0 h	4.0 h
	Analyse von drei Java Swing Applikationen, à 2 Stunden	6.0 h	8.75 h
Au3	Research zu Evaluationsverfahren im Bereich von Java Web Frameworks	12.0 h	9.75 h
	Definition des Evaluationsverfahrens	8.0 h	12.25 h
	Evaluation von vier Java Web Frameworks, à 5 Stunden	20.0 h	21.0 h
Au4	Analyse der IT Architektur der ZKB	8.0 h	4.5 h
	Prüfen ob die Integration möglich ist bei vier Java Web Frameworks, à 2.5 Stunden	10.0 h	6.0 h
Au5	Analyse der Komponenten von vier Java Web Frameworks und prüfen ob die Implementation möglich ist, à 3 Stunden	12.0 h	7.5 h
Au6	Anforderungen an den Prototyp definieren	8.0 h	5.5 h
	Testfälle für den Prototyp definieren	8.0 h	4.5 h
	Umsetzung des Prototypen	20.0 h	16.75 h
	Empfehlung eines Java Web Frameworks	4.0 h	2.5 h
	Total	144.0 h	137.75 h

Tabelle D.4.: Geplante Zeit für gestellte Aufgaben

Arbeitspaket	Geplant	Effektiv
Kick-off Meeting	1.0 h	2.0 h
Design-review Meeting	1.0 h	2.75 h
Schlusspräsentation	1.0 h	1.0 h
Ausserterminliche Betreuungszeit	7.0 h	4.0 h
Total	10.0 h	9.75 h

Tabelle D.5.: Geplante Zeit für Betreuung

Arbeitspaket	Geplant	Effektiv
Planung von Terminen	2.0 h	8.5 h
Druckenlassen der Dokumentation	4.0 h	4.0 h
Kommunikation mit der Schulleitung und dem Dozenten	2.5 h	2.25 h
Übrige administrative Aufgaben	5.0 h	14.75 h
Total	13.5 h	29.5 h

Tabelle D.6.: Geplante Zeit für administrative Aufgaben

https://github.com/sushicutta/Diplomarbeit/wiki/Arbeitsprotokoll

Zusätzliche Informationen zum Ablauf der Diplomarbeit werden ebenfalls im Wiki erfasst und sind unter der URL ersichtlich:

https://github.com/sushicutta/Diplomarbeit/wiki/

D.4. Meilensteine

Die Meilensteine entsprechen dem vorgegebenen Ablauf einer Diplomarbeit aus dem Einschreibe- und Bewertungssystem der HSZ-T (EBS)[1]. Die Projekt Termine wurden alle gemäss Reglement eingehalten, siehe Tabelle D.7.

[1] https://ebs.hsz-t.ch/

Datum	Meilenstein	Ort
14. März 2011	Die Aufgabenstellung zur Diplomarbeit wurde eingereicht	EBS
15. März 2011	Freigabe der Diplomarbeit	EBS
21. März 2011	Inhaltliches Kick-off Meeting	Panter llc
13. April 2011	Offizielles Kick-off Meeting	HSZ-T
04. Mai 2011	Design-Review Meeting	HSZ-T
15. Juni 2011	Abgabe der Dokumentation	HSZ-T
29. Juni. 2011	Schlusspräsentation	HSZ-T

Tabelle D.7.: Projekt Meilensteine

D.5. Erreichte Ziele

Es wurden alle Ziele gemäss den erwarteten Resultaten der Aufgabenstellung erreicht. Die einzelnen Punkte sind hier aufgeführt, siehe Tabelle D.8 auf Seite 109.

Resultat	Ziel	Stand
Resultat - R1	Es soll eine Liste der erkannten GUI Paradigmen vorliegen.	erreicht
Resultat - R2	Es soll eine Liste der verwendeten Swingkomponenten vorliegen.	erreicht
Resultat - R3	Es soll eine Rangliste der vier Java Web Frameworks, in der Anordnung entsprechend ihrer Eignung, vorliegen.	erreicht
Resultat - R4	Es sollen die Java Web Frameworks aufgelistet werden, welche für einen Einsatz in der IT Infrastruktur der Zürcher Kantonalbank in Frage kommen.	erreicht
Resultat - R5	Es sollen die Java Web Frameworks aufgelistet werden, welche die notwendigen Swingkomponenten und GUI Paradigmen unterstützen.	erreicht
Resultat - R6	Ein lauffähiger Prototyp.	erreicht
Resultat - R7	Es soll eine Empfehlung eines Java Web Frameworks, für den möglichen Einsatz in der Zürcher Kantonalbank, ausgesprochen werden.	erreicht

Tabelle D.8.: Übersicht der erreichten Ziele

E. 18 Anforderungen an Web Frameworks nach AgileLearn

Folgende Anforderungen stammen aus dem Dokument "*18 Anforderungen an Webframeworks - OpenDoc*", siehe [ap11], und wurden zusammengefasst. Die Anforderungen sind hier aufgelistet und mit einer ID in der Form {Soll} - {Laufnummer} versehen, da die Anforderungen als Soll-Kriterien für die Evaluation der Java Web Frameworks dienen.

Soll-01 - Zugriffskontrolle Ein Webframework sollte EntwicklerInnen verschiedene Mechanismen bereitstellen, um die Anwendung vor fremden und unerlaubten Zugriff schützen zu können.

Authentifizierung/Autorisierung: Üblicherweise werden im Vorfeld Rollen für verschiedenen Gruppen festgelegt. Das Ziel einer sicheren Webanwendung ist es, bestimmte Bereiche einer Seite abzusichern und die Rechte aller Benutzer je nach Rolle einzuschränken. Anhand der Rolle wird deren Benutzer für die festgelegten Bereiche autorisiert.

Vertraulichkeit / Verschlüsselung: Sensible Daten, wie Passwörter und Personendaten, müssen vor dem Zugriff und der Kenntnisnahme von Dritten geschützt werden. Hierfür werden die Daten während der Übertragung verschlüsselt. Für eine sichere Datenübertragung werden meist Verschlüsselungsprotokolle wie SSL (Secure Sockets Layer), sowie dessen Nachfolger TLS (Transport Layer Security) eingesetzt. Diese gelten als relativ sicher und sind bei Transaktionen bei einer Bank unverzichtbar.

