Bachelorarbeit

Fahim Aleaf

Entwicklung eines Reportmoduls für
Ad-hoc-Abfragen als Komponente eines
webbasierten Open Source Vertriebssystems

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Fahim Aleaf

Entwicklung eines Reportmoduls für Ad-hoc-Abfragen als Komponente eines webbasierten Open Source Vertriebssystems

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Jörg Raasch
Zweitgutachter: Prof. Dr. rer. nat. Bettina Buth

Kooperation mit Firma: novomind AG

Abgegeben am 14. November 2008

Fahim Aleaf

Thema der Bachelorarbeit

Entwicklung eines Reportmoduls für Ad-hoc-Abfragen als Komponente eines webbasierten Open Source Vertriebssystems

Stichworte

CRM, EJB, Java EE, JSF, Open-Source, Reporting, Testen, Vertriebssystem

Kurzzusammenfassung

Diese Arbeit beschäftigt sich mit der Weiterentwicklung eines webbasierten Vertriebssystems. Dabei soll unter Berücksichtigung aller Anforderungen auf prototypischer Weise ein Reportingmodul entwickelt werden, um dem bisherigen Vertriebssystem Analysefunktionen bereitzustellen. Zu diesen Funktionen gehören u. a. Kundenauswertungen nach benutzerdefinierten Kriterien, Exportmöglichkeiten für operative Daten und die Erstellung von Statistiken.

Fahim Aleaf

Title of the paper

Development of an ad hoc reporting module as a component of a webbased open source distribution system

Keywords

CRM, EJB, Java EE, JSF, Open-Source, Reporting, Data Warehouse-System, Testing, distribution system

Abstract

This thesis is about the further development of a webbased distribution system. In consideration of all requirements, the goal of this prototypical devolopment is to supply analytical functions to the previous distribution system. Part of the features of this modul are customersanalysis on ad hoc criteria, exportoptions for operational data and ascertain statistics.

Danksagung

Allen, die mich bei dieser Arbeit unterstützt haben, sei herzlich gedankt.

Insbesondere Prof. Dr. Jörg Raasch für die Anregungen, die vielen wertvollen Gespräche und Diskussionen und das wichtige Know-how in der Projektarbeit. Prof. Dr. Bettina Buth danke ich für die Betreuung als Zweitprüfer und die exzellente Vermittlung von Wissen in den Bereichen Software Engineering und Testing.

Ein ganz besonderer Dank geht an Herrn Peter Samuelsen für die Ermöglichung dieser Projektarbeit sowie die kritische Durcharbeitung meiner Ergebnisse und die vielen Hinweise und Verbesserungsvorschläge. Auch bei allen Mitarbeitern der novomind AG möchte ich mich bedanken, die mir stets mit Rat und Tat zur Seite standen.

Weiteren Dank schulde ich allen Kommilitonen unseres Semesters, insbesondere

Herrn Mehdi Afridi, Herrn Mahmut Cevik, Herrn Markus Wernicke, Herrn Ahmet Inci, Herrn Dennis Winter, Herrn Sahabettin Alkan, Herrn Ali Kilic, Herrn André Schmer, Herrn Remzi Musici

für ihre Hilfsbereitschaft sowie für die freundliche Arbeitsatmosphäre während meines Studiums.

Weiterhin möchte ich mich auch bei meinen Eltern bedanken, die immer für mich da waren und mir jederzeit während des Studiums den Rücken stärkten.

Mein besonderer Dank geht an meine Freundin Magda Sadlucki für ihren kritischen Blick auf den Text und ihre Geduld bei meiner Abwesenheit am Computer.

Martin Sadowski und André Schmer: Euch danke ich für die gute Zeit! Ohne eure Unterstützung wäre mir der Einstieg in das Projekt nicht leicht gefallen.

Fahim Aleaf, November 2008

Inhaltsverzeichnis

Abbildungsverzeichnis **8**

Tabellenverzeichnis **9**

Quelltextverzeichnis **10**

1 Einleitung **11**
1.1 Motivation . . . 11
1.2 Problemstellung und Zielsetzung . . . 11
1.3 Umfeld . . . 12
1.4 Gliederung der Arbeit . . . 12

2 Grundlagen **14**
2.1 Betriebswirtschaftliches Konzept . . . 14
2.1.1 Customer Relationship Management . . . 14
2.1.2 Relationship Marketing . . . 15
2.1.2.1 Sicherheitsstreben . . . 15
2.1.2.2 Unternehmenswachstum . . . 16
2.1.2.3 Gewinn . . . 17
2.1.2.4 Fazit . . . 18
2.1.3 Reporting . . . 19
2.1.3.1 Historie . . . 20
2.1.3.2 Data Warehouse-System . . . 21
2.1.3.3 Betriebliche Statistik-Reports . . . 23
2.1.3.4 Betriebliche Daten-Reports . . . 23
2.1.3.5 Fortgeschrittene Anforderungen . . . 24
2.2 Realisierungskonzept . . . 25
2.2.1 Architektur . . . 26
2.2.1.1 Container . . . 27
2.2.1.2 Application Components . . . 27
2.2.1.3 Services . . . 27
2.2.2 Verwendete Implementierung . . . 28
2.2.3 Erwähnenswerte Services/APIs . . . 28

2.2.3.1 JavaServer Faces (JSF) 28
2.2.3.2 Java Naming Directory Interface (JNDI) 29
2.2.3.3 Java Persistence API (JPA) 29

3 Fachkonzept **32**
3.1 Bisheriges System 32
3.1.1 Bewertung aus Benutzersicht 32
3.1.2 Bewertung aus Entwicklersicht 33
3.2 Anforderungsanalyse 34
3.2.1 Fachliche Anforderungen 34
3.2.2 Technische Anforderungen 39
3.2.2.1 Gesamtsystem 40
3.2.2.2 Report Modul 40
3.3 Anforderungsspezifikation 40
3.3.1 Funktionale Anforderungen 41
3.3.2 Nichtfunktionale Anforderungen und Qualitätskriterien 43

4 Marktanalyse **45**
4.1 Untersuchung existierender CRM-Systeme 45
4.1.1 Salesforce Professional Edition 46
4.1.2 SugarCRM Enterprise Edition 47
4.1.3 vTiger CRM 49
4.2 Untersuchung existierender Reporting Tools 51
4.2.1 JasperReports 51
4.2.2 Pentaho Reporting (ehemals JFreeReport) 52
4.2.3 DataVision 54
4.3 Fazit 55
4.3.1 Zusammenfassung CRM-Systeme 55
4.3.2 Zusammenfassung Reporting Tools 56

5 Vision **58**
5.1 Kurzfristige Ziele 58
5.2 Zukünftige Ziele 59

6 Architektur **61**
6.1 Fachlich 62
6.1.1 Datenebene 62
6.1.2 Business Logik-Ebene 66
6.1.3 Darstellungsebene 69
6.2 Technisch 69

7 Realisierung **71**

7.1 Systembeschreibung . . . 71
7.1.1 Selektionsmodul . . . 71
7.1.1.1 Lösung . . . 71
7.1.2 Statistikmodul . . . 76
7.1.2.1 Lösung . . . 76
7.1.3 Exportmodul . . . 78
7.1.3.1 Lösung . . . 79
7.2 Prototypischer Entwicklungsprozess . . . 82
7.2.1 Ziele . . . 82
7.2.2 Spezifikation . . . 83
7.2.3 Machbarkeitsbeweis . . . 85

8 Testkonzept **87**
8.1 Planung . . . 88
8.2 Analyse . . . 89
8.3 Realisierung . . . 91
8.4 Auswertung . . . 92
8.4.1 SQL-Anweisungen und Dateninkonsistenz . . . 94
8.4.2 Export und Betriebssystemabhängigkeit . . . 94
8.4.3 Selektionen und Grenzwerte . . . 94
8.4.4 Zusammenfassung von Schlüsseltabellen . . . 95
8.4.5 Navigation und Performance . . . 95

9 Resümee **98**
9.1 Zusammenfassung . . . 98
9.2 Ausblick . . . 99
9.3 Methodische Abstraktion . . . 100

A Die zugehörige CD **101**

Glossar **102**

Abkürzungsverzeichnis **104**

Literaturverzeichnis **105**

Abbildungsverzeichnis

2.1 Kundenbindungs(miss)erfolge in der New Economy 18
2.2 Data Warehouse-System 22
2.3 Prozesse bei einem Data Warehouse-System 25
2.4 Übersicht der Java EE 5 Architektur 26
2.5 JavaServer Faces Request/Response-Zyklus 29

3.1 Anwendungsfälle für das Modul Reporting 35

4.1 Reporting in Salesforce 47
4.2 Reporting in SugarCRM 48
4.3 Reporting in vTiger CRM 50
4.4 Ablauf des Reportings bei JasperReports 52
4.5 Beispiel-Report erstellt durch Pentaho Reporting 53
4.6 Report Designer Werkzeug von DataVision 55

6.1 ER-Diagramm des Reporting Moduls 63
6.2 OR-Mapping der Datenbanktabellen 67
6.3 Klassendiagramm des Report Moduls 68
6.4 Verteilungsdiagramm des Systems 70

7.1 Interaktionen zwischen Benutzer und System beim Erstellen von Selektionen 72
7.2 Erstellung einer Selektion. Schritt 1: Wählen des Reporttypes. 74
7.3 Erstellung einer Selektion. Schritt 2: Zusammensetzen der Kriterien. 74
7.4 Erstellung einer Selektion. Schritt 3+4: Wahl des Layouts und Export. 75
7.5 Aktivitäten bei der Durchführung von Statistiken 77
7.6 Admin-Bereich: Einsehen von Statistiken 77
7.7 Benutzerstatistiken im Statistikmodul 78
7.8 Exportieren von Kontaktdaten in Office-Vorlagen 80
7.9 Hinzufügen, Entfernen und Editieren von Office-Vorlagen 81
7.10 Vorgehensweise im Projekt 84

8.1 Test-Qualität und Software-Qualität 87
8.2 Fundamentaler Testprozess nach dem ISTQB 88
8.3 Monitoring und Messungen mit dem Testwerkzeug Glassbox 93

Tabellenverzeichnis

7.1 Platzhalter für Microsoft Office Vorlagen 82

8.1 Logischer Testfall Selektion (Teil 1) 90
8.2 Logischer Testfall Selektion (Teil 2) 90
8.3 Logischer Testfall Selektion (IS ONE OF Beziehung) 90
8.4 Konkreter Testfall Selektion (Teil 1) 91
8.5 Konkreter Testfall Selektion (Teil 2) 91
8.6 Konkreter Testfall Selektion (IS ONE OF Beziehung) 91
8.7 Minimale Mehrfachbedingungsüberdeckung der Operatoren bei Selektionen . 92

Quelltextverzeichnis

2.1 Beispiel des EntityManagers 30
2.2 Beispiel von EntityManager Queries 31
2.3 SQL vs. JPQL 31

6.1 Beispieldaten der EnumerationType-Tabelle 64
6.2 Beispieldaten der Report-Tabelle 64
6.3 Beispieldaten der ReportProperties-Tabelle 65
6.4 Beispieldaten der ReportLayout-Tabelle 65
6.5 Beispieldaten der LayoutAttribute-Tabelle 65
6.6 Beispieldaten der Tenant-Tabelle 66
6.7 Beispieldaten der Employee-Tabelle 66

7.1 Erstellen einer SQL-Query mittels der Criteria API 76

1 Einleitung

1.1 Motivation

Vor allem bei wettbewerbsintensiven Märkten entscheidet ein effizientes Kundenmanagement über Erfolg oder Misserfolg eines Unternehmens. Der erste Schritt zum Erfolg ist demnach eine Anpassung der Geschäftsprozesse an die Kundenwünsche. Für die novomind AG heißt das, die seit Jahren eingesetzte Standardsoftware durch ein innovatives webbasiertes CRM-System abzulösen. Als Ergebnis entstand ein operatives CRM-System, welches in nächster Zeit als Open Source Software veröffentlicht werden soll.
Bevor diese Software veröffentlicht wird, benötigt dieses System analytische Fähigkeiten für die vorhandenen operativen Daten. Damit das CRM-System diese Funktionen unterstützt, wird ein innovatives Berichtswesen benötigt, welches auf die Prozessabläufe und Wünsche der Anwender angepasst wird. Im Gegensatz zu der zuvor genutzten Standardsoftware soll dieses Berichtswesen durch ein benutzerfreundliches User Interface dem Anwender ermöglichen, komplexe Auswertungen auf bequeme Art und in kurzer Zeit durchzuführen.

1.2 Problemstellung und Zielsetzung

Das Ziel dieser Arbeit ist, das bisher entwickelte CRM-System um ein Bereich Reporting zu erweitern. Dazu muss zunächst ein betriebswirtschaftliches Konzept erarbeitet werden, um alle Anforderungen der Bereiche Vertrieb, Marketing und Controlling innerhalb der novomind AG zu verstehen. Durch dieses Wissen soll in einer anschließenden Marktanalyse untersucht werden, wie der Bereich Reporting erfolgreicher CRM-Systeme realisiert wurde.
Als Ergebnis soll ein Reportmodul entstehen, das für die Geschäftsprozesse der novomind AG optimert ist. Dazu soll aus architektonischer Sicht Platz für Erweiterungsmöglichkeiten geschaffen werden, um zukünftige Anforderungen mit wenig Aufwand zu realisieren.
Um dieses Ziel erreichen zu können, müssen einige Bedingungen eingehalten werden. Zum Beispiel darf die bestehende 3-Schichten-Architektur des CRM-Systems durch die Erweiterung des Reportmoduls nicht verändert werden. Zudem soll diese Erweiterung nahtlos in das webbasierte CRM-System integriert werden. Das heißt, dass das CRM-System für den Benutzer optisch identisch aussieht, jedoch um analytische Funktionen ergänzt wird.

1.3 Umfeld

Diese Arbeit ist die letzte von drei Arbeiten, die im Hause der novomind AG entstanden ist. In der ersten Arbeit von Martin Sadowski (Sadowski, 2008) wurde das neue Vertriebssystem konzipiert und in prototypischer Vorgehensweise entwickelt. Die Arbeit von André Schmer (Schmer, 2008) befasst sich mit der Weiterentwicklung, Evaluierung und Einführung dieses Systems. Um alle Anforderungen eines analytischen CRM-Systems zu erfüllen, beschäftigt sich diese Arbeit, im Gegensatz zu den Vorhergehenden, mit einem fachspezifischen Bereich (vgl. Sadowski, 2008, S.14).

Die novomind AG besteht seit 1999 und entwickelt Software für eine prozess- und kostenoptimierende Kundenkommunikation. Dazu gehören unter anderem Email-Management-Systeme, virtuelle Berater sowie Systeme zur interaktiven Echtzeitkommunikation im Internet, die in nahezu allen Branchen zum Einsatz kommen. Zusätzlich realisiert und betreut die novomind AG im Bereich Services komplexe E-Business-Applikationen für namenhafte Unternehmen. Zu den Kunden gehören mittelständische sowie Großunternehmen aller Branchen wie z. B. das Bundesministerium und Otto.

Nachdem die genannten Ziele erreicht wurden, wird die Veröffentlichung des CRM-Systems als Open Source Software angestrebt.

1.4 Gliederung der Arbeit

Im zweiten Kapitel *Grundlagen* werden die Konzepte des Reportmoduls aus betriebswirtschaftlicher und technischer Sicht beschrieben. Dabei wird ein Wissen vermittelt, um die Anforderungen an ein Reportmodul aus vertrieblicher Sicht verständlich zu machen. Zusätzlich werden einige Technologien beschrieben, die in dieser Arbeit Verwendung finden.

Im darauffolgenden Kapitel wird im *Fachkonzept* das alte Vertriebssystem analysiert. Anschließend werden die Anforderungen der Benutzer und Führungskräfte anhand von Szenarien dokumentiert und analysiert. Das Ergebnis ist eine Anforderungsspezifikation, die den vertraglich vereinbarten Umfang des Reportmoduls beschreibt.

Um ein gebrauchstaugliches Modul zu entwickeln, werden im Kapitel *Marktanalyse* einige erfolgreiche CRM-Systeme untersucht und miteinander verglichen. Anschließend werden einige Open Source Bibliotheken näher untersucht, die bei der Entwicklung des Moduls behilflich sein könnten.

Nachdem in Kapitel fünf das Ziel dieser Arbeit in einer kurzfristigen und zukünftigen *Vision* dargestellt wird, wird in Kapitel sechs die *Architektur* aus fachlicher und technischer Sicht

beschrieben. Dabei werden anhand von UML-Diagrammen die einzelnen Schichten der Architektur näher erläutert.

Im Kapitel *Realisierung* werden die Ergebnisse der Entwicklung präsentiert. Dabei werden die notwendigen Schritte zur Erstellung eines Reports anhand von Screenshots visualisiert. Zudem wird in diesem Kapitel die prototypische Vorgehensweise beschrieben und der Erfolg in diesem Projekt bewertet.

Bevor im letzten Kapitel das Ergebnis dieser Arbeit reflektiert wird und Anregungen für eine Weiterentwicklung gegeben werden, wird im Kapitel *Testkonzept* ein Testplan aufgestellt und durchgeführt, um die Qualität des Gesamtsystems zu messen. Dazu wurde unter anderem das System durch ein Testwerkzeug analysiert, um potentielle Verbesserungsmöglichkeiten zu finden.

2 Grundlagen

Bevor in den folgenden Kapiteln auf die Anforderungen und Spezifikationen von Reports eingegangen wird, sollen in diesem Kapitel zunächst einige Grundlagen vermittelt werden. Im ersten Teil dieses Kapitels wird auf Customer-Relationship-Management (CRM) im Allgemeinen eingegangen. Danach wird der Bereich Marketing behandelt, damit Sinn und Zweck sowie die Anwendungsgebiete von Reports deutlich werden. Im dritten Teil dieses Kapitels wird der Begriff Report erörtert. Nach der Begriffsabgrenzung folgt die Beschreibung von Data Warehouse-Systemen.

2.1 Betriebswirtschaftliches Konzept

2.1.1 Customer Relationship Management

Da das Kundenbeziehungsmanagement (CRM) den Grundstein für das bisher entwickelte CRM-System bildet, sollen an dieser Stelle einige Grundlagen aufgegriffen werden. Für eine ausführlichere Beschreibung soll auf die Arbeit von Martin Sadowski verwiesen werden (Sadowski, 2008, S.13ff).

Nahezu alle Unternehmen heben heutzutage die Kundenorientierung als zentralen Leitgedanken ihres Handelns hervor. Dabei wird in der Umsetzung der Kundenorientierung oft der Begriff Customer Relationship Management (CRM) verwendet. CRM ist ein Teil des Relationship Marketing und steht dabei in der Praxis für ein informationstechnisches Konzept, welches dazu dient, Kundenbeziehungen mit Hilfe von Software zu analysieren und zu steuern (vgl. Hippner und Wilde, 2007).

Insbesondere verspricht man sich durch den Einsatz von CRM-Systemen eine strukturierte

- Analyse,
- Planung,
- Durchführung und Kontrolle,

- Initiierung,
- Stabilisierung,
- Intensivisierung und Wiederaufnahme

von Geschäftsbeziehungen zu den Kunden (vgl. Höschel, 2007, S.13) (vgl. Diller, 2007).

2.1.2 Relationship Marketing

Innerhalb der letzten Jahrzehnte fand ein Wechsel von einer transaktions- zu einer beziehungsorientierten Sichtweise im Bereich Marketing statt. Beim Transaktionsmarketing geht es darum, ein Produkt bzw. eine Dienstleistung mit den „4 Ps“ (Produce, Price, Promotion, Place) zu vermarkten, ohne auf die Wünsche des Kunden einzugehen. Ziel dieser Marketingmethode ist, innerhalb kürzester Zeit einen Umsatzerfolg zu erwirtschaften (vgl. Diller, 2007, S.99).
Das Beziehungsmarketing (Relationship Marketing) dagegen verfolgt das Ziel, alle Unternehmensaktivitäten an den Wünschen und Bedürfnissen des Kunden auszurichten. Hauptgrund für diesen Wandel ist der immer stärker werdende Wettbewerb in nahezu allen Branchen (vgl. Bruhn, 2007, S.6).

Durch eine kundenorientierte Unternehmensstrategie entstehen Vor- und Nachteile für das Unternehmen. Grob gesehen kann man die Absichten eines Unternehmens im Bereich Marketing wie folgt aufteilen (vgl. Diller, 2007, S.101ff):

- Mehr Sicherheit für das Unternehmen
- Mehr Wachstum für das Unternehmen
- Mehr Gewinn für das Unternehmen

Im Folgenden soll erörtert werden, inwieweit das Beziehungsmarketing die oben genannten ökonomischen Effekte unterstützt.

2.1.2.1 Sicherheitsstreben

Um ein Unternehmen erfolgreich zu führen, bedarf es mehr als einer nützlichen Idee. Es muss sichergestellt werden, dass das Unternehmen langfristig Erfolg erzielt. Diese Erfolge entstehen in der Regel durch langfristige Bindungen mit Kunden.
Durch eine verstärkte Kundenbindung entsteht ein *Immunisierungseffekt*. Da der Kunde im ständigen Kontakt mit dem Unternehmen ist, nimmt er sich nicht die Zeit, sich bei Wettbewerbern über Alternativen zu informieren. Voraussetzung für diesen Effekt ist natürlich eine

positive Einstellung zum Unternehmen. Zusätzlich zu der Immunisierung entsteht ein gewisser *Toleranzeffekt*. Durch die Bemühung des Unternehmens zur Aufrechterhaltung der Beziehungen entsteht für den Kunden ein positives Bild. Der Kunde toleriert dadurch Fehler des Unternehmens und das Abwerben durch einen Konkurrenten wird erschwert.

Um eine langfristige Bindung mit dem Kunden beizubehalten, ist auch das Feedback des Kunden von Vorteil. Durch das Beziehungsmarketing entsteht eine höhere *Beschwerdebereitschaft*. Ohne jegliche Kundenbindung sind unzufriedene Kunden dem Unternehmen meist nicht bekannt. Missmutige Kunden wandern ab, ohne das das Unternehmen entsprechend reagieren kann. Durch ein rechtzeitiges Feedback kann sich das Unternehmen auf solche Kunden einstellen und Maßnahmen ergreifen. Durch die Beschwerdebereitschaft entsteht auch eine größere *Auskunftsbereitschaft*. Die Kunden sind bereit, Auskunft zu geben und beteiligen sich zum Beispiel an Analysen oder Marktforschungen (vgl. Diller, 2007, S.104ff).

Der wahrscheinlich wichtigste Aspekt für kundenorientierte Marketingmethoden ist das Vertrauen des Kunden. Durch die Bindung mit dem Kunden festigt und vertieft sich die Vertrauensbasis für die Zukunft. Durch das Vertrauen des Unternehmens zum Kunden, fühlt sich der Kunde verpflichtet, dem Unternehmen ebenfalls zu vertrauen („Vertrauensspirale“). Durch diesen Kreislauf gibt der Kunde Informationen weiter und verzichtet auf Kontrolle (vgl. hierzu ausführlich: Hannemann, 2002, S.18ff).

Beziehungsmarketing bringt jedoch auch Nachteile mit sich: Durch eine zu starke Vertrauensbasis mit dem Kunden stärkt sich bei dem Unternehmen die *Inflexibilität*. Durch das Vertrauen können schneller Konflikte entstehen. Dies folgert, dass das Unternehmen sich an den Kunden anpassen muss. Dieser würde sich zum Beispiel diskriminiert fühlen, wenn das Unternehmen einen Konkurrenten des bisherigen Kunden akquiriert und möglicherweise als wichtigeren Kunden einstuft.
Sicherheitsrisiken können auch durch eine gemeinsame *Veralterung* des Kunden und des Unternehmens entstehen. Durch eine zu starke Bindung an den Kunden wird das Unternehmen zu träge, um neue Kundenpotenziale zu erschließen (vgl. Diller, 2007, S.106).

2.1.2.2 Unternehmenswachstum

Kundenbindung kann auch vieles zum Unternehmenswachstum beitragen, welches am Absatz oder Umsatz gemessen wird. Dieses Wachstum entsteht durch *Kundenpenetration* und durch *Kundenempfehlungen*.

Konzentriert sich der Kunde auf ein Unternehmen bei seinen Aktivitäten, so gewinnt dieses Unternehmen Marktanteile gegenüber seinen Mitbewerbern. Dadurch kann der Absatz des

Kunden auf dieses Unternehmen überführt werden. Ähnlich verhält sich dies bei Instrumenten wie zum Beispiel bei der Bahncard oder Bonusprogrammen wie Miles & More. Durch diese Mittel wird der Kunde an das Unternehmen gebunden und ist dazu bereit, intensivere (das heißt häufigere und teurere) Käufe bei diesem Unternehmen durchzuführen. Diese Art der *Kundenpenetration* „[...] ist auch Hintergrund des sog. *Permission Marketing*, mit dem mit expliziter Erlaubnis des Kunden dieser regelmäßig über bestimmte Angebote und Serviceleistungen eines Anbieters informiert wird, was Kaufanstöße auslöst." (Diller, 2007, S.107) Als Anwendungsfall kann man hier das Versenden von Newslettern nennen, welches ein neues Kaufinteresse bei bisherigen Kunden weckt.

Durch die stärkere Kundenbindung entsteht gleichermaßen ein Anstieg der Kundenempfehlungen. „Dies liegt darin, dass gebundene Kunden [...] bevorzugt als Meinungsführer und als Informationsquelle und Kaufinitiatoren bei bisher unerschlossenen Kundenkreisen fungieren" (Diller, 2007, S.109). Als Grund für dieses Vorgehen der Kunden nennt Diller die bessere Kenntnis des Unternehmens und die gesammelten Erfahrungen durch die Nutzung der Produkte des Unternehmens.

Die Kundenbindung kann das Wachstum eines Unternehmens behindern, falls das Unternehmen ein negatives Bild für den Kunden liefert. So können Fehler (bzw. nicht gelungene Produkte oder Dienstleistungen) bei Kunden für negative Mund-zu-Mund-Werbung sorgen (vgl. Diller, 2007, S.107ff).

2.1.2.3 Gewinn

Durch *Kosteneinsparungen* in der Kundenbearbeitung und *Erlössteigerungen* lässt sich durch Beziehungsmarketing zusätzlich der Gewinn steigern.

Bei Bestandskunden ist keine Kundenaquisition mehr nötig. Das heißt, dass Kosten für Besuche, Vorbereitungen, Entwürfe und Vorleistungen für die Entstehung von Geschäftsbeziehungen eingespart werden. Durch die starke Kundenbindung sind auch Einsparungen von Werbekosten möglich, da Bestandkunden über Direktwerbung ansprechbar sind.

Erlössteigerungen lassen sich durch gezieltes Up- und Cross-Selling erwirtschaften. Beim Up-Selling versucht man einem Bestandskunden ein höherwertiges Produkt zu verkaufen. Dagegen versucht man beim Cross-Selling durch die Angebotsvielfalt zusätzliche Produkte zu verkaufen.

Um diese Kundenbindungen zu pflegen und Gewinne zu erzielen, sind bestimmte Maßnahmen erforderlich. Es entstehen *Kundenbindungskosten* für kundenspezifische Wünsche und Vergünstigungen für Bestandskunden. Zudem sind Kosten für Veranstaltungen und Marketingtools notwendig um die Beziehungen aufrechtzuerhalten (vgl. Diller, 2007, S.110ff). Wie gefährlich Kundenbindungskosten für das Unternehmen sein können, soll Abbildung 2.1

deutlich machen. In der Phase des E-Commerce-Hypes (1998-2001) ist es vielen Unternehmen der Branchen Lebensmittel und Bekleidung nicht gelungen, die Werbungskosten für die Neukundengewinnung durch die Einkäufe gebundener Kunden einzuholen. Die meisten Kunden wechselten das Unternehmen, bevor die Werbungskosten wieder ausgeglichen wurden.

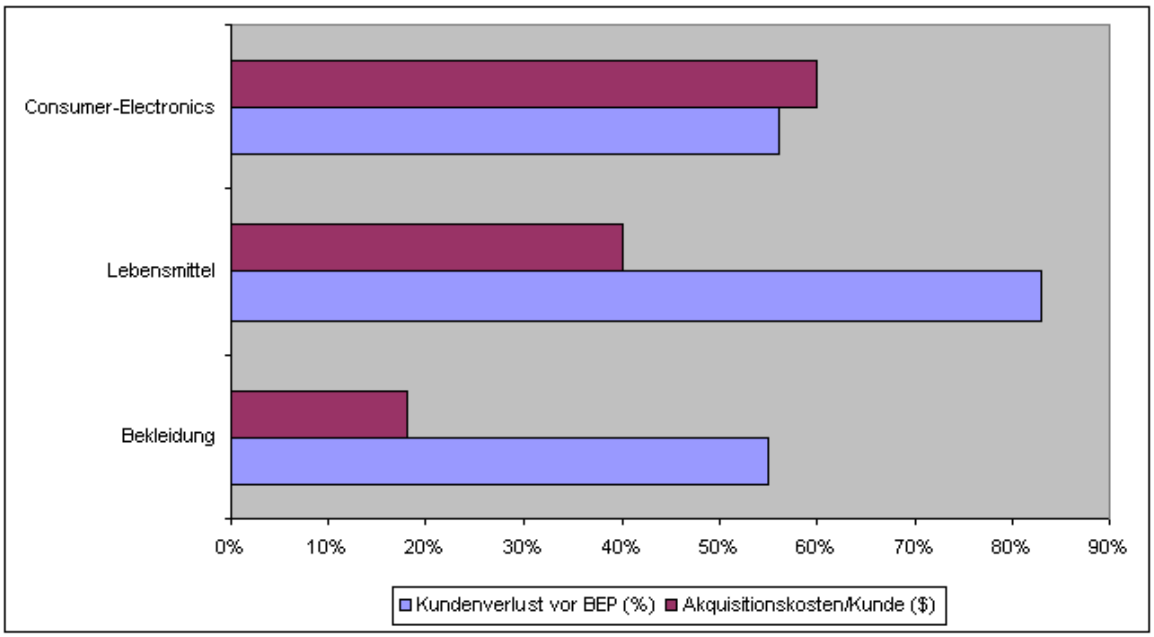

Abbildung 2.1: Kundenbindungs(miss)erfolge in der New Economy. Quelle: (Reichheld und Schefter, 2000)

2.1.2.4 Fazit

Die oben genannten Wirkungen, die durch Beziehungsmarketing entstehen, sind die Grundlagen für ein erfolgreiches Unternehmen. Infolgedessen hat sich das Beziehungsmarketing gegenüber dem Transaktionsmarketing behauptet und diesen weitgehend verdrängt.
Um allerdings Beziehungsmarketing durchführen zu können, sind folgende Grundsteine notwendig (vgl. Diller, 2007, S.116):

- Eine Datenbasis mit kundenbezogenen Daten
- Ein operatives Softwareprogramm welches an kundenbezogenen Prozessabläufen angepasst ist
- Analytische Tools, um die kundenbezogenen Daten analysieren zu können

Die ersten beiden Grundsteine sind durch die Arbeiten von Martin Sadowski (Sadowski, 2008) und André Schmer (Schmer, 2008) bereits geschaffen. Die Datenbasis wird aus dem vorherigen Vertriebssystem übernommen, so dass keine Neuanlage notwendig ist. Das entwickelte CRM-System stellt zusätzlich zur Datenbasis ein generisches Datenmodell zu Verfügung. Dieses Modell erlaubt, benutzerdefinierte und kundenbezogene Eigenschaften für Kontakte oder auch Unternehmen festzulegen. Das System bleibt dadurch flexibel in den Einsatzgebieten und muss nicht ständig durch Entwickler an die aktuelle Marktlage angepasst und funktional erweitert werden (vgl. hierzu ausführlich: Schmer, 2008, S.23ff).
Die Anpassung des CRM-Systems an die Prozessabläufe kann durch die prototypische Entwicklung belegt werden (vgl. Sadowski, 2008, S.53). Durch das frühzeitige Feedback der Anwender wird dadurch eine Fehlentwicklung ausgeschlossen.

Was letztendlich fehlt, und im Rahmen dieser Arbeit behandelt wird, ist der Grundstein für eine Analyse der kundenbezogenen Daten. Dieser Grundstein soll durch das Modul Reporting abgedeckt werden.

2.1.3 Reporting

Nur Aufgrund von fundierten Informationen können unternehmerische Entscheidungen getroffen werden. Diese Entscheidungen werden meist durch die Qualität der Informationen eines operativen Systems beeinflusst. Ein leistungsfähiges Berichtswesen kann dafür sorgen, qualitativ hochwertige Daten zu filtern und zu analysieren. Durch diese Maßnahmen können rechtzeitig Indikatoren bereitgestellt werden, die eine Bewertung und gegebenenfalls eine Korrektur der Unternehmensstrategie zulassen (vgl. Offe, 1999, S.5).

Da das Berichtswesen von unterschiedlichen Anwendergruppen genutzt wird, entstehen auch recht unterschiedliche Anforderungen, die ein Berichtswesen erfüllen soll.
Mitarbeiter aus dem Bereich *Controlling*, wie u. a. Geschäftsführer, interessieren sich für erbrachte Leistungen von Mitarbeitern oder für „Verkaufsschlager“ des Unternehmens. Somit erwarten Mitarbeiter *statistische Auswertungen*, die einem bei der Organisation und Führung hilfreich sind.
Aus den Bereichen *Marketing & Vertrieb* dagegen hat man wenig Nutzen von statistischen Auswertungen. In diesen Bereichen ist das Interesse für *datenorientierte Auswertungen* viel größer. Bei diesen Auswertungen liegt der Fokus bei der Kundenbeziehung und Kundenkommunikation.

Nach einer Beschreibung der Geschichte des Berichtswesens erfolgt eine kurze Abgrenzung zu Data Warehouse-Systemen. Anschließend wird auf statistisch- sowie datengetriebene Reports genauer eingegangen.

2.1.3.1 Historie

Die Fortschritte des computergesteuerten Berichtswesen lassen sich nach P. Chamoni (Chamoni und Gluchowski, 2005, S.3ff) in fünf historische Bereiche einteilen:

- Die ersten Berichtswesen sind in den 60er Jahren entstanden. Damals wurden sogenannte Management Information Systems (MIS) aufgebaut, mit dem Ziel, durch Auswertungen aus hierarchischen Datenbanksystemen den Planungs- und Kontrollprozess zu optimieren.
  Danach wurden zum Teil relationale Datenbanksysteme eingesetzt, so dass SQL-Anweisungen diese Auswertungen teilweise ersetzten. Diese Anweisungen wurden in periodischen Abständen ausgeführt und zwischengespeichert, damit ein Zugriff auf die Daten zu jeder Zeit möglich war.
- Aufgrund der großen Aufwände bei Auswertungen sind Decisions Support Systems (DSS) entstanden. Bei diesem Systemen wird die Unterstützung von Planungs- und Entscheidungsprozessen fokussiert. Als Beispiel kann man hier Tabellenkalkulationsprogramme nennen, die viele Möglichkeiten zur Datenauswertung bieten. Durch die Nutzung von DSS sind jedoch auch Nachteile entstanden: Durch Rich-Client-Anwendungen (RCP) werden meist lokale Datenbestände benutzt, so dass keine Konsistenz oder Aktualität der Daten gewährleistet werden kann.
- Als nächstes kam der Wunsch, dass man Daten durch Auswertungen nicht nur visuell darstellt, sondern auch unter verschiedenen Aspekten betrachten kann. Durch sogenannte Executive Information Systems (EIS) wurden eigene Systeme entwickelt, die intuitiv bedienbar waren und durch das Berichtswesen die Ursachen für die Auswertungsergebnisse präsentierten.
- Da der Aufwand für den Aufbau eines EIS viel zu groß war, haben sich Query-Tools der Datenbankhersteller etabliert. Mit wenig Kapitalaufwand konnten Auswertungen durch diese Anwendungen erfolgen. Mittlerweile bieten die meisten Datenbankhersteller diese Tools an, mit denen Mitarbeiter Auswertungen durchführen können.
- Durch wachsende Anforderungen seitens der Unternehmen sind Query-Tools heutzutage nicht mehr ausreichend. Da mittlerweile an allen Arbeitsplätzen mit dem Berichtswesen gearbeitet wird, haben nach P. Gluchowski (Gluchowski, 2005, S.193ff) Unternehmen mit folgenden Problemen zu kämpfen:
  - Durch komplexe Abfragen entstehen hohe Netzbelastungen, die andere Benutzer bei der Arbeit behindern können.
  - Die Server werden stark belastet, da Anwender ihre Berichte jederzeit durchführen können.

- Da keine zentrale Verwaltung der Berichte möglich ist, entsteht ein hoher Aufwand für den Administrator.
- Anwender müssen geschult werden, damit sie Abfragen in der Datenbankanwendung erzeugen können.

Um alle oben genannten Probleme zu lösen, wurde das Data Warehouse bzw. Business Intelligence-System erfunden.

2.1.3.2 Data Warehouse-System

> *„Ein Data Warehouse repräsentiert eine von den operativen Datenbanken getrennte Decision Support-Datenbank (Analyse-Datenbank), die primär zur Unterstützung des Entscheidungsprozesses im Unternehmen genutzt wird. Ein Data Warehouse wird immer multidimensional modelliert und dient zur langfristigen Speicherung von historischen, bereinigten, validierten, synthetisierten, operativen, internen und externen Datenbeständen“ (Kurz, 1999, S.50).*

Demnach ist ein Data Warehouse-System ein eigenständiges System, welches verschiedene Analysemöglichkeiten über einen eigenen Datenbestand ermöglicht. Diese Daten können aus einem operativen System, wie z. B. das bisher entwickelte CRM-System, stammen. Da ein Data Warehouse-System als eigenständiges System fungiert, ist auch eine gewisse Interopabilität vorhanden. So lassen sich z. B. Daten verschiedener Systeme für Analysezwecke in einem Data Warehouse-System zusammenfassen. Die wesentlichen Merkmale eines Data Warehouse-Systems sind nach P. Chamoni (Chamoni und Gluchowski, 2005, S.14) folgende:

- *Themenorientierung*: Ein Data Warehouse-System konzentriert sich auf die Schwerpunkte der Entscheidungsunterstützung. Das operative System dagegen ist für die Prozessabläufe optimiert.
- *Vereinheitlichung*: Da die Daten eines Unternehmens meist aus verschiedenen (u. a. externen) Quellen stammen, werden in Data Warehouse-Systemen diese Daten normiert.
- *Zeitorientierung*: Durch die Aufnahme von Zeitbezügen können die Daten einer bestimmten Periode ausgewertet werden.
- *Beständigkeit*: Durch die langfristige Speicherung von Daten ist es möglich, Indikatoren für neue Unternehmensstrategien festzulegen.

Die Abbildung 2.2 zeigt eine typisches Data Warehouse-System.

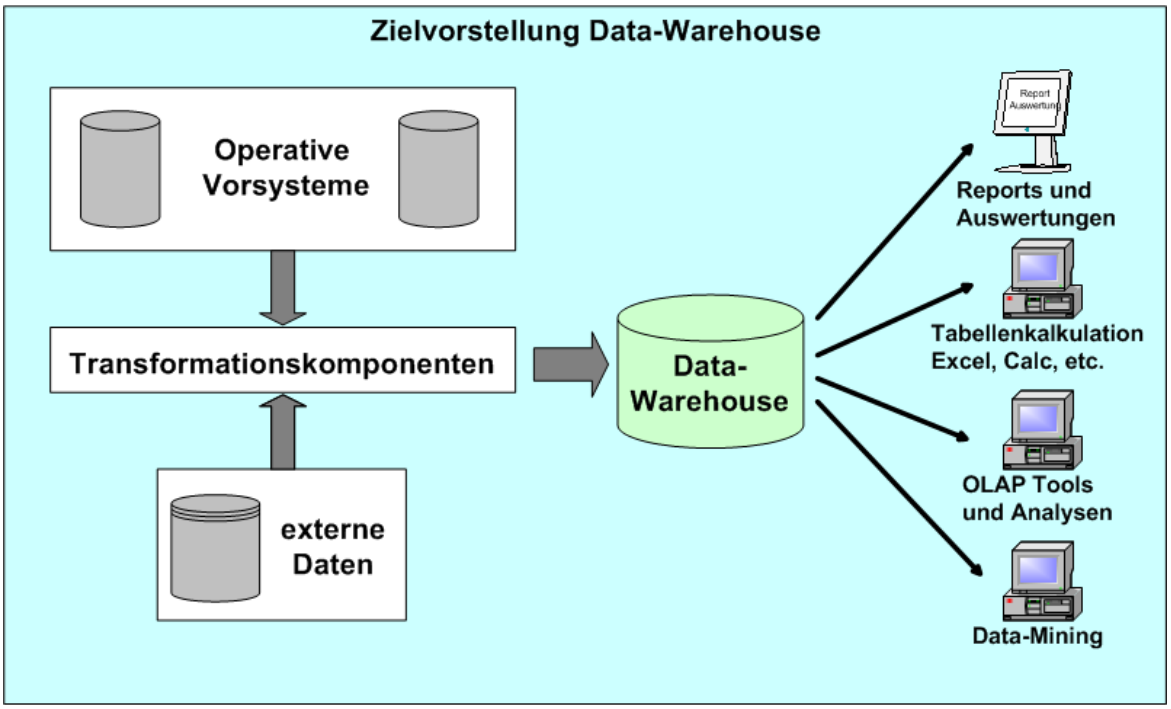

Abbildung 2.2: Data Warehouse-System. Quelle: (Wikipedia, 2008a)

2.1.3.3 Betriebliche Statistik-Reports

Mit zunehmender Konkurrenz im Wettbewerb in allen Geschäftsbereichen wird es immer wichtiger, möglichst aktuelle Kennzahlen und Statistiken eines Unternehmens zu analysieren. Diese Auswertungen stellen die Basis für zukünftige Maßnahmen, um den Erfolg des Unternehmens zu steigern. Reports bieten eine Möglichkeit, Auswertungen nach beliebigen Kriterien vorzunehmen. So verschafft man sich durch Reports einen gegenwärtigen Überblick über das Unternehmen und deren Umfelder, wie z.B. Kundenkennzahlen, Projektkennzahlen oder auch Mitarbeiterkennzahlen. Als Kennzahlen bezeichnet man im üblichen Maßzahlen, welche größtenteils dazu eingesetzt werden, um qualitative Verbesserungen durch quantitative Messungen innerhalb des Unternehmens zu initiieren.
Eine Statistik fasst quantitative Daten zu Tabellen, grafischen Darstellungen oder Kennzahlen zusammen. Diese Art des Reports wird in den meisten Fällen intern genutzt, um empirische Daten über das Unternehmen und deren Mitarbeiter zu sammeln.

2.1.3.4 Betriebliche Daten-Reports

Um jedoch nicht nur Geschäftsprozesse innerhalb des Unternehmens zu analysieren und zu verbessern, sondern auch zum Beispiel Kundenmaßnahmen zur Verbesserung des Geschäftsverhältnisses durchführen zu können, bedarf es mehr als einer reinen Kennzahlenanalyse. In diesem Fall will man durch einen Report nicht nur Statistiken erstellen, sondern auch die betroffenen Daten dieser Statistiken einsehen und auswerten. Dieser kleine, jedoch enorm wichtige Unterschied wird anhand des folgenden Beispiels deutlich:

Beispielfragen Statistik-Reports:

- Wie viele Projekte wurden im letzten Quartal abgeschlossen?
- Wie viele potentielle Kunden sind einem Mitarbeiter zugeordnet?
- Wie viele Kunden mit einem Umsatz von über 10 Millionen Euro erhalten einen Newsletter?

Alle oben genannten Beispiele führen zu Kennzahlen, welche innerhalb des Unternehmens für die Statistik zwar wichtig sind, jedoch nicht wirklich zum Unternehmenserfolg (vor allem im Bezug zur Umwelt) beitragen. Um erfolgreiche Maßnahmen durchführen zu können, benötigt man nicht nur die Kennzahlen, sondern zusätzlich die Daten, welche durch die Kennzahlen repräsentiert werden.

So ergibt sich bei datengetriebenen Reports folgende Fragestellung:

- Welche Projekte wurden im letzten Quartal abgeschlossen?

- Welche Kunden sind einem Mitarbeiter zugeordnet?
- Welche Kunden mit einem Umsatz von über 10 Millionen Euro erhalten einen Newsletter?

So will ein Mitarbeiter des Unternehmens nicht nur wissen, wie viele Kunden einen Newsletter erhalten, sondern welche Kunden dies sind. Ein Geschäftsführer möchte dementsprechend nicht nur wissen, wie viele Projekte abgeschlossen wurden, sondern um welche Projekte es sich dabei handelt, um die Auftraggeber eventuell zu kontaktieren.

Demnach kann man sagen, dass „einfache“ Reports durch Kennzahlen diesen Anforderungen nicht standhalten würden. Der Grund dafür ist, dass mit den im vorherigen Kapitel beschriebenen Marketingmethoden die Beziehungen zum Kunden die größte Rolle spielen. Reports müssen demnach auch kundenorientiert gestaltet werden.

2.1.3.5 Fortgeschrittene Anforderungen

> *„Was wir brauchen, bekommen wir nicht. Was wir bekommen, verstehen wir nicht. Was wir verstehen, erhalten wir nicht.“* (Reinecke u. a., 2001b, S.690)

Klagen dieser Art hört man oft von Führungskräften, die keine korrekten Managemententscheidungen treffen können, weil ihnen nicht die richtigen Informationen zur richtigen Zeit vorliegen.
Um die richtigen Informationen liefern zu können, muss zunächst analysiert werden, welche Instrumente in der Praxis eingesetzt werden. Eine standardisierte schriftliche Befragung von 276 Unternehmen in Deutschland und der Schweiz hat ergeben, dass in der Praxis Branchen- bzw. Kokurrenzanalysen am häufigsten durchgeführt werden. Seltener, aber dennoch regelmäßig, kommen strategische Produkt- und Kundenportfolios zum Einsatz (vgl. Reinecke u. a., 2001a, S.76f). Um solche Portfolios zu erstellen, werden verschiedene Analysetechniken genutzt, die (vor allem in Hinblick auf das Kundenbindungsmanagement) erfordern, dass verschiedene Kennzahlenanalysen zu einem Kennzahlensystem verknüpft werden (vgl. Palloks-Kahlen, 2001, S.522f).

Ein Data Warehouse-System bietet eine gute Basis, um fortgeschrittene Anforderungen zu erfüllen. In der Abbildung 2.3 werden einige Prozesse und Funktionen eines Data Warehouse-Systems dargestellt.
Wie bereits im Kapitel 2.1.3.2 beschrieben, besitzt ein Data Warehouse-System eine eigene analytische Datenbank. Diese Datenbasis wird in regelmäßigen Abständen mit den Daten des operativen Datensystems gefüllt. Um Analysefunktionen zu nutzen, wird eine Query Engine genutzt, welche als Schnittstelle der analytischen Datenbank dient. Die Ergebnisse der Query Engine werden seperat in einer Datenbasis gesichert, damit z. B. Vergleiche

zwischen Datenbeständen oder auch Zeitreihenanalysen möglich sind. Um die Daten auszuwerten, werden zwei verschiedene Möglichkeiten angeboten. Ad-hoc Auswertungen dienen für spontane Auswertungen, die der Benutzer spezifizieren muss. Als Resultat erhält dieser eine nicht gestaltete Liste von Ergebnissen.

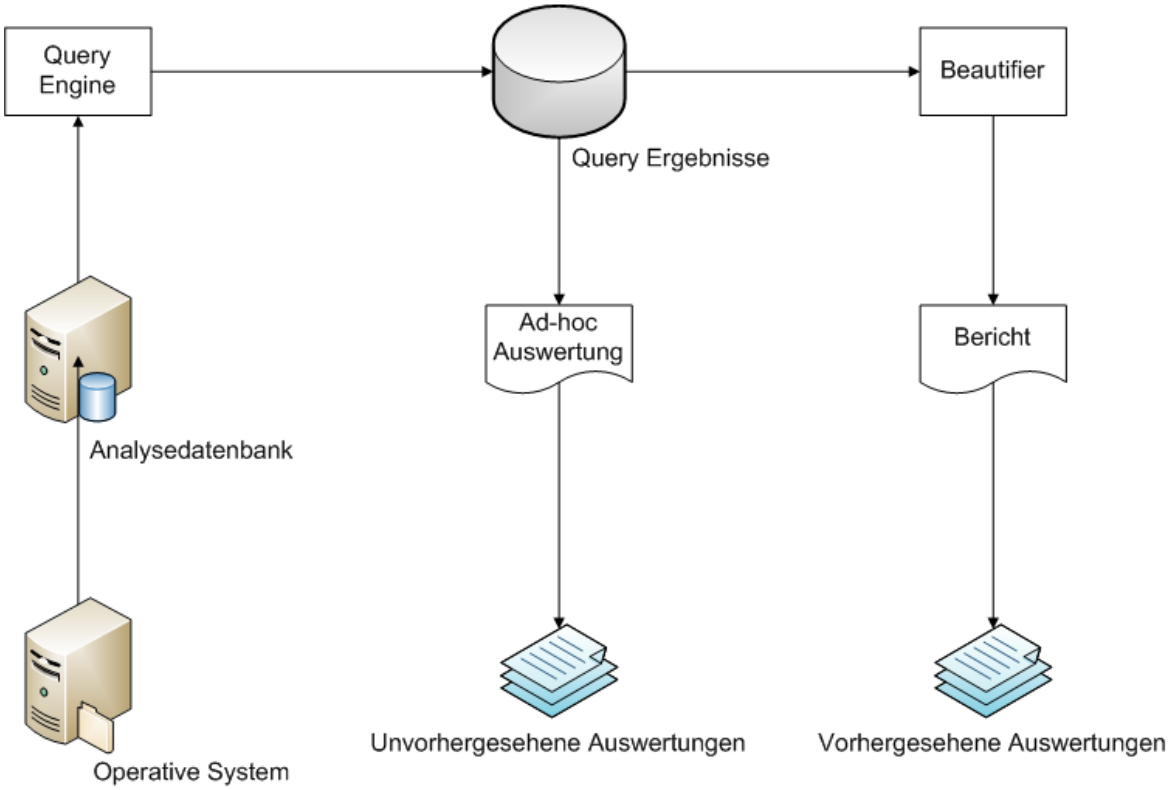

Abbildung 2.3: Prozesse bei einem Data Warehouse-System

Außer Ad-hoc Auswertungen werden noch sogenannte vorhergesehene Auswertungen angeboten. Diese Auswertungen werden regelmäßig angestoßen und sind für den Benutzer stets abrufbar. Da es sich hierbei um geplante Auswertungen handelt, wird ein Beautifier angesetzt, damit das Ergebnis grafisch aufbereitet wird. Auswertungen dieser Art werden z. B. für Präsentationen, Portfolios oder Marketingzwecke genutzt. Zudem bietet sich dadurch die Möglichkeit, verschiedene Auswertungen über Zeitreihen oder Zusammenhänge zu vergleichen.

2.2 Realisierungskonzept

Durch die technischen Anforderungen wurden der Realisierung des CRM-Systems enge Grenzen gesetzt. So soll das System in der Programmiersprache Java entwickelt und aus-

schließlich Open Source Komponenten verwendet werden, damit keine Hindernisse für die Veröffentlichung des Systems am Open Source Markt entstehen (vgl. Sadowski, 2008, S.30).

Java Plattform, Enterprise Edition (Java EE), ist eine Spezifikation zur Entwicklung von verteilten, mehrschichtigen Anwendungen und wird im bisher entwickeltem CRM-System verwendet. Sie basiert auf der bereits bewährte Java Plattform, Standard Edition (Java SE), und ist aktuell in der Version 5 (Mai 2006) verfügbar.
Diese Spezifikation bietet ein Umfeld für eine Softwarearchitektur, die aus mehreren modularen Komponenten (Services) bestehen kann (vgl. Shannon, 2006, S.1).

2.2.1 Architektur

Die Abbildung 2.4 enthält einen Überblick der Java EE Architektur, welche im Folgenden beschrieben wird.

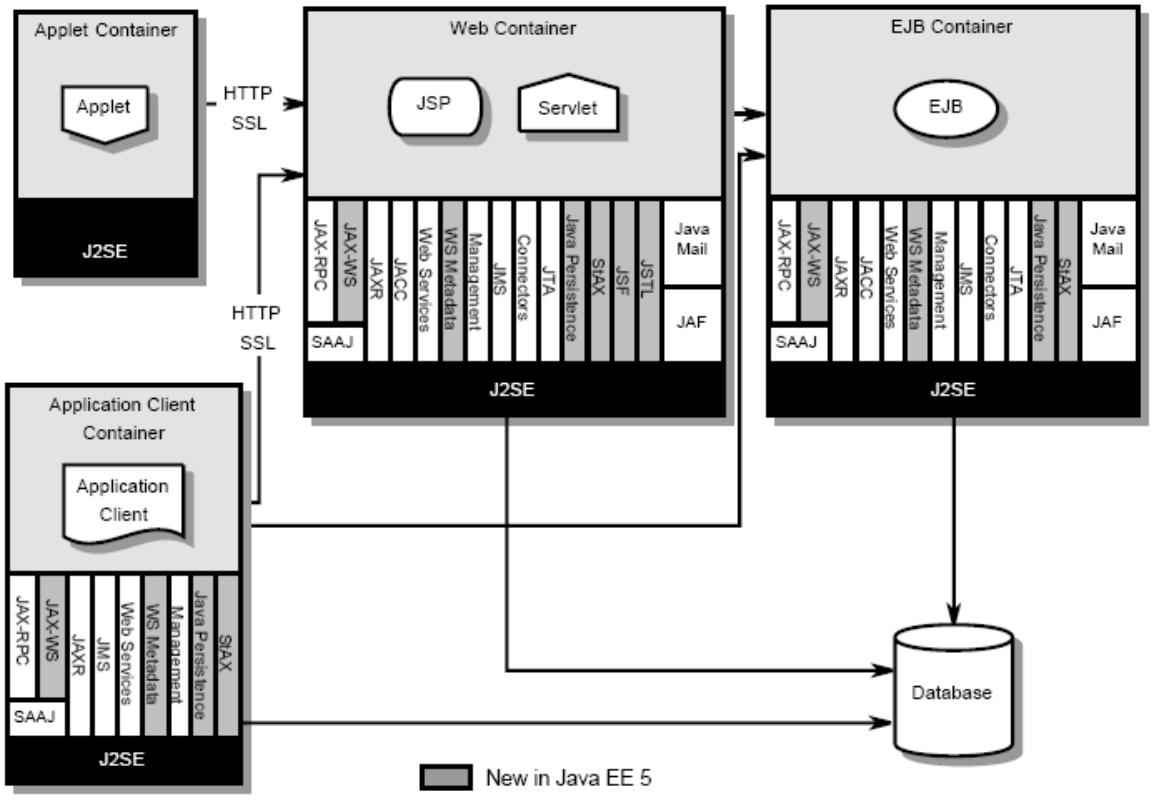

Abbildung 2.4: Übersicht der Java EE 5 Architektur. Quelle: (Shannon, 2006, S. 6)

2.2.1.1 Container

In der Java EE Architektur gibt es insgesamt vier Container, die jedoch nicht alle eingesetzt werden müssen. Mit einem Container bezeichnet man eine Java EE Laufzeitumgebung (Java EE runtime environment), welche eigenständig ausgeführt wird und nur über eigene Schnittstellen mit anderen Containern oder der Datenbank kommuniziert. Diese Container stellen für ihre Komponenten (Application Components) diverse Services bereit.

2.2.1.2 Application Components

In der Java EE Spezifikation werden vier verschiedene Komponenten definiert (vgl. Shannon, 2006, S.6). Diese Komponenten laufen stets innerhalb eines Containers und können ausschließlich über die Schnittstellen des Containers miteinander kommunizieren. Zwischen folgenden Komponenten wird unterschieden:

- Applet: GUI Komponenten die in einem Webbrowser laufen (z. B. HTML-Seiten)
- Application Client: Rich-Client Anwendungen, die auf einem Rechner laufen
- JSP, Servlet u.a.: Komponenten um z. B. HTML-Seiten zu generieren
- EJB: Enterprise Java Beans, welche die Geschäftslogik repräsentieren

An dieser Stelle möchte ich auf die Bachelorarbeit von Martin Sadowski (Sadowski, 2008) verweisen, welche sich ausführlich mit der wohl wichtigsten Komponente, Enterprise Java Beans (EJB), beschäftigt.

2.2.1.3 Services

Services sind verschiedene Technologien, welche die Java Platform Standard Edition um Funktionalität und Interoperabilität erweitern. Diese Services werden von den jeweiligen Containern (siehe Abb.2.4) für die Application Components bereitgestellt.
Einige Services werden in Kapitel 2.2.3 näher beschrieben.

2.2.2 Verwendete Implementierung

Da Java EE eine Grundlage für eine Softwarearchitektur spezifiziert, sollte für die Entwicklung einer Web Anwendung ein Java EE-konformer Application Server verwendet werden. Die im entwickelten System verwendete Implementierung ist der JBoss Application Server in der Version 4.2.2 von der Firma Red Hat.
JBoss ist der populärste Open Source Java EE Application Server und wurde von Sun Microsystems zertifiziert, nachdem diese Implementierung alle 23.000 Tests der Sun Microsystems Compatibility Test Suite für Java EE 1.4 erfolgreich bestanden hat (vgl. Sun, 2004).

Diese Implementierung bietet zahlreiche Features, wie zum Beispiel (vgl. RedHat, 2008):

- Unterstützung vieler Standards (wie z. B. EJB 3.0)
- Hohe Zuverlässigkeit durch Clustering, Caching, Load Balancing u.v.m.
- Service-orientierte Architektur
- Einfache Verwaltung durch eine Management-Console (JMX Services)
- Hohe betriebssystem- und hardwareunabhängige Interoperabilität

2.2.3 Erwähnenswerte Services/APIs

Java EE bietet zahlreiche Services, welche die APIs der Java, Standard Edition, funktional erweitern. Die JavaServer Faces API ermöglicht eine einfache und schnelle Entwicklung von grafischen Oberflächen. Das Java Naming and Directory Interface definiert eine einheitliche Schnittstelle für verteilte Anwendungen. Die Java Persistence API stellt eine einheitliche und datenbankunabhängige Schnittstelle für das objektrelationale Mapping (OR-Mapping) bereit.

2.2.3.1 JavaServer Faces (JSF)

JavaServer Faces ist ein serverseitiges UI-Framework, um Java-basierte Webanwendungen zu entwickeln. Die Hauptkomponenten von JSF sind (vgl. Sun, 2007, S.285):

- Eine API für UI-Komponenten sowie deren Management (Zustände, Event-Handling, Konversionen u.v.m.)
- Zwei JavaServer Pages (JSP) Bibliotheken, um UI-Komponenten innerhalb von JSP-Seiten darzustellen

Durch die JSF-Architektur (vgl. Schmer, 2008, S.12) und die zusätzlichen Bibliotheken lassen sich in kurzer Zeit serverseitige Benutzerschnittstellen erstellen. Eine JSF-Applikation wartet dementsprechend auf eine Anfrage durch den Client und antwortet mit generiertem HTML (vgl. Abb. 2.5). Der Vorteil dabei ist, dass zwischen JSF und HTML keine Kopplung entsteht, wodurch ein Rendering in andere Client-Sprachen ebenfalls denkbar wäre.

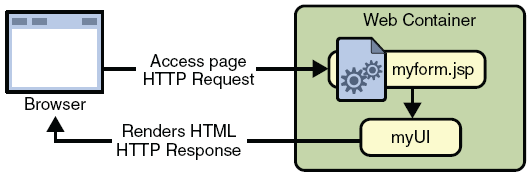

Abbildung 2.5: JSF Request/Response-Zyklus. Quelle: (Sun, 2007, S. 286)

Folgende Vorteile ermöglichen JavaServer Faces (vgl. Sun, 2007, S.285):

- Koppeln von Server-Application und Ereignissen, die durch UI-Komponenten ausgelöst werden
- Binden von UI-Komponenten an serverseitige Daten
- Erstellen von wiederverwendbaren und erweiterbaren Komponenten
- Speichern und Wiederherstellen vom UI-Zustand unabhängig vom Request/Response-Zyklus

2.2.3.2 Java Naming Directory Interface (JNDI)

Damit verteilte Komponenten über ihre Container (siehe Abb. 2.4) miteinander kommunizieren können, wird ein Vermittler benötigt. Durch JNDI ist ein Interface spezifiziert, welches Anwendungen Zugriff auf Namens- und Verzeichnisdienste ermöglicht.
Der Vorteil dieses Services ist die einheitliche Schnittstelle für Java-Entwickler. JDNI unterstützt durch das Service Provider Interface (SPI) nahezu alle wichtigen Namens- und Verzeichnisdienste wie z. B. LDAP (Lightweight Directory Access Protocol) oder CORBA (Common Object Request Broker Architecture) (vgl. Stark, 2005, S.180).

2.2.3.3 Java Persistence API (JPA)

Da für die Durchführung von Reports bzw. Selektionen Datenbankabfragen (SQL-Anweisungen) notwendig sind, werden bequeme Zugriffsmöglichkeiten benötigt.

Die grundlegendste Schnittstelle zum Persistieren von Daten in einer Datenbank ist für Java-Anwendungen die Java Database Connectivity (JDBC). Über sogenannte JDBC-Treiber, welche je nach DBMS-Hersteller variieren, kann man von einer Java-Anwendung SQL Anweisungen an die Datenbank senden. Um auch Ergebnisse zu erhalten, wie z. B. bei SELECT-Anweisungen, erhält man in der Java-Anwendung ein java.sql.ResultSet-Objekt zurück, mit dem man über Zeilen- und Spaltenindizes auf die abgefragten Daten zugreifen kann.

Die Java Persistence API (JPA) wird seit Java EE 5 für das Persistieren von Daten als objektorientierte Lösung angeboten. Sie ist der Nachfolger der EJB Container Managed Persistence, welche sich aufgrund ihrer Komplexität nicht bewährt hat (Heider u. a., 2006, S.3). JPA sorgt dafür, dass sogenannte Entities (welche Bestandteile von EJB 3.0 sind) in einer relationalen Datenbank persistiert werden. Entities sind dabei „Plain Old Java Objects“ (POJO), welche über Annotations, einfache Metadaten, mit Datenbanktabellen verknüpft werden (vgl. hierzu ausführlich: Sadowski, 2008, S.18ff).

Um die Entities in der Datenbank zu persistieren, kommt das im JPA enthaltene Interface javax.persistence.EntityManager zum Einsatz. Der EntityManager bietet Operationen, um Daten über Entities zu löschen, zu speichern oder zu aktualisieren. Im Beispiel 2.1 wird demnach durch die Entität Employee ein Tabelleneintrag erstellt. Danach wird die Entität aktualisiert und letztendlich wird dieser Eintrag gelöscht.

```
1  // Erstellen einer neuen Entity:
2  Employee employee = new Employee(10);
3  employee.setLastname("Miller");
4  employee.setFirstname("Thomas");
5
6  // Initialisierung des EntityManagers:
7  @PersistenceContext
8  EntityManager em;
9
10 // Persistieren durch den EntityManager:
11 em.persist(employee);
12
13 // Aktualisieren durch den EntityManager:
14 em.merge(employee);
15
16 // Entfernen durch den EntityManager:
17 em.remove(employee);
```

Listing 2.1: Beispiel des EntityManagers

Zusätzlich bietet der EntityManager Funktionen, um Datenbankanfragen zu erstellen. Da JPA datenbankunabhängig ist, kommt hier die Java Persistence Query Language (JPQL) zum Einsatz. Alternativ kann man auch die datenbankspezifische Sprache (z. B. SQL für MySQL oder PL/SQL für Oracle) für Abfragen verwenden, jedoch wird dies aufgrund der Datenbankunabhängigkeit von JPA nicht empfohlen (Heider, 2007, S.9). Das Beispiel 2.2 macht den Unterschied zwischen JPQL und SQL deutlich.

```
1  // Initialisierung des EntityManagers:
2  @PersistenceContext
3  EntityManager em;
4
5  // Erstellen einer DB-Abfrage in JPQL:
6  Query q = em.createQuery("SELECT e FROM Employee e");
7  List<Employee> employees = q.getResultList();
8
9  // Erstellen einer DB-Abfrage in SQL:
10 Query q = em.createQuery("SELECT * FROM employee e");
11 List<Employee> employees = q.getResultList();
```

Listing 2.2: Beispiel von EntityManager Queries

Auf den ersten Blick ähneln sich JPQL und SQL sehr stark. Der einzige sichtbare Unterschied ist, dass SQL über Tabellen und JPQL über Entities selektiert. Das folgende Beispiel 2.3 soll die Vorteile von JPQL verdeutlichen.

```
1  -- SQL Anweisung ohne Outer Join
2  SELECT * FROM employee e, address a WHERE e.address_id = a.id or e.address_id IS NULL
3
4  -- SQL Anweisung mit Outer Join
5  SELECT * FROM employee e LEFT OUTER JOIN address a ON (e.address_id = a.id)
6
7  -- JPQL Anweisung mit Outer Join
8  SELECT p FROM employee e LEFT JOIN e.address a
```

Listing 2.3: SQL vs. JPQL

Da bei EJB 3.0 die Beziehung einer Entity zu anderen Entities über Annotations definiert wird (vgl. Sadowski, 2008, S.19), ist es nicht mehr notwendig, zu definieren, über welche Spalten ein *JOIN* zweier Tabellen stattfinden soll. Die Java Persistence API führt über Annotations alle Tabellen zusammen und transformiert den JPQL Ausdruck in eine äquivalente SQL-Anweisung.

Durch EJB 3.0 und JPQL wird das Erstellen von SQL-Anweisungen stark vereinfacht. Zusätzlich bietet der bisher genutzte Anwendungsserver JBoss (siehe Kapitel 2.2.2) ein java-basiertes Persistenz-Framework namens Hibernate.
Da bei Reports bzw. Selektionen die Abfragen dynamisch zur Laufzeit erzeugt werden müssen, wird die Criteria API von Hibernate verwendet. Diese API bietet eine objektorientierte Lösung zur Erstellung von Datenbankabfragen. In Kapitel 7.1.1.1 werden die Vorteile dieser API näher erläutert.

3 Fachkonzept

Um die Anforderungen des zu entwickelnden Reporting-Moduls zu spezifizieren, muss zunächst ein Fachkonzept geschaffen werden. Dabei werden die Funktionen im Bereich Reporting im alten Vertriebssystem untersucht und analysiert. Anschließend werden die Anforderungen an das neue Modul durch die zukünftigen Benutzer erfasst und spezifiziert.

3.1 Bisheriges System

Bei dem alten Vertriebssystem handelt es sich um eine Standardsoftware von einem Softwarehersteller. Diese Software ist seit mehreren Jahren im Einsatz und wurde seitdem ständig weiterentwickelt, um den steigenden Anforderungen gerecht zu werden. Dabei handelt es sich um eine Desktop-Anwendung, zu der nur eine begrenzte Anzahl an Benutzern Zugang hatten.

3.1.1 Bewertung aus Benutzersicht

Bei einem Interview mit Anwendern aus verschiedenen Bereichen sind folgende Bewertungen entstanden:

- *Veraltete Software*: Die Software ist nach einem jahrelangen Einsatz veraltet und erfüllt nicht mehr alle funktionalen Anforderungen mit Zufriedenheit.
- *Gebrauchstauglichkeit*: Alle Anwendungsfälle lassen sich bisher auch umsetzen, jedoch mit starken Einschränkungen in der Usability.
- *Geschäftsprozesse*: Die internen Geschäftsprozesse des Unternehmens haben sich seit der Einführung der Software geändert, so dass die Software nicht alle Prozesse optimal unterstützt.
- *Exportformat*: Bei Selektionen werden von den Anwendern in den meisten Fällen alle Details exportiert. Eine Auswahl von bestimmten Eigenschaften ist nur mit umständlichen Umwegen möglich.

- *Selektionsdauer*: Selektionen haben bis zu 45 Minuten gedauert. In dieser Zeit kann mit dem System nicht weitergearbeitet werden.
- *Selektionsablauf*: Nach einer Selektion und dem anschließenden Export in das CSV-Format wird diese Ergebnisliste von Hand weitergepflegt. Viele Mitarbeiter, die keinen Zugang zum System haben, haben diese Liste erweitert oder auch Datensätze gelöscht.

Zusammenfassend kann beurteilt werden, dass durch die Software nach allen denkbaren Kriterien selektiert werden kann. Aufgrund der starken Einschränkung in der Gebrauchstauglichkeit sind einige Funktionen den Benutzern nicht bekannt, so dass der Bereich Selektion im alten Vertriebssystem nie wirklich akzeptiert wurde.

3.1.2 Bewertung aus Entwicklersicht

Da die meisten Anwender mit dem alten Vertriebssystem ausschließlich im Bereich CRM (bzw. Relationship Marketing) tätig sind, ist eine einseitige Sichtweise entstanden. So sind mir zusätzlich folgende Defizite aufgefallen:

- *Selektionskriterium*: Bei den Kriterien gibt es immer nur eine „Von...Bis“-Beziehung. Aus diesem Grund ist es nur über Umwege möglich, Selektionen nach bestimmten Kriterien durchzuführen. Zusätzlich muss dadurch auf die alphabetische Sortierung der möglichen Werte geachtet werden. Das folgende Beispiel soll dies verdeutlichen.

  Beispielselektion: Gesucht werden alle Kunden, die in Germany *oder* Italy ansässig sind.

  Lösung: Selektiere alle Kunden,
  deren Feld „Land“ die Werte Germany bis Italy enthält *und*
  deren Feld „Land“ nicht die Werte Ghana bis Israel enthält.

  Da, zwischen allen Kriterien in diesem Selektionsmodul immer Konjunktionen entstehen, wäre folgende Lösung falsch:

  Selektiere alle Kunden,
  deren Feld „Land“ die Werte Germany bis Germany enthält *oder*
  deren Feld „Land“ Italy bis Italy enthält.

  Da bei einer Konjunktion alle Aussagen wahr sein müssen, damit das Ergebnis wahr ist, entsteht zwischen den Kriterien eine *UND*-Beziehung. Das heißt im oben genannten Fall müsste der Kunde in Germany und in Italy ansässig sein, was praktisch unmöglich ist und zu einer leeren Ergebnismenge führen wird.

- *Selektionsdauer*: Selektionen dauern bis zu 45 Minuten. Abhängig von der Dauer ist die Anzahl der Ergebnisse, die man durch eine Selektion erhält. Selbst bei Selektionen, die zu keinem Ergebnis führen, entsteht eine unnötige Wartezeit.
- *Usability*: Selektionen werden wie Bausteine zusammengesetzt. Man kann in Folge dessen sehr schlecht ablesen, was eine Selektion aussagt.

Im Großen und Ganzen kann vorweg erwähnt werden, dass sich die Selektion in diesem System von einer Selektion anderer (CRM-)Systeme sehr stark abgrenzt (vgl. hierzu ausführlich Kapitel 4.1).

3.2 Anforderungsanalyse

Die Anforderungsanalyse ist anhand von zahlreichen Interviews mit zukünftigen Benutzern des Report Moduls entstanden.

3.2.1 Fachliche Anforderungen

Die zusätzlichen Anforderungen, die das zu entwickelnde Modul erfüllen soll, werden anhand von Anwendungsfällen (siehe Abb. 3.1) in Form von User Stories beschrieben.

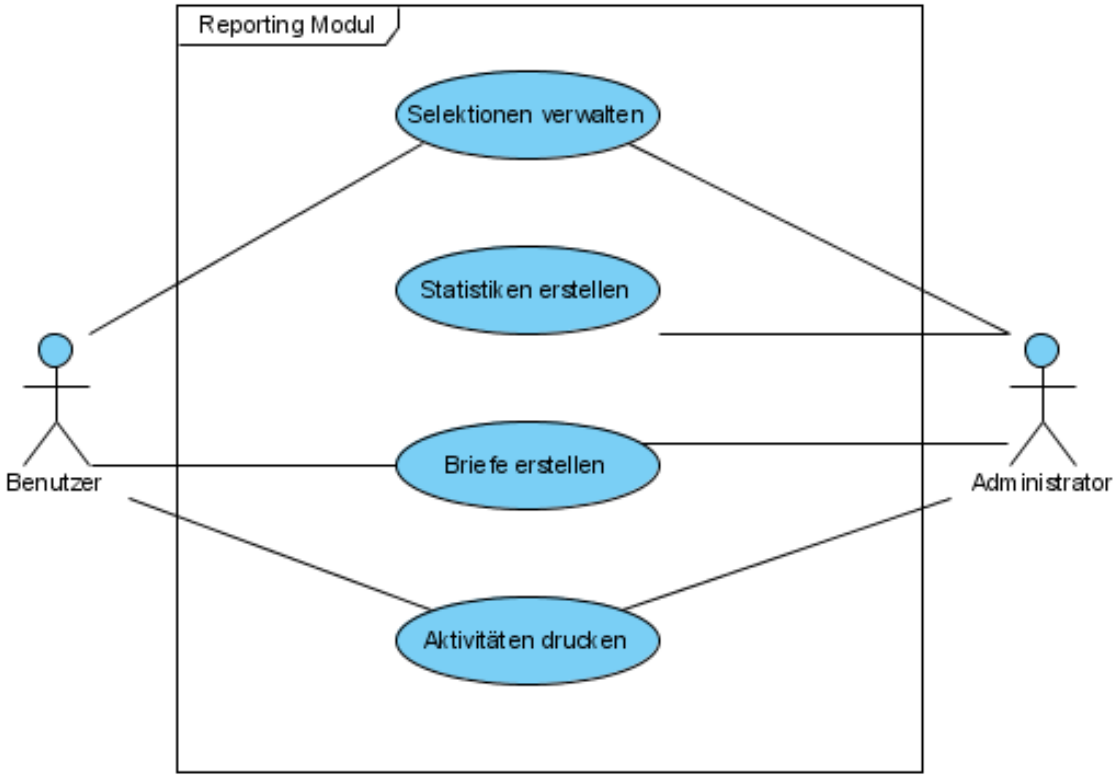

Abbildung 3.1: Anwendungsfälle für das Modul Reporting

Anwendungsfall Nr. 1.0: Selektionen verwalten (Anlegen eines Reports)
Zusammenfassung: Dieser Anwendungsfall beschreibt einen typischen Ablauf für die Erstellung einer Selektion.
Vorbedingungen: Der Benutzer muss Zugang zum System haben. Zusätzlich benötigt er ein Recht für den Zugang zum Reporting-Bereich.
Beschreibung des Szenarios: Der Benutzer möchte eine Selektion erstellen. Folgende Möglichkeiten sollen ihm für eine Selektion zur Auswahl stehen: Accounts, Contact, Activities oder Opportunities. Anschließend kann der Benutzer die Kriterien spezifizieren und dem System mitteilen, welche Datensätze dieses Moduls er benötigt. Als nächstes soll der Benutzer diesen Report für eine spätere Durchführung speichern können. Dann soll der Benutzer auswählen können, welche Eigenschaften der Datensätze (z. B. nur der Nachname und die Adresse) exportiert werden sollen. Diese Reports sollen dann genutzt werden um z. B. Serienbriefe oder Mailings zu erstellen.
Ausnahmen: **E1:** Eine Selektion könnte zu keinem Ergebnis führen. In diesem Fall ist kein Export möglich. **E2:** Beim Speichern des Reports soll eine Hinweismeldung auftreten, falls ein Report mit diesem Namen bereits existiert. Der Benutzer soll die Möglichkeit haben, einen anderen Namen einzugeben oder aber auch den alten Report zu überschreiben.

Anwendungsfall Nr. 1.1: Selektionen verwalten (Editieren eines Reports)
Zusammenfassung: Dieser Anwendungsfall beschreibt das Editieren eines Reports.
Vorbedingungen: Siehe Anwendungsfall Nr. 1.0. Zusätzlich: Der zu editierende Report muss bereits existieren.
Beschreibung des Szenarios: Der Benutzer möchte einen bereits gespeicherten Report aufrufen und editieren. Nachdem der Report aufgerufen wurde, soll der Benutzer die Kriterien ändern können. Anschließend soll es eine Möglichkeit zur Speicherung und Durchführung des Reports geben.
Ausnahmen: **E1:** Beim Speichern des Reports soll eine Hinweismeldung auftreten, falls ein Report mit diesem Namen bereits existiert. Der Benutzer soll die Möglichkeit haben, einen anderen Namen einzugeben oder aber auch den alten Report zu überschreiben. **E2:** Beim Speichern des Reports soll eine Fehlermeldung auftreten, falls der Benutzer versucht ein Report eines anderen Benutzers zu überschreiben. In diesem Fall soll der Benutzer die Möglichkeit haben, seinem Report einen anderen Namen zu geben.

Anwendungsfall Nr. 1.2: Selektionen verwalten (Löschen eines Reports)
Zusammenfassung: Dieser Anwendungsfall beschreibt das Löschen eines Reports.
Vorbedingungen: Siehe Anwendungsfall Nr. 1.1.
Beschreibung des Szenarios: Der Benutzer möchte einen bereits gespeicherten Report löschen.
Ausnahmen: **E1:** Der Report bietet keine Option zum löschen. Diese Ausnahme soll eintreten, wenn der Benutzer nicht der Ersteller des Reports ist.

Anwendungsfall Nr. 1.3: Selektionen verwalten (Kampagne erstellen)
Zusammenfassung: Dieser Anwendungsfall beschreibt das Starten einer Kampagne.
Vorbedingungen: Die Selektion muss bereits abgeschlossen sein.
Beschreibung des Szenarios: Der Benutzer hat einen Report durchgeführt und besitzt eine Liste von Kontaktpersonen (Contacts), die an einer Veranstaltung teilnehmen. Damit diese Information nicht verloren geht, soll der Benutzer die Möglichkeit haben, eine Kampagne (in Form von Aktivitäten) zu erstellen, die alle Benutzer enthält.
Ausnahmen: **E1:** Eine Selektion könnte zu keinem Ergebnis führen. In diesem Fall kann man keine Kampagnen erstellen.

Anwendungsfall Nr. 2.0: Statistiken erstellen
Zusammenfassung: Dieser Anwendungsfall beschreibt die Durchführung von Statistiken.
Vorbedingungen: Der Benutzer muss Zugang zum System haben. Zusätzlich benötigt er ein Recht für den Zugang zum Admin- und zum Statistik-Bereich.
Beschreibung des Szenarios: Der Administrator möchte gerne Statistiken einsehen. Um Statistiken durchzuführen, soll er zunächst auswählen können, welchen Zeitraum die Anwendung für die Statistik berücksichtigen soll. Zwischen folgenden Statistiken soll der Administrator auswählen können: Statistiken der Benutzer und Statistiken der Aktivitäten.
Ausnahmen: Keine Ausnahmen.

Anwendungsfall Nr. 3.0: Briefe erstellen
Zusammenfassung: Dieser Anwendungsfall beschreibt das Erstellen von Briefen.
Vorbedingungen: Der Benutzer muss Zugang zum System haben. Zusätzlich benötigt er ein Recht für die Nutzung der Export-Funktion.
Beschreibung des Szenarios: Der Benutzer möchte gerne einen Kunden (Contact) eines Unternehmens (Account) schriftlich benachrichtigen. Er soll die Möglichkeit haben, aus der Anwendung heraus eine Briefvorlage (oder auch Fax) zu erstellen, welche bereits alle wichtigen Informationen enthält.
Ausnahmen: Keine Ausnahmen.

Anwendungsfall Nr. 4.0: Aktivitäten drucken
Zusammenfassung: Dieser Anwendungsfall beschreibt das Drucken von Aktivitäten eines Unternehmens.
Vorbedingungen: Der Benutzer muss Zugang zum System haben. Zusätzlich benötigt er ein Recht für die Nutzung der Druckfunktion.
Beschreibung des Szenarios: Der Benutzer befindet sich auf einer Reise zu einem Kunden (Contact) eines Unternehmens (Account). Um sich während der Reise über die aktuellen Aktivitäten (Activities) bei diesem Unternehmen zu erkundigen, möchte er diese drucken. Um dies zu bewerkstelligen, soll der Benutzer zunächst die Detail-Seite des Accounts aufrufen. Dort soll es eine Möglichkeit geben, die Aktivitäten eines bestimmten Zeitraums in einer Druckansicht zu überführen.
Ausnahmen: **E1:** Das Unternehmen besitzt keine Aktivitäten im angegebenen Zeitraum. In diesem Fall sollen lediglich die Unternehmensdaten angezeigt werden.

3.2.2 Technische Anforderungen

Die technischen Anforderungen betreffen die Umwelt des Entwicklers sowie die hardwarenahen Anforderungen.

3.2.2.1 Gesamtsystem

Browserkompatibilität: Die gesamte Anwendung soll für den Internet Explorer 7 optimiert werden. Eine einwandfreie Funktion mit anderen Browsern wäre optimal, ist jedoch nicht zwingend notwendig.

Bibliothek: Da die Anwendung in naher Zukunft als Open Source Software veröffentlicht wird, sollen ausschließlich Bibliotheken genutzt werden, die unter einer Open Source Lizenz vertrieben werden.

Entwicklungsumgebung: Es sollen ausschließlich frei verfügbare Entwicklungsumgebungen genutzt werden.

3.2.2.2 Report Modul

Design/Layout: Bei der Entwicklung soll das aktuelle Design berücksichtigt werden.

Entwicklungssprache: Das Modul soll in der Sprache des bisher entwickelten Systems realisiert werden.

Architektur: Das Modul soll in die aktuelle Architektur eingebunden werden und die Effizienz dieser Architektur instand halten.

Datenbank: Das Modul soll in das aktuelle Datenmodell integriert werden.

3.3 Anforderungsspezifikation

Die Anforderungsspezifikation ist eine Abgrenzung der Anforderungsanalyse. Die Anforderungen werden in drei verschiedenen Gruppen klassifiziert: muss (must be), soll (should be) und Komfort (nice to have). Muss-Kriterien müssen erfüllt sein, da sonst ein wirtschaftlicher Einsatz nicht möglich ist (Proof of Concept). Soll-Kriterien sind für einen professionellen Praxiseinsatz notwendig. Komfort-Kriterien können ohne Nebenwirkungen auf spätere Versionen verschoben werden (vgl. Raasch u. a., 2007, S.11). Jede Anforderung wird zusätzlich

mit einem Datum versehen, da viele Anforderungen durch das evolutionäre Prototyping (siehe Kapitel 7.2) während der Entwicklungszeit entstanden.

3.3.1 Funktionale Anforderungen

F001: Selektionen durchführen [10.04.2008]

Das Selektionsmodul sollte folgende Module unterstützen:

- Accounts (must be)
- Contacts (must be)
- Activities (should be)
- Opportunities (should be)

Das heißt, nach den oben genannten Modulen soll eine Selektion durchführbar sein. Über folgende Eigenschaften der „must be“-Module sollten Kriterien aufgestellt werden können:

Kriterien von Accounts (Unternehmen):

- Accounttype: Die Unternehmensart (z.B. Kunde oder Lieferant)
- City: Der Ort, in dem das Unternehmen ansässig ist
- Country: Das Land, in dem das Unternehmen ansässig ist
- Generic Fields: Die aktuellen generischen Felder sowie alle Felder, die in Zukunft durch den Administrator erstellt werden könnten.
- Industry: Das Gewerbe des Unternehmens
- Owner: Der interne Zuständige für dieses Unternehmen
- Zip Code: Die Postleitzahl der Hauptadresse des Unternehmens

Kriterien von Contacts (Kontaktpersonen eines Unternehmens):

- Accounttype: Die Unternehemsart (z.B. Kunde oder Lieferant)

- Buyinginterest: Das Kaufinteresse der Kontaktperson
- City: Der Ort, in dem das Unternehmen ansässig ist
- Country: Das Land, in dem das Unternehmen ansässig ist
- Generic Fields: Die aktuellen generischen Felder sowie alle Felder, die in Zukunft durch den Administrator erstellt werden könnten.
- Industry: Das Gewerbe des Unternehmens
- Owner: Der interne Zuständige für dieses Unternehmen
- Zip Code: Die Postleitzahl der Hauptadresse des Unternehmens

Bei der Erstellung der Kriterien soll der Benutzer auswählen können, ob ein Kriterium einen bestimmten Wert, eine endliche Anzahl von möglichen Werten enthält, einen bestimmten Wert nicht oder auch gar keinen Wert enthält.

F002: Selektionen speichern/löschen [10.04.2008]
Da Selektionen oft wiederverwendet werden, soll es die Möglichkeit geben, Selektionen speichern und löschen zu können.

F003: Selektionen exportieren [10.04.2008]
Nachdem Selektionen durchgeführt werden, sollen diese in ein Format exportiert werden, welches einen Import in andere Anwendungen (z.B. Office-Anwendungen) zulässt.

F004: Selektionen Exportattribute [10.04.2008]
Es sollen sich alle denkbaren Attribute (z. B. Name oder Adresse) der Module exportieren lassen.

F005: Contacts Exportieren [10.04.2008]
Es soll eine einfache Möglichkeit geben, mit der Benutzer ihre Briefvorlagen mit den Kontaktdaten füllen können.

F006: Selektionen Rechtemanagement [15.05.2008]
Das Löschen einer Selektion darf nur durch den Ersteller erfolgen. Selektionen anderer Ersteller sollen sich auch nicht überschreiben lassen. Zudem darf der Ersteller der

Selektion auswählen, ob er diese auch für andere Benutzer zugängig machen möchte, oder ob nur er die von ihm erstellte Selektion sehen und benutzen darf.

F007: Aktivitäten drucken [10.07.2008]

Es soll eine Möglichkeit geben, Aktivitäten eines Unternehmens zu drucken. Dabei soll der Benutzer optional angeben können, ob die zu exportierenden Aktivitäten innerhalb eines bestimmten Zeitraums liegen sollen.

F008: Selektionen Kampagnen hinzufügen [01.08.2008]

Da Selektionen meistens mit Kampagnen verbunden sind, soll es die Möglichkeit geben, eine Kampagne in Form von Aktivitäten nach einer Selektion anzulegen.

F009: Statistiken erstellen [15.08.2008]

Der Administrator soll die Möglichkeit haben, Benutzer- und Aktivitäts-Statistiken durchzuführen. Bei Benutzer-Statistiken soll das Opportunitiy-Modul sowie das Nutzverhalten aller Module (d. h. anlegen oder editieren von Activities, Accounts, Contacts oder Opportunities) berücksichtigt werden. Bei Aktivitäts-Statistiken soll die Industrie (Industry) und das Kaufinteresse (Buyinginterest) berücksichtigt werden.

3.3.2 Nichtfunktionale Anforderungen und Qualitätskriterien

Die nichtfunktionalen Anforderungen werden nach der DIN-Norm DIN 6672 festgelegt. Diese Norm beinhaltet u. a. die Qualitätsmerkmale für Software nach ISO/IEC 9126 (vgl. Usability-Net, 2006).

NF001: Funktionalität

Die gesamte Anwendung sowie das zusätzliche Report Modul soll einwandfrei funktionieren. Dazu müssen z.B. alle Selektionen zu richtigen Ergebnissen führen und allen nicht-Autorisierten der Zugriff verweigert werden.

NF002: Zuverlässigkeit

Das Modul soll selbst auf fehlerhafte Eingaben angemessen reagieren und insgesamt stabil laufen. Fehler sollen nicht zu Abstürzen des Moduls oder der Anwendung führen.

NF003: Benutzbarkeit

Das Modul soll selbst für nicht geschulte Anwender einfach zu bedienen sein, so dass keine Aufwendungen für ausführliche Einarbeitungen anfallen. Das Modul soll den Benutzer bei der Arbeit unterstützen und nicht behindern. Eine Anforderung für die Erfüllung dieses Kriteriums ist das Aufrechterhalten des bisherigen Designs, so dass dieses Modul für den Anwender nicht als solches erfasst wird.

NF004: Effizienz

Die Erstellung eines Reports soll in kurzer Zeit möglich sein. Die Dauer für die Durchführung des Reports ist zunächst nebensächlich, da sie die Dauer des Altsystems (bis zu 45 Minuten) voraussichtlich nicht überschreitet.

NF005: Änderbarkeit

Das Modul soll unter den wichtigsten architektonischen Gesichtspunkten entwickelt werden, so dass eine Erweiterung mit wenig Aufwand möglich ist. Änderungen sollten in Form von Erweiterungen implementiert werden, damit das Offen-Geschlossen-Prinzip nicht verletzt wird (vgl. Meyer, 1988).

NF006: Übertragbarkeit

Das Modul soll nahtlos in die aktuelle Anwendung integriert werden. Unter den Aspekten der bisherigen Anwendung soll eine Portierbarkeit zu anderen Datenbanksystemen angestrebt werden.

4 Marktanalyse

In diesem Kapitel soll der praxisbezogene Einsatz von Reports in CRM-Systemen untersucht werden. Dabei gibt es zwei verschiedene Blickwinkel: Aus der fachlichen Sicht werden einige CRM-Systeme mit Reporting-Funktionen analysiert. Anschließend werden aus technischer Sicht einige Reporting Tools untersucht. Dabei handelt es sich um Fertigprodukte, die in das aktuelle CRM-System integriert werden könnten.

4.1 Untersuchung existierender CRM-Systeme

In der Bachelorarbeit von Martin Sadowski wurden bereits einige existierende CRM-Systeme auf ihre Funktionen untersucht (vgl. Sadowski, 2008, S.31ff). Im Folgenden soll der Bereich Reporting einiger Systeme untersucht werden. Dabei werden nicht zuletzt nur Open Source CRM-Systeme wie in der Arbeit von Martin Sadowski (vgl. Sadowski, 2008) analysiert, sondern auch kommerzielle Systeme, die sich im CRM-Markt etabliert haben. Der Grund dafür ist, dass in dieser Untersuchung keine vollständigen Lösungen bzw. Implementierungen für das Report Modul gesucht werden, sondern eher in der Praxis eingesetzte Lösungen validiert werden, um neue Ideen und Verbesserungen ausfindig zu machen. Diese neue Ideen unterstützen die anschließende Entwicklungsphase.

4.1.1 Salesforce Professional Edition

Hersteller	Salesforce.com
Website	http://www.salesforce.com/de
Demo	https://login.salesforce.com (nur für registrierte Nutzer)
Version	unbekannt

Salesforce.com ist Marktführer im Bereich CRM und bietet ihre Applikation als Mietsoftware an. Für eine Demonstration ist eine Registrierung notwendig, wodurch man 30 Tage Zugang zu einem Testsystem hat.
Salesforce bietet in allen Versionen (Professional oder Group Edition) die Reportfunktion an.

Das gesamte Modul verbreitet sich in diesem System über drei Bereiche:

- Dashboard:
  Das Dashboard befindet sich auf der Startseite wie in Abbildung 4.1 zu sehen ist. In diesem Bereich erhält man einen Einblick über aktuelle Statistiken des Unternehmens. Das Dashboard lässt sich manuell passen, damit einige Statistiken aus- oder eingeblendet werden können.

- Berichte:
  Der Bereich Berichte ermöglicht die Selektion von Datensätzen. Hier werden zwischen öffentlichen und privaten Berichten unterschieden. Die Berichterstellung verbreitet sich über sechs Seiten, damit alle Anforderungen des Benutzers erfüllt werden können.

- Export:
  Exporte einzelner oder mehrerer Objekte sind an fast allen Stellen möglich. Die Daten lassen sich in das CSV-Format exportieren.

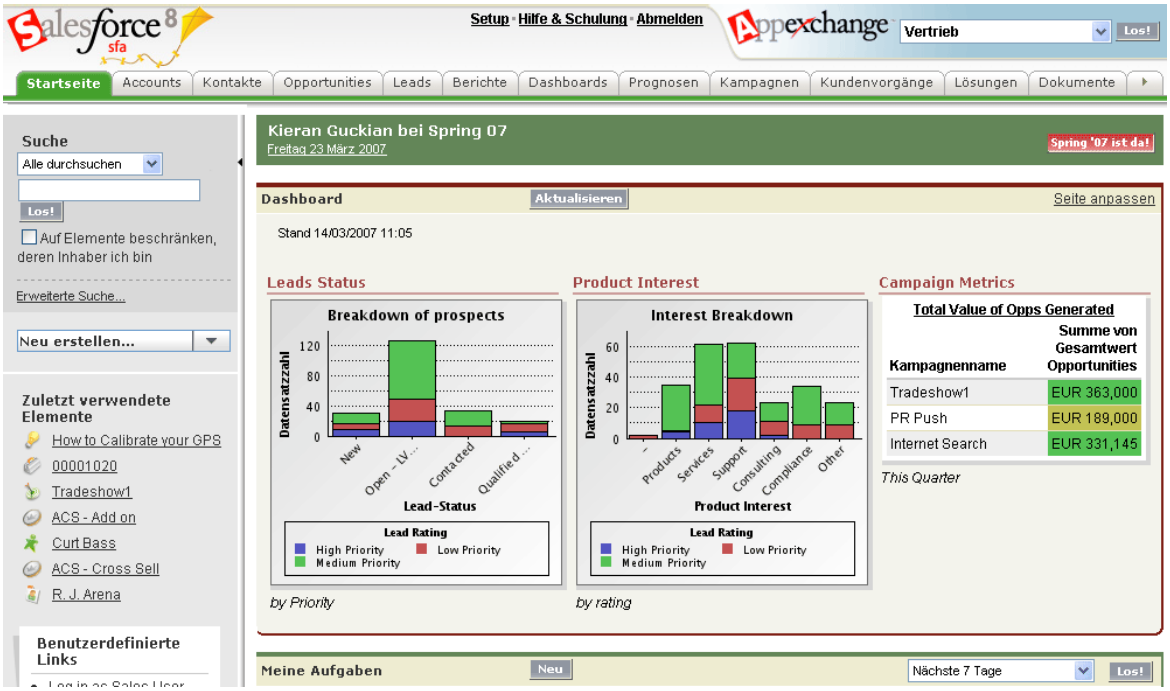

Abbildung 4.1: Reporting in Salesforce

4.1.2 SugarCRM Enterprise Edition

Hersteller	SugarCRM Inc.
Website	http://www.sugarcrm.com/crm/
Demo	http://demo.sugarcrm.com (will / will)
Version	5.1.0

Die Firma SugarCRM bietet ihr CRM-Systeme in drei verschiedenen Versionen an, wovon eine Version als Open Source Community Edition vertrieben wird. Da das Modul Reporting in der Community Edition nicht zur Verfügung steht, wird die Enterprise Edition untersucht.

Das Modul Reporting verbreitet sich in diesem System ebenfalls über drei Bereiche:

- Home:

  Über den Reiter Home kann man über einen zweiten Reiter die „Sales Page“ (Verkaufsangelegenheiten) und „Marketing & Support Page“ (Aktivitäten) einsehen. Diese beiden Seiten bieten zahlreiche Statistiken zur Auswahl.

- Berichte:

  Der Bereich Selektion befindet sich hinter einen eigenen Reiter „Berichte“, wie in Abb. 4.2 zu sehen ist. Dort wird der Benutzer durch ein sechsseitiges Menü geführt, falls er einen neuen Bericht erstellen will. Unter anderem kann ausgewählt werden, welche Filter/Kriterien gesetzt werden sollen und nach welcher Spalte die Liste sortiert werden soll.

- Export:

  Exporte einzelner Objekte sind auch in der Community Version möglich. Dazu muss man über den Reiter des Objekts (z.B. Kontakte, Unternehmen oder Aktivitäten) die einzelnen Objekte markieren und den Export in Gang setzen. Anschließend werden alle Daten in das CSV Format übertragen.

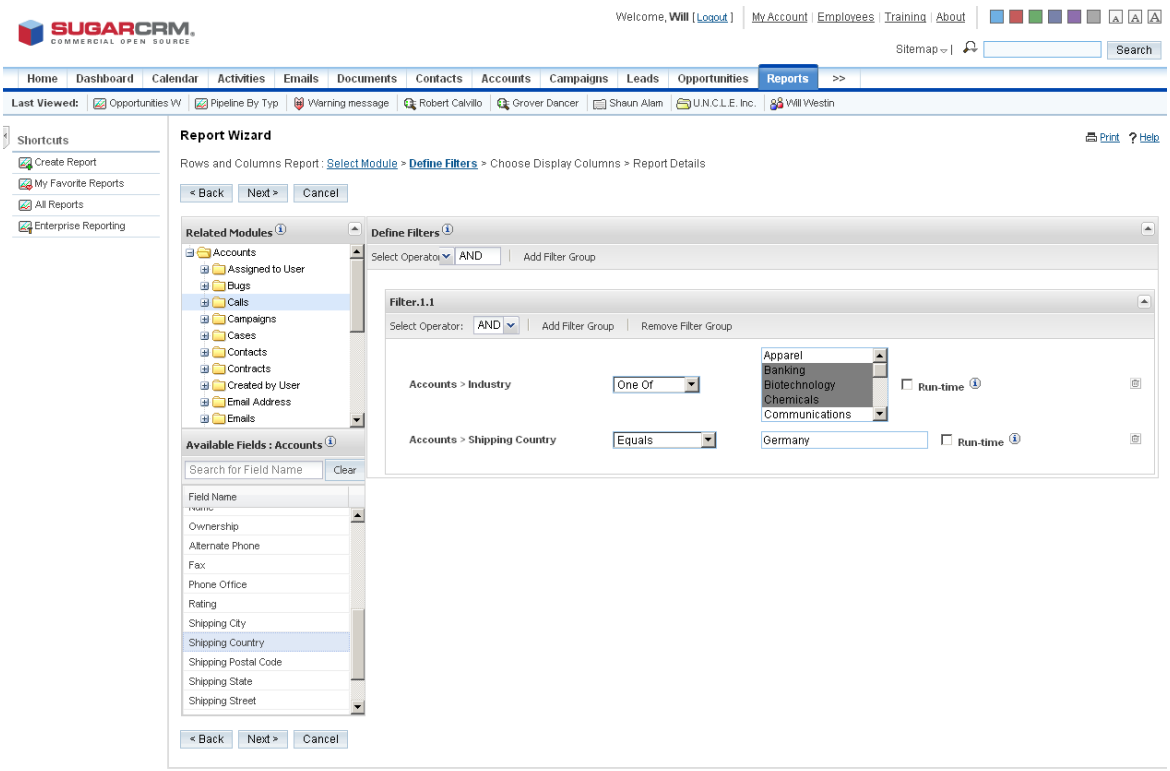

Abbildung 4.2: Reporting in SugarCRM

4.1.3 vTiger CRM

Hersteller	vtiger.com
Website	http://vtiger.com
Demo	https://vtigercrm5x.at.crm-now.de (admin / admin)
Version	5.0.4

vTiger CRM ist, im Gegensatz zu SugarCRM, ein reines Open Source CRM-System. Die Basis dieser Anwendung beruht auf SugarCRM. Aus diesem Grund sind einige Merkmale, auch im Bereich Reports, identisch.

Auch hier besteht das Modul Reporting aus drei Bereiche:

- Auswertung Cockpit:
  Über den Reiter Auswertung kann man über einen zweiten Reiter die Seite „Cockpit“ aufrufen. Auf dieser Seite lassen sich diverse Statistiken aufrufen und ebenfalls grafisch anzeigen.
- Auswertung Berichte:
  Auf der Seite „Berichte“ lassen sich alle vorhandenen Berichte einsehen und auch neue Berichte erstellen. Dazu wird man durch ein siebenseitiges Menü geführt um den Bericht zu spezifizieren (siehe Abb. 4.3).
- Export:
  Hier gibt es keine erwähnenswerten Unterschiede im Vergleich zu der Export-Funktion in SugarCRM.

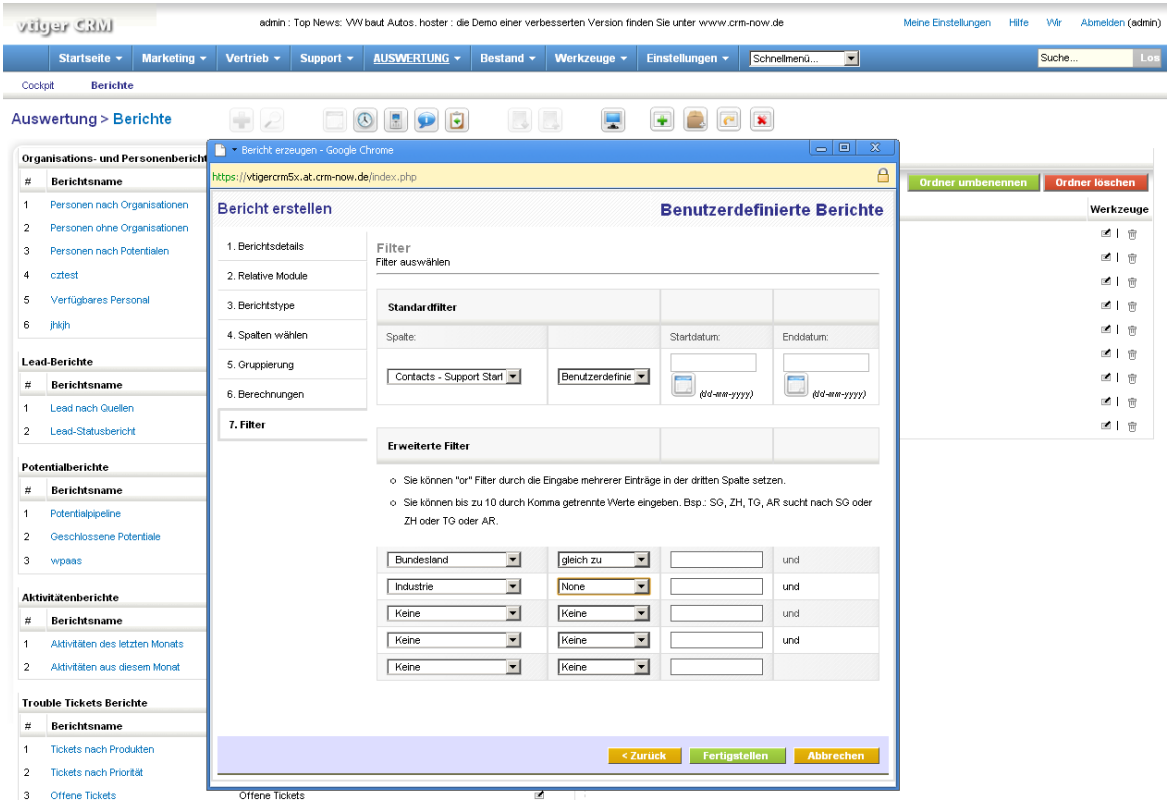

Abbildung 4.3: Reporting in vTiger CRM

4.2 Untersuchung existierender Reporting Tools

In diesem Abschnitt werden einige java-basierte Open Source Reporting Tools untersucht. Dabei wird in Frage gestellt, ob und inwieweit diese Tools die Entwicklung des Report Moduls unterstützen könnten.

4.2.1 JasperReports

Hersteller	JasperSoft
Website	http://jasperforge.org
Version	3.0.0 (19. März 2008)

Mit JasperReports erhält man eine Javabibliothek, welche sich problemlos in Java-Anwendungen integrieren lassen soll. Diese Bibliothek stammt von der Firma JasperSoft, welche 2005 gegründet wurde. JasperReports existiert jedoch bereits seit 2001.

JasperSoft bietet ein Designer-Werkzeug, iReport, um zunächst Report-Templates zu erstellen. Diese Templates spezifizieren das Layout des Reports und die SQL Anweisung, welche die gewünschten Daten zurückliefert. Anschließend kann zur Laufzeit durch den Jasper Compiler von JasperReports ein Exemplar dieses Templates erstellt werden, welches alle gewünschten Daten enthält (siehe Abb. 4.4).

Folgende Aspekte zeichnen diese Bibliothek aus (vgl. JasperSoft, 2006):

- Export nach PDF, HTML, XLS, CVS, RTF, TXT und XML möglich
- Integrierter Viewer, um sich Reports nach der Erstellung anzuschauen
- Unterstützt zusätzlich die EJB-Spezifikation (u.a. Hibernate und POJOs)
- Grafische Darstellung von Statistiken möglich
- Bietet iReport als kostenloses Werkzeug um Templates zu erstellen
- Subreports für z. B. verschachtelte SQL Anweisungen

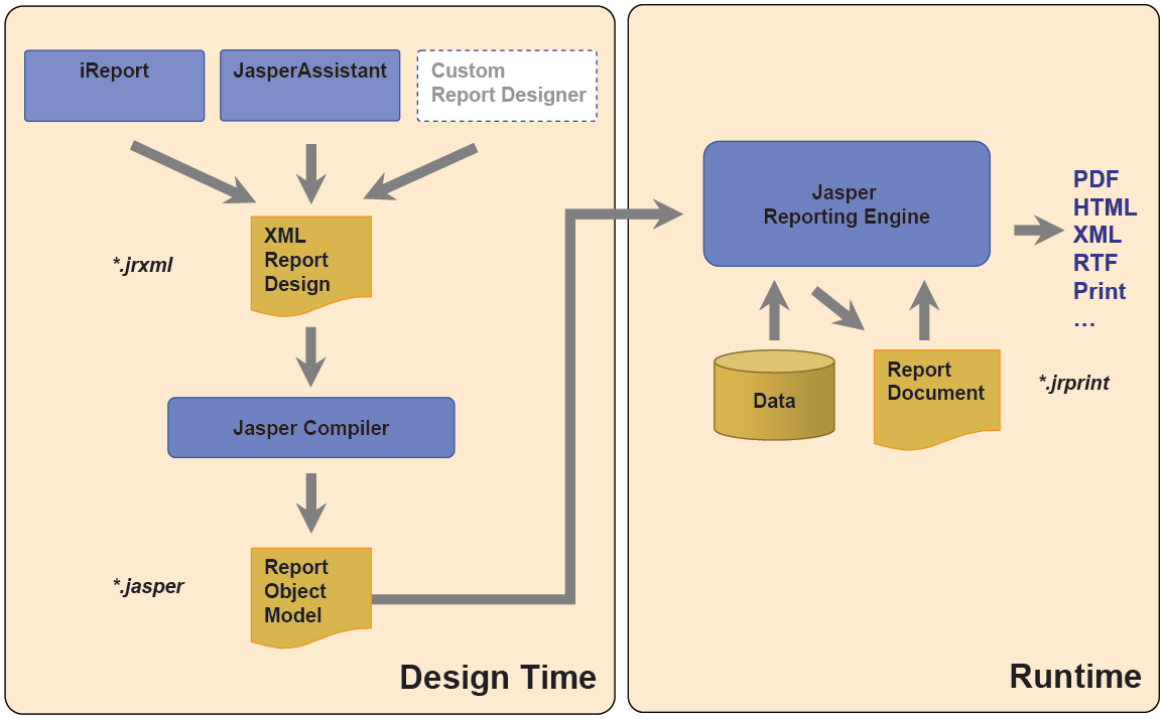

Abbildung 4.4: Ablauf des Reportings bei JasperReports

4.2.2 Pentaho Reporting (ehemals JFreeReport)

Hersteller	Pentaho
Website	http://reporting.pentaho.org
Version	1.7.1 (29. August 2008)

Pentaho Reporting entstand durch die Übernahme von JFreeReport im Januar 2006. Pentaho ist hauptsächlich Anbieter von verschiedenen Open Source Business Intelligence Anwendungen.

Auch Pentaho bietet einen kostenlosen Report Designer, um Report Templates mit wenig Aufwand zu erstellen. Durch die Unterstützung von JDBC-Treibern werden zahlreiche relationale Datenbankmanagementsysteme (RDBMS) unterstützt.

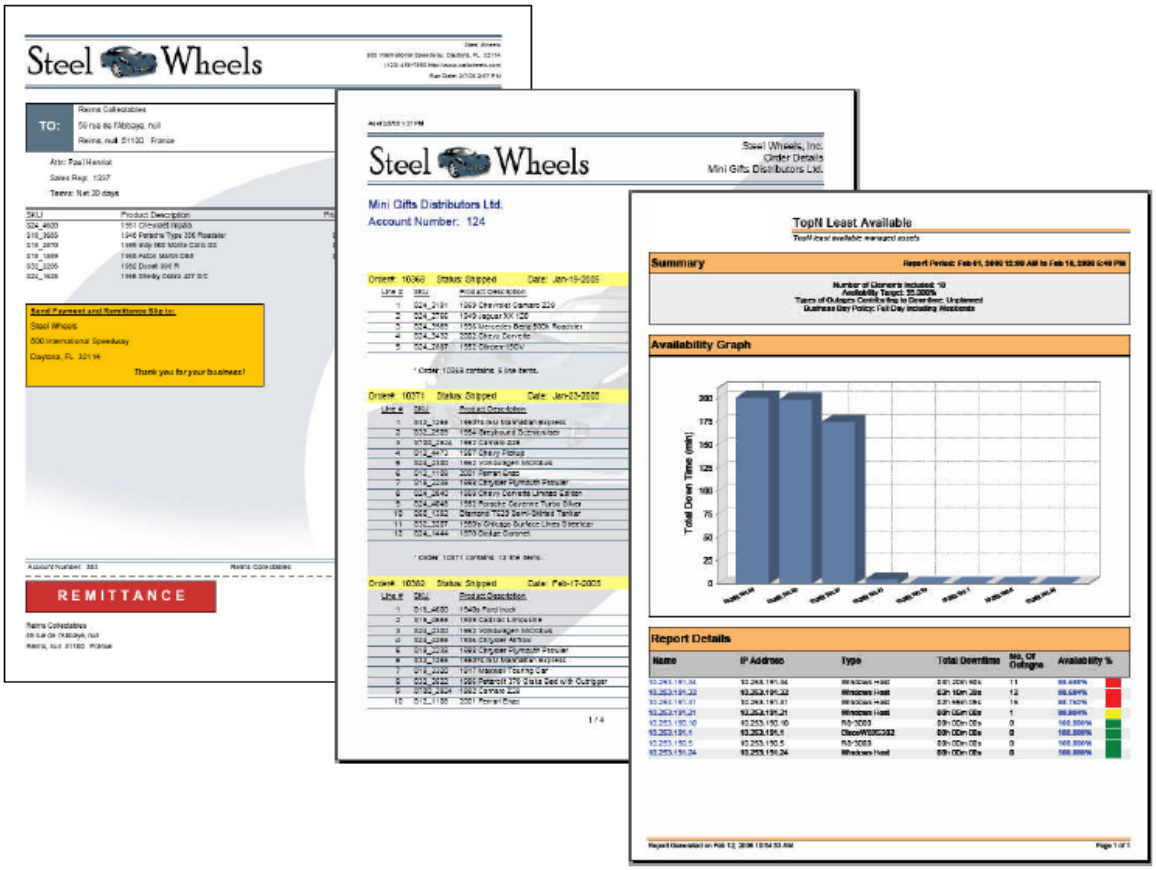

Abbildung 4.5: Beispiel-Report erstellt durch Pentaho Reporting

Folgende Aspekte zeichnen diese Bibliothek aus (vgl. Pentaho, 2007):

- Export nach PDF, HTML, XLS, RTF und TXT
- Integrierte Webanwendung, um Reports zur Laufzeit selbst zusammenzustellen
- Unterstützt reine SQL Anweisungen
- Grafische Darstellung von Statistiken möglich
- Bietet Report Designer als kostenloses Werkzeug um Templates zu erstellen
- Subreports für z. B. verschachtelte SQL Anweisungen

4.2.3 DataVision

Hersteller	DataVision
Website	http://datavision.sourceforge.net
Version	1.2.0 (13. Juli 2008)

DataVision ist ein Reporting Tool, welches dem kommerziellen Reporting Tool Crystal Reports (vom Hersteller Business Objects / SAP) ähnelt. Dieses Tool wird ebenfalls unter einer Open Source Lizenz vertrieben.

DataVision arbeitet gleichermaßen wie die bereits vorgestellten Reporting Tools. Über einen GUI-basiertes Werkzeug (siehe Abb. 4.6) lassen sich zunächst Report Templates erstellen, die dann zur Laufzeit durch einen speziellen Compiler mit Daten gefüllt werden.

Folgende Aspekte zeichnen diese Bibliothek aus (vgl. DataVision, 2008):

- Export nach PDF, HTML, XLS, XML, DocBook und LaTeX2e
- Unterstützt reine SQL Anweisungen
- Bietet Report Designer als kostenloses Werkzeug, um Templates zu erstellen
- Subreports für z. B. verschaltete SQL Anweisungen

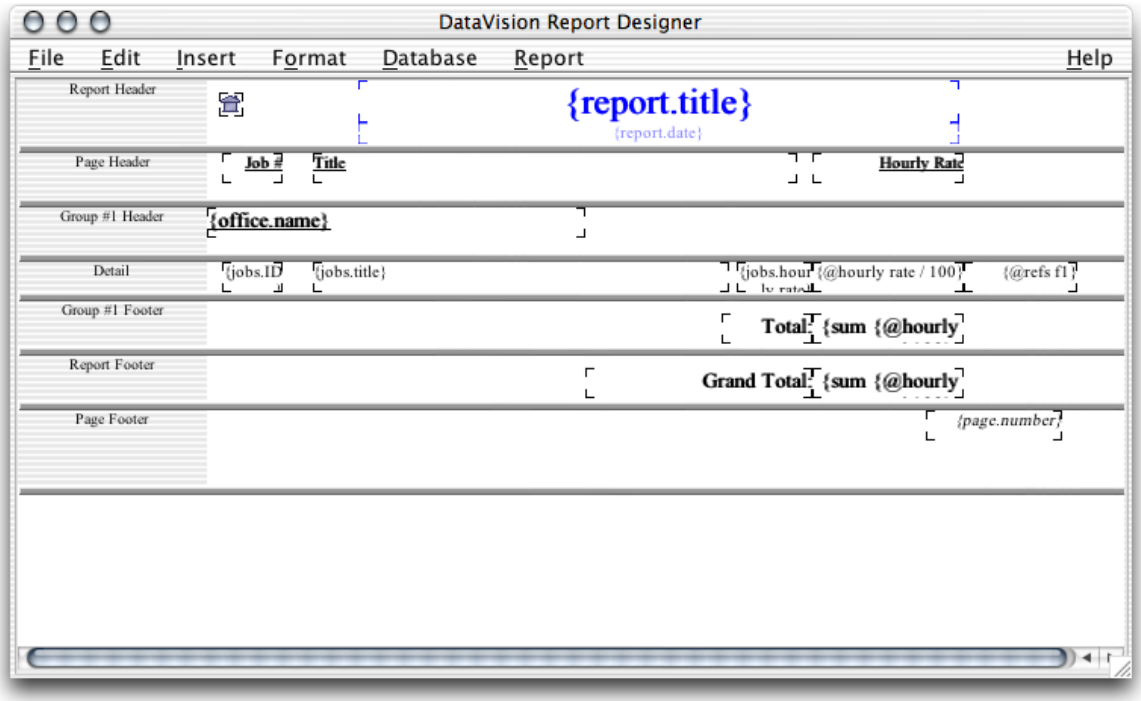

Abbildung 4.6: Report Designer Werkzeug von DataVision

4.3 Fazit

4.3.1 Zusammenfassung CRM-Systeme

Alle oben genannten CRM-Systeme bieten ähnliche Funktionen, die auf unterschiedliche Weise bedient werden müssen. Salesforce bietet als kommerzielle Software und Marktführer die meisten Funktionen im Bereich Reporting. Dennoch werden nicht alle gewünschten Anforderungen erfüllt. Im Folgenden werden die drei Bereiche der fachlichen Anforderungsanalyse (siehe Kapitel 3.2.1) mit den Funktionen der CRM-Systeme verglichen.

- Selektion erstellen/verwalten:
  Jedes der oben genannten CRM-Systeme erfüllt alle funktionalen Anforderungen. Der Benutzer kann Kriterien jeder Art und über alle Eigenschaften eines Moduls (z. B. Contacts, Accounts, Activities oder Opportunities) setzen. Obwohl alle Funktionen unterstützt werden, gibt es in diesem Bereich starke Einschränkungen in der Benutzer-

freundlichkeit. Bei allen Systemen ist die Erstellung einer Selektion sehr zeitaufwändig und ohne ausreichende Vorkenntnis nicht möglich. Es werden z. B. in einigen Systemen Kenntnisse der Aussagenlogik vorausgesetzt. Durch die Nutzung der Aussagenlogik wird eine einheitliche, korrekte und klare Struktur geschaffen, die alle denkbaren Anforderungen erfüllen kann.
Jedoch können auch Fehler entstehen, wenn die Benutzer mit der Aussagenlogik nicht vertraut sind. Solche Fehler sind jedoch meistens auf unzureichende Kenntnisse der Benutzer zurückzuführen.

- Briefe erstellen:
  Eine direkte Brieffunktion gibt es in keinem System. Lediglich ein Export nach CVS ist möglich, was einen späteren Import in beliebige Office-Anwendungen möglich. Da dies jedoch zeitaufwändig ist, würde diese Lösung keine Akzeptanz gewinnen.

- Aktivitäten drucken:
  In den Anforderungen wird gewünscht, dass die Aktivitäten übersichtlich ausgedruckt werden können. In allen oben genannten Systemen ist ausschließlich ein Ausdruck in das CVS Format möglich. Die oben genannten Systeme würden diesen Anforderungen nicht standhalten.

4.3.2 Zusammenfassung Reporting Tools

Alles in allem verhalten sich die oben genannten Reporting Tools bis auf einige Unterschiede identisch. Alle bieten ein Werkzeug zum Erstellen von Templates, einen Compiler zum Füllen dieser Reports und viele Exportformate, die sehr nützlich sind.
Vor allem JasperReports scheint jedoch etwas ausgereifter zu sein, da hier die EJB-Spezifikation unterstützt wird. Da EJB im bisher entwickelten System zum Einsatz kommt, wäre somit JasperReports die einzige Komponente, die in Frage kommt. Bei den anderen Reporting Tools würde eine Verletzung der bisher aufgestellten Architektur (Schmer, 2008, S.33) auftreten, da wir die Datenbankzugriffe nicht mittels der Java Persistence API (siehe Kapitel 2.2.3.3), sondern durch direkte Kommunikation mit der Datenbank durch den Java

Database Connector (JDBC) durchführen würden. Eine datenbankunabhängige Implementierung wäre somit ebenfalls ausschlossen, da Zugriffe über JDBC datenbankspezifische SQL Anweisungen verlangen.

Abschließend nach der Analyse der Reporting Tools und der Bewertung wird dennoch eine Individualentwicklung des Bereichs Reporting angestrebt. Die Gründe sind die Folgenden:

JasperReports ist ein mächtiges Reporting Tool, welches alle Anforderungen erfüllen könnte. Dennoch würde dieses Tool die Entwicklung des Bereichs Selektion eher behindern als unterstützen. Es werden bei diesem Tool zwar POJOs unterstützt, jedoch nicht im ausreichenden Maße. Pro erstelltem Report-Template wird immer nur ein einziger Klassentyp unterstützt. Das heißt, wenn wir eine Liste von Contacts haben und den zugehörigen Account (welcher einen anderes POJO repräsentiert) haben wollen, müssen sogenannte Sub-Reports zum Einsatz kommen. Sub-Reports sind eigenständige Report-Templates, die in die eigentlichen Report-Templates integriert werden. Durch diese Kombination von Reports und Sub-Reports entsteht ein zusätzlicher Arbeitsaufwand, welcher natürlich auch steigt, wenn die Anwendung erweitert wird (Adelchi, 2004, S.6).
Zusätzlich wird die gesamte Anwendung schwergewichtiger, da die JasperReports Library viele andere Bibliotheken voraussetzt (Adelchi, 2004, S.34).

Durch eine individuelle Lösung könnte ein Modul entstehen, welches auf die EJB-Spezifikation abgestimmt ist und auf die Bedürfnisse des Auftraggebers angepasst ist. Dies führt zu einem Modul, welches sich auf die wesentlichen Anforderungen konzentriert und leichter zu erweitern bzw. zu warten ist. JasperReports ist in einigen Bereichen „oversized“(z. B. in den Exportmöglichkeiten und Designvorlagen) und an anderen Stellen „undersized“(z. B. die EJB-Unterstützung).

5 Vision

In diesem Kapitel werden zunächst die Ziele der Entwicklung beschrieben. Alle beschriebenen Ziele sollten demnach erfüllt werden, damit das Reporting-Modul seinen Zweck erfüllt. Die zukünftigen Ziele beschreiben die Visionen, die als nächstes angestrebt werden sollen, um einen professionellen Einsatz in allen Bereichen zu gewährleisten.

5.1 Kurzfristige Ziele

Das kurzfristige Ziel beschreibt die nächsten Schritte, welche zu einer Anwendung führen, die allen funktionalen Anforderungen Stand hält. Es muss gewährleistet werden, dass zum Beispiel alle Selektionen zu richtigen Ergebnissen führen. Syntaktisch falsche Selektionen müssen verhindert werden, damit die Benutzer diese verfälschten Daten nicht weiterverarbeiten.

Als nächstes sollten alle nichtfunktionalen Anforderungen erfüllt werden. Der gesamte Bereich Reporting sollte benutzerfreundlich gestaltet werden, damit die Benutzer ohne Einarbeitung zurechtkommen.
Als Grundlage dient hier eine unterstützende Sichtweise des Systems im Gegensatz zu einer ablaufsteuernden Sichtweise. Bei der ablaufsteuernden Sichtweise wird der Benutzer gezwungen, sich an bestimmte Arbeitsabläufe zu halten. Für den Benutzer wäre die Arbeit mit diesem System langatmig, da bei solchen Systemen der Mensch zu einem Arbeitsablauf „programmiert“ wird. Eine ablaufsteuernde Sichtweise kam bei dem Altsystem zum Einsatz und hat sich wie in Kapitel 3.1 beschrieben nicht bewährt.
Bei einer unterstützenden Sichtweise bleibt die Ablaufkontrolle in der Hand des Benutzers.

Er kann z. B. auswählen, wann er einen Ablauf unterbrechen möchte, welche Schritte eines Ablaufs er erledigen möchte und wie der Ablauf eines Prozesses verlaufen soll (vgl. Raasch, 2007, S.9).

Um alle diese Ziele zu erreichen und frühestmöglich während der Entwicklung ein Feedback von den Anwendern zu erhalten, wird bewusst Prototyping (siehe dazu ausführlich Kapitel 7.2) als Entwicklungsmethode verwendet.

5.2 Zukünftige Ziele

Im Laufe der Zeit entstehen immer wieder neue Anforderungen, die sowohl durch neue Märkte als auch strukturelle und strategische Veränderungen innerhalb eines Unternehmens entwickelt werden. Um auf alle diese Anforderungen vorbereitet zu sein, sollte der Einsatz eines Data Warehouse-Systems in Betracht gezogen werden.

> *„Wollen Unternehmen zukünftig erfolgreich sein, ist es nötig, die zur Verfügung stehenden Informationen sinnvoll zu nutzen. Aus technischer Sicht ist dies derzeitig über den Einsatz von Data Warehouse Systemen und Analysewerkzeugen realisierbar, was für die Notwendigkeit eines Einsatzes spricht.“*(Blume, 2007, S.90)

Demnach ist es vor allem bei wachsenden Unternehmen erforderlich, alle denkbaren Daten analytisch zu betrachten, um neue Potentiale zu entdecken. Für kleine Unternehmen nennt Katrin Blume einige Aspekte, die beachtet werden sollten (vgl. Blume, 2007, S.85ff):

- *Komplexität:* Durch ein Data Warehouse-System wird ein bisheriges System um ein komplexes Modul erweitert. Dadurch entstehen u. a. Defizite in der Wartbarkeit und Portierbarkeit des Gesamtsystems.
- *Einsatz*: Laut der Befragung einiger Unternehmen werden Data Warehouse-Systeme überwiegend bei Großunternehmen eingesetzt. Kleine Unternehmen halten ihre Daten für übersichtlich und sehen keine Vorteile durch die Nutzung von Data Warehouse-Systemen (vgl. Blume, 2007, S.62ff).

- *Aufwand u. Einarbeitung*: Der Aufwand zur Integration eines Data Warehouse-Systems ist wesentlich kleiner als der Nutzen, den man durch den Einsatz erhält. Dennoch kann eine vollständige Integration bis zu zwölf Monate andauern.

Die oben genannten Probleme verhindern einen schnellen und kurzfristigen Einsatz eines Reportmoduls in das aktuelle CRM-System. Dennoch sollte bei zukünftigen Anforderungen und daraus entstehenden Weiterentwicklungen ein Wechsel zu einem Data Warehouse-System in Betracht gezogen werden.
Es gibt bereits einige Open Source Data Warehouse-Systeme auf dem Markt, wie z.B. die Business Intelligence Suite von der Firma Pentahoo (vgl. Pentaho, 2008) oder auch die Data Warehouse-Lösung von Infobright (vgl. Infobright, 2008).

6 Architektur

> *„Eine Softwarearchitektur bezeichnet die Modelle und die konkreten Komponenten eines Softwaresystems in ihrem statischen und dynamischen Zusammenspiel. Sie kann selbst als explizites Modell dargestellt werden. Eine Softwarearchitektur beschreibt ein konkretes System in seinem Anwendungskontext.“* (Züllighoven u. a., 1998, S.324)

Diese Softwarearchitektur beschreibt demnach eine notwendige Grundlage, die geschaffen werden muss, um ein System zu realisieren, dass allen Anforderungen entspricht. Es gibt unzählige Definitionen der Softwarearchitektur, die sich in der Regel ergänzen (vgl. CMU, 2005).

Die Architektur muss in erster Linie alle nicht funktionalen Anforderungen berücksichtigen (vgl. Keller, 2002, S.12). Sollten diese Anforderungen (siehe Kapitel 3.3.2) durch eine konzipierte Architektur nicht erfüllt werden, muss diese Architektur erneut evaluiert werden. Der Evaluierungsprozess wird durch das Prototyping (siehe Kapitel 7.2) unterstützt. Zu den nichtfunktionalen Anforderungen gehören Beschreibungen der Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit des Systems.
Durch diese Bedingungen, die eine Architektur erfüllen muss, werden Randbedingungen geschaffen, die den Aufbau des Software-Systems stark beeinflussen können.

In der architektonischen Phase wird demnach ein Fundament für die Realisierung des Reportmoduls geschaffen.

6.1 Fachlich

Um die Anforderungen der fachlichen Analyse in Kapitel 3.3 zu erfüllen, wird zunächst eine fachliche Architektur geschaffen.

Das gesamte Modul wird in drei unterschiedliche Komponenten aufgeteilt: Selektion, Export und Statistik. Damit eine möglichst hohe Kohäsion entsteht, werden Abhängigkeiten und Kompositionen zwischen den Komponenten vermieden. Unterstützt wird dies durch die EJB-Architektur, welche beschreibt, dass für jede erstellte Service-Klasse eine Schnittstelle definiert werden muss.

Selektion Die Selektionskomponente befasst sich mit dem Hauptbestandteil des Reportmoduls. Über benutzerdefinierte Kriterien soll ein Benutzer operative Daten aus dem System auswerten und exportieren können. Der Zugang zu dem Selektionsbereich soll nur bestimmten Benutzern ermöglicht werden. Die Komponente lässt sich in die Bereiche „Kontakte-Reports“ und „Unternehmen-Reports“ aufteilen.

Export Die Exportkomponente beschäftigt sich mit dem Export von Kontaktdaten in vorhandene Office-Vorlagen. Diese Komponente soll so gestaltet werden, dass der Administrator ohne großen Aufwand neue Vorlagen in das System einbinden kann.

Statistik Um Informationsdaten des Unternehmens auszuwerten, werden Statistiken in den Bereichen Activities (Aktivitäten), Opportunities (Verkaufsgelegenheiten) und Users (Benutzer) benötigt. Für diese Auswertungen soll ein Start- und End-Datum definiert werden können.

6.1.1 Datenebene

Für die Selektionskomponente und für die Statistikkomponente ist eine Erweiterung des bisherigen Datenmodells erforderlich. Eine Datenerhaltungsbasis für die Exportkomponente wurde bereits im Zuge der Qualitätssicherung (siehe Kapitel 8.4.4) geschaffen. Die Abbildung 6.1 zeigt die Erweiterung des Datenmodells.

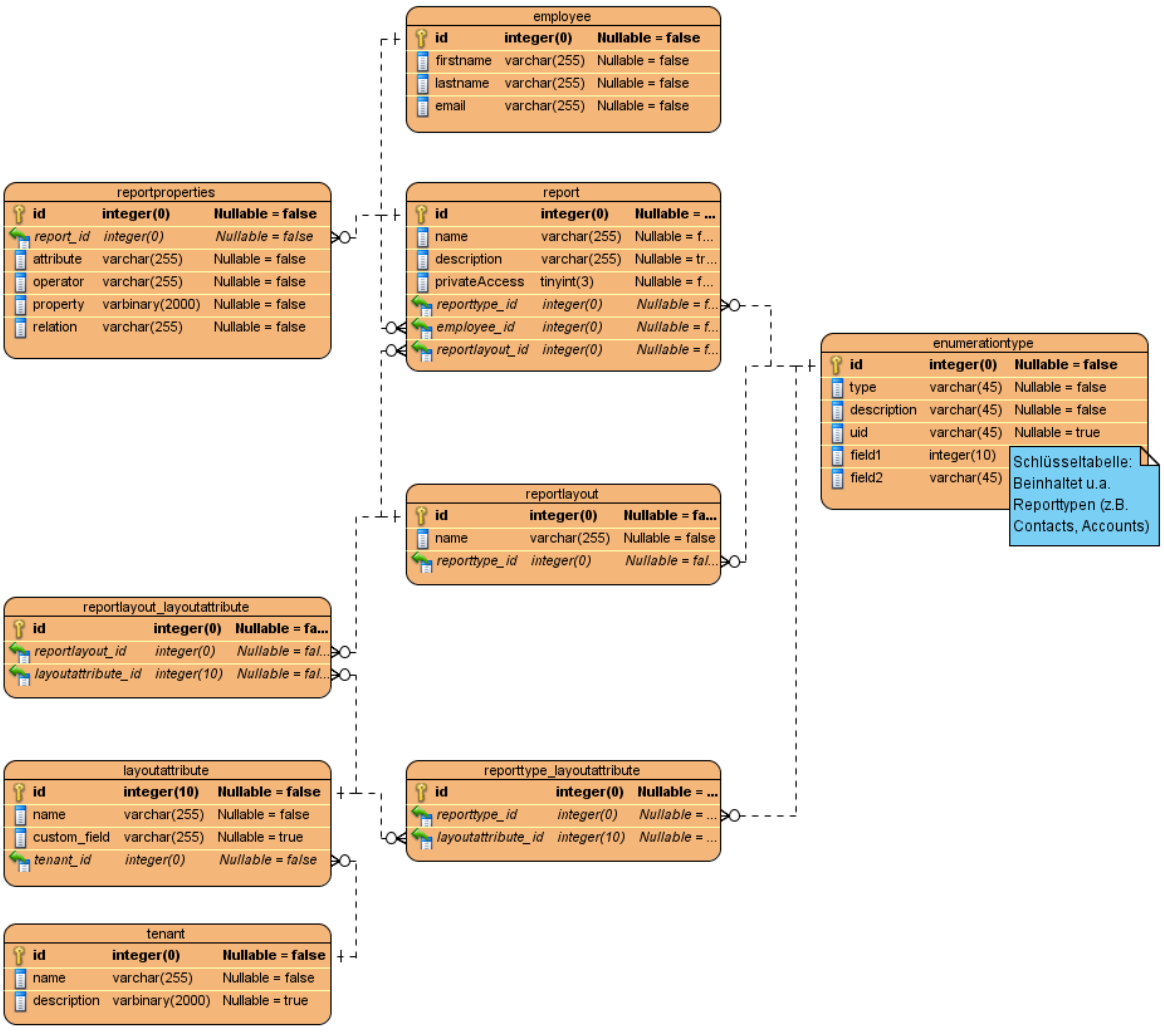

Abbildung 6.1: ER-Diagramm des Reporting Moduls

- **enumerationtype:** Die `enumerationtype`-Tabelle wurde im Rahmen der Qualitätssicherung erstellt, um mehrere Schlüsseltabellen abzulösen (siehe dazu ausführlich Kapitel 8.4.4). Dabei sind die Felder `type` und `description` die einzigen Pflichtfelder. Über die Restlichen können weitere spezifische Daten eines Typs eingetragen werden. Beispiel:

```
id = 2
type = "reporttype"
description = "contacts"
uid = null
field1 = null
field2 = null
```

Listing 6.1: Beispieldaten der EnumerationType-Tabelle

- **report:** In der `report`-Tabelle werden Selektionen für einen späteren Aufruf gespeichert. Die Markierung `privateAccess` definiert, ob es sich dabei um eine öffentliche (und somit für alle sichtbare) oder private Selektion handelt. Weiterhin besitzt eine Selektion n `reportproperties`, die den Kriterien des Reports entsprechen. Ein `reporttype` definiert, ob es sich um ein Kontakte-Report oder ein Unternehmens-Report handelt.
  Beispiel:

```
id = 5
name = "Alle VIP-Kunden"
description = "Alle VIP-Kunden die zu einem Meeting eingeladen werden sollen"
privateAccess = true
reporttype_id = 2 /* Referenzierung auf Reporttyp. Hier: Contacts */
employee_id = 14 /* Referenzierung auf den aktuellen Benutzer (Ersteller) */
reportlayout_id = 3 /* Referenzierung auf ein Layout für die Ausleitung */
```

Listing 6.2: Beispieldaten der Report-Tabelle

- **reportproperties:** Jeder Eintrag dieser Tabelle ist einem Eintrag in der `report`-Tabelle zugewiesen. Durch `reportproperties` werden die Kriterien einer Selektion bestimmt. Das Feld `relation` definiert die Beziehung zur nächsten Relation. Da zwischen Kriterien immer Konjunktionen (UND-Beziehungen) bestehen, ist dieses Feld vorerst zwecklos.

Beispiel:

```
id = 12
attribute = "Kundentyp"
operator = "is" /* Alternativ: is not, is one of, none */
property = "VIP"
relation = "AND" /* Beziehung zum nächsten Kriterium */
report_id = 5 /* Referenzierung auf ein Report */
```

Listing 6.3: Beispieldaten der ReportProperties-Tabelle

- **`reportlayout`:** Diese Tabelle dient zum Speichern von Layouts. Durch ein Layout wird definiert, welche Eigenschaften einer Kontaktperson/eines Unternehmens in ein Office-Format exportiert werden sollen.

  Beispiel:

```
id = 10
name = "Kontakt Adressdaten"
reporttype_id = 2 /* Referenzierung auf Reporttyp. Hier: Contacts */
```

Listing 6.4: Beispieldaten der ReportLayout-Tabelle

- **`layoutattribute`:** In dieser Tabelle werden Eigenschaften eingetragen, welche für den Export der Daten notwendig sind. Da ein `layoutattribute`, wie z. B. das Attribute *Owner* (der Verantwortliche eines Kontaktes/Unternehmens), in Selektionen mit den `reporttype` contacts und accounts verwendet wird, besteht eine m:n-Beziehung zwischen den Tabellen `layoutattribute` und `reporttype`. Durch das Anlegen von generischen Feldern wird diese Tabelle ebenfalls erweitert, damit diese Felder ebenfalls exportiert werden können.

  Beispiel:

```
id = 4
name = "Email-Adresse"
reporttype_id = 2 /* Referenzierung auf Reporttyp. Hier: Contacts */
custom_field = null /* Bei generischen Feldern: accounts oder contacts */
tenant_id = null /* Nötig um die Mandantendaten bei generischen Feldern zu
    berücksichtigen */
```

Listing 6.5: Beispieldaten der LayoutAttribute-Tabelle

- **`tenant:`** Diese Tabelle repräsentiert die Mandanten des CRM-Systems. Genauere Beschreibungen des Businessprofils werden in der Arbeit von Martin Sadowski erläutert (Sadowski, 2008, S.42ff).
  Beispiel:

```
id = 1
name = "Großunternehmen Nr.1"
description = "Dies ist der bisherige einzige Mandat in diesem System"
```

Listing 6.6: Beispieldaten der Tenant-Tabelle

- **`employee:`** Ein `employee` repräsentiert einen Mitarbeiter eines Unternehmens. Da eine Selektion stets einem Benutzer (und darüber einem Mandanten) zugeordnet wird, steht diese Tabelle in relation zu der `report`- und `reportlayout`-Tabelle (vgl. hierzu ausführlich: Sadowski, 2008, S.42ff).
  Beispiel:

```
id = 1
firstname = "Max"
lastname = "Mustermann"
email = "max@mustermann.de"
/* Diese Tabelle besitzt weitere, für uns nicht relevante Daten */
```

Listing 6.7: Beispieldaten der Employee-Tabelle

Als nächstes werden die in diesem Kapitel genannten Tabellen durch das OR-Mapping in POJOs („Plain Old Java Objects“) überführt (siehe Abbildung 6.2). Wie in Kapitel 2.2.3.3 beschrieben, soll jegliches persistieren und abrufen von Daten nur noch über JPA und den EntityManager erfolgen.

6.1.2 Business Logik-Ebene

Wie in Kapitel 2.2.1.2 beschrieben, wird die gesamte Business Logik im EJB-Container implementiert. Diese Klassen befinden sich in Abbildung 6.3 unterhalb der *ServiceLocator*-Klasse, welche für die Verbindung der Darstellungsschicht und der Business Logik zuständig ist. Bei diesen Klassen handelt es sich um zustandslose Session-Beans.

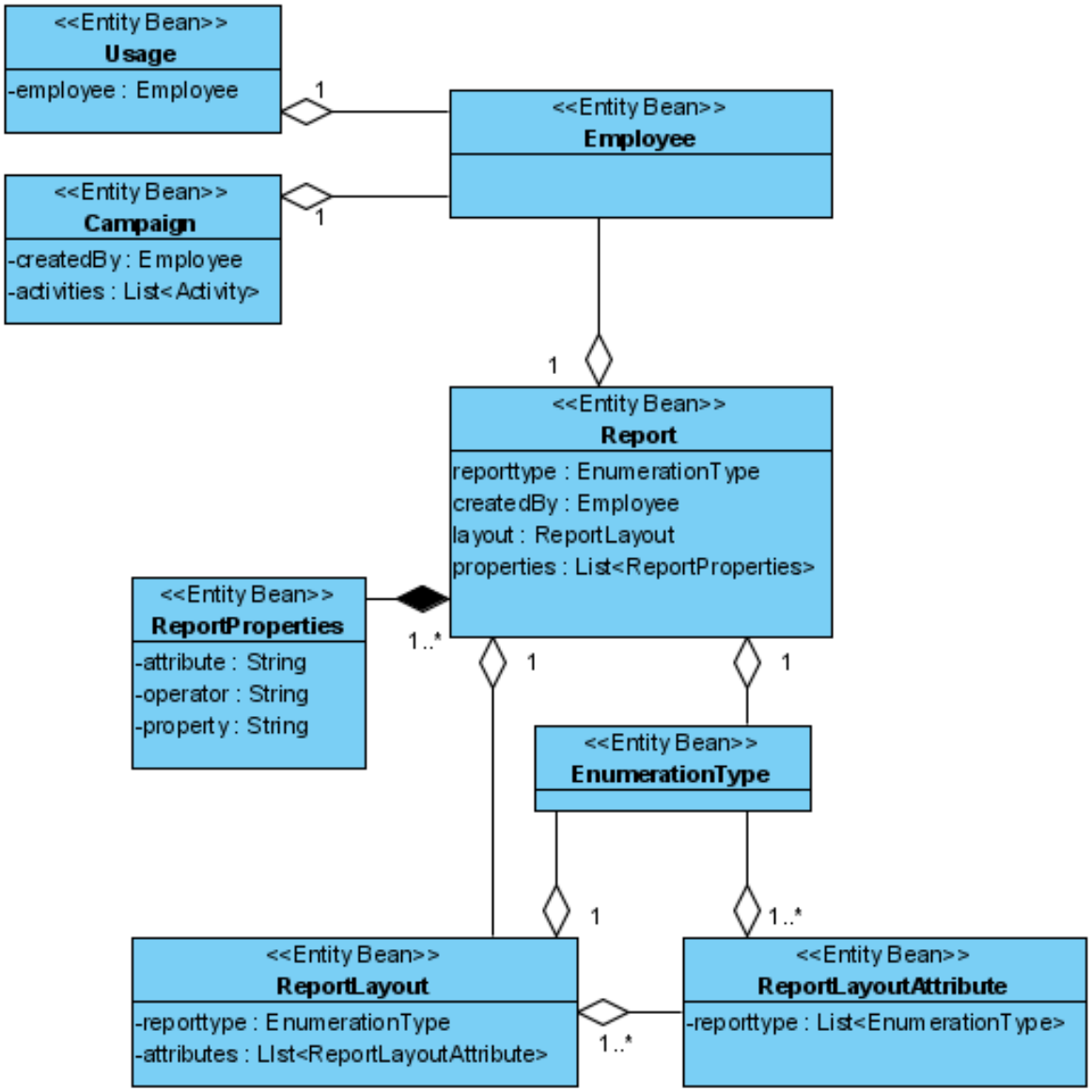

Abbildung 6.2: OR-Mapping der Datenbanktabellen

Beans dieser Art können keine Informationen speichern, was zur Folge hat, dass Informationen immer über Parameter übergeben werden müssen.
Hinter dem *ServiceLocator* befindet sich die *ReportFacade*. Über diese Fassade können alle Services in Bezug zu Reports aufgerufen werden. Bis auf die Selektionskomponente wird die Logik jeder Komponente in einer Session Bean implementiert. Da Selektionen verschiedener Typen identische Methoden besitzen können, werden diese Methoden in einer abstrakten Oberklasse *ReportService* herausfaktorisiert.

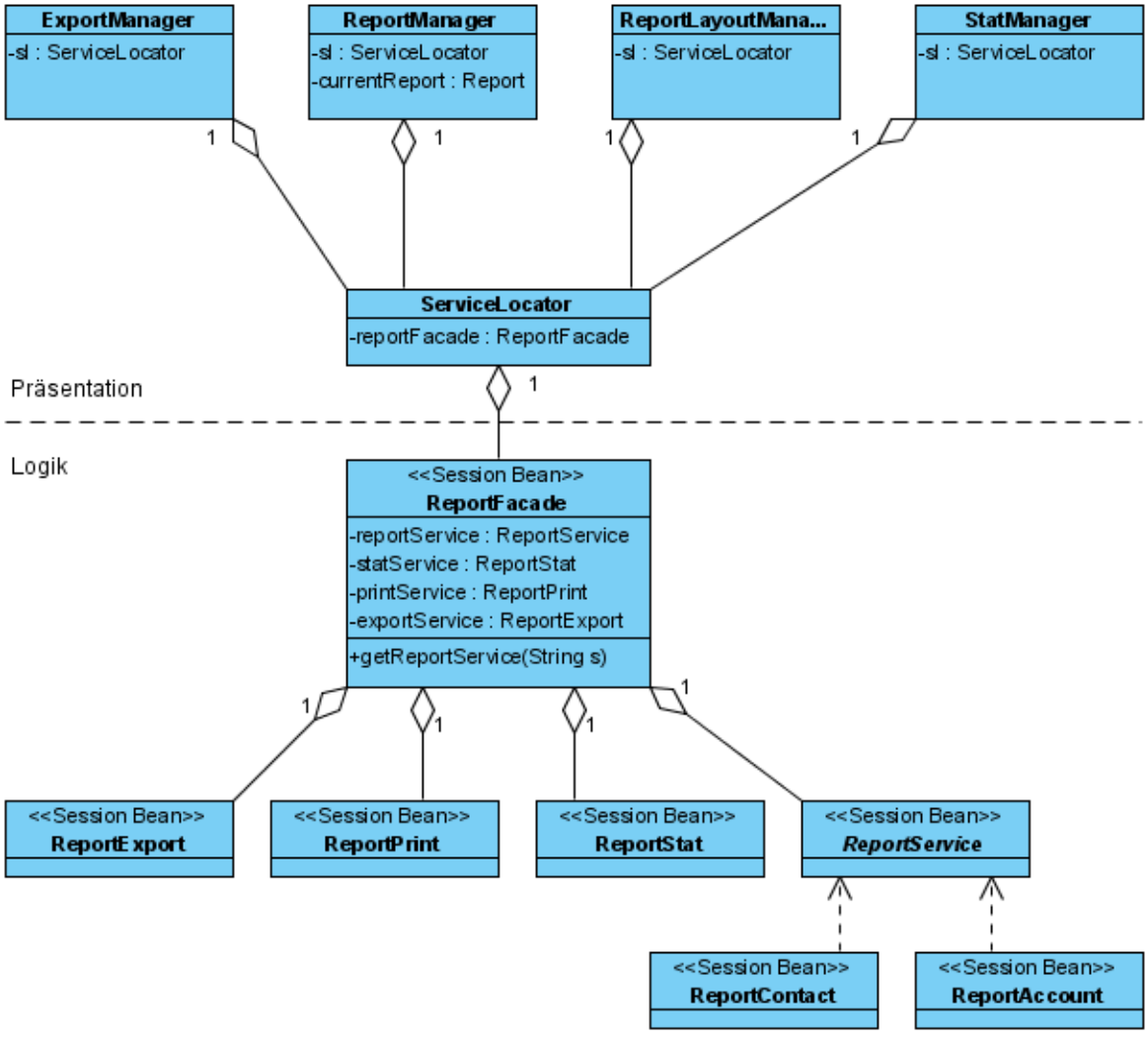

Abbildung 6.3: Klassendiagramm des Report Moduls

6.1.3 Darstellungsebene

Die obere Hälfte der Abbildung 6.3 repräsentiert die Präsentationsschicht. Über das Service Locator Pattern (siehe hierzu ausführlich: Sun, 2002) wird die Verbindung zur Business Logik hergestellt.
Die Darstellungsebene wird mit Hilfe von JavaServer Faces realisiert, wodurch eine schnelle Erstellung von grafischen Oberflächen ermöglicht wird. Die in der Abbildung dargestellten Klassen dienen dabei als sogenannte Backing Beans für das User Interface, welche durch XHTML-Seiten realisiert werden. Eine XHTML-Seite sendet bei jeder Anfrage ihre Daten an die Backing Bean, welche dann mit einer entsprechenden Reaktion antwortet (siehe Kapitel 2.2.3.1).

6.2 Technisch

Aus technischer Sicht werden bewusst keine Änderungen vorgenommen. Java Enterprise Edition spezifiziert eine Architektur welche Datenhaltung, Logik und Präsentation strikt voneinander trennt. Momentan wird eine 3-Schichten Architektur verwendet, welche sich jedoch problemlos auf 4 Schichten erweitern lässt, da das JNDI-Protokoll (siehe Kapitel 2.2.3.2) zwischen Web Container und EJB Container für verteilte Systeme verwendet werden kann.

Als Datenbankserver kommt weiterhin MySQL in der Version 5.0 zum Einsatz. JBoss (siehe Kapitel 2.2.2) wird als Applikationsserver weiterverwendet und als Client-Browser soll die Anwendung zumindest für den Internet Explorer 7 optimiert sein.

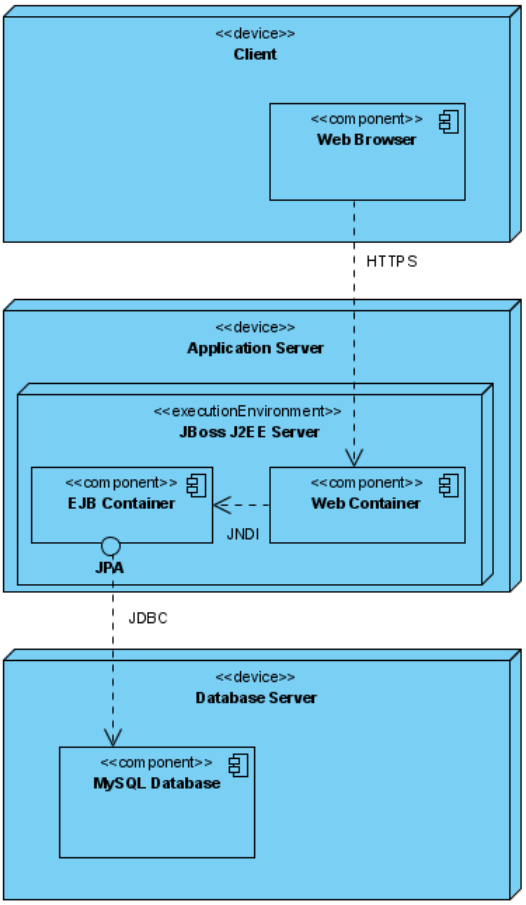

Abbildung 6.4: Verteilungsdiagramm des Systems

7 Realisierung

In diesem Kapitel werden die Ergebnisse der Entwicklungsphase präsentiert. Dazu werden die zuvor entworfenen Anforderungen reflektiert und mit den Ergebnissen verglichen. Anschließend wird im zweiten Teil der prototypische Entwicklungsprozess während des Projektes näher beschrieben.

7.1 Systembeschreibung

7.1.1 Selektionsmodul

Das Selektionsmodul ist der komplexeste und umfangreichste Teil des Reporting-Moduls. Selektionen sollen Ad-hoc-Abfragen ermöglichen. Das heißt, dass Benutzer nach eigenen Wünschen Kriterien festlegen und miteinander kombinieren können.
Da durch Selektionen die Daten des operativen Systems aus der Datenbank exportiert werden, ist dieser Bereich nur bestimmten Benutzern zugänglich und somit durch das Rechtesystem geschützt. In der Abbildung 7.1 wird die Durchführung einer Selektion anhand eines Sequenzdiagrammes dargestellt, welches im folgenden Abschnitt näher erläutert wird.

7.1.1.1 Lösung

Wie man der Abbildung 7.1 entnehmen kann, gibt es mindestens 30 Schritte beim Ausführen einer Selektion. Der Benutzer muss dabei fünf Schritte bewältigen:

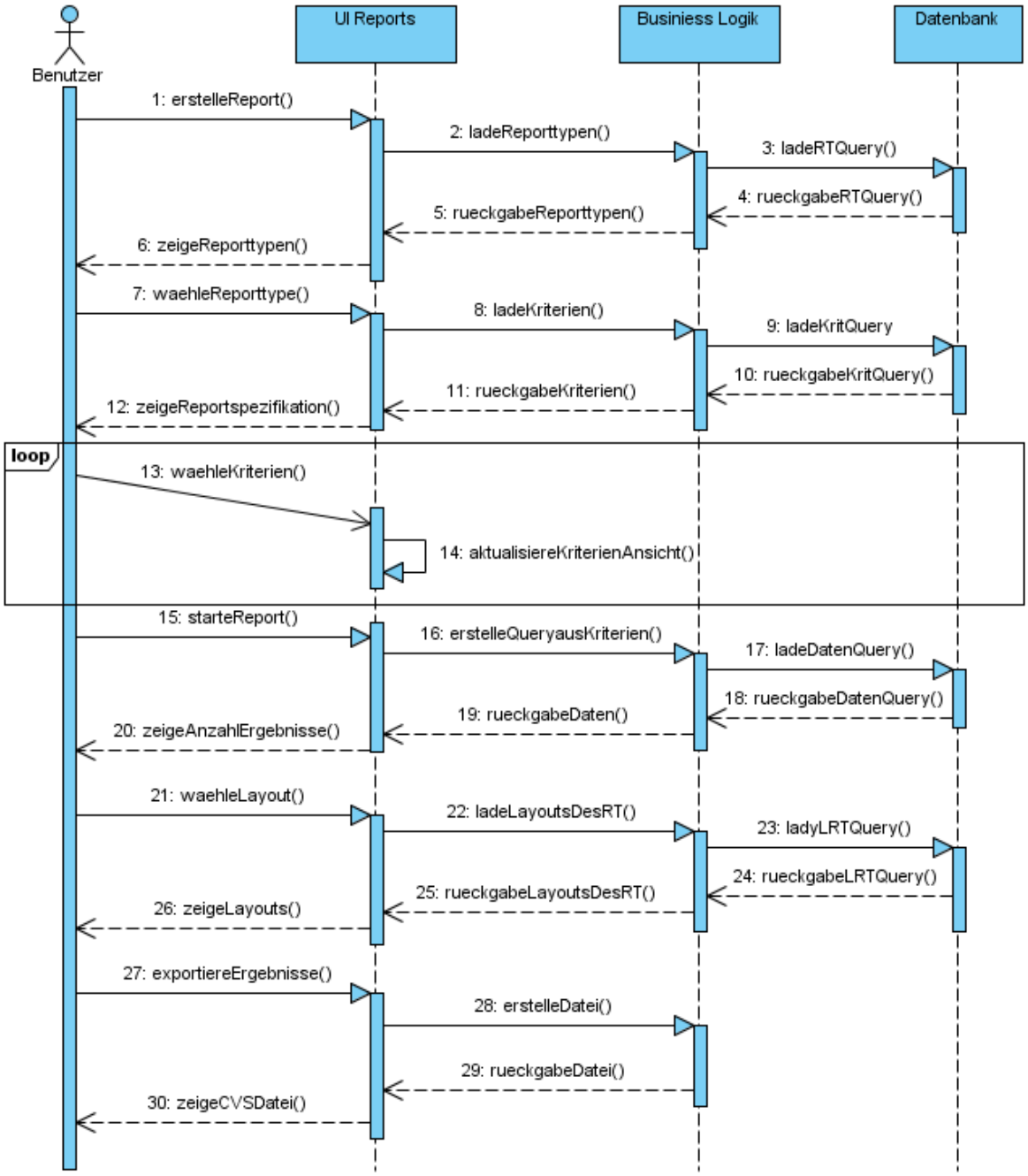

Abbildung 7.1: Interaktionen zwischen Benutzer und System beim Erstellen von Selektionen

- *erstelleReport():* In diesem Schritt muss der Benutzer den Report-Bereich betreten und eine neue Selektion erstellen. Alternativ kann der Benutzer auch vorhandene Reports öffnen.
- *waehleReporttype():* Als nächstes muss der Benutzer auswählen, was er letztendlich exportieren möchte. Es stehen zum Beispiel Reports für Accounts oder für Contacts zur Verfügung.
- *waehleKriterien():* Hier werden die möglichen Attribute für den Reporttypen angezeigt. Der Benutzer hat die Möglichkeit, Kriterien zu erstellen und zu verketten. Die Form der Darstellung ähnelt dabei stark der booleschen Algebra, damit die Benutzer schnell damit zurechtkommen.
- *waehleLayout():* Sobald die Selektion durchgeführt wurde, kann der Benutzer auswählen, welche Eigenschaften exportiert werden sollen. Alternativ kann er sich auch für ein zuvor erstelltes Layout-Template entscheiden. Diese Templates besitzen bereits fest definierte Eigenschaften.
- *exportiereErgebnisse():* Als Letztes muss der Benutzer den Export anstoßen. Der Benutzer erhält eine CSV-Datei, welche sich durch den anerkannten Standard (siehe dazu ausführlich: NetworkWorkingGroup, 2005) in nahezu alle Büroprogramme importieren lässt. Diese Datei beinhaltet alle selektierten Daten.

Im weiteren Verlauf werden einige der oben genannten Benutzerschritte grafisch dargestellt. Wie man in den Abbildungen 7.2, 7.3 und 7.4 erkennen kann, wird der Benutzer durch ein Menü mit vier Schritten geführt. Bei dem vierten Schritt handelt es sich um einen Bereich, den man nur bei Reports mit dem Type „Contacts“ betreten kann. Dieser Schritt erlaubt es, eine Kampagne für jede Kontaktperson anzulegen, damit die Auswertungen in Form von Aktivitäten protokolliert werden.

Im zweiten Schritt in Abbildung 7.3 kann man zusätzlich auswählen, ob die Selektion gespeichert werden soll. Mittels einer asynchronen Datenübertragung zwischen Server und Browser (AJAX) werden beim Speichern zusätzliche Optionen angezeigt, wie z. B. die Zugriffsberechtigung (privater oder öffentlicher Zugriff), der Name und die Beschreibung der Selektion.

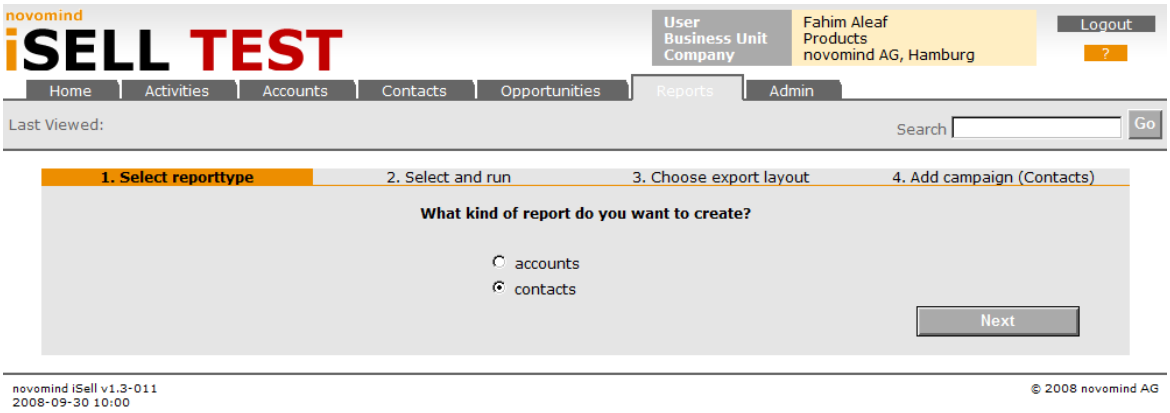

Abbildung 7.2: Erstellung einer Selektion. Schritt 1: Wählen des Reporttypes.

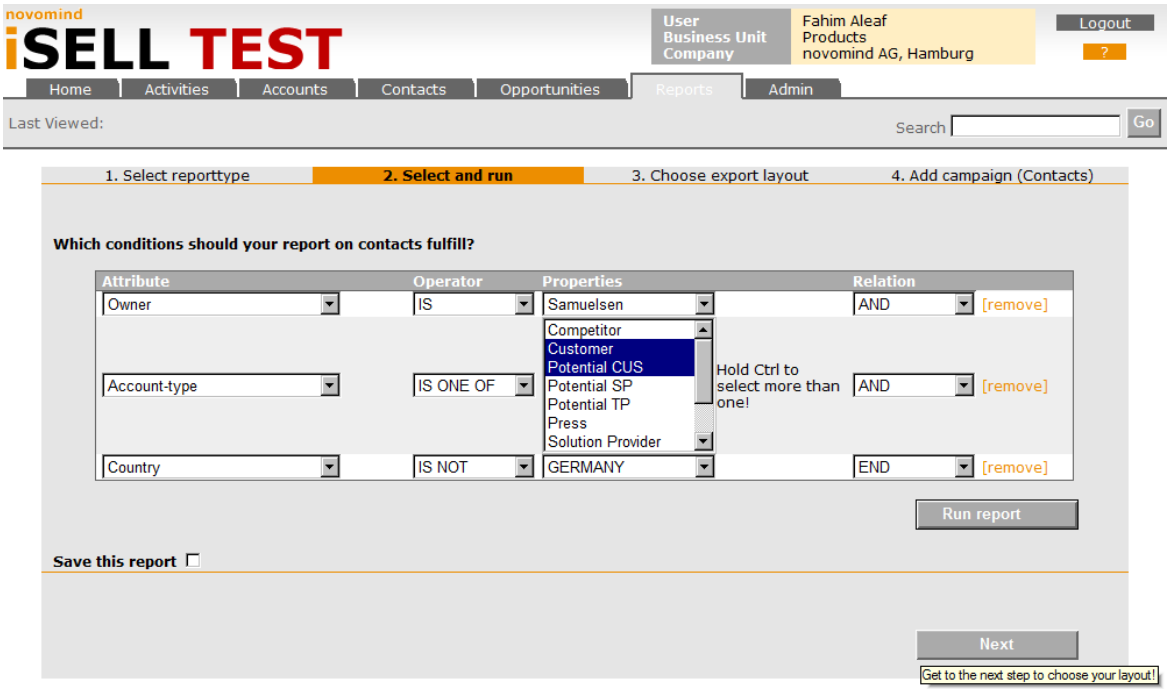

Abbildung 7.3: Erstellung einer Selektion. Schritt 2: Zusammensetzen der Kriterien.

Bei der Wahl des Layouts in Abbildung 7.4 kann der Benutzer Eigenschaften auswählen, die im Export angezeigt werden sollen. Da oft dieselben Exporte angestoßen werden, gibt es die Möglichkeit, sogenannte Layout Templates anzulegen und diese immer wieder aufzurufen. Diese Templates besitzen eine Menge von Eigenschaften, die in der Abbildung 7.4 angezeigt werden.

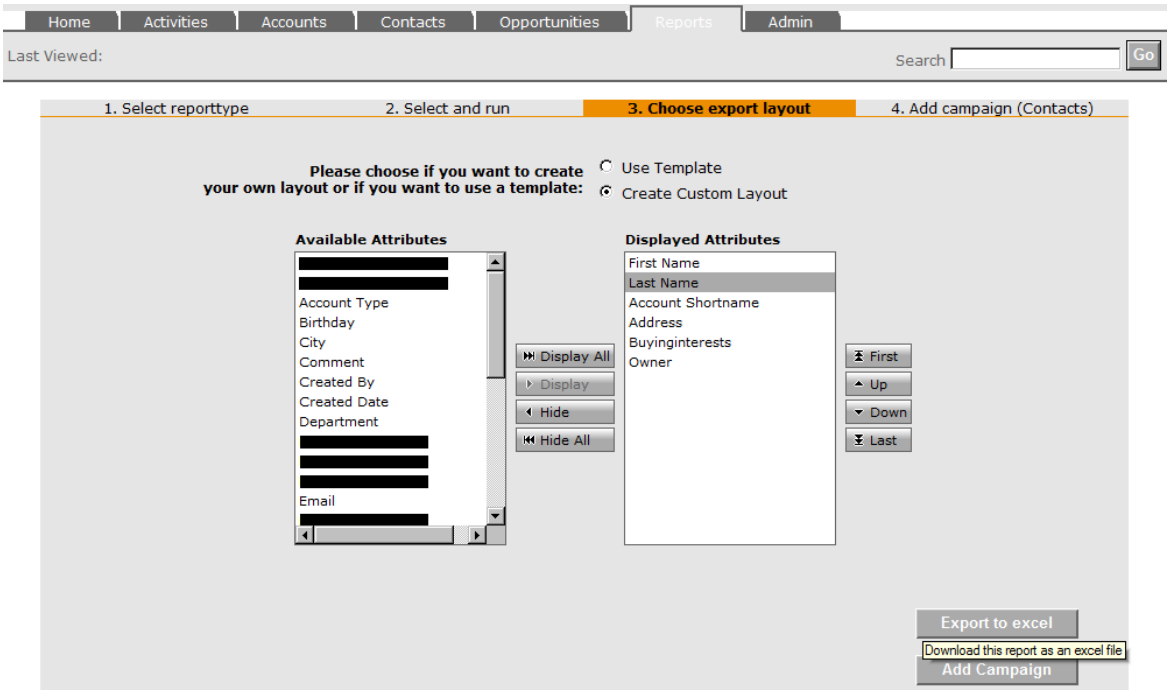

Abbildung 7.4: Erstellung einer Selektion. Schritt 3+4: Wahl des Layouts und Export.

Um die ausgewählten Kriterien in Abbildung 7.3 umzusetzen, muss für diese ein Äquivalent in der Datenbanksprache geschaffen werden, welches syntaktisch korrekt ist. In der bisherigen Form werden SQL-Anweisungen in Form von Strings (deutsch: Zeichenketten) in Java verarbeitet. Da diese Form der Verarbeitung eine hohe Fehleranfälligkeit hat, wird die von Hibernate angebotene Criteria API verwendet. Dabei handelt es sich um eine objektorientierte Möglichkeit, um JPQL-Anweisungen zu generieren. Vor allem bei Ad-hoc-Abfragen ist dieses Framework von Vorteil, da keine direkten Manipulationen von Strings für die Datenbankabfragen notwendig sind (vgl. Bauer und King, 2005, S.142). Listing 7.1 soll dies verdeutlichen.

```
1  // Initialisierung des Criteria-Objektes:
2  org.hibernate.Session session_;
3  org.hibernate.Criteria crit = session_.createCriteria(Contact.class, "contactAlias");
4
5  // Beliebige Anzahl an Kriterien definieren:
6  crit.add(Restrictions.eq("lastname", "Müller"));
7
8  // Durchführen der Anweisung. Automatische Überführung in JPQL -> SQL
9  List<Contact> result = crit.list();
10
11 // Die generierte Anweisung lautet: SELECT c FROM Contact c WHERE c.lastname="Müller";
```

Listing 7.1: Erstellen einer SQL-Query mittels der Criteria API

7.1.2 Statistikmodul

Um eine Metrisierung des Unternehmens zu ermöglichen, werden Statistiken erhoben und analysiert (siehe hierzu: Kapitel 2.1.3.3). Diese Funktion soll nur einem kleinen Benutzerkreis zugänglich sein, und wird deshalb durch das Rechtesystem geschützt.
Wie man dem Aktivitätsdiagramm in Abbildung 7.5 entnehmen kann, gibt es zwei verschiedene Statistiken. Jede Statistik benötigt dabei als Parameter ein Start- und ein Enddatum.

7.1.2.1 Lösung

Da dieses Modul nur wenigen Nutzern (meist Führungskräften) zur Verfügung steht, befindet es sich im Admin-Bereich und wird über ein Recht vor unberechtigtem Zugriff geschützt. Nach der Eingabe von einem Start- und Enddatum kann man jeweils Benutzer- oder Aktivitätsstatistiken einsehen (siehe Abbildung 7.6). Um eine Benutzerstatistik zu erstellen, ist eine Erweiterung des Datenmodells notwendig. Die Tabelle `usage` protokolliert das Nutzverhalten der Benutzer des CRM-Systems. Bei dem Nutzverhalten wird erhoben, welcher Nutzer zu welcher Zeit mit welchem Modul (Account, Contact, Activity oder Opportunity) gearbeitet hat.

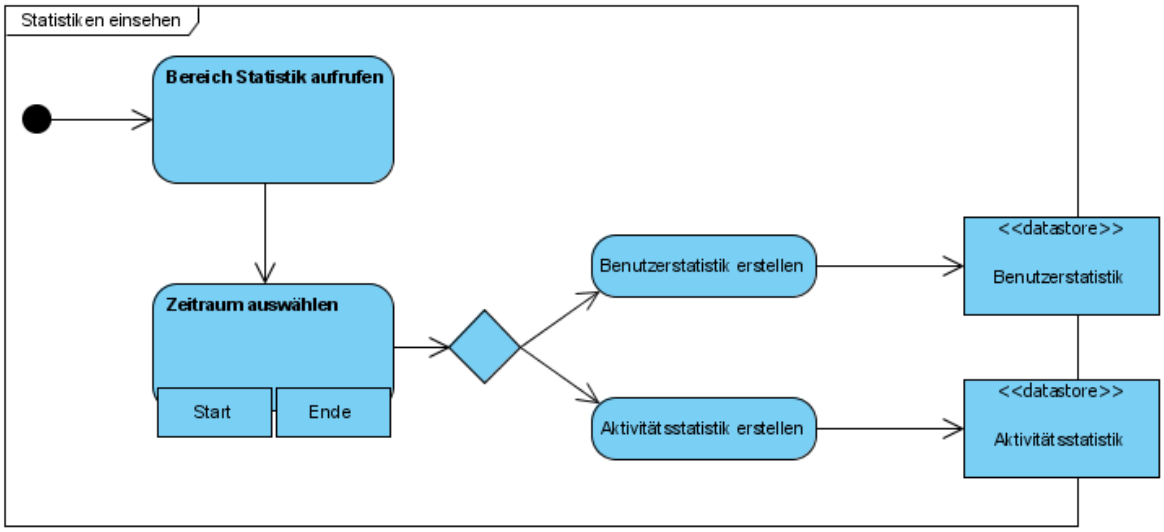

Abbildung 7.5: Aktivitäten bei der Durchführung von Statistiken

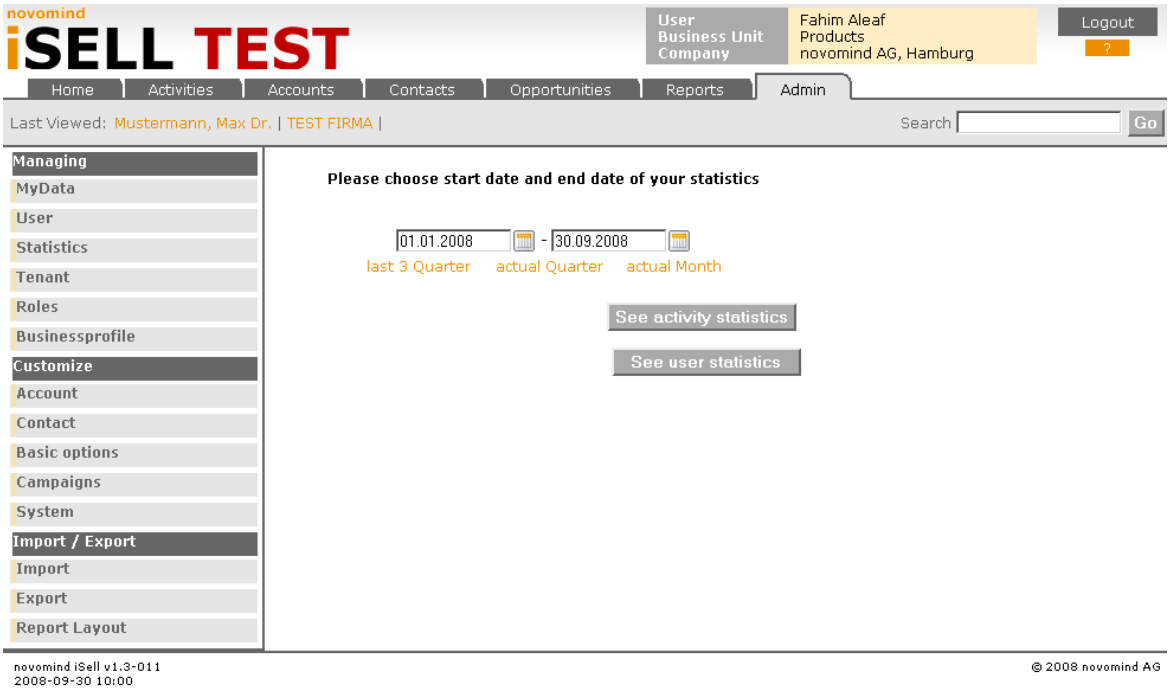

Abbildung 7.6: Admin-Bereich: Einsehen von Statistiken

Zusätzlich wird dargestellt, wieviel Umsatz jeder Benutzer in dem Zeitraum erwirtschaftet hat, wie in Abbildung 7.7 zu sehen ist.

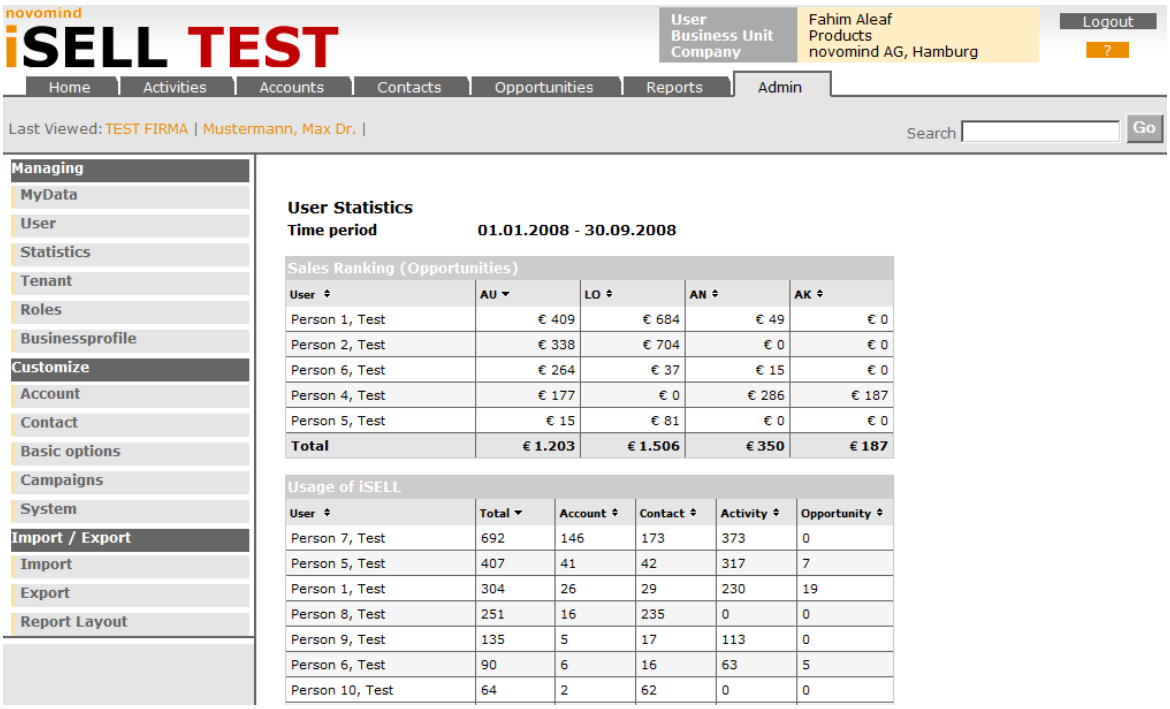

Abbildung 7.7: Benutzerstatistiken im Statistikmodul

Die Aktivitätsstatistik bezieht sich auf die Aktivitäten des CRM-Systems. Dort wird angezeigt, wie viele offene Aktivitäten in einem Zeitraum vorhanden sind. Dabei werden die Aktivitäten pro Industrie und Kaufinteresse dargestellt.

7.1.3 Exportmodul

Durch das Exportmodul erhält der Benutzer eine automatisierte Möglichkeit, MS-Office-Dateien mit Daten eines Contacts zu erstellen. Diese „Assistenzfunktion“ wird benutzt, um Briefe oder Versandaufkleber zu erstellen und auszudrucken. Nur ein ausgewählter Benutzerkreis soll diese Funktion nutzen können, so dass die Integration dieses Moduls in das Rechtesystem vor unberechtigtem Zugriff schützen soll.

Da innerhalb der novomind AG Microsoft Office bzw. Microsoft Word als Textverarbeitungsprogramm genutzt wird, ist es notwendig, die Daten eines Contacts im CRM-System in ein MS-Office kompatibles Format zu überführen. Bei dem zurzeit genutzten Dateiformat Microsoft Document (*.doc) handelt es sich um ein proprietäres Format, welches eine Bearbeitung ohne Microsoft Anwendungen nicht ermöglicht. Demnach ist die automatisierte Überführung von Kontaktdaten in Dateien dieser Formate nicht möglich.

Seit 2007 stellt Microsoft ein neues Format namens Office Open XML zur Verfügung, welches im August 2008 trotz Einwände einiger Gremien als ISO-Standard veröffentlicht wurde (vgl. ISO, 2008). Durch die Installation eines von Microsoft bereitgestellten Updates ist es möglich, dieses neue Format mit älteren Microsoft Office Versionen zu nutzen. Da es sich bei Office Open XML Dateien (mit der Endung *.docx) um ein offenes XML-basiertes Format handelt, ergibt sich die Möglichkeit zur Manipulation von Word Dokumenten. Zudem besteht durch diesen offenen Standard eine gewisse Interopabilität, die es zulässt, dass die Dateien mit verschiedenen Systemen (in unserem Fall das CRM-System und Microsoft Word 2003) zusammenarbeiten können (vgl. Murray, 2006, S.35).

7.1.3.1 Lösung

Über die Detailansicht eines Kontaktes hat der Benutzer die Möglichkeit, die Daten des Kontaktes in Office-Vorlagen zu exportieren. Dazu muss er über den Link „Microsoft Office Word“ das passende Template auswählen. Dieses Template wird dann mit den Daten gefüllt und an den Benutzer zurückgesendet, wie in Abbildung 7.8 zu sehen ist.

Da sich Office-Vorlagen mit der Zeit verändern können, sollte zumindest der Administrator des CRM-Systems die Möglichkeit haben, diese Vorlagen zu verwalten. Um dies zu ermöglichen, wurden die notwendigen Dateinamen, die auf die Office-Templates referenzieren, vollständig in eine einzige Schlüsseltabelle (siehe hierzu Kapitel 8.4.4) überführt. Somit hat der Administrator die Möglichkeit, diese Dateinamen im Admin-Bereich über den Menüpunkt Basic Options zu pflegen, wie in Abbildung 7.9 zu sehen ist. Da hier nur die Referenzierungen auf die Office-Vorlagen gepflegt werden, muss die Datei selbst auf dem Server zur Verfügung stehen.

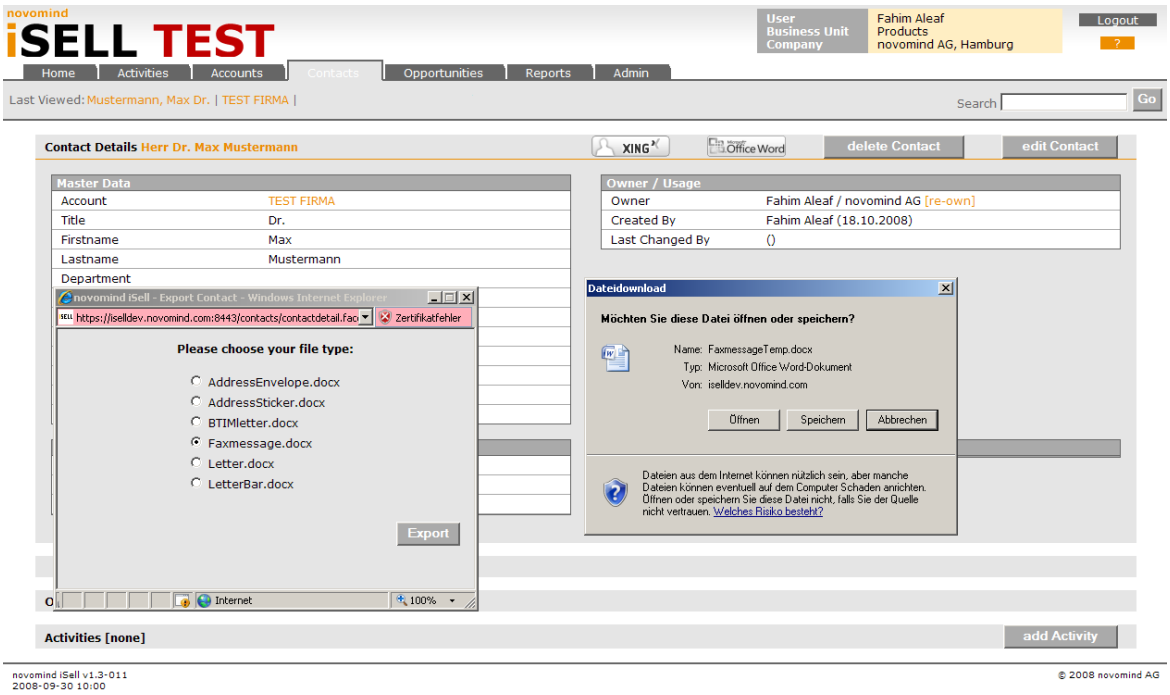

Abbildung 7.8: Exportieren von Kontaktdaten in Office-Vorlagen

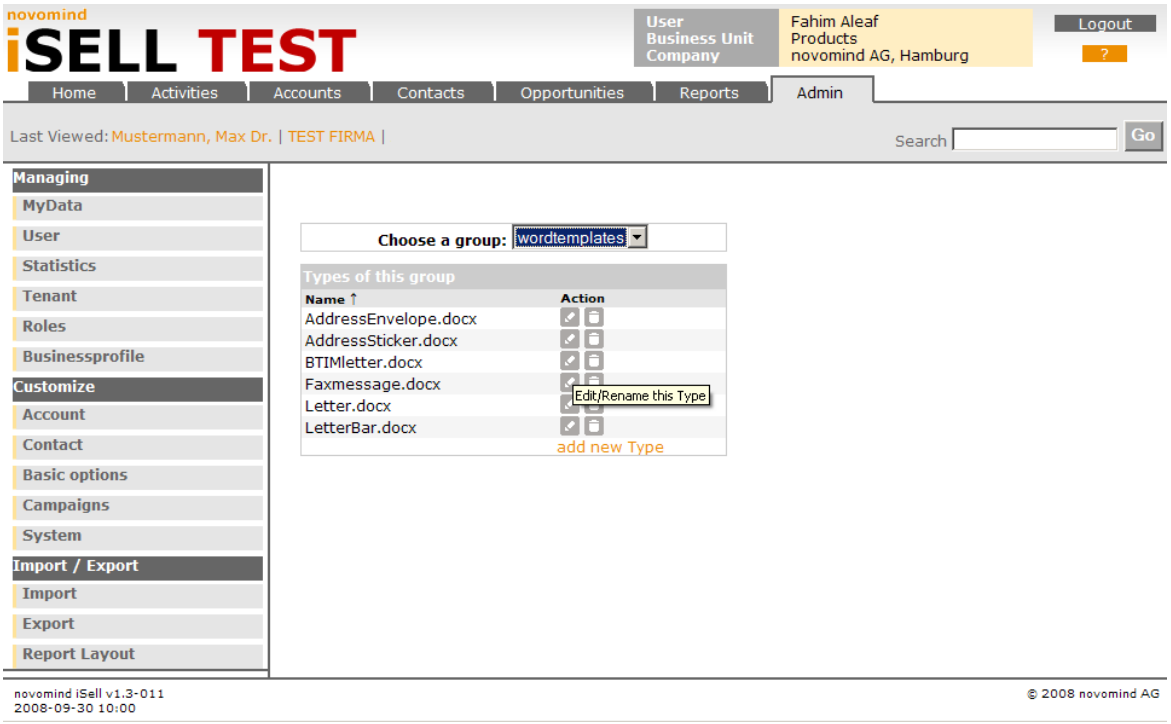

Abbildung 7.9: Hinzufügen, Entfernen und Editieren von Office-Vorlagen

Damit die Benutzer die Office-Vorlagen nach eigenen Bedürfnissen erstellen können, wurden Platzhalter eingeführt, die durch bestimmte Daten des CRM-Systems ersetzt werden können. Durch diese Vorgehensweise ist in der Zukunft kein Eingriff durch Entwickler notwendig. Eine Liste der aktuellen Platzhalter ist in der Tabelle 7.1 zu sehen.

Zum Einsatz kommt hier die Simple API for XML (SAX), welche bereits in der Java Standard Edition vorhanden ist. Diese API besitzt einen Parser um XML Dateien zu bearbeiten (vgl. McLaughlin, 2002, S.43).

[COMPANY]	–>	Firmenname
[COUNTRY]	–>	Land
[CREATOR_EMAIL]	–>	Email-Adresse des aktuellen Benutzers
[CREATOR_NAME]	–>	Vollständiger Name des aktuellen Benutzers
[FULL_NAME]	–>	Vollständiger Name des Kontaktes
[COUNTRY]	–>	Land
[SALUT_ENVELOPE]	–>	Anrede auf dem Briefumschlag. Zum Beispiel: Herrn
[SALUT_NAME_ENVELOPE]	–>	Anrede und Name. Zum Beispiel: Herrn Dr. Max Mustermann
[SALUTATION_LETTER]	–>	Briefanrede und Name. Z.B.: Sehr geehrter Herr Mustermann
[STREET_NO]	–>	Straße und Hausnummer
[ZIP_CITY]	–>	Postleitzahl und Stadt

Tabelle 7.1: Platzhalter für Microsoft Office Vorlagen

7.2 Prototypischer Entwicklungsprozess

7.2.1 Ziele

> *„Maximum Feedback for Minimum Effort.“* (Snyder, 2003, S.12)

Prototyping ist in der Softwareentwicklung ein iterativer Prozess, der die Kommunikation zwischen Entwickler und Benutzer unterstützt. Demnach ist das Ziel leicht zu beschreiben: Man versucht durch möglichst wenig Aufwand möglichst viel Resonanz von Benutzern oder Auftraggebern zu erhalten. Nur durch viel Informationsaustausch kann gewährleistet werden, dass alle mit dem Ergebnis zufrieden sein werden.
Folgende Vorteile bietet das Prototyping nach C.Snyder (vgl. Snyder, 2003, S.12):

- Versorgung der Entwickler mit Feedback der Benutzer, meist schon bevor viel Aufwand in die Entwicklung investiert wurde.
- Unterstützung einer schnellen und iterativen Entwicklung. Durch ständiges Feedback kann man experimentierfreudige Ergebnisse liefern.
- Innerhalb des Entwicklungsteams und vor allem mit dem Kunden wird mehr kommuniziert.

- Technische Erfahrungen werden nicht benötigt. So können erfahrene und unerfahrene Benutzer/Entwickler besser zusammenarbeiten.
- Förderung der Kreativität im Entwicklungsprozess.

Im Rahmen dieser Arbeit ist die Entwicklung des Reportmoduls in einem iterativen und prototypischen Prozess entstanden. Der Prozess kann wie folgt beschrieben werden:

- Wöchentlich gab es eine Veröffentlichung des aktuellen Standes auf einem Testsystem.
- Wöchentlich gab es ein Meeting zwischen Entwickler und Projektleiter, welcher zugleich ein zukünftiger Benutzer ist, in dem über den aktuellen Stand diskutiert wurde.
- Mindestens einmal im Monat gab es einen Meilenstein, bei dem der Projektleiter die zukünftigen Benutzer aufgefordert hat, den bisherigen Stand zu testen und Feedback zu geben. Dabei ging es hauptsächlich um das Testen der Gebrauchstauglichkeit des Systems.

7.2.2 Spezifikation

Die Prozesse der prototypischen Entwicklung können in drei Arten unterteilt werden:

Exploratives Prototyping untersucht die Anforderungen der Benutzer. Dabei werden die Ziele durch direkte Kommunikation mit den Benutzern protokolliert. Anhand von grafischen Benutzerschnittstellen entstehen Einblicke in das zu entwickelnde System, damit sich die Benutzer an der Erstellung und Gestaltung des Systems beteiligen können.

Experimentelles Prototyping wird während der Entwurfsphase angewendet. Es werden dabei Möglichkeiten zur Realisierung erforscht und Probleme erkannt.

Evolutionäres Prototyping beschreibt die anschließende Weiterentwicklung des Systems. Schrittweise wird gemäß Feedback des Benutzers das System weiterentwickelt.

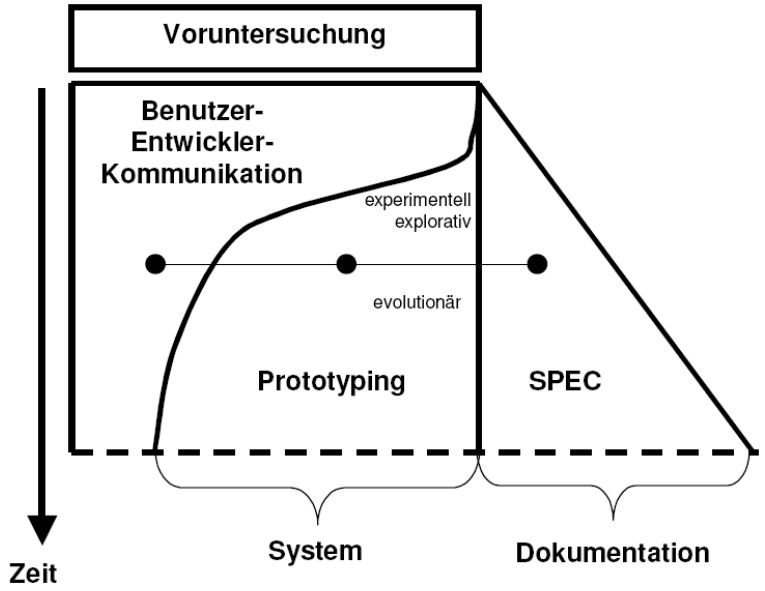

Abbildung 7.10: Vorgehensweise im Projekt. (Quelle: Lilienthal u. a., 2007, S. 6)

Die Kommunikation mit den Benutzern ist in der Anfangsphase des Projektes am größten, wie in Abbildung 7.10 zu sehen ist.
Während der Anforderungsanalyse (siehe Kapitel 3.2.1) wurde mit den Benutzern gemeinsam ein Papier-Prototyp entworfen, um eine grafische Benutzerschnittstelle des Systems zu entwickeln und zu validieren. Dabei wurden alle Probleme der Benutzer erfasst, die das alte Vertriebssystem betreffen. Ebenso wurden Verbesserungsvorschläge protokolliert.
Anschließend wurden im experimentellen Prototyping einige Möglichkeiten zur Realisierung dieses Papier-Prototypen untersucht und verifiziert. Dadurch sind einige Prototypen entstanden, die zusammen mit dem Projektleiter diskutiert wurden. Durch dieses experimentelle Vorgehen ist anschließend eine Benutzerführung mit vier Schritten entstanden (siehe Abbildung 7.2 - 7.4), damit ein Benutzer zu jeder Zeit weiß, in welchem Teil des Reports er sich befindet.
Danach wurde das evolutionäre Prototyping verwendet, um Schrittweise alle zusätzlichen Anforderungen, die während der Entwicklung entstehen, zu erfüllen. Zusätzliche Wünsche (wie z. B. das Anhängen einer Kampagne nach dem Report) wurden mit den Benutzern besprochen und in den aktuellen Prototypen integriert.

7.2.3 Machbarkeitsbeweis

Spätestens zum Ende der Entwicklungsphase haben sich folgende Fragen gestellt:

- Entwickeln wir das richtige Produkt?
- Entwickeln wir das Produkt richtig?

Um die Machbarkeit des entwickelten Prototypen nachzuweisen, muss man dieses Projekt validieren und verifizieren. Bei der Validation wird überprüft, ob das Ergebnis alle in Kapitel 3.2.1 protokollierten Anforderungen erfüllt. Die Verifikation dagegen untersucht die Korrektheit des Systems. Um beide Nachweise zu liefern, bedarf es einer dynamischen Analyse des Prototypen (vgl. Buth, 2008, S.100ff).

Da die Benutzer ständig Zugang zu einem Testsystem mit einem aktuellen Prototypen hatten, kann man die Validation als abgeschlossen sehen. Verbesserungsvorschläge sind während der Entwicklung direkt in die Entwicklungen eingeflossen, so dass schlussendlich keine Wünsche mehr offen waren. Durch eine Vergleichsanalyse des Prototypen mit dem Altsystem konnte schnell festgestellt werden, dass der entwickelte Prototyp zumindest alle Funktionen des Altsystems erfüllt. So wurden zum Beispiel alle vorhandenen Selektionen/Reports des Altsystems in den neuen Prototypen zu Testzwecken überführt.

Verifiziert wurde der Prototyp in einer Schulung, bei der allen zukünftigen Anwendern der Bereich Reporting vorgestellt wurde. Anschließend mussten alle Benutzer zwei Reports erstellen. Daraus ergab sich eine Diskussion, inwieweit dieses Modul seinen Zweck erfüllt. Dabei wurden Vorschläge geäußert, die zu einigen Verbesserungen in der grafischen Darstellung führten.

Allumfassend kann man sagen, dass das Reportmodul, vor allem im Vergleich zum Reporting im Altsystem, für positive Resonanz innerhalb der novomind AG führte. Positive Äußerungen gab es vor allem zu der Bedienbarkeit, so dass komplexe Reports mit wenig Aufwand durchgeführt werden konnten.

Dennoch bleibt die Frage offen, inwieweit das Reportmodul die Anforderungen anderer Unternehmen erfüllt. Da eine Auswilderung des CRM-Systems in den Open Source Markt geplant ist, sollte das Reportmodul auch Anforderungen anderer Nutzer Stand halten. Zum

einen kann man sagen, dass durch die Unterstützung von generischen Feldern nahezu alle Anforderungen im Bereich Selektionen erfüllt werden können. Doch vor allem bei betriebswirtschaftlich orientierten Unternehmen sind die Anforderungen an ein Reportmodul weitaus größer (siehe hierzu Kapitel 2.1.3.5). Eine Integration eines Data Warehouse-Systems sollte daher als nächstes angestrebt werden. Der jetzige Prototyp bietet dafür eine Basis, die sich problemlos erweitern lässt.

8 Testkonzept

Um zu gewährleisten, dass die entwickelte Software für einen Produktiveinsatz bereit ist, sind verschiedene Tests notwendig. Diese Tests dienen als Beweis für eine lauffähige Software. Dabei ist das Ziel dieser Tests, eine möglichst hohe Testabdeckung zu erreichen und somit die Qualität der Software zu messen. Dass wenige Fehler im Test nicht unbedingt auf gute Software zurückzuführen sind, soll Abbildung 8.1 zeigen. Demnach können auch mangelhafte Tests wenige Fehler verursachen, so dass der Eindruck entsteht, dass gute Software entwickelt wurde.

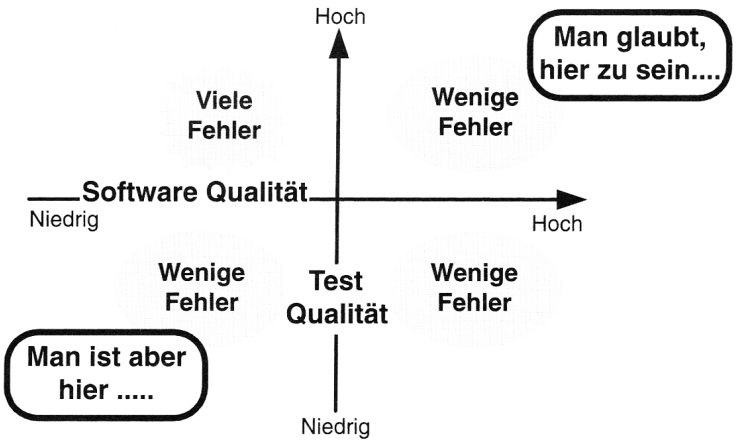

Abbildung 8.1: Test-Qualität und Software-Qualität. Quelle: (Siedersleben, 2003, S. 288)

8.1 Planung

Um alle Bereiche der Anwendung zu prüfen, wird dieser Test nach dem fundamentalen Testprozess des International Software Testing Qualification Board (ISTQB) durchgeführt (siehe Abbildung 8.2). In diesem Kapitel wird beschrieben, welche Bereiche der Anwendung getestet werden sollen. Dabei wird ein vorbeugender Ansatz verfolgt, bei dem der Tester sich bereits bei Projektbeginn mit der Testplanung und dem Testdesign auseinandersetzt (vgl. Spillner und Linz, 2005, S.182).

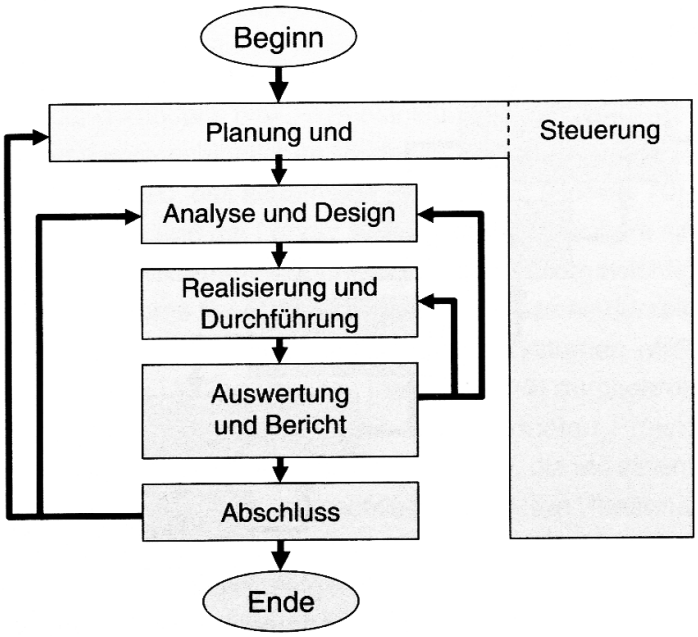

Abbildung 8.2: Fundamentaler Testprozess nach dem ISTQB. Quelle: (Spillner und Linz, 2005, S. 20)

Das CRM-System iSell besteht aus folgenden Modulen:

- Accountverwaltung
- Contactverwaltung
- Activityverwaltung
- Opportuntityverwaltung

- Reports
- Administration

Das Modul Reports soll in diesem Test intensiv geprüft werden. Dabei sollen keine Fehler entstehen, die zu einem Absturz der gesamten Anwendung führen. Da Selektionen am häufigsten mit zwei bis fünf Kriterien durchgeführt werden, soll sich der Test auf diese Anzahl spezialisieren. Selektionen dieser Art sollen in einer angemessenen Zeit zu Ergebnissen führen.
Bei allen Modulen reagiert die Anwendung träge. Dieses Performanceproblem soll gelöst werden, da unter diesen Umständen kein Produktiveinsatz möglich ist.

Für die Tests steht eine Testumgebung auf einer virtuellen Maschine bereit. Die Datenbasis der Testumgebung wird in regelmäßigen Abständen mit den Daten des Produktivsystems aktualisiert, damit ausschließlich mit aktuellen Daten getestet wird. Nach einem Abnahmetest wird der bisherige Softwarestand eingefroren und auf das Produktivsystem übertragen. Währenddessen bleibt dem Tester ein Zeitfenster von 30 Minuten um einige Tests im Produktivsystem erneut durchzuführen.

8.2 Analyse

Zu diesem Zeitpunkt ist die Datenübernahme vom alten Vertriebssystem zum neuen CRM-System abgeschlossen. Die Daten wurden nach dem Import von allen Benutzern überprüft, so dass einige Nachbesserungen notwendig waren. Da keine Testumgebung für das alte Vertriebssystem vorhanden ist, kann dieses System für Testzwecke nicht genutzt werden. Dies folgert, dass für das neue Reportmodul im neuen CRM-System keine Vergleichsdaten für Tests zur Verfügung stehen. So können zum Beispiel identische Selektionen im Alt- und Neusystem nicht miteinander verglichen werden.
Um das Modul dennoch auf Korrektheit zu prüfen, werden aus diesem Grund die Selektionen analysiert und mit gleichwertigen SQL-Anweisungen verglichen (siehe hierzu Tabelle 8.1 und 8.2). Ein Ausnahmenfall ist die *IS ONE OF*-Beziehung. Hier ist es möglich, mehrere Attribute auszuwählen. Auf Grund dessen wird dieser Bereich nochmal gesondert getestet (siehe

Logischer Testfall	1	2
Eingabeoperator x	x = IS	x = IS ONE OF
Erwartetes Ergebnis (SQL: WHERE-condition)	attr = prop	attr = prop1 OR attr = prop2

Tabelle 8.1: Logischer Testfall Selektion (Teil 1)

Logischer Testfall	3	4
Eingabeoperator x	x = IS NOT	x = NONE
Erwartetes Ergebnis (SQL: WHERE-condition)	attr!=prop	attr=' ' OR attr IS NULL

Tabelle 8.2: Logischer Testfall Selektion (Teil 2)

Tabelle 8.3). Da zusätzlich bei der Kriterienwahl diverse Kombinationen von *IS-, IS ONE OF-, IS NOT-, NONE*-Beziehungen auftreten können, wird in einem Whitebox-Test die minimale Mehrfachbedingungsüberdeckung getestet. Zudem wird der entwickelte Programmcode mit Hilfe des Plugins Checkstyle einer Review unterzogen.
In einem Blackbox-Test wird ein zustandsbezogener Test durchgeführt um zu gewährleisten, dass die Navigation zwischen den Seiten einwandfrei funktioniert.

Um die vorhandenen Performanceprobleme zu lösen, wird auf ein Testwerkzeug zur dynamischen Analyse zurückgegriffen, welches Probleme und ihre Ursachen aufdecken soll.

Logischer Testfall	1	2	3
Eingabeoperator x	x = 0	0 < x < max(Attribute)-1	x = max(Attribute)
Erwartetes Ergebnis	0	1 bis max(Kontakte)-1	max(Kontakte)

Tabelle 8.3: Logischer Testfall Selektion (IS ONE OF Beziehung)

Logischer Testfall	1	2
Eingabeoperator x	x = IS	x = IS ONE OF
Erwartetes Ergebnis (SQL: WHERE-condition)	buyinginterest=none	country=germany OR country=italy

Tabelle 8.4: Konkreter Testfall Selektion (Teil 1)

Logischer Testfall	3	4
Eingabeoperator x	x = IS NOT	x = NONE
Erwartetes Ergebnis (SQL: WHERE-condition)	accounttype!=customer	city=' ' OR city IS NULL

Tabelle 8.5: Konkreter Testfall Selektion (Teil 2)

8.3 Realisierung

In den Tabellen 8.4, 8.5 und 8.6 sind die konkreten Testfälle von Kapitel 8.2 abgebildet. Hierbei handelt es sich um angewandte Testfälle mit aktuellen Daten des CRM-Systems. Um die möglichen Kombinationen der Operatoren zu testen, wurden diese in der Tabelle 8.7 aufgeführt. Bei der minimalen Mehrfachbedingungsüberdeckung wurden die Ausdrücke getestet, bei denen die Änderung eines Wahrheitwertes eines atomaren Ausdrucks den Wahrheitswert des zusammengesetzten Ausdrucks ändert (vgl. Buth, 2008, S.113). Laut diesem Überdeckungstest wurden bei vier Operatoren/Ausdrücken mindestens fünf verschiedene Kombinationen getestet, um zu gewährleisten, dass alle Operatoren miteinander kombinierbar sind.

Bei dem Zustandsübergangstest wurden alle Zustände im Bereich Reporting erfasst. Anschließend wurden diese Zustände durch die Übergänge verbunden und alle möglichen Wege von einem Startpunkt bis zum Ziel durchlaufen.

Konkreter Testfall	1	2	3
Eingabeoperator x	0	10	87
Erwartetes Ergebnis	0	1326	15983

Tabelle 8.6: Konkreter Testfall Selektion (IS ONE OF Beziehung)

IS	IS ONE OF	IS NOT	NONE	IS ∧ IOF ∧ IN ∧ NO
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Tabelle 8.7: Minimale Mehrfachbedingungsüberdeckung der Operatoren bei Selektionen

In einem Systemtest wurde die Testumgebung mit dem Testwerkzeug Glassbox gestartet, um das Gesamtsystem zur Laufzeit anhand von Messwerten zu analysieren und zu überprüfen (vgl. Glassbox, 2008). Um brauchbare und repräsentative Messwerte aufzuzeichnen, wurde 30 Minuten an dem CRM-System gearbeitet. Die Ergebnisse konnten anschließend über den Webbrowser eingesehen werden (siehe hierzu Abbildung 8.3). In einem Abnahmetest wurden die meisten durchgeführten Tests in der Produktivumgebung wiederholt. Zu diesem Zeitpunkt sollte das System bereits fehlerfrei sein.

8.4 Auswertung

Da aus zeitlichen Gründen eine Automatisierung der Tests nicht möglich war, kann keine Angabe zur Häufigkeit der Testdurchführung gemacht werden. Es soll aber erwähnt werden, dass durch die Tests einige Fehler aufgedeckt wurden und durch die anschließenden Korrekturen die Qualität der Software erhöht wurde. Im Folgenden werden einige Fehler näher beschrieben.

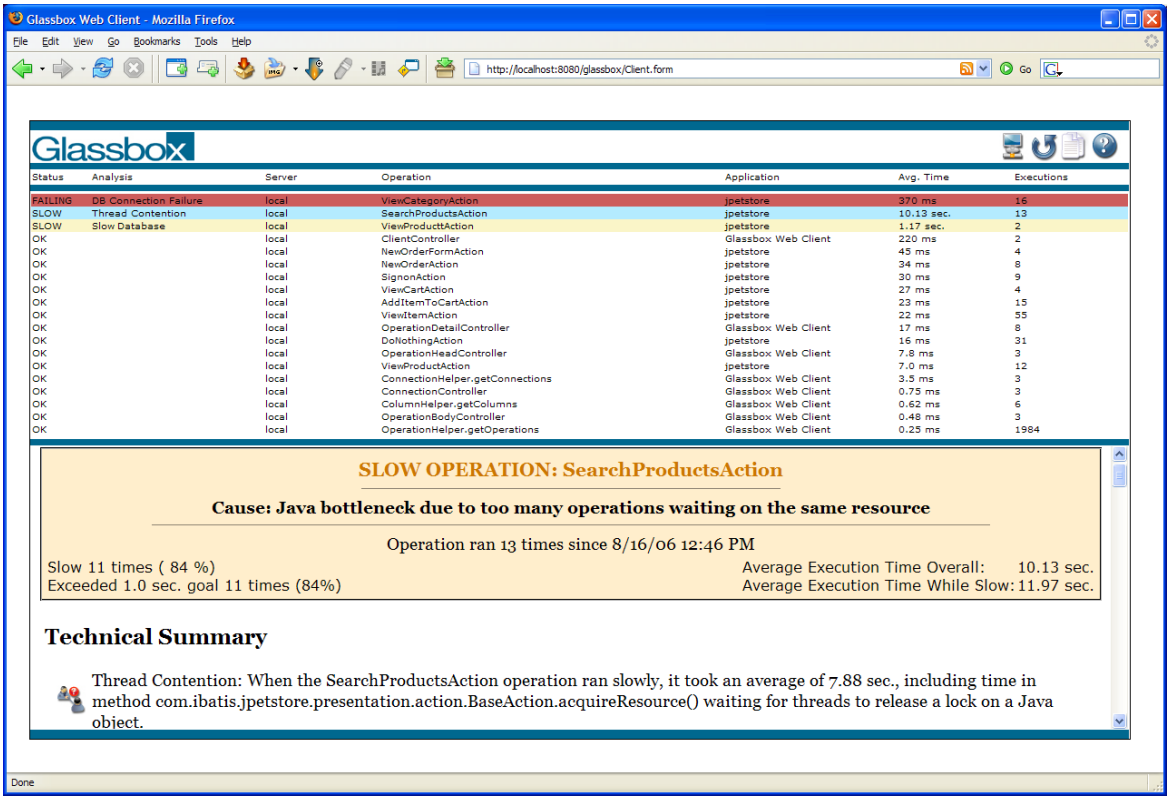

Abbildung 8.3: Monitoring und Messungen mit dem Testwerkzeug Glassbox. Quelle: (Glassbox, 2008)

8.4.1 SQL-Anweisungen und Dateninkonsistenz

Der Bereich Selektion bietet für einen Benutzer eine einfache Möglichkeit, um komplexe Datenbankabfragen zu erstellen. Bei der Durchführung einer Selektion wurde festgestellt, dass einige Datensätze in das Ergebnis nicht miteinbezogen werden, obwohl alle Kriterien für diesen Datensatz erfüllt sind. Dasselbe Problem trat bei der globalen Suchoption auf. Um dieses Problem zu lösen, mussten die generierten Datenbankabfragen analysiert werden. Festgestellt wurde, dass Datensätze teilweise inkonsistent waren. So hatten einige Daten den Wert *null* in Spalten, die normalerweise als Pflichtfelder definiert sind.

8.4.2 Export und Betriebssystemabhängigkeit

Tests wurden zunächst lokal auf der Entwicklungsmaschine unter Microsoft Windows XP durchgeführt. Anschließend wurden die Tests auf dem Testsystem unter Debian Linux wiederholt. Bei dem Export der Daten nach einer Selektion wurde festgestellt, dass Umlaute und Ligaturen falsch dargestellt werden. Grund dafür war die fehlende Zeichenkodierung, die festgelegt werden sollte. Somit wurde beim Export die Microsoft-Codierung CP1252 verwendet, mit der Linux-basierte Betriebssysteme nicht umgehen können (vgl. Wikipedia, 2008b). Um eine Betriebssystemunabhängigkeit zu ermöglichen, wird in Zukunft die standardisierte Codierung ISO 8859-1 verwendet.

8.4.3 Selektionen und Grenzwerte

Durch die Grenzwertanalyse wurde ein Fehler bei den Selektionskriterien aufgedeckt. Wenn bei Selektionen kein Kriterium angegeben wurde, erhielt der Benutzer kein Ergebnis. Dieser erwartet jedoch, dass alle Datensätze exportiert werden, wenn kein Kriterium angegeben wurde.

8.4.4 Zusammenfassung von Schlüsseltabellen

Im Rahmen eines Code-Reviews hat sich eine Optimierung bei den Datenbanktabellen ergeben. Zu diesem Zeitpunkt gab es u. a. folgende Tabellen:

- buyinginterest: Kaufinteresse einer Kontaktperson
- activity_type: Typ einer Aktivität
- accounttype: Typ eines Unternehmens
- bc_roles: Rolle einer Kontaktperson in einem Buyingcenter
- countries: Landessitz des Unternehmens
- industry: Industrie des Unternehmens
- addresstype: Adresse des Unternehmens
- reporttype: Art des Reports
- wordtemplates: Dateinamen von MS-Office Vorlagen

Alle oben genannten Tabellen besitzen eine feste Anzahl an Einträgen, die sich vorerst nicht ändern. Um das Datenbankschema übersichtlicher zu gestalten, wurden diese Tabellen zu einer Tabelle zusammengeführt. Entstanden ist daraus eine Tabelle *enumerationtype*, welche sich zusätzlich über den Admin-Bereich pflegen lässt (siehe hierzu Abb. 7.9).

8.4.5 Navigation und Performance

Seit Projektbeginn gab es bereits Probleme in der Performance. Nach einer Anfrage an den Server musste der Benutzer vier bis zehn Sekunden auf die Antwort warten. Dementsprechend langsam war dadurch der Seitenaufbau der CRM-Anwendung. Zudem war der Prozessor des Servers nach jeder Anfrage voll ausgelastet.

Eine Analyse mit dem Testwerkzeug Glassbox hat folgendes ergeben:

- Im Durchschnitt musste der Client fünf Sekunden auf eine Antwort warten.
- Bei jedem „Klick“ wurde die Datenbank voll ausgelastet.
- Bei jedem „Klick“ wurden identische Anfragen bis zu 1.000 Mal an die Datenbank geschickt.
- Bei jedem „Klick“ wurden bis zu 10.000 Anfragen an die Datenbank geschickt.
- Da die Anwendung auf die Antwort der ausgelasteten Datenbank wartet, entstand hier ebenfalls eine volle Auslastung.

Hauptgrund für diese Auslastung durch Datenbankanfragen war ein Programmierfehler im Rechtesystem (siehe hierzu ausführlich: Schmer, 2008, S.31). In diesem System entscheidet ein Recht darüber, ob ein an das Recht gekoppelter Bereich für den Benutzer ein- oder ausgeblendet werden soll. Zu diesem Zeitpunkt gab es ca. 40 Rechte und somit 40 unterschiedliche Bereiche. Das Problem bestand darin, dass pro Klick des Benutzers für jede Rechteüberprüfung eine Datenbankabfrage abgeschickt wurde. Diese hat die Daten und Rechte des Benutzers in der Business Logik aktualisiert. Da der Benutzer nach dem Login in das System eine Session startet, die bis zum Logout gültig ist, reicht es diese Daten nur ein Mal zu laden und global festzuhalten. Durch diese Maßnahme sind bei einem Regressionstest durch Glassbox folgende Ergebnisse entstanden:

- Im Durchschnitt muss der Client weniger als eine Sekunde auf eine Antwort warten.
- Der Datenbankprozess lastet den Prozessor bis zu 30% aus.
- Bei jedem „Klick“ werden ca. ein dutzend identische Anfragen an die Datenbank geschickt.
- Bei jedem „Klick“ werden bis zu 100 Anfragen an die Datenbank geschickt.
- Weder die Datenbank, noch die Anwendung wird voll ausgelastet.

Diese Werte werden im Monitoring von Glassbox nicht beanstandet. Durch einige Optimierungen von Datenbankanweisungen konnten diese Werte verbessert werden.

Abschließend kann gesagt werden, dass durch diese Fehlerbehebungen die Qualität der Software verbessert wurde. Durch die Lösung des Performanceproblems ist eine höhere Akzeptanz entstanden, so dass einem Produktivbetrieb innerhalb der novomind AG nichts mehr im Weg steht.

9 Resümee

An dieser Stelle soll zunächst die Arbeit und das Ergebnis reflektiert werden. Anschließend wird im Ausblick erwähnt, welche Schritte als nächtes erforderlich sind, um die Anwendung marktreif zu machen.
Im letzten Teil, der methodischen Abstraktion, werden die methodische Vorgehensweise und die Erfahrungen beschrieben, die während der Arbeit entstanden sind.

9.1 Zusammenfassung

Ziel dieser Arbeit war die Entwicklung eines Reportmoduls für Ad-hoc-Abfragen. Dieses Modul sollte in das bisher entwickelte CRM-System iSell integriert werden.
Bevor die Anforderungen durch Führungskräfte und zukünftige Benutzer aufgenommen wurden, wurde ein betriebswirtschaftliches Konzept erarbeitet, um alle Anforderungen im Bereich Vertrieb und Marketing zu verstehen. Durch dieses Konzept sind zusätzliche Anforderungen entstanden, welche für die Entwicklung eines Data Warehouse-Systems sprechen.
Ausschlaggebend für den Erfolg dieses Projektes war, neben der Korrektheit des Systems, eine benutzerfreundliche Gestaltung des User Interfaces, mit dem alle Benutzer zurecht kommen.
Um dies zu erreichen, wurden zunächst Report-Bereiche einiger namenhafter CRM-Systeme untersucht. Ergebnis dieser Untersuchung war, dass alle Systeme den Bereich Reports ähnlich konzipiert haben. Demnach besitzen alle Systeme Funktionen um Kontaktdaten zu exportieren, Statistiken zu erstellen und Ad-hoc-Abfragen durchzführen. In einer anschließenden Untersuchung wurden zusätliche Hilfsmittel analysiert, welche die Entwicklung eines benutzerfreundlichen Reportmoduls unterstützen könnten. Aufgrund der geringen

Unterstützung der neuen Java Technologien (u.a. EJB 3.0) wurde eine Individualentwicklung angestrebt.
Durch die Eigenentwicklung wurde das Reportmodul in das bisherige System integriert, so dass als nächstes eine Erweiterung dieses Prototypen zu einem Data Warehouse-System möglich ist. In einem anschließenden Test wurden Fehler beseitigt und die gesamte Anwendung optimiert, so dass einem Produktiveinsatz in der novomind AG nichts mehr im Weg steht.

9.2 Ausblick

Das CRM-System iSell befindet sich zu diesem Zeitpunkt in der Version 1.3-011. Um das System Marktreif zu machen, sollte als nächstes das Reportmodul zu einem Data Warehouse-System erweitert werden. Nur durch eine eigene analytische Datenstruktur und analytische Tools kann sichergestellt werden, dass der Bereich Reporting in allen Unternehmensbereichen Gebrauch findet. In der zukünftigen Vision (siehe Kapitel 5.2) werden bereits einige Open Source Data Warehouse-Systeme genannt, die evaluiert werden sollten. Das entwickelte Reportmodul bietet dafür eine solide Basis, wodurch eine Wiederverwendung der Komponenten möglich ist. Da ein vollständiges Data Warehouse-System umfangreiche Funktionen bietet, kann das entwickelte Data Warehouse-System als Plugin für den Open Source Markt angeboten werden, damit weiterhin eine leichtgewichtige Version von iSell zur Verfügung steht.
Zusätzlich sollte man sich mit dem Thema Open Source beschäftigen. Damit letztendlich keine proprietäre Software entsteht, sollte zumindest das Problem mit der Browser- und Datenbankkompatibilität gelöst werden (siehe hierzu ausführlich: Schmer, 2008, S.73f). Zudem sollte man sich mit den Lizenzen (und dadurch Verpflichtungen) der bisher genutzten Open Source Bibliotheken vertraut machen.

9.3 Methodische Abstraktion

Da ich in diesem Projekt zum ersten Mal mit prototypischen Ansätzen entwickelt habe, kann ich keine Vergleiche stellen. Dennoch gibt es erwähnenswerte Aspekte, die ich nicht außer Acht lassen möchte.
Durch die prototypische Entwicklung steht man den zukünftigen Benutzern sehr nah, so dass eine Fehlentwicklung fast ausgeschlossen werden kann. Letztendlich ist in diesem Projekt ein Prototyp entstanden, der alle Anforderungen durch das benutzerfreundliche User Interface mit vollster Zufriedenheit erfüllt. Durch eine abschließende Verifikation der Anwendung (siehe Kapitel 7.2) sind die letzten Verbesserungsvorschläge eingegