Evaluation eines Java Web Frameworks zur Ablösung bestehender Java Swing Applikationen

Diplomarbeit in Informatik

Studierender - Roman Würsch

Auftraggeber - Bernhard Mäder, ZKB

Projektbetreuer - Beat Seeliger

Experte - Marco Schaad

HSZ-T - Technische Hochschule Zürich

März 2011 bis Juni 2011

Zusammenfassung

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.

Inhaltsverzeichnis

1. 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

2. 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 Infenieur FH abgeschlossen.

2.1. 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, habe ich mich auf die männliche Form beschränkt. Selbstverständlich sind bei allen Formulierungen beide Geschlechter angesprochen.

2.2. 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]

2.3. Bewertungskriterien

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

[1] Gemäss Beschluss am Kick-off Meeting, siehe Kapitel E Kick-off Protokoll auf S. 97

3. 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, die das Internet als Vertriebskanal bietet, ein. In der Finanzindustrie wurde dieser Schritt mit dem Online-Banking gegen Ende der Neunzigerjahre gemacht. Dabei wurden die Dienstleistungen, die beim klassischen Bankschalter bezogen werden konnten, mit einer Web Applikation, über das Internet, nutzbar gemacht. Das 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 käuflich erwerbliche Softwarelösungen nicht vollends aus. Um den Vorsprung zur Konkurenz auszubauen, entwickelt die Finanzindustrie vielmals Computerprogramme, welche die Automatisierung ihrer Prozesse ermöglicht, oder mit denen Arbeiten in ihrem Kerngeschäft gemacht werden können.

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

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.

3.1. Motivation

Die Zürcher Kantonalbank setzt bei der Entwicklung von Software als Inhouse Lösungen auf Java Swing Applikationen und auf Java Web Applikationen. Die Java Web Applikationen basieren auf dem Web Framework Apache Struts 1.3 mit einer von der ZKB erstellten Erweiterung namens ZKB Internet Plattform (ZIP). Da eine Lösung als Java Web Applikationen 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 ist mitlerweilen in die Jahre gekommen. Es basiert auf dem Prinzip von "request und response". Wann immer eine Aktion von einem User in der Applikation gemacht wird, sei es das Ansteuern eines Links oder das Absenden eines Formulars, wird die Webseite, welche die Applikation repräsentiert, neu geladen. Heutzutage gibt es Frameworks, welche einem ermöglichen eine Web Applikation zu entwicklen, die sich bei der Bedienung wie eine klassische Desktop Applikation anfühlen. Dies wird durch die Technik von Asynchronous JavaScript and XML (Ajax) realisiert.

Bei Ajax können Daten einer Web Applikation, ohne die komplette Webseite neu zu laden, verändert werden. Dies erlaubt es Web Applikationen, auf Benutzer Aktionen schneller zu reagieren, da vermieden wird, dass statische Daten, die sich unter Umständen nicht geändert haben, immer wieder über das Netzwerk übertragen werden müssen.

Apache Struts 1.3 unterstützt Ajax nicht direkt, das kann aber über Erweiterungen ermöglicht werden. Es gibt Java Web Frameworks, welche diese Technik out of the box mitbringen. Eine Evaluation soll zeigen welches dieser Java Web Frameworks am besten für einen möglichen Einsatz geeignet ist.

3.2. 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 anhand eines strukturierten Vorgehens eine Menge von bestehenden Java Swing Applikationen der Zürcher Kantonalbank auf deren Funktionalitäten der Benutzeroberfläche eruiert werden. Aufgrund der Anforderungen der IT-Architektur der Zürcher Kantonalbank sollen Java Web Frameworks auf die Validität der erhobenen Funktionalitäten verifiziert werden. Die Java Web Frameworks, welche diesen Ansprüchen genügen, sollen auf einen möglichen Einsatz in der Informatik Infrastruktur der Züricher Kantonalbank untersucht werden. Durch die Implementierung eines Prototyen soll gezeigt werden, dass die Umsetzung möglich ist. Schlussendlich soll eine Empfehlung für ein Java Web Framework ausgesprochen werden, für den Fall, dass eine bestehen-

de Java Swing Applikation in eine Web Applikation umgebaut werden soll. Ebenso sollen die gewonnen Erkenntnisse, für zukünftige Revidierungen im Bereich der Java Web Frameworks der IT-Architektur der Zürcher Kantonalbank, als Grundlage dienen können.

3.3. Struktur

tbd

3.4. Danksagung

tbd

4. Java Swing Applikationen

Swing kommt aus dem Hause Oracle, ehemals SUN, 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.

4.1. 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 als Abstraktion des User Interfaces dient.

4.1.1. Komponenten

Gemäss [Sch10], kennt Swing drei Hirarchien von Komponenten: Top-Level-, Intermediate- und Atomic Components. Unter Atomic Components versteht man einzelne Bausteine wie ein *javax.swing.JButton*, ein einfach Knopf, oder ein *javax.swing.JTextField*, ein einfaches Textfeld. 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 Komponente in zwei Teile. 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.

4.1.2. Layouts

Viele Komponenten in Swing haben ein Layout, vorallem 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]

4.1.3. Eventbasierte Kommunikation der Komponenten

Das Programmiermodell mit Java Swing ist Eventbasiert. Dabei können Events definiert werden, z.B. ein Mausklick auf einen Knopf, bei welchen eine Aktion ausgeführt werden soll, z.B. das speichern eines Dokuments. Die eventbasierte Kommunikation zwischen den Swing Komponenten ist nach dem Observer Design Pattern implementiert, siehe [Wik11h].

4.1.4. Hilfsmittel

Die Klasse *javax.swing.SwingUtilities* bietet eine vielzahl von Hilfsfunktionen, welche bei der Entwicklung von Swing Applikationen verwendet werden können. Zusätzlich bietet Swing es eine handvoll weiterer 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 $java.swing.SwingWorkder < T, V >$, um asynchrone Tasks zu verarbeiten.

4.2. Pluggable Look & Feel

Das Erscheinungsbild und das Verhalten der Swing Komponenten, auch genannt 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.

4.3. Multithreading

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

4.3.1. Asynchrone Tasks

Damit der EDT bei länger dauernden Tasks nicht blockert wird, stellt Swing das Konzept vom SwingWorker zur Verfügung, siehe [Wik10c]. Dabei können Tasks, zum Beispiel eine lang andauernde Berechnung, an einen neuen Thread deligiert werden. Die benötigte Klasse heisst: $javax.swing.SwingWorker < T, V >$.

5. 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 [Wik11i] und [All02] S. 2.

5.1. Grundlagen

Der Begriff RIA ist mit der Entwicklung des Internets entstanden und wird heute oft verwendet. Für viele Leute ist RIA ein synonmy für Webanwendungen, welche mit Ajax realisiert werden. Ajax bietet die Möglichkeit eine "reichhaltige" 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 5.1.

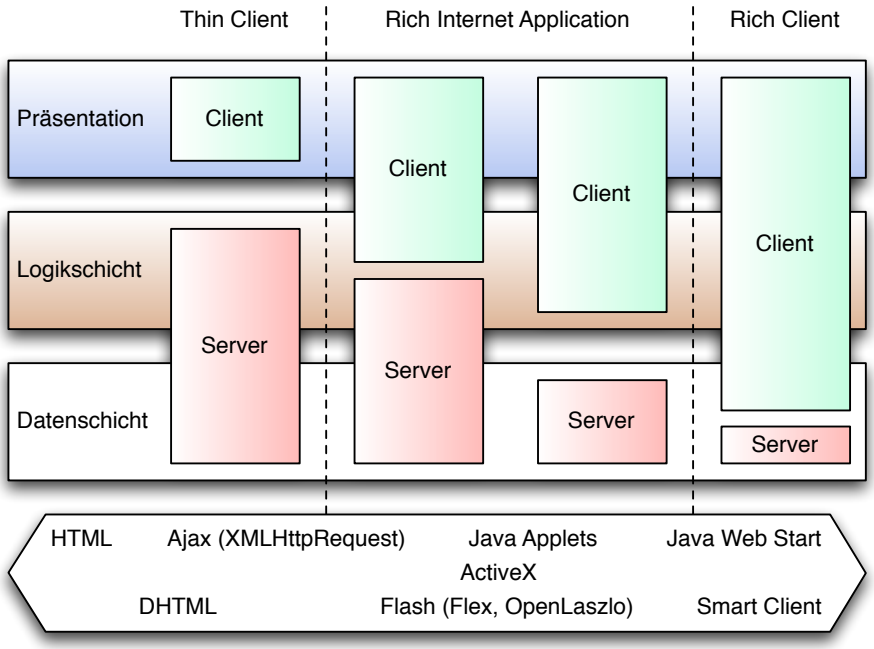

Abbildung 5.1.: Web Anwendungen (nach [Sch07] S. 6f. und [LW] S. 5)

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.

5.2. Klassifikation

Es wird eine Klassifikation aller RIA in die Klassen der Plugin orientierten, der Client orientierten und der Browser orientierten gemacht.

5.2.1. Plugin orientiert

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 [Wik11j] und [OWL11].

5.2.2. Client orientiert

Unter Client orientiert versteht man lediglich, dass alle Desktopanwendungen, welche entweder mit den Internet kommunizieren oder zumindes über dieses ausgeliefert werden, zu dieser Kategorie gehören. Dies trifft nur zu, in der Annahme, dass Desktopanwendungen eine “reichhaltige” Bedienoberfläche anbieten, siehe [Wik11i]. Eine Java Swing Applikation könnte also auch als Client orientierte RIA klassifiziert werden.

5.2.3. Browser orientiert

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 Jahrtausenwende wurden klassische Webanwendung nach dem Prinzip von Request - Response 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 5.2. Der zeitliche Ablauf sieht mit einem Unified Modeling Language (UML) Sequenzdiagramm wie folgt aus, siehe Abbildung 5.3. 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 war, siehe [DC05] S. 44ff.

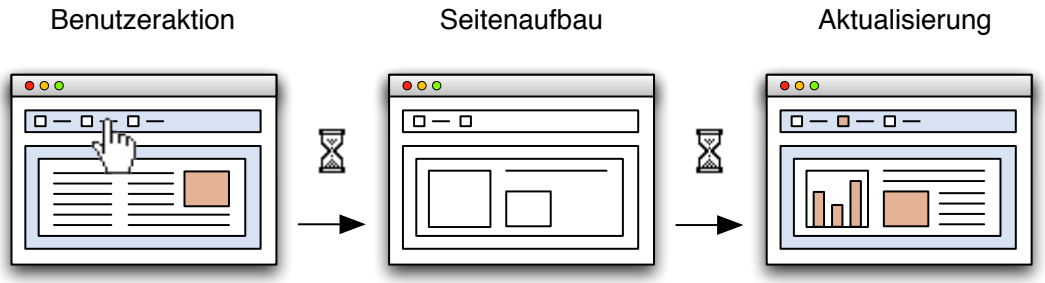

Abbildung 5.2.: Klassische Webanwendung aus der Usersicht (nach [Sch07] S. 10)

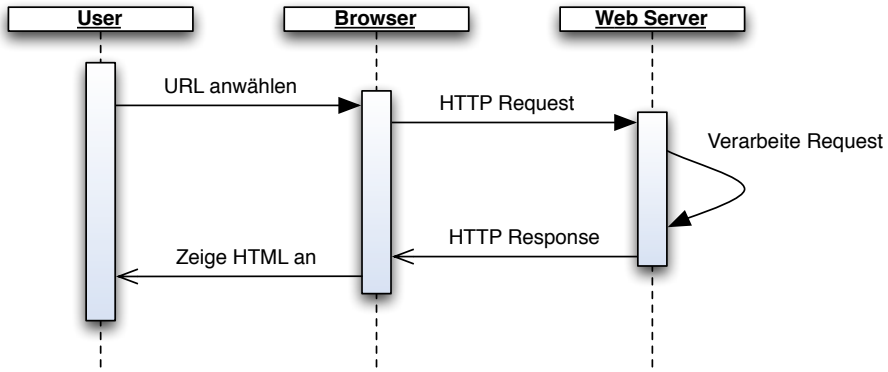

Abbildung 5.3.: HTTP Request als UML Sequenzdiagramm (nach [Mug06] S. 10)

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 5.4. Der zeitliche Ablauf sieht mit einem UML Sequenzdiagramm wie folgt aus, siehe Abbildung 5.5. Das Prinzip funktioniert dadurch, dass eine zusätzliche Schicht zwischen dem Browser und Server eingerichtet wird. Diese Schicht, ich nenne sie hier Ajax-Engine, 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 im Hintergrund getätigte Interaktionen verarbeiten.

Abbildung 5.4.: Webanwendung mit Ajax aus der Usersicht (nach [Sch07] S.12)

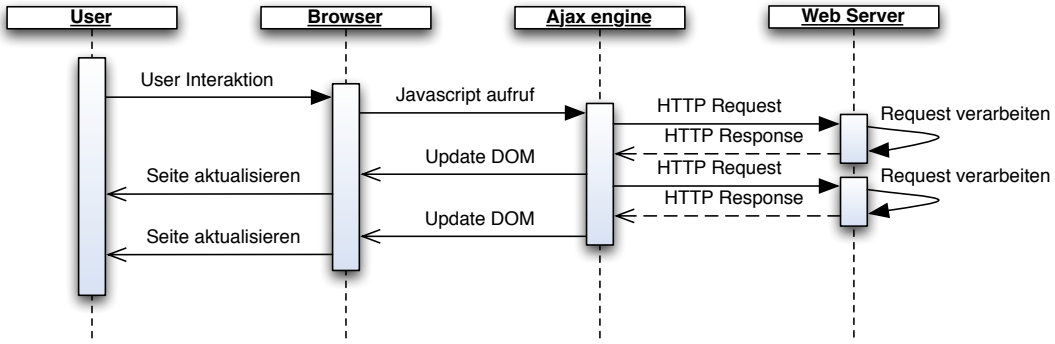

Abbildung 5.5.: Ajax Request als UML Sequenzdiagramm

Security

Nach [Wik11i] gelten Browser orientierte RIA, welche auf Webstandards[1] basieren, als relativ sicher. Mögliche Sicherheitslöcher stellen vorallem der jeweils verwendete Browser, in der die Applikation dargestellt wird, und Attacken nach dem Prizip von Social Engineering[2].

Suchmaschinenoptimierung

Suchmaschinenoptimierung dient dazu im Ranking einer Suchmaschine besser abzuschliessen. Dies ist vorallem bei Unternehmen von Bedeutung, welche das Internet als Vertriebkanal 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.

[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 [Wik11l]

5.3. 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.

5.3.1. 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.

5.3.2. 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 den Browser orientierten RIA, siehe Abbildung 5.6

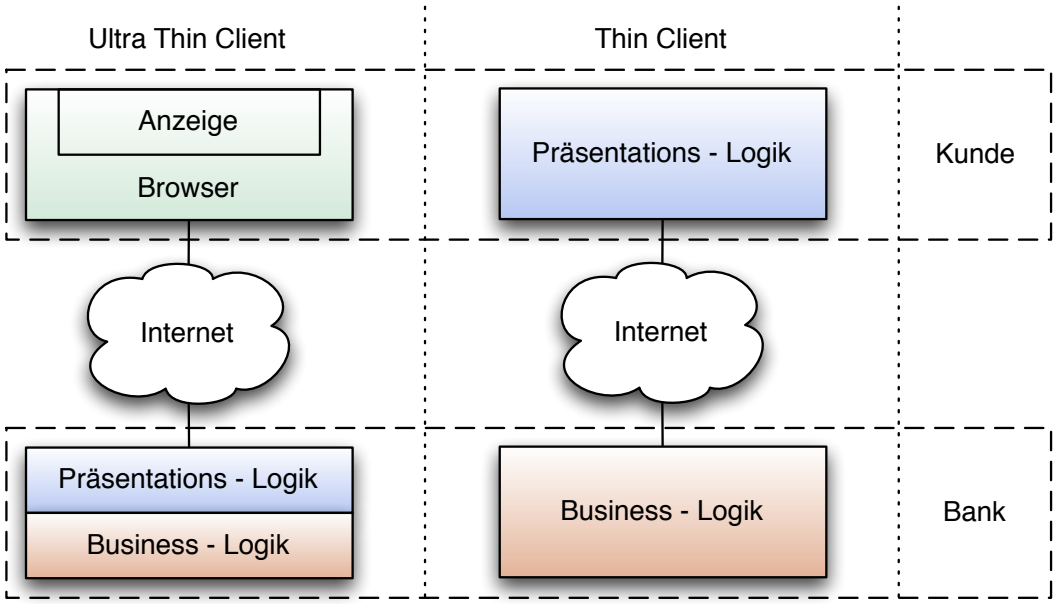

Abbildung 5.6.: Möglicher Aufbau von Web Applikationen (nach [uS10] S. 141)

6. 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 Ansprüchen der IT-Infrastruktur genügt. Die Informationen, welche hier beschrieben werden, stammen aus dem Handbuch der ZKB IT-Architektur, siehe [uS10]. In der Abbildung 6.1 ist die Standardkonfiguration, wie sie heute verwendet wird, ersichtlich.

6.1. 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 auf allen ZKB 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.

Die Verbindung zur Präsentations-Logik und auch das Load-Balancing wird mit dem Modul[1] *mod_jk*[2] gemacht.

6.2. Präsentations-Logik

Für die Präsentations-Logik wird der Servlet/JSP Container JBoss Web[3] verwendet, der ein Bestandteil des JBoss AS[4] ist und auf der Open Source Software Apache Tomcat[5] 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.

[1] Der Apache HTTP Server 2 hat einen modularen Aufbau, womit er durch Module erweitert werden kann.
[2] Für Informationen zum Apache Tomcat Connector, siehe [Fou10]
[3] JBoss Web basiert auf Apache Tomcat 6.0 und implementiert die Standards Servlet 2.5 und JavaServer Pages 2.1, siehe [RHM08]
[4] JBoss AS steht für JBoss Application server, siehe [RHM11]
[5] Für mehr Informationen zu Apache Tomcat, siehe [Fou11]

Für die Skalierung der Web Applikation können weitere JBoss AS Instanzen dazugeschalten werden. In der standard Konfiguration 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.

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

6.3. 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 dazugeschalten werden. In der standard Konfiguration 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.

Die Kommunikation zwischen der Web Applikation und der ZKB spezifischen Middleware findet ebenfalls hier in der Business-Logik statt.

6.4. Datenschicht

Die Datenschicht wird meistens mit einem RDBMS gemacht. Normalerweise wird Oracle eingesetzt. Je nach grösse der Applikation kann es sein, dass mehrere verschiedene RDBMS zum Einsatz kommen.

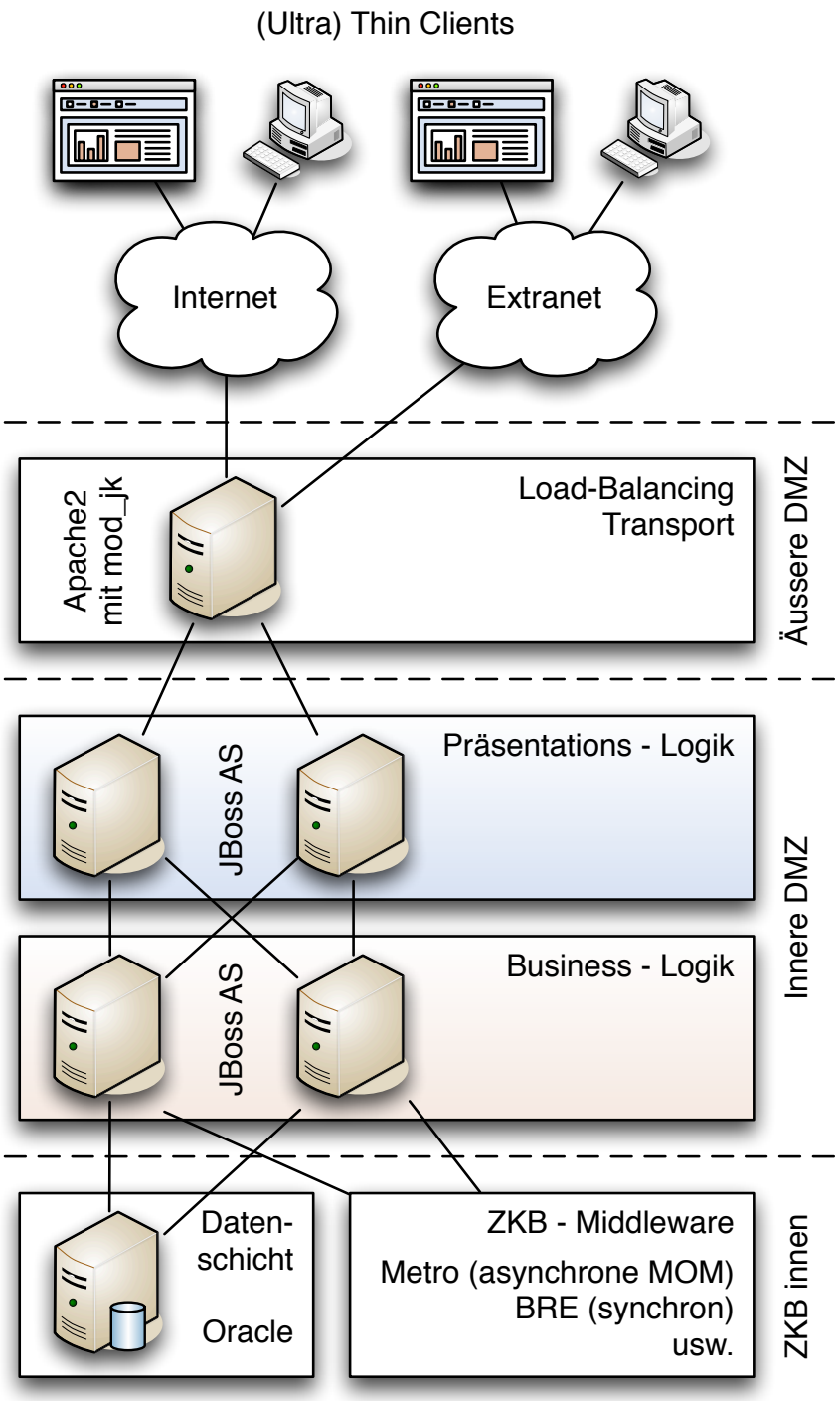

Abbildung 6.1.: Architektur von Internet-Applikationen (nach [uS10] S. 145)

7. 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.

7.1. 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 7.1. In meiner Arbeit werde diese Methoden zur Analyse von Elementen der grafischen Benutzeroberfläche einsetzten.

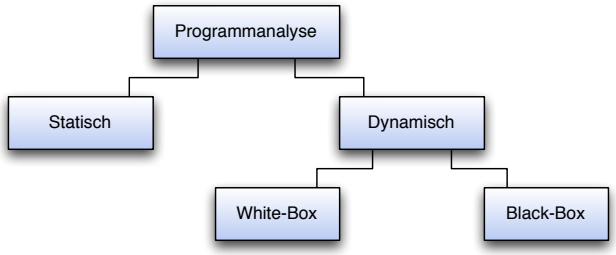

Abbildung 7.1.: Überblick über konventionelle Testmethoden (nach [Pep05] S. 5)

7.1.1. 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 Entwicklungsumbegung (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 [Wit08] befasst sich der Author damit, was man bei Grafische Benutzeroberfläche (GUI) lastiger Software durch eine statische Programmanaylse in Erfahrung bringen kann. Nach Aussage des Authors 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]

7.1.2. 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 7.1 Zeile 4. Dabei wird auf einen dreifachen Mausklick abgefragt. 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 7.1: Spezialfall - dreifacher Mausklick

[1] [Wit08] Zusammenfassung - Seite 1

7.1.3. Probleme bei der Verwendung von Bibliotheken

Gemäss [Wit08] S. 1f. besteht ebenfalls 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 meisst um ein Vielfaches grösser sind als das zu analysierende Programm. Der Aufwand wird durch die grössere Menge an Sourcecode zudem erhöht.

7.2. Wahl des Verfahrens

Die Wahl des Verfahrens der Analyse soll wie in der Abbildung 7.2 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.

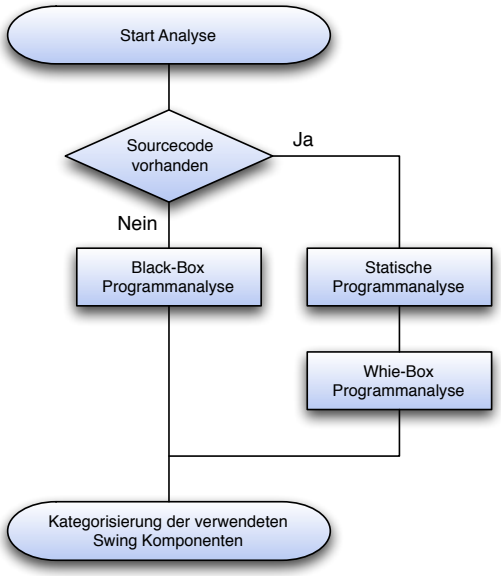

Abbildung 7.2.: Wahl des Verfahrens der Programmanalyse

8. 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.

8.1. 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 [Wik11g]. Diese Methode stammt aus dem Bereich der quantitativen Analysemethoden der Eintscheidungstheorie. Um eine möglichst präzise objektive Gewichtung der einzelnen Faktoren zu erhalten wurde 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.

8.2. Gewichtete Nutzwertanalyse

Bei der gewichteten NWA wird eine Menge von Alternativen, auf deren Nutzen, miteinander verglichen. Der Vergleich wird über n vergleichbare Kriterien geführt. Dabei werden die einzelnen Kriterien mit einem Erfüllungsgrad e_i bewertet. Die Skala der Erfüllungsgrade ist in der Tabelle 8.1 ersichtlich.

Jedes Kriterium wird durch einen Gewichtungsfaktor g_i versehen, was die Präferenz des Kriteriums wiederspiegelt. Dabei gilt: Die Gewichte gi werden so gewählt, dass ihre Summe 1 (100%) ergibt, siehe Formel 8.1.

$$Gewicht := \sum_{i=1}^{n} g_i = 1 \tag{8.1}$$

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

Tabelle 8.1.: Skala der Erfüllungsgrade

Der Nutzwert ergibt sich durch die Formel 8.2:

$$Nutzwert := \sum_{i=1}^{n} e_i \cdot g_i \tag{8.2}$$

Für jede zu evaluierende Alternative soll geprüft werden, ob eines der KO-Kriterien erfüllt ist, was zu einem Ausschluss der Alternative führen würde. Falls das nicht der Fall ist, wird der Nutzwert der Alternative berechnet. Diejenige Alternative mit dem grössten Nutzwert entspricht am meisten den Anforderungen.

8.2.1. Anschauliches Beispiel

Als anschauliches Beispiel sollen zwei Alternativen - Auto A und B - miteinander verglichen werden. Sie sollen auf deren Kriterien - Leistung, Aussehen und Alltagstauglichkeit - verglichen werden. In der Tabelle 8.2 ist das Beispiel ersichtlich. Das Resultat zeigt, dass das Auto A dem Auto B gegenüber bevorzugt werden soll, da der Nutzwert $5.0 > 4.7$ ist.

Kriterien	Gewichtung g	e_A	Wertigkeit A	e_B	Wertigkeit B
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 8.2.: Beispiel einer Nutzwertanalyse

8.3. 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. Totzdem 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

8.3.1. 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.

8.3.2. 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 verlgichen, was eine grössere Bedeutung in der gestellten Frage hat. Die Skala geht von 1 bis 9, siehe Tabelle 8.3.
- 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 8.3.

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

Tabelle 8.3.: Skala der Vergleichsgrade

8.3.3. 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[1], dies ist ein Java Programm mit dem der gesammte Prozess des AHP abgebildet und berechnet werden kann. Das Programm wird unter den Bedingungen der GNU General Public License vertrieben.

8.4. Kombination beider Methoden

Diese beiden Methoden können auch kombiniert eingesetzt werden, siehe [Buc05], da der AHP relativ komplex in der Umsetztung 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 plausable Evaluation ermöglicht.

In der Abbildung 8.1 ist der komplette Ablauf einer Evaluation mit der kombinierten Methode ersichtlich.

8.4.1. Verbesserung der Gewichtung

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

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

- Die Sensitivitätsanalyse, siehe [Wik11k].

  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 Vergesserung der Gewichtung. Das Vorgehen wird iterativ, bis zur Unterschreitung einer definierten Schwelle des Inkonsistenzfaktors, fortgeführt.

- Statistische Methoden, wie der Mittelwert oder der Median, siehe [Wik11e] und [Wik11f].

  Dabei wird der Prozess der Gewichtung von mehr als einer Person durchgeführt. Das Ergebniss wird aufgrund der statistischen Methoden normalisiert.

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

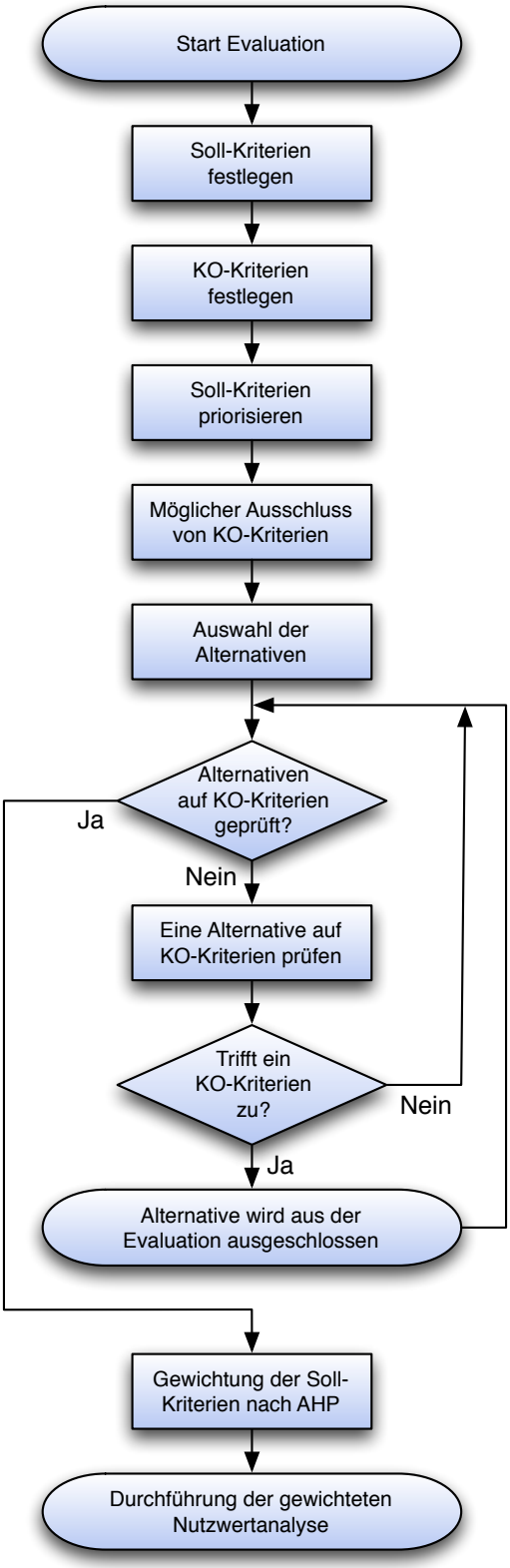

Abbildung 8.1.: Ablauf einer Evaluation mit der kombinierten Methode aus Nutzwertanalyse und AHP

9. 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 7 (Methoden zur Analyse von Java Swing Applikationen, S. 23ff) angewendet. Als Ergebniss wird eine Kategorisierung der verwendeten Swing Komponenten aufgelistet.

9.1. 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 9.1 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 9.1.: Zu analysierende Java Swing Applikationen

9.1.1. Begründung

Strukti Live Strukti Live 1.2 wurde gewählt, weil es eine Applikation ist, die von der ZKB veröffentlicht wurde. Somit werden bei der Anlayse keine sensiblen Informationen preisgegeben.

Strukti Online Strukti Online 2.10.0 wurde gewählt, weil eine mögliche Migration zur Web Applikation schon einmal in der ZKB zur Disskusion stant.

Hedo Tool Hedo Tool 1.0.710 wurde gewählt, da die Applikation über ein hochstehendes GUI verfügt.

9.2. Durchführung der Analyse

Die Wahl des Verfahrens zur Analyse von Java Swing Applikationen, soll, wie im Kapitel 7 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[1] 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 miteinbezogen 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 [Wik11b], 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 desshalb 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

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

9.3. Strukti Live

Strukti Live ist ein Lerntool der Zürcher Kantonalbank für strukturierte Produkte. Die Applikation kann auf dem Internetauftritt [2] der ZKB heruntergeladen werden. Es wurde die aktuelle Version 1.2 für die Analyse verwendet.

In der Tabelle 9.2 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 9.2.: Verwendete Bibliotheken von Strukti Live 1.2

9.3.1. Gefundene Komponenten

Top-Level-Komponenten

- *javax.swing.JDialog*, siehe Abbildung 9.2
- *javax.swing.JFrame*, siehe Abbildung 9.1

Intermediate-Komponenten

- *javax.swing.JPanel*, siehe Abbildung 9.1
- *javax.swing.JRootPane*, siehe Abbildung 9.1
- *javax.swing.JScrollPane*, siehe Abbildung 9.1
- *javax.swing.JTabbedPane*, siehe Abbildung 9.3

Atomic-Komponenten

- *javax.swing.JButton*, siehe Abbildung 9.1
- *javax.swing.JCheckBox*, siehe Abbildung 9.1

[2] Unter der Internetadresse `http://www.zkb.ch/struktilive` sind alle nötigen Informationen zu Strukti Live enthalten.

- *javax.swing.JLabel*, siehe Abbildung 9.1
- *javax.swing.JRadioButton*, siehe Abbildung 9.2
- *javax.swing.JSlider*, siehe Abbildung 9.2
- *javax.swing.JTextField*, siehe Abbildung 9.2
- *javax.swing.JToolTip*, siehe Abbildung 9.3

Spezielle Komponenten

- *javax.swing.JLabel* als externer Link, siehe Abbildung 9.3
- *org.jfree.chart.JFreeChart*, siehe Abbildung 9.2
- *org.xhtmlrenderer.simple.XHTMLPanel*, siehe Abbildung 9.3

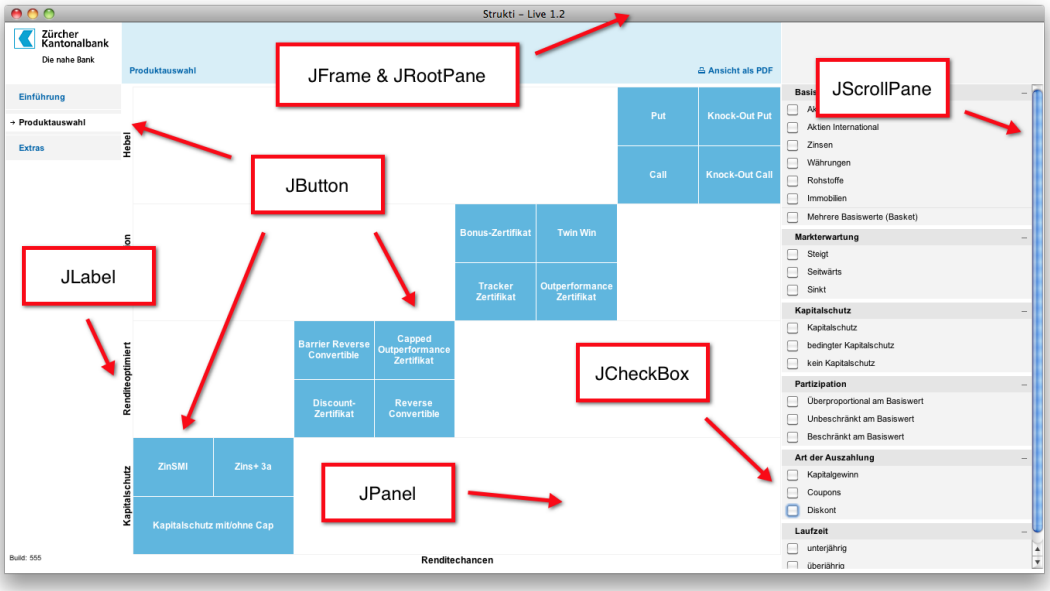

Abbildung 9.1.: Strukti Live 1.2 - Screenshot I

9.3.2. Gefundene Design-Patterns

Observer Die Menüführung wurde durch das Observer-Pattern, siehe [Wik11h], implementiert. Dabei wird in einem Modell der jeweilige Status der Applikation durch das Drücken eines *javax.swing.JButton* angepasst, wobei sich die jeweiligen *javax.swing.JPanel*, welche auf Änderungen des Modells höhren, sich neu zeichnen.

Abbildung 9.2.: Strukti Live 1.2 - Screenshot II

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

9.3.3. 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 Panel als Titelleiste funktioniert. In der Titelleiste kann über ein Minus- oder Plussymbol ein weiterer Panel ein- oder ausgeklappt werden. Zudem gibt es in der Titelleiste ein Info-Button, wo ein Dialog mit Hintergrundinformationen geöffnet werden kann.

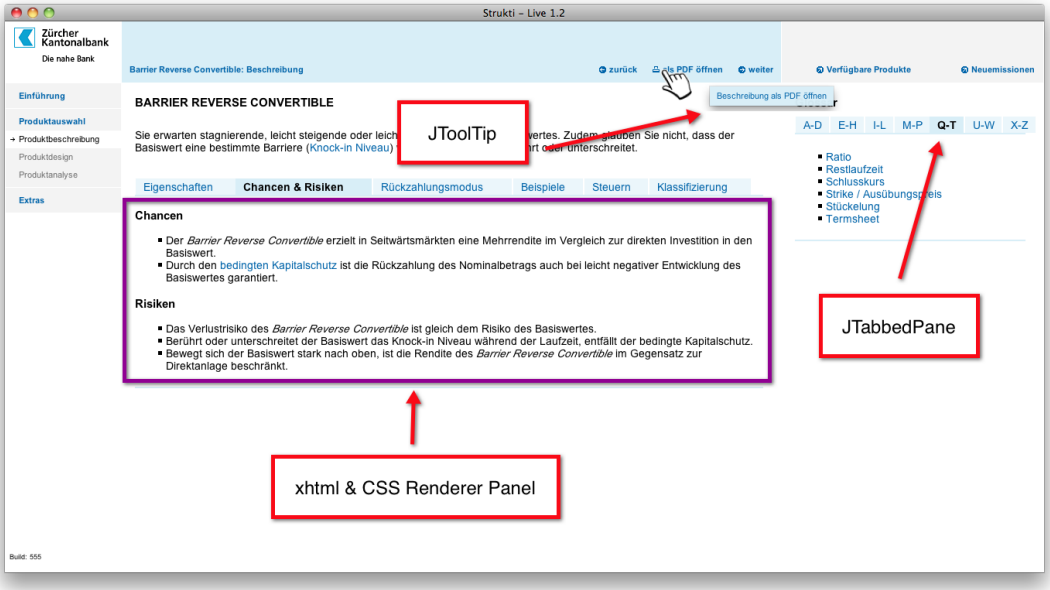

Abbildung 9.3.: Strukti Live 1.2 - Screenshot III

9.4. Strukti Online

Strukti Online ist ein Emissionstool der ZKB für strukturierte Produkte. Die Applikation wird aktuell nur zum 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.

In der Tabelle 9.3 sind alle verwendeten Bibliotheken, welche eine Interaktion mit dem GUI haben, ersichtlich, und ob deren Sourcecode zugänglich ist.

9.4.1. Gefundene Komponenten

Top-Level-Komponenten

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

Intermediate-Komponenten

- $javax.swing.JLayerdPane$
- $javax.swing.JPanel$

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 9.3.: Verwendete Bibliotheken von Strukti Online 2.10.0

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

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

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*
- *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*

9.4.2. Gefundene Design-Patterns

MVC Das Zusammenspiel von Model, View und Kontroller wurde strikte nach dem MVC-Design-Pattern implementiert, siehe [Amr08] S. 15ff.

Observer Einfache Aktuallisierungen zwischen Komponenten wurden strikte nach dem Observer-Design-Pattern implementiert, 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.

9.4.3. 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 Click angezeigt werden kann. Die JButtons können entsprechend ihrer Laufzeit (x-Achse) und der Underlyings[3] 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 Panel als Titelleiste funktioniert. In der Titelleiste kann über ein Minus- oder Plussymbol ein weiterer Panel ein- oder ausgeklappt werden.

[3] Auch Basiswerte genannt, siehe [Wik10a]

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

9.5. 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.

In der Tabelle 9.4 sind alle verwendeten Bibliotheken, welche eine Interaktion mit dem GUI haben, ersichtlich, und ob deren Sourcecode zugänglich ist.

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 9.4.: Verwendete Bibliotheken von Hedo Tool 1.0.710

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

9.5.1. 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*

9.5.2. Gefundene GUI Paradigmen

Design-Patterns

MVC Das Zusammenspiel von Model, View und Kontroller wurde strikte nach dem MVC-Design-Pattern implementiert, siehe [Amr08] S. 15ff.

Observer Einfache Aktuallisierungen zwischen Komponenten wurden strikte nach dem Observer-Design-Pattern implementiert, siehe [Amr08] S. 2ff.

Worker Asynchrone Jobs, die viel Rechenzeit in Anspruch nehmen, wurden mit $javax.swing.SwingWo$ $V, T >$ gelöst.

"neue" Komponenten

Es wurden keine "neuen" Komponenten indentifiziert.

9.6. Liste der Komponenten

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

9.6.1. Top-Level-Komponenten

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

9.6.2. Intermediate-Komponenten

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

9.6.3. Atomic-Komponenten

- $javax.swing.JButton$
- $javax.swing.JCheckbox$
- $javax.swing.JComboBox$
- $javax.swing.JFileChooser$
- $javax.swing.JFormattedTextField$
- $javax.swing.JLabel$
- $javax.swing.JMenuBar$
- $javax.swing.JMenu$
- $javax.swing.JMenuItem$
- $javax.swing.JPasswordField$
- $javax.swing.JProgressBar$
- $javax.swing.JRadioButton$
- $javax.swing.JSeparator$
- $javax.swing.JSlider$
- $javax.swing.JSpinner$
- $javax.swing.JTable$
- $javax.swing.JTextArea$
- $javax.swing.JTextField$
- $javax.swing.JTextPane$
- $javax.swing.JToolTip$

9.6.4. Spezielle Komponenten

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

9.7. Liste der GUI Paradigmen

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

9.7.1. Design-Patterns

MVC Siehe [Amr08] S. 15ff.

Observer 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.

Worker Asynchrone Jobs, die viel Rechenzeit in Anspruch nehmen, werden mit $javax.swing.SwingWo$
$V, T >$ gelöst.

9.7.2. "Neue" Komponenten

Button-Matrix JButtons werden in einer Matrix angeordnet, um sie entsprechend ihrer Funktion zu ordnen.

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

10. Evaluation der Java Web Frameworks

Dieses Kapitel behandelt die Evaluation der Java Web Frameworks. Es wird die kombinierte Methode aus dem Kapitel 8 (Methoden zur Entscheidungsfindung bei einer Evaluation, S. 27ff) angewendet. Mit der definition von sinnvollen Rahmenbedingungen soll die Evaluation durchgeführt und das Resultate in Form einer Rangliste vorgelegt werden.

10.1. Rahmenbedingungen für die Evaluierung

Die Rahmenbedingungen für die Evaluierung werden über Soll-Kriterien und KO-Kriterien festgelegt.

10.1.1. Soll-Kriterien

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}.

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 der Fragestellung heraus, haben sie 18 Anforderungen an ein Web Framework ausgearbeitet, welche öffentlich in einem Google-Doc[2] ersichtlich sind.

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

Folgende Anforderungen stammen aus dem Dokument "*18 Anforderungen an Web-frameworks - OpenDoc*", siehe [ap11], und werden im Anhang C (18 Anforderungen an Web Frameworks nach AgileLearn, S. 85ff) zusammengefasst und mit einer ID versehen.

10.1.2. KO-Kriterien

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}.

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.*" [3]

Dabei gibt es eine Hintertür:

"*Grundsätze sind verbindliche Vorgaben (Konventionen), von denen nur in begründeten Ausnahmen abgewichen werden kann.*" [4]

Das Dokument wurde analysiert und alle Grundsätze, welche für diese Diplomarbeit relevanten sind, werden im Anhang D (Grundsätze der ZKB IT-Architektur, S. 91ff) aufgelistet und mit einer ID versehen.

10.2. 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. Desshalb sollen die Soll-Kriterien in eine Hirarchie entsprechend ihrer Wichtigkeit für die ZKB aufgeteilt werden. Die Hirarchie wird in drei Gruppen aufgeteilt:

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

- Wichtig
- Nice to have
- Unwichtig

Es sollen nur die “wichtigen” Soll-Kriterien in die Evaluation mit einbezogen werden.

10.2.1. Wichtige Aspekte im Software-Lebenszyklus für die 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 10.1 liegen die Hauptkosten im Software-Lebenszyklus bei der Maintenance-Phase. Somit können die Kosten für requirements Engineering, Design, Programming und Integration vernachlässigt werden.

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

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

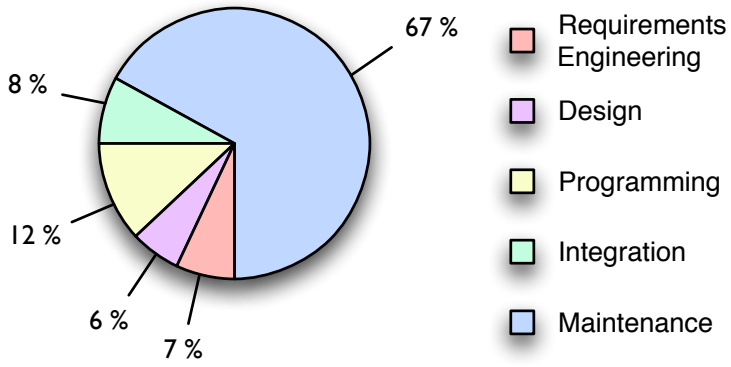

Abbildung 10.1.: Ungefähre relative Kosten der Phasen des Software-Lebenszyklus (nach [Sch99] S. 11 und [OB])

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 10.1 gezeigt.

Die Soll-Kriterien mit zwei oder mehr + werden als “wichtig”, mit einem + als “nice to have” und ohne + als “nicht wichtig” priorisiert.

Anforderung	(Si)	(Ko)	(Re)	(Im)
Soll-01 - Zugriffskontrolle (Authentifizierung/Authorisation/Rollenverwaltung)	++			
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 10.1.: 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

10.2.2. Wichtig

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

Soll-03 - Modulare Architektur Um wärend der Maintenance-Phase des Software-Lebenszyklus

notwendige Anpassungen an einer Applikation machen zu können, ist eine modulare Architekur von grossem Vorteil. Zudem ist das auch wärend der Entwicklungs-Phase hilfreich.

Soll-06 - Testing Um wärend 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 brillianten Dokumentation hat bestimmt höhere Chancen sich längerfristig am Markt durchzusetzten.

Soll-13 - Community Damit die notwendigen Ressourcen, für die Umsetztung 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.

10.2.3. Nice to have

Soll-02 - Form-Validierung Eine saubere Form-Validierung 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-Unterstuetzung Eine IDE-Unterstützung ist zu präferieren, soll aber kein Hinderniss für den Einsatz eines guten Web Frameworks sein.

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

10.2.4. Unwichtig

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 ebenbürdige 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 git es viele Object Relational Mapping (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 Mit Scaffolding und Rapid Prototyping kann in der Programmier-Phase des Software-Lebenszyklus die Arbeit erleichtert werden. In der Maintenance-Phase ergibt sich daraus kein Benefit.

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 Nachfrage nach Caching

Soll-11 - View-Engine Da Java eine Objekt Orientierte Sprache ist, wird das schon von der Sprache her unterstützt.

Soll-15 - Kosten fuer 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 fuer 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.

10.3. 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.

10.3.1. Begründung

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

- RIA - Frameworks
- MVC - Frameworks
- 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 den RIA Frameworks. Gemäss Kick-off Protokoll Beschluss soll dieses Framework als Alternative gewählt werden. Zudem wird die ULC, Canoo RIA Suite in den schweizer Banken Credit Suisse und UBS eingesetzt.

A-2 - Struts 1.3.10 mit ZIP-Framework

MVC - Framework, das in der ZKB bereits eingesetzt wird.

A-3 - Vaadin 6.5.7

Vaadin zählt zu den Java Script/HTML/CSS - Cross-Compiler Frameworks und baut auf Google Web Toolkit (GWT) auf.

A-4 - Apache Wicket 1.4.17

Ein Framework das in der ZKB bereits eingesetzt wird. An dieser Stelle ist zu erwähnen, dass der Einsatz aufgrund von Ausnahmegenehmigungen eingesetzt werden darf. Diese müssten im Falle einer erfolgreichen Evaluation ebenfalls beantragt werden.

10.4. KO-Kriterien die nicht beachtet werden sollen

Damit die Durchführung der Evaluation einen Sinn ergibt, sollen einige KO-Kriterien nicht beachtet werden. In Absprache mit dem Fachbetreuer der ZKB 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.

10.5. Prüfen der zu beachtenden KO-Kriterien

Es soll eine Prüfung aller Frameworks statt finden, ob es gegen definierte KO-Kriterien verstösst. Falls das der Fall ist, wird das Framework von der Evaluation ausgeschlossen. In Absprache mit dem Fachbetreuer der ZKB gibt es gewisse KO-Kriterien, die im Rahmen der Diplomarbeit ausser Kraft treten, diese sollen nicht beachtet werden.

10.5.1. A-1 - ULC, Canoo RIA Suite

Leider wurde wärend der Analyse festgestellt, dass die ULC, Canoo RIA Suite nur in einer Java Virtual Machine[5] lauffähig ist. Das wird über die Technik von Java-Web-Start[6] oder mit der Hilfe von einem Java-Applet[7] 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 8.1 wird diese Alternative aus der Evaluation ausgeschlossen.

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.

[5] Die Java Virtual Machine ist der Teil der JRE, der für die Ausführung des Java-Bytecodes verantwortlich ist, siehe [Wik11d].

[6] 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].

[7] Ein Java-Applet ist ein Computerprogramm, das in der Programmiersprache Java verfasst wurde und normalerweise in einem Webbrowser ausgeführt wird, siehe [Wik11c].

Falls sich die Grundsätze der IT-Architektur der ZKB in der Zunkunft 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.

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

Es wurde keine KO-Kriterien gefunden, die für diese Alternative zutreffen.

10.5.3. A-3 - Vaadin 6.5.7

Es wurde keine KO-Kriterien gefunden, die für diese Alternative zutreffen.

10.5.4. A-4 - Apache Wicket 1.4.17

Es wurde keine KO-Kriterien gefunden, die für diese Alternative zutreffen.

10.6. 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, vewendet werden. Das Resultat der Nutzwertanalyse soll dargestellt werden.

10.6.1. Bestimmen der Kriterien

Die Kriterien, welche in die Evaluation miteinbezogen werden, und somit für jede Alternative untersucht werden soll, sind die “wichtigen” Soll-Kriterien:

- Soll-01 - Zugriffskontrolle (Authentifizierung/Authorisation/Rollenverwaltung)
- Soll-03 - Modulare Architektur
- Soll-06 - Testing
- Soll-12 - Dokumentation
- Soll-13 - Community
- Soll-18 - AJAX-Unterstützung

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

Es werden alle "wichtigen" Soll-Kriterien miteinander verglichen. Die Vergleichsmatrix und die daraus resultierende Gewichtung ist in der Tabelle 10.2 und der Abbildung 10.2 ersichtlich. Für die Werte in der Vergleichsmatix wird die Skala der Vergleichsgrad, siehe Tabelle 8.3, verwendet.

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

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

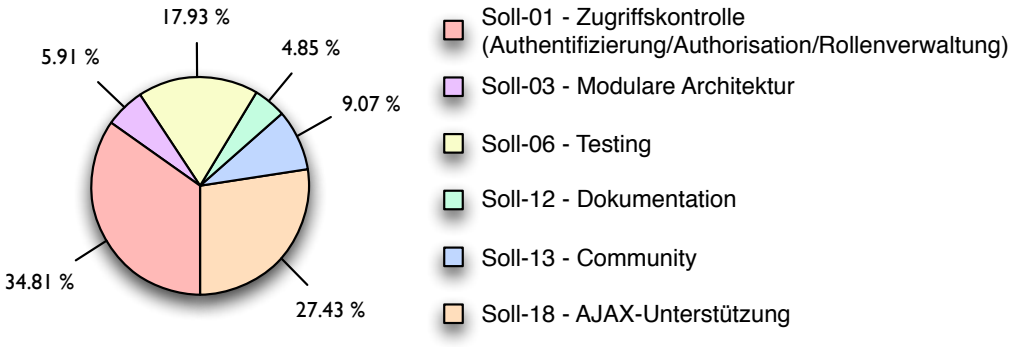

Abbildung 10.2.: Gewichtung der Soll-Kriterien nach der Methode des AHP

10.6.3. Bestimmen des Erfüllungsgrades

Für die Werte zur Bestimmung der Erfüllungsgrade wird die Skala der Erfüllungsgrade, siehe Tabelle 8.1, verwendet.

10.7. Resultat

10.7.1. A-1 - ULC, Canoo RIA Suite

Wurde aufgrund der KO-Kriterien KO-18 und KO-19 aus der Evaluation ausgeschlossen.

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

Siehe Tabelle 10.3.

Kriterien	Gewichtung g	Erfüllungsgrad e	Wertigkeit $_{A-2}$
Soll-01	34.81 %	x	y
Soll-03	5.91 %	x	y
Soll-06	17.93 %	x	y
Soll-12	4.85 %	x	y
Soll-13	9.07 %	x	y
Soll-18	27.43 %	x	y
Ergebnis	100.00 %		z

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

10.7.3. A-3 - Vaadin 6.5.7

Siehe Tabelle 10.4.

10.7.4. A-4 - Apache Wicket 1.4.17

Siehe Tabelle 10.5.

Kriterien	Gewichtung g	Erfüllungsgrad e	Wertigkeit $_{A-3}$
Soll-01	34.81 %	x	y
Soll-03	5.91 %	x	y
Soll-06	17.93 %	x	y
Soll-12	4.85 %	x	y
Soll-13	9.07 %	x	y
Soll-18	27.43 %	x	y
Ergebnis	100.00 %		z

Tabelle 10.4.: Nutzwertanalyse der Alternative A-3 - Vaadin 6.5.7

Kriterien	Gewichtung g	Erfüllungsgrad e	Wertigkeit $_{A-4}$
Soll-01	34.81 %	x	y
Soll-03	5.91 %	x	y
Soll-06	17.93 %	x	y
Soll-12	4.85 %	x	y
Soll-13	9.07 %	x	y
Soll-18	27.43 %	x	y
Ergebnis	100.00 %		z

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

11. Integration in die ZKB Infrastruktur möglich?

12. Implementierung der gefundenen Swingkomponenten möglich?

13. Proof of Concept - Umsetzung durch einen Prototypen

14. Empfehlung für ein Java Web Framework

15. Fazit und Ausblick

16. Reflektion

Die Arbeit war in meinen Auge ein voller Erfolg, da ich alle Ziele erreicht und viel gelernt habe. Ich bin bereit für einen neuen Lebensabschnitt.

Wieso NWA und AHP?

Wieso nur drei Swing Applikationen?

Wieso nur vier Java Web Frameworks?

Wieso die 18 Anforderungen gemäss AgileLearn

Wieso Sicherheit, Kosten der Maintenance, Ressourcen und Image?

Die Evaluation hätte auch anders kommen können.

Wieso sind keine Lasttests gemacht worden, resp. die Skalierung nicht angeschaut worden.

A. 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.

A.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.

A.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.

A.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.

A.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.

A.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.

B. 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.

B.1. Detailanalyse der Aufgabenstelltung

In der Detailanalyse der Aufgabenstellung werden die definierten Aufgaben der Aufgabenstellung untersucht und deren Resultate ausgearbeitet. Aufgrund dieser Erkenntnisse wird eine Grobplanung festgelegt. Damit die Nachvollziehbarkeit garantiert ist, werden die Termine, Meilensteine und Arbeitsschritte aufgelistet.

Aufgabe - A1 *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.

Aufgabe - A2 *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.

Aufgabe - A3 *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 miteinbezogen 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 Rahmenbedigungen 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.

Aufgabe - A4 *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.

Aufgabe - A5 *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.

Aufgabe - A6 *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 Paradigment 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.

B.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 Reflektion, soll der effektiv geleistete Aufwand aufgezeigt werden.

B.2.1. Grobplanung

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

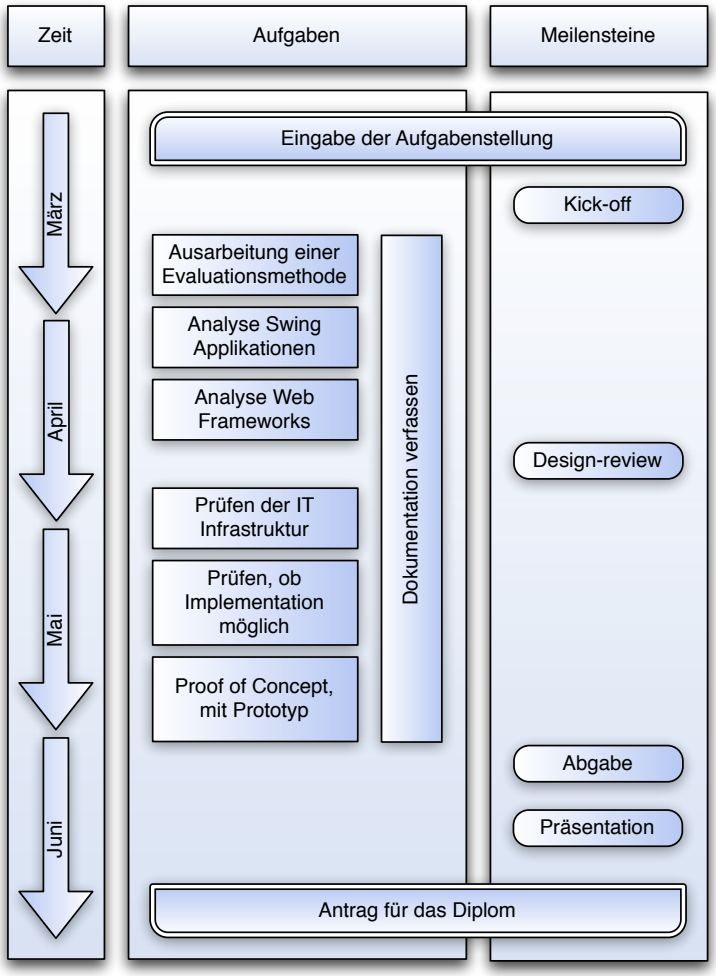

Abbildung B.1.: Chronologischer Ablauf der Diplomarbeit

B.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 B.1 ersichtlich.

Arbeitspaket	Geplant	Effektiv
Präsentationen	20.0 h	...
Gestellte Aufgaben gemäss Aufgabenstellung	144.0 h	...
Abzugebende Dokumente	94.0 h	...
Fachbetreuung durch den Dozenten	10.0 h	...
Administrative Aufgaben	13.5 h	...
Total	281.5 h	...

Tabelle B.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 B.2 ersichtlich.

Arbeitspaket	Geplant	Effektiv
Kick-off Präsentation ca. 5 Minuten	2.5 h	4.0 h
Design-review Präsentation ca. 10 Minuten	5.0 h	...
Schlusspräsentation ca. 20 Minuten	10.0 h	...
Prototyp Demo ca. 5 Minuten	2.5 h	...
Total	20.0 h	...

Tabelle B.2.: Geplante Zeit für Präsentationen

Gestellte Aufgaben gemäss Aufgabenstellung

Die einzelnen Aufgaben sollen in deren Teilaufgaben unterteilt werden. Die Planung ist in der Tabelle B.3 ersichtlich.

Aufgabe	Arbeitspaket	Geplant	Effektiv
Aufgabe - A1	Research zu Analyseverfahren von Java Swing Applikationen	8.0 h	...
	Definition des Analyseverfahrens	4.0 h	...
	Analyse von drei Java Swing Applikationen, à 4 Stunden	12.0 h	...
Aufgabe - A2	Research zu Java Swing Komponenten	4.0 h	...
	Analyse von drei Java Swing Applikationen, à 2 Stunden	6.0 h	...
Aufgabe - A3	Research zu Evaluationsverfahren im Bereich von Java Web Frameworks	12.0 h	...
	Definition des Evaluationsverfahrens	8.0 h	...
	Evaluation von vier Java Web Frameworks, à 5 Stunden	20.0 h	...
Aufgabe - A4	Analyse der IT Architektur der ZKB	8.0 h	...
	Prüfen ob die Integration möglich ist bei vier Java Web Frameworks, à 2.5 Stunden	10.0 h	...
Aufgabe - A5	Analyse der Komponenten von vier Java Web Frameworks und prüfen ob die Implementation möglich ist, à 3 Stunden	12.0 h	...
Aufgabe - A6	Anforderungen an den Prototyp definieren	8.0 h	...
	Testfälle für den Prototyp definieren	8.0 h	...
	Umsetzung des Prototypen	20.0 h	...
	Empfehlung eines Java Web Frameworks	4.0 h	...
	Total	144.0 h	...

Tabelle B.3.: Geplante Zeit für gestellte Aufgaben

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 B.4 ersichtlich.

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

Tabelle B.4.: 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 B.5 ersichtlich.

Arbeitspaket	Geplant	Effektiv
Kick-off Meeting	1.0 h	1.0 h
Design-review Meeting	1.0 h	...
Schlusspräsentation	1.0 h	...
Ausserterminliche Betreeuungszeit	7.0 h	...
Total	10.0 h	...

Tabelle B.5.: Geplante Zeit für Betreuung

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.

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

Tabelle B.6.: Geplante Zeit für administrative Aufgaben

B.3. Arbeitsschritte

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

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/

B.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 B.7.

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
xx. xx. 2011	Abgabe der Dokumentation	HSZ-T
xx. xx. 2011	Schlusspräsentation	HSZ-T

Tabelle B.7.: Projekt Meilensteine

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

B.5. Erreichte Ziele

Es wurden alle Ziele gemäss den erwarteten Resultaten der Aufgabenstellung erreicht. Die einzelnen Punkte sind hier analog der Aufgabenstellung aufgeführt, siehe Tabelle B.8.

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.	offen
Resultat - R4	Es sollen die Java Web Frameworks aufgelistet werden, welche für einen Einsatz in der IT Infrastruktur der Zürcher Kantonalbank in Frage kommen.	offen
Resultat - R5	Es sollen die Java Web Frameworks aufgelistet werden, welche die notwendigen Swingkomponenten und GUI Paradigmen unterstützen.	offen
Resultat - R6	Ein lauffähiger Prototyp.	offen
Resultat - R7	Es soll eine Empfehlung eines Java Web Frameworks, für den möglichen Einsatz in der Zürcher Kantonalbank, ausgesprochen werden.	offen

Tabelle B.8.: Übersicht der erreichten Ziele

C. 18 Anforderungen an Web Frameworks nach AgileLearn

Folgende Anforderungen stammen aus dem Dokument "*18 Anforderungen an Webframeworks - OpenDoc*", siehe [ap11], und werden hier zusammengefasst.

C.1. Auflistung der Anforderungen

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 (Authentifizierung/Authorisation/Rollenverwaltung) 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 Proto-

typing, 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.

D. Grundsätze der ZKB IT-Architektur

Die Grundsätze sind aus dem *Handbuch der IT-Architektur*, siehe [uS10], der ZKB entnommen.

D.1. Auflistung der Grundsätze

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.

[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

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

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.

[9] [uS10] Kapitel 12.2.1 - *Einsatz von Frameworks*, Seite 140
[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

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.

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-/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.

[20] [uS10] Kapitel 12.3.7 - *Client-Technologien*, Seite 144
[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

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.

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“.

[30] [uS10] Kapitel 12.4.3 - *Einsatz von Enterprise Java Beans*, Seite 150
[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

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.

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.

[40] [uS10] Kapitel 13.13 - *Einsatz von Open Source Software*, Seite 206
[41] [uS10] Kapitel 13.13 - *Einsatz von Open Source Software*, Seite 206
[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

E. Kick-off Protokoll

E.1. Anwesende Personen

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

Tabelle E.1.: Anwesende Personen

E.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.

F. Design-review Protokoll

F.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 F.1.: Anwesende Personen

F.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 6.5.7
  - 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.

G. Abkürzungsverzeichnis

AHP	Analytic Hierarchy Process
Ajax	Asynchronous JavaScript and XML
CORBA	Common Object Request Broker Architecture
EBS	Einschreibe- und Bewertungssystem der HSZ-T
EDT	Event Dispatch Thread
GUI	Grafische Benutzeroberfläche
GWT	Google Web Toolkit
HSZ-T	Hochschule für Technik Zürich
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Integrierte Entwicklungsumgebung
JDBC	Java Database Connectivity
JFC	Java Foundation Classes
JRE	Java Runtime Environment
MVC	Model View Controller
MVP	Model View Presenter
NWA	Nutzwertanalyse
OO	Objektorientierung
ORM	Object Relational Mapping
RDBMS	Relational Database Management System
RIA	Rich Internet Application
RMI	Remote Method Invocation
SOAP	Simple Object Access Protocol
UML	Unified Modeling Language
URL	Uniform Resource Locator
ZIP	ZKB Internet Plattform
ZKB	Zürcher Kantonalbank
ZSI	Zentrale Server Infrastruktur

H. Abbildungsverzeichnis

I. Tabellenverzeichnis

J. Listingverzeichnis

K. 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].

[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.

[Fou10] The Apache Software Foundation. The apache tomcat connector. http://tomcat.apache.org/connectors-doc/, 2010. [Online; 21. April 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].

[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].

[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.

[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.

[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].

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

[Wik11c] Wikipedia. Java applet. http://de.wikipedia.org/w/index.php?title=Java-Applet&oldid=86761109, März 2011. [Online; 08. Mai 2011].

[Wik11d] Wikipedia. Java virtual machine. http://de.wikipedia.org/w/index.php?title=Java_Virtual_Machine&oldid=86955036, März 2011. [Online; 08. Mai 2011].

[Wik11e] Wikipedia. Median. http://de.wikipedia.org/w/index.php?title=Median&oldid=87404521, April 2011. [Online; 08. Mai 2011].

[Wik11f] Wikipedia. Mittelwert. http://de.wikipedia.org/w/index.php?title=Mittelwert&oldid=87198455, April 2011. [Online; 08. Mai 2011].

[Wik11g] Wikipedia. Nutzwertanalyse. http://de.wikipedia.org/w/index.php?title=Nutzwertanalyse&oldid=86980368, März 2011. [Online; 31. März 2011].

[Wik11h] Wikipedia. Observer (entwurfsmuster). http://de.wikipedia.org/w/index.php?title=Observer_(Entwurfsmuster)&oldid=86993850, März 2011. [Online; 04. April 2011].

[Wik11i] Wikipedia. Rich internet application. http://de.wikipedia.org/w/index.php?title=Rich_Internet_Application&oldid=86143678, März 2011. [Online; 05. April 2011].

[Wik11j] Wikipedia. Rich internet application. http://en.wikipedia.org/w/index.php?title=Rich_Internet_application&oldid=417575667, März 2011. [Online; 05. April 2011].

[Wik11k] Wikipedia. Sensitivitätsanalyse. http://de.wikipedia.org/w/index.php?title=Sensitivit%C3%A4tsanalyse&oldid=87406951, April 2011. [Online; 08. Mai 2011].

[Wik11l] 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.