Soll-02 - Form-Validierung Das Verarbeiten von Formularen bzw. die Handhabung von Benutzereingaben und -aktionen gehört zu den täglichen Aufgaben der Webentwicklung. Die Logik für server- und clientseitiges Validieren ist im Idealfall nur einmal implementiert.

Server-Side Validation: Das Webframework soll die Mögichkeit bieten, eingegebene Daten einfach zu überprüfen. Dabei soll für jede Eigenschaft eines Datenmodells (Model) ein Wertebereich definierbar sein, zusätzlich soll geprüft werden, ob die Eingabe erforderlich ist. Die Programmierlogik soll minimal sein. Nach dem Senden der Daten wird alles überprüft und ggf. entsprechende Fehlermeldungen zurückgegeben.

Client-Side Validation: Das Webframework soll wenn möglich schon auf dem Client validieren. Wenn möglich, sollen die Daten gleich bei oder kurz nach der Eingabe überprüft werden, wobei die Logik auf dem Server implementiert ist. Beispiel: Ist der Anmeldename schon vergeben. Dadurch werden Serverressourcen gespart und der Benutzer hat ein direktes Feedback.

Soll-03 - Modulare Architektur Eine Webanwendung stellt ein Zusammenspiel verschiedenster Internettechnologien dar, die wiederum hinsichtlich ihrer Entwicklung einem stetigen Wandel unterliegen. Die Herausforderung für die Entwickler ist es, die Entwicklung der Technologien im Auge zu behalten um gegebenenfalls Neuheiten oder Änderungen im System anzupassen. Eine Webanwendung sollte daher in all ihren Bestandteilen möglichst wartbar bleiben.

Soll-04 - Schnittstellen und Webservices Interoperabilität beschreibt den Austausch von Informationen verschiedener Softwaresysteme. Es gibt verschiedene Technologien, mit denen ein Informationsaustausch umgesetzt werden kann - REST, SOAP und RPC sind am weitesten verbreitet. Das Webframework sollte Mechanismen und Funktionen zur Umsetzung von Schnittstellen bereitstellen.

Soll-05 - MVC-Entwurfsmuster Besonders im Web hat sich das Model-View-Controller Entwurfsmuster als quasi Standard-Architekturmuster für Webanwendungen etabliert und hält daher Einzug bei den meisten Webframeworks. Es dient zur Strukturierung der Software in drei Einheiten. Das Model (Datenmodell) enthält die Geschäftslogik - die Informationen die dargestellt werden. Der Controller (Steuerung) ist die Schnittstelle zwischen der View und dem Datenmodell. Er nimmt Benutzeraktionen entgegen (z.B. Formulardaten) und leitet sie an ein bestimmtes Datenmodell weiter. Der Controller führt dann Operationen wie speichern, ändern und löschen auf dem Datenmodell (besser gesagt dem Objekt) aus. Die View ist die Präsentationsschicht und stellt die Daten dar, die es vom Controller entgegennimmt. Jedoch sollte eine View nicht ohne einen Controller neue Objekte erzeugen oder speichern (Trennung von Logik und Darstellung). Mit dem MVC-Entwurfsmuster können u.a. ProgrammiererInnen und DesignerInnen während der Entwicklung unabhängig voneinander arbeiten.

Soll-06 - Testing Testing ist mit test-driven development (TDD), vor allem in der agilen Softwareentwicklung, ein fester Bestandteil während der Projektentwicklung. Vor der Implementierung überprüft der Programmierer, mittels Unit-Tests, konsequent das Verhalten jeglicher Komponenten. Gerade kritische Prozesse und Transaktionen (zum Beispiel eine Banküberweisung) sollten ausgiebig getestet werden. Das Webframework soll die Möglichkeit von Unit-Tests bieten.

Soll-07 - Internationalisierung und Lokalisierung Viele Webanwendungen richten sich mittlerweile an ein internationales Publikum. Dank der Offenheit und der weiten Verbreitung des Internets lassen sich dadurch sehr einfach neue Zielgruppen (BenutzerInnen) erschließen. Jedoch gilt es, einige Vorraussetzungen und Besonderheiten bei der Internationalisierung von Webanwendungen zu beachten. Zunächst müssen sämtliche Texte übersetzt und möglicherweise in neue Datenbanken ausgelagert werden. Hierbei ist zu berücksichtigen, dass es länderspezifische Zeichensätze, Zahlen, Datum und Währungswerte gibt. Hinzu kommt, dass unter Umständen auch spezielle Grafiken neu erstellt werden müssen.

Soll-08 - Object Relational Mapping (ORM) Die meisten Webframeworks unterstützen das objektorientierte Programmierparadigma. Im Zusammenhang mit relationalen Datenbanken kommt es allerdings zu grundlegenden Problemen, weil der Zustand und das Verhalten eines Objekts nicht in einer relationalen Datenbank gespeichert werden kann. Dies ist auf die beiden widersprüchlichen Konzepte von objektorientierter Programmierung (OOP) und relationalen Datenbankmanagementsystemen (RDMS) zurückzuführen. Im Gegensatz zu objektorientierten Datenbanken, werden beim ORM die Tabellen aus der Datenbank als Klassen abgebildet (gemappt). Durch die Klassenabbildung wird für den/die ProgrammiererIn wieder die gewohnte OOP-Umgebung geschaffen. Wenn ein Objekt erstellt oder geändert wird, ist der Mapper verantwortlich, um diese Informationen in der Datenbank zu speichern und auch ggfs. wieder zu löschen. ORM bildet somit eine Schnittstelle zwischen OOP und relationalen Datenbanken.

Unterstützung von Transaktionen: Bei der Speicherung von Informationen in mehreren Datenbanktabellen muss sicher gestellt sein, dass entweder alle oder keine Informationen gespeichert werden.

Soll-09 - Scaffolding / Rapid Prototyping Mit Scaffolding ist das automatische Generien von den sogenannten CRUD-Pages (Create, Read, Update, Delete) gemeint.

Scaffolding und das dadurch verstandene Rapid Prototyping ist ein wesentlicher Aspekt der agilen Softwareentwicklung (zum Beispiel in Verwendung mit der Scrum-Methodik). Gerade zu Beginn eines Projekts eignet sich das Rapid Prototyping, da Änderungen in den Models umgehend in die Views eingebunden werden können.

Soll-10 - Caching Neben der Übertragunsgeschwindigkeit gibt es eine Menge anderer Faktoren, die für eine leistungsstarke Webanwendung entscheidend sind. Ein Aspekt ist das Caching - das Zwischenspeichern von Daten, die häufig verwendet bzw. aufgerufen werden.

Das Caching kann meist auf verschiedenen Ebenen implementiert werden: Auf dem Client, auf dem Webserver und auf der Datenbankebene. Dabei soll das Webframework

diese Mechanismen unterstützen und es ermöglichen, diese einfach zu aktivieren und zu kalibrieren.

Soll-11 - View-Engine View Engines werden eingesetzt, um das Arbeiten mit den Views zu erleichtern. Sie unterstützen vor allem das Templating - das Erstellen von Vorlagen. Durch typisierte Views wird eine Webanwendung robuster, weil weniger fehleranfällig; durch partial Views (oft auch nur als partials bezeichnet) werden einzelne Elemente oder Bereiche der Benutzeroberfläche wiederverwendbar gemacht. Diese Vorlagen können von mehreren Views genutzt werden. Dadurch wird deutlich weniger redundanter Code erstellt. Dies ist ein wichtiger Aspekt in Bezug auf das Don't repeat yourself (DRY) Prinzip.

Soll-12 - Dokumentation Eine Webanwendung ist aus Entwicklersicht eine Zusammenfassung verschiedenster Technologien (Webframeworks, Bibliotheken, Schnittstellen). Über das Application Programming Interface (API) haben Programmierer Zugriff auf die verfügbaren Funktionen. Nicht selten erstreckt sich die Dokumentation einer API über mehrere hundert Seiten. Für Programmierer ist es daher von enormer Bedeutung, dass die Bibliotheken gut strukturiert und verständlich beschrieben sind. Schlecht oder gar nicht dokumentierte Technologien erhöhen die Fehlerquote und sind oft ausschlaggebend für einen nicht flüssigen Workflow.

Soll-13 - Community Die beteiligten Personen in den Foren, Mailing-Listen oder Wikis bilden im Zusammenhang mit Webframeworks die Community. Es findet dabei ein Wissensaustausch statt, der weit über die standard-Dokumentation hinaus geht. Die Nutzer helfen sich gegenseitig bei Problemen und Fehlern und oft hinterlassen sie mit ihren Einträgen wiederum einen Lösungsansatz, der zukünftig von anderen wieder aufgegriffen werden kann. Es ist daher wichtig, dass den Anwendern eine Möglichkeit geboten wird, sich auszutauschen, gemeinsam Fehlermeldungen zu deuten und Lösungsansätze zu entwickeln.

Soll-14 - IDE-Unterstuetzung In der Softwareentwicklung ist die IDE das Basiswerkzeug für die Programmierer. Die Entwicklungsumgebung stellt den Entwicklern verschiedene Komponenten zur Verfügung, wie Editor, Compiler, Linker oder Debugger. Hinzu kommen mit Syntaxhighlighting, Refactoring und Code Formatierung weitere wichtige Funktionen, die die Entwickler in vielerlei Hinsicht enorm unterstützen.

Soll-15 - Kosten fuer Entwicklungswerkzeuge Je nach verfügbaren finanziellen Mitteln spielen die Kosten von Entwicklungswerkzeugen und Technologien durchaus eine Rolle. Bei beiden Faktoren hat man meist die Wahl zwischen kostenlosen (OpenSource) und kommerziellen Produkten. Je nach Webanwendung müssen somit Lizenzgebühren für

die Entwicklungswerkzeuge, Server zum Ausführen der Anwendung und Datenbankserver berücksichtigt werden. Dabei ist es wichtig die richtige Mischung verschiedener Komponenten zu finden.

Soll-16 - Eignung fuer agile Entwicklung Die herkömmlichen Methoden der Software-Entwicklung werden heute oft durch neue agile Methoden, wie Extreme Programming oder Scrum abgelöst. Sie fokussieren auf das Wesentliche und stehen für deutlich mehr Flexibilität in der Entwicklungsphase als konventionelle Methoden. Verschiedene Technologien unterstützen die agilen Methoden. Refactoring, Testing spielt dabei eine wichtige Rolle und soll unterstützt werden.

Soll-17 - Lernkurve fuer EntwicklerInnen Zur Umsetzung einer komplexen Webanwendung wird den Entwicklern ein Wissen über verschiedenste Bereiche der Softwareentwicklung abverlangt. Glücklicherweise gibt es für die Webentwicklung keinen einheitlichen Standard, der festlegt, wie eine Webanwendung entwickelt werden muss. Das Internet bietet für jeden Bereich eine Auswahl unterschiedlicher Technologien an. Daher müssen vor der Entwicklung mehrere Entscheidungen getroffen werden - hinsichtlich Programmiersprache, Javascript-Framework oder Datenbankserver. Es werden daher oft Technologien gewählt, die leicht zu erlernen und verstehen sind.

Wichtig ist, dass man einen schnellen Einstieg bekommt und Erfolge bald sichtbar werden, um die Motivation der Entwickler zu erhöhen.

Soll-18 - AJAX-Unterstuetzung Durch das Aufkommen von Javascript-Bibliotheken wie jQuery und Prototype haben sich vollkommen neue Möglichkeiten eröffnet mit Javascript auf dem Client zu arbeiten und mit AJAX wurde die Kommunikation zwischen Server und Client im Web revolutioniert. Die direkte Unterstützung eines Javascript-Frameworks ist für ein Webframework sinnvoll und erwünscht.

Darüber hinaus sollte die unterstützte Bibliothek lose gekoppelt sein, und damit austauschbar. Sogenanntes unobtrusive Javascript, bei dem auch bei abgeschaltetem Javascript die Anwendung funktioniert, ist auch eine Anforderung.

F. Grundsätze der ZKB IT-Architektur

Die Grundsätze sind aus dem *Handbuch der IT-Architektur*, siehe [uS10], der ZKB entnommen. Die Grundsätze sind hier aufgelistet und mit einer ID in der Form {KO}-{Laufnummer} versehen, da die Grundsätze als KO-Kriterien für die Evaluation der Java Web Frameworks dienen.

KO-01 [1] Applikationen sollen als N-Tier Applikationen designed und implementiert werden.

KO-02 [2] Objektorientierung (OO) soll innerhalb der Informatik für Neuentwicklungen durchgängig angewandt werden.

KO-03 [3] Eine neue Applikation (oder eine neue Komponente einer bestehenden Applikation) ist mehrsprachfähig zu realisieren.

KO-04 [4] Neue Applikationen sind Unicode-fähig zu realisieren.

KO-05 [5] Eine Applikation muss in mehreren Instanzen lauffähig sein.

KO-06 [6] Der ZKB GUI Style Guide ist in allen ZKB IT-Projekten anzuwenden.

KO-07 [7] Die Zentrale Server Infrastruktur (ZSI) ist als Server-Plattform für Applikationen, welche Windows-, Unix-, Linux-basierte Server einsetzen, zu verwenden.

KO-08 [8] RMI kann für reine Java-Anwendungen eingesetzt werden.

KO-09 [9] Für Java-Applikationen (Internet, Extranet und Intranet) wird das ZIP-Framework eingesetzt.

[1] [uS10] Kapitel 2.2.2 - *N-Tier Applikationen*, Seite 20
[2] [uS10] Kapitel 2.2.10 - *Objektorientierung*, Seite 22
[3] [uS10] Kapitel 3.3 - *Mehrsprachigkeit*, Seite 36
[4] [uS10] Kapitel 3.3 - *Mehrsprachigkeit*, Seite 37
[5] [uS10] Kapitel 3.9.1 - *Skalierbarkeit / Ausfallsicherheit / Perfomance*, Seite 49
[6] [uS10] Kapitel 4.4.1 - *User Interface*, Seite 55
[7] [uS10] Kapitel 5.2 - *Verwendung der Zentralen Server Infrastruktur ZSI*, Seite 57
[8] [uS10] Kapitel 9.2 - *Java RMI*, Seite 71
[9] [uS10] Kapitel 12.2.1 - *Einsatz von Frameworks*, Seite 140

KO-10 [10] Die Validierung und Plausibilisierung der Eingaben erfolgt immer abschliessend auf dem bankseitigen Applikations-Server. Es ist aber durchaus möglich, dass sich auf der Client-Seite eine Logik zur Überprüfung und Validierung der Eingaben für den Benutzerkomfort befindet.

KO-11 [11] Die Business-Logik in einer Internet-Applikation ist so auszulegen, dass diese von verschiedenen Präsentations-Logiken im Rahmen von Ultra-Thin- und Thin-Client-Applikationen genutzt werden kann.

KO-12 [12] Die Internet-Applikationen der ZKB werden nicht mit Browser Abhängigkeiten versehen und orientieren sich an den neutralen Standards der W3C-Komission.

KO-13 [13] Für Ultra-Thin-Client-Applikationen wird als Session-Mechanismus die Cookie oder die URL-Rewriting-Methode angewendet.

KO-14 [14] Einfache Internet-Applikationen mit dem Schwerpunkt Information können ausschliesslich Java Server Pages verwenden.

KO-15 [15] Komplexe Internet-Applikationen verwenden eine Kombination von Java Server Pages und mindestens einem Servlet als Dispatcher-Mechanismus.

KO-16 [16] Die Internet-Applikationen funktionieren auch eingeschränkt, ohne dass die Skript-Funktion im Browser aktiviert ist.

KO-17 [17] ActiveX wird wegen der Möglichkeit für direkte Zugriffe auf das Betriebssystem nicht eingesetzt.

KO-18 [18] Es werden keine neuen Applikationen als Java Applets entwickelt. Der Einsatz von Applets beschränkt sich auf einfache Funktionen wie Börsen- oder News-Ticker.

KO-19 [19] Internet-Applikation werden ohne Plugins entwickelt.

KO-20 [20] Für Ultra Thin Clients bzw. Browser-basierende Applikationen muss das aktuelle, Struts-basierende HTML-Client-Framework der ZKB Internet Plattform verwendet werden.

[10] [uS10] Kapitel 12.3.5 - *Client-/Server-Schemata von Internet-Applikationen*, Seite 141
[11] [uS10] Kapitel 12.3.5 - *Client-/Server-Schemata von Internet-Applikationen*, Seite 141
[12] [uS10] Kapitel 12.3.6.1 - *Browser-Abhängigkeiten*, Seite 142
[13] [uS10] Kapitel 12.3.6.2 - *Session-Mechanismus*, Seite 142
[14] [uS10] Kapitel 12.3.6.4 - *Einfache Internet-Applikationen*, Seite 143
[15] [uS10] Kapitel 12.3.6.5 - *Komplexe Internet-Applikationen*, Seite 143
[16] [uS10] Kapitel 12.3.6.6 - *Verwendung von JavaScript beziehungsweise ECMAScript*, Seite 143
[17] [uS10] Kapitel 12.3.6.6 - *Einsatz von ActiveX und Cookies*, Seite 143
[18] [uS10] Kapitel 12.3.6.8 - *Applets*, Seite 144
[19] [uS10] Kapitel 12.3.6.9 - *Browser-Plugins*, Seite 144
[20] [uS10] Kapitel 12.3.7 - *Client-Technologien*, Seite 144

KO-21 [21] Neue Internet- und Extranet-Applikationen müssen sich an das Layering gemäss nachfolgender Grafik halten. Für Intranet-Applikationen ist die Validator-Komponente fakultativ.

KO-22 [22] Eine Applikation muss ohne Änderung von der Intranet-Anwendung zur Extra- oder Internet-Applikation gemacht werden können. Es geschieht dies lediglich durch das Vorschalten der Validator-Komponente.

KO-23 [23] Der ZKB Standard-Web-Server ist der Apache HTTP-Server.

KO-24 [24] Der Application-Server wird als die integrierte technische Middleware für die Unterstützung von Java Server Pages (JSP), Servlets, Enterprise Java Beans (EJB) und der sicheren Kommunikation zwischen Client und Server eingesetzt.

KO-25 [25] Der ZKB Standard-J2EE-Application-Server ist der JBoss Application Server.

KO-26 [26] Der J2EE-Application-Server wird in der ZSI eingesetzt.

KO-27 [27] Die Serverplattform für den Einsatz von Web-Application-Servern für Internet, Intranet udn Extranet-Applikationen ist Linux.

KO-28 [28] Die technischen Services wie Session Management, Load Balancing, Transaction Management und Instance Pooling werden vom J2EE Application Server zur Verfügung gestellt.

KO-29 [29] Das Muster JSP/Servlets mit EJBs ist anzuwenden, wenn die Applikation eine umfangreiche, komplexe Business-Logik aufweist, die Business-Logik wiederverwendbar sein soll, mehrere unterschiedliche Clients (Browser (Ultra-Thin-)(HTML), Thin-(Java), Mobile, ...) mit einer Business Logik bedient werden müssen, hohe Anforderungen an die Skalierbarkeit gestellt werden und ein lange Lebenszyklus der Applikation erwartet wird.

KO-30 [30] Das Muster JSP/Servlets ohne EJBs ist anzuwenden, wenn die Applikation eine einfache Business-Logik aufweist, nur einen Client, zum Beispiel ein Browser (Ultra-Thin-)-Interface unterstützt, niedrige Anforderungen an die Skalierbarkeit stellt und nur eine vergleichsweise kurzer Lebenszyklus der Applikation erwartet wird.

[21] [uS10] Kapitel 12.3.8 - *Layering von Internet-Applikationen*, Seite 145
[22] [uS10] Kapitel 12.3.8 - *Layering von Internet-Applikationen*, Seite 146
[23] [uS10] Kapitel 12.3.10.2 - *Web Server*, Seite 146
[24] [uS10] Kapitel 12.4.2 - *Architektur beim Einsatz von Application-Servern*, Seite 147
[25] [uS10] Kapitel 12.4.2 - *Architektur beim Einsatz von Application-Servern*, Seite 148
[26] [uS10] Kapitel 12.4.2 - *Architektur beim Einsatz von Application-Servern*, Seite 148
[27] [uS10] Kapitel 12.4.2 - *Architektur beim Einsatz von Application-Servern*, Seite 148
[28] [uS10] Kapitel 12.4.2 - *Architektur beim Einsatz von Application-Servern*, Seite 150
[29] [uS10] Kapitel 12.4.3 - *Einsatz von Enterprise Java Beans*, Seite 150
[30] [uS10] Kapitel 12.4.3 - *Einsatz von Enterprise Java Beans*, Seite 150

KO-31 [31] In der ZKB werden folgende Standards für Web Services eingesetzt: SOAP, WSDL, W3C X Schema (XSD, WXS).

KO-32 [32] Alle Neuentwicklungen sind konsequent in Client-/Server-Komponenten aufzuteilen.

KO-33 [33] Die gewünschte Isolation der Systemteile wird durch eine Anwendungsstruktur mit Trennung in Model (Verarbeitung, Datenhaltung), View (Benutzeroberfläche) und Controller (Organisation, Ereignisvermittlung) erreicht.

KO-34 [34] Jede Applikation ist dafür verantwortlich, dass diejenigen Daten, für die sie den Lead hat, validiert sind.

KO-35 [35] Jede Applikation benutzt Datenvalidierung um sicherzustellen, dass sie die erhaltenen Daten verarbeiten kann und dass ihr Betrieb nicht gefährdet ist (Stabilität / Verfügbarkeit).

KO-36 [36] Benutzereingaben werden so früh wie möglich validiert.

KO-37 [37] Für die Entwicklung neuer Applikationen und beim neuen Design bestehender Applikationen wird Java eingesetzt.

KO-38 [38] Zusätzliche Java-Klassenbibliotheken, also solche, die nicht im JDK enthalten sind, werden nur in begründeten Ausnahmen eingesetzt. Normalerweise müssen solche Bibliotheken dem 100%-Pure-Java-Grundsatz entsprechen. Ausnahmen können systemnahe Funktionen für Security und dergleichen sein.

KO-39 [39] .NET-basierte Applikationen werden in der ZKB Informatik nicht entwickelt oder zur Entwicklung in Auftrag gegeben ausser Kleinapplikationen der Kategorie „Office“.

KO-40 [40] Die Nutzung von Open Source Software ist erlaubt.

KO-41 [41] Open Source Software unterliegt denselben Kriterien wie kommerzielle Software. Sie muss evaluiert, registriert, homologiert, intern supported und gepflegt werden.

[31] [uS10] Kapitel 12.9.2 - *Standards*, Seite 171
[32] [uS10] Kapitel 13.1 - *Client/Server-Konzept*, Seite 175
[33] [uS10] Kapitel 13.2 - *Anwendungsstruktur (MVC/Model, View, Controller)*, Seite 175
[34] [uS10] Kapitel 13.9.7.2 - *Allgemeine Grundsätze*, Seite 189
[35] [uS10] Kapitel 13.9.7.2 - *Allgemeine Grundsätze*, Seite 189
[36] [uS10] Kapitel 13.9.7.3.1 - *Datenvalidierung an der Benutzerszchnittstelle*, Seite 191
[37] [uS10] Kapitel 13.10 - *Programmiersprachen und Entwicklungsumgebungen*, Seite 192
[38] [uS10] Kapitel 13.10.4.8 - *Zusätzliche Java Klassenbibliotheken*, Seite 195
[39] [uS10] Kapitel 13.11 - *Einsatz von .NET-basierten Applikationen*, Seite 196
[40] [uS10] Kapitel 13.13 - *Einsatz von Open Source Software*, Seite 206
[41] [uS10] Kapitel 13.13 - *Einsatz von Open Source Software*, Seite 206

KO-42 [42] Produktiv eingesetzte Open Source Software muss durch angemessene Informationsquellen unterstützt sein (Newsgroups, Mailinglists, FAQ-Listen, WebSites, User Groups).

KO-43 [43] Die Weiterentwicklung von produktiv eingesetzter Open Source Software ausserhalb der ZKB muss öffentlich einsehbar sein und aktiv verfolgt werden können.

KO-44 [44] Die Lizenzbedingungen einer Open Source Software müssen vor dem Einsatz geprüft werden und von der Zürcher Kantonalbank akzeptiert werden können.

[42] [uS10] Kapitel 13.13.1 - *Kriterien für die Evaluation von Open Source Software*, Seite 207
[43] [uS10] Kapitel 14.14.1 - *Kriterien für die Evaluation von Open Source Software*, Seite 207
[44] [uS10] Kapitel 13.13.1 - *Kriterien für die Evaluation von Open Source Software*, Seite 207

G. Kick-off Protokoll

G.1. Anwesende Personen

Funktion	Person	Anwesend
Student	Roman Würsch	Ja
Auftraggeber	Bernhard Mäder, ZKB	Nein
Projektbetreuer	Beat Seeliger	Ja
Experte	Marco Schaad	Nein
Vertreter Studiengang Informatik	Olaf Stern	Ja

Tabelle G.1.: Anwesende Personen

G.2. Beschlüsse

- Gemäss Email von Herr Stern vom 17. März 2011
  - "Sie führen in Absprache mit Ihrem Betreuer ein inhaltliches Kick-Off durch. Gibt Ihr Betreuer sein OK anschliessend, fahren Sie mit der Bearbeitung der Arbeit fort (kein Zeitverlust für Sie)."
  - "An dem von Ihnen gebuchten Termin am 13. April führen wir ein verkürztes Kick-Off durch, dieses wird auch als das formale Kick-Off in EBS eingetragen und protokolliert."
- Gemäss Email von Herr Stern vom 15. März 2011: Es wurden die geforderten Änderungen an der Aufgabenstellung angepasst.
- Es soll ein RIA - Framework (z.B. ULC) mit in die Evaluation genommen werden.
- Es soll ein MVC - Framework (z.B. Struts) mit einbezogen werden.
- Es werden die Bewertungskriterien für die Bachelor Arbeit verwendet.

- Der Zeitplan ist straff, sportlich, sollte aber machbar sein.

H. Design-review Protokoll

H.1. Anwesende Personen

Funktion	Person	Anwesend
Student	Roman Würsch	Ja
Auftraggeber	Bernhard Mäder, ZKB	Nein
Projektbetreuer	Beat Seeliger	Ja
Experte	Marco Schaad	Ja
Vertreter Studiengang Informatik	Matthias Bachmann	Ja

Tabelle H.1.: Anwesende Personen

H.2. Beschlüsse

- Vorschlag von Matthias Bachmann: Es soll eine Sensitivitätsanalyse für die Gewichtung der Soll-Kriterien angewendet werden. Aufgrund der Abgrenzung in der Aufgabenstellung "*Umfragen, Erhebungen sowie Feldstudien werden nicht durchgeführt.*" wird das nicht gemacht. Zudem arbeitet der Student nicht intern in der Zürcher Kantonalbank, was ihn daran hindert eine Gewichtung durch die Mitarbeiter der ZKB-Architektur zu machen.
- Es sollen folgende Frameworks evaluiert werden:
  - ULC, Canoo RIA Suite
  - Apache Struts 1.3.10 mit ZIP
  - Vaadin 6.6.0
  - Apache Wicket 1.4.17

- Die Aufgabenstellung soll im EBS angepasst werden. Der Satz "*Eine Auflistung von etablierten Java Web Frameworks*" soll durch den Satz "*Ergebnisse der Evaluation von etablierten Java Web Frameworks*" ersetzt werden.
- Auftraggeber ist zufrieden, gemäss Abstimmung vom 19.04.2011.
- Die Einladung zum Schlusstermin erfolgt durch Roman Würsch.
- Gute Kriterien und Analysemethode gefunden.
- KO-Kriterien von der ZKB erhalten
- Gute Literatur Recherche
- Gute Präsentation, Achtung in Zukunft mit Farben (Titel sind nicht lesbar gewesen)
- Es soll die Thematik "Skalierung" der jeweiligen Frameworks erwähnt werden. evtl. mit einem Lasttest, falls nicht möglich in der Reflektion abgrenzen.

I. Abkürzungsverzeichnis

AHP	Analytic Hierarchy Process
API	Application Programming Interface
Ajax	Asynchronous JavaScript and XML
BRE	Business Request Exchange
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
EBS	Einschreibe- und Bewertungssystem der HSZ-T
EDT	Event Dispatch Thread
FAQ	Frequently Asked Questions
GUI	Grafische Benutzeroberfläche
GWT	Google Web Toolkit
HSZ-T	Hochschule für Technik Zürich
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrierte Entwicklungsumgebung
JDBC	Java Database Connectivity
JFC	Java Foundation Classes
JRE	Java Runtime Environment
JSP	JavaServer Pages
MOM	Message oriented Middleware
MVC	Model View Controller
MVP	Model View Presenter
NWA	Nutzwertanalyse
OO	Objektorientierung
ORM	Object Relational Mapping
PDF	Portable Document Format
PNG	Portable Network Graphics
RDBMS	Relational Database Management System
RIA	Rich Internet Application
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
SVG	Scalable Vector Graphics
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
ZIP	ZKB Internet Plattform
ZKB	Zürcher Kantonalbank
ZSI	Zentrale Server Infrastruktur

J. Abbildungsverzeichnis

K. Tabellenverzeichnis

L. Listingverzeichnis

M. Literaturverzeichnis

[AG] Canoo Engineering AG. Ulc architektur guide. http://ulc.canoo.com/developerzone/ULCArchitectureGuide.pdf. [Online; 08. Mai 2011].

[All02] Jeremy Allaire. Macromedia flash mx—a next-generation rich client. http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf, März 2002.

[Amr08] Dr. Beatrice Amrhein. Design pattern für graphische benutzeroberflächen. http://www.sws.bfh.ch/~amrhein/Skripten/Swing/Pattern.pdf, Februar 2008.

[ap11] agilelearn's posterous. 18 anforderungen an webframeworks - opendoc. http://agilelearn.posterous.com/18-anforderungen-an-webframeworks-opendoc, März 2011. [Online; 29. März 2011].

[BS11] Browser-Statistik.de. Browser-marktanteile im januar 2011. http://www.browser-statistik.de/marktanteile/2011/januar/, 2011. [Online; 07. Juni 2011].

[Buc05] Jörg Bucher. Ahp und nwa - vergleich und kombination beider methoden. http://community.easy-mind.de/page-77.htm#kombinierte, Januar 2005. [Online; 31. März 2011].

[DC05] Darren James Dave Crane, Eric Pascarello. *Ajax in Action*. Manning, Greenwich, CT , USA, 2005.

[Des10] Jesse Desjardins. You suck at power point! http://www.slideshare.net/jessedee/you-suck-at-powerpoint, 2010. [Online; 17. April 2011].

[DOS09a] Studienleiter Informatik Dr. Olaf Stern. Ablauf diplomarbeit. https://ebs.hsz-t.ch/files/ebs_files/Reglemente/Kreditsystem/Diplomarbeit/Ablauf-Bachelorarbeit_Studiengang-Informatik-der-HSZ-T_V1.3.pdf, Juni 2009.

[DOS09b] Studienleiter Informatik Dr. Olaf Stern. Bewertungskriterien diplomarbeit. https://ebs.hsz-t.ch/files/ebs_files/Reglemente/

Kreditsystem/Diplomarbeit/Bewertungskriterien-Bachelorarbeit_Studiengang-Informatik-der-HSZ-T_V1.0.xls, Mai 2009.

[Fou] Apache Software Foundation. Apache wicket - setup tomcat. https://cwiki.apache.org/WICKET/setup-tomcat.html. [Online; 15. Mai 2011].

[Fou08a] Apache Software Foundation. Installing struts with your servlet container. http://struts.apache.org/1.3.10/userGuide/installation-tc.html, 2008. [Online; 15. Mai 2011].

[Fou08b] The Apache Software Foundation. Package org.apache.struts.taglib.html. http://struts.apache.org/1.x/struts-taglib/apidocs/org/apache/struts/taglib/html/package-summary.html#package_description, 2008. [Online; 11. Mai 2011].

[Fou08c] The Apache Software Foundation. Tag reference sheet - html. http://struts.apache.org/1.x/struts-taglib/tagreference.html#struts-html.tld, 2008. [Online; 11. Mai 2011].

[Fou10a] The Apache Software Foundation. The apache tomcat connector. http://tomcat.apache.org/connectors-doc/, 2010. [Online; 21. April 2011].

[Fou10b] The Apache Software Foundation. Component reference. http://wicketstuff.org/wicket14/compref/, 2010. [Online; 29. Mai 2011].

[Fou11] The Apache Software Foundation. Apache tomcat. http://tomcat.apache.org/, 2011. [Online; 21. April 2011].

[fTZ09] Hochschule für Technik Zürich. Richtlinie poster zur bachelorarbei. https://ebs.hsz-t.ch/files/ebs_files/Reglemente/Kreditsystem/Diplomarbeit/Richtlinie-Poster-Bachelorarbeit_Studiengang-Informatik-der-HSZ-T_V1-pdf.pdf, Mai 2009.

[Ges] Schweizer Informatik Gesellschaft. Arbeitsplätze: Wo arbeiten Informatikfachleute? http://www.i-s.ch/index.php?id=is242. [Online; 29. März 2011].

[Gro10] Marko Groenroos. *Book of Vaadin*. Vaadin Ltd, Ruukinkatu 2-4, FI-20540 Turku, Finland, 2010.

[Hö01] Prof. Dr. Uwe Höft. Swot-analyse. http://www.fh-brandenburg.de/~hoeft/toolbox/swot.htm, Februar 2001. [Online; 08. Juni 2011].

[Inc10] Google Inc. Gbst uses google web toolkit to improve productivity and create a rich user experience. http://google-web-toolkit.googlecode.com/files/CaseStudy-GBST-Uses-GWT.pdf, 2010.

[JGG06] ehemaliger Studienleiter Informatik Jean-Gabriel Gander. Bestimmungen für die diplomarbeit. https://ebs.hsz-t.ch/files/ebs_files/Reglemente/Kreditsystem/Diplomarbeit/Bestimmungen_fuer_DA_V2.pdf, Mai 2006.

[Len10] Karsten Lentzsch. Jgoodies forms - build better screens faster. http://www.jgoodies.com/freeware/forms/index.html, 2010. [Online; 04. April 2011].

[Ltd11a] Vaadin Ltd. remotegate online cash management - six interbank clearing (switzerland). http://vaadin.com/who-is-using-vaadin/showcase/-/asset_publisher/2Spd/content/six-remotegate?redirect=/who-is-using-vaadin, 2011. [Online; 18. Mai 2011].

[Ltd11b] Vaadin Ltd. Vaadin - security alerts. http://vaadin.com/security-alerts, 2011. [Online; 11. Mai 2011].

[Ltd11c] Vaadin Ltd. Vaadin pro. http://vaadin.com/pro, 2011. [Online; 15. Mai 2011].

[Ltd11d] Vaadin Ltd. Vaadin sampler. http://demo.vaadin.com/sampler, Mai 2011. [Online; 15. Mai 2011].

[LW] Frank Ramirez Luke Wroblewsk. Web application solutions: A designer's guide. http://www.lukew.com/resources/WebApplicationSolutions.pdf.

[Mor03] Maxime Morge. Java analytic hierarchy process. http://www.di.unipi.it/~morge/software/JAHP.html, 2003. [Online; 21. April 2011].

[Mug06] Christian Mugg. Http - basics. http://mugg.at/fhsg/2005-2006/2006-02-03/folien_http-basics.pdf, Februar 2006.

[OB] Robin J. Adams Emre Tunar N. Dwight Barnette Osman Balci, William S. Gilley. Software life cycle models. http://courses.cs.vt.edu/~csonline/SE/Lessons/LifeCycle/index.html. [Online; 22. April 2011].

[Ora11] Oracle. A visual guide to swing components (java look and feel). http://download.oracle.com/javase/tutorial/ui/features/components.html, 2011. [Online; 14. April 2011].

[OWL11] Stat OWL. Rich internet application market share - ria market penetration and global usage. http://www.statowl.com/custom_ria_market_penetration.php, 2011. [Online; 05. April 2011].

[Pep05] Florian Pepping. Softwareanalyse: Begriffe und techniken. http://www2.cs.uni-paderborn.de/cs/ag-engels/ag_dt/Courses/Lehrveranstaltungen/Siemens-Seminar/Ausarbeitungen/Pepping-Florian.pdf, Mai 2005.

[RF10] Olaf Siefart Roland Förher, Carl-Eric Menzel. *Wicket, Komponentenbasierte Webanwendungen in Java*. dpunkt.verlag, Ringstrasse 19B, 69115 Heidelberg, 2010.

[RHM08] LLC Red Hat Middleware. Jbossweb documentation. http://docs.jboss.org/jbossweb/2.1.x/index.html, 2008. [Online; 21. April 2011].

[RHM11] LLC Red Hat Middleware. Jboss application server. http://www.jboss.org/jbossas, 2011. [Online; 21. April 2011].

[Saa80] Thomas L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw Hill Higher Education, Columbus, OH 43272, USA, 1980.

[Sch99] Stephen R. Schach. *Software Engineering*. McGraw-Hill, Boston MA, 1999.

[Sch07] Stephan Schuster. Erweiterung des web-frameworks wings durch die integration von optionalem ajax. http://wingsframework.org/doc/papers/Diplomarbeit_Stephan_Schuster.pdf, März 2007.

[Sch10] Boris Schäling. Kapitel 2: Swing. http://www.highscore.de/java/aufbau/swing.html, 2010. [Online; 14. April 2011].

[s.r] JetBrains s.r.o. Static code analysis. http://www.jetbrains.com/idea/documentation/static_code_analysis.html. [Online; 07. April 2011].

[SV99] Fritz Lienhard Stefan Vogler. Willkommen bei der zkb. http://www.markenexperte.ch/upload/pdf/willkommen-thexis.pdf, 1999. [Online; 22. April 2011].

[uS10] Team IT-Architektur Konzepte und Standards. *Handbuch der IT-Architektur*. Zürcher Kantonalbank, Bahnhofstrasse 9, 8010 Zürich, 7.5 edition, Februar 2010. Allgemeine Bestimmungen zur IT-Architektur, innerhalb der Zürcher Kantonalbank.

[Wal01] Ernest Wallmüller. *Software-Qualitätsmanagement in der Praxis*. Carl Hanser Verlag GmbH & Co. KG, Postfach 86 04 20, 81631 München, Deutschland, 2001.

[Wel99a] Don Wells. Acceptance Tests. http://www.extremeprogramming.org/rules/functionaltests.html, 1999. [Online; 29. Mai 2011].

[Wel99b] Don Wells. User Stories. http://www.extremeprogramming.org/rules/userstories.html, 1999. [Online; 29. Mai 2011].

[Wik10a] Wikipedia. Basiswert. http://de.wikipedia.org/w/index.php?title=Basiswert&oldid=78073732, August 2010. [Online; 19. April 2011].

[Wik10b] Wikipedia. Java web start. http://de.wikipedia.org/w/index.php?title=Java_Web_Start&oldid=75585629, Juni 2010. [Online; 08. Mai 2011].

[Wik10c] Wikipedia. Swingworker. http://en.wikipedia.org/w/index.php?title=SwingWorker&oldid=386414199, September 2010. [Online; 04. April 2011].

[Wik10d] Wikipedia. Tag-library. http://de.wikipedia.org/w/index.php?title=Tag-Library&oldid=73655431, April 2010. [Online; 29. Mai 2011].

[Wik11a] Wikipedia. Analytic hierarchy process. http://de.wikipedia.org/w/index.php?title=Analytic_Hierarchy_Process&oldid=87277232, Januar 2011. [Online; 04. April 2011].

[Wik11b] Wikipedia. Apache wicket. http://de.wikipedia.org/w/index.php?title=Apache_Wicket&oldid=88558267, Mai 2011. [Online; 18. Mai 2011].

[Wik11c] Wikipedia. Entwurfsmuster. http://de.wikipedia.org/w/index.php?title=Entwurfsmuster&oldid=87357381, April 2011. [Online; 18. April 2011].

[Wik11d] Wikipedia. Java applet. http://de.wikipedia.org/w/index.php?title=Java-Applet&oldid=86761109, März 2011. [Online; 08. Mai 2011].

[Wik11e] Wikipedia. Java archive. http://de.wikipedia.org/w/index.php?title=Java_Archive&oldid=84676141, Januar 2011. [Online; 29. Mai 2011].

[Wik11f] Wikipedia. Java virtual machine. http://de.wikipedia.org/w/index.php?title=Java_Virtual_Machine&oldid=86955036, März 2011. [Online; 08. Mai 2011].

[Wik11g] Wikipedia. Javaserver pages. http://de.wikipedia.org/w/index.php?title=JavaServer_Pages&oldid=83743292, Januar 2011. [Online; 29. Mai 2011].

[Wik11h] Wikipedia. Median. http://de.wikipedia.org/w/index.php?title=Median&oldid=87404521, April 2011. [Online; 08. Mai 2011].

[Wik11i] Wikipedia. Mittelwert. http://de.wikipedia.org/w/index.php?title=Mittelwert&oldid=87198455, April 2011. [Online; 08. Mai 2011].

[Wik11j] Wikipedia. Nutzwertanalyse. http://de.wikipedia.org/w/index.php?title=Nutzwertanalyse&oldid=86980368, März 2011. [Online; 31. März 2011].

[Wik11k] Wikipedia. Observer (entwurfsmuster). http://de.wikipedia.org/w/index.php?title=Observer_(Entwurfsmuster)&oldid=86993850, März 2011. [Online; 04. April 2011].

[Wik11l] Wikipedia. Paretoprinzip. http://de.wikipedia.org/w/index.php?title=Paretoprinzip&oldid=88214860, April 2011. [Online; 29. Mai 2011].

[Wik11m] Wikipedia. Rich internet application. http://de.wikipedia.org/w/index.php?title=Rich_Internet_Application&oldid=86143678, März 2011. [Online; 05. April 2011].

[Wik11n] Wikipedia. Rich internet application. http://en.wikipedia.org/w/index.php?title=Rich_Internet_application&oldid=417575667, März 2011. [Online; 05. April 2011].

[Wik11o] Wikipedia. Sensitivitätsanalyse. http://de.wikipedia.org/w/index.php?title=Sensitivit%C3%A4tsanalyse&oldid=87406951, April 2011. [Online; 08. Mai 2011].

[Wik11p] Wikipedia. Social engineering (sicherheit). http://de.wikipedia.org/w/index.php?title=Social_Engineering_(Sicherheit)&oldid=87231859, April 2011. [Online; 05. April 2011].

[Wit08] Markus Wittlinger. Gui-analysen und bibliotheken. http://www2.informatik.uni-stuttgart.de/iste/ps/Lehre/SS2008/HS_Programmanalysen/gui.wittlinger.pdf, Juli 2008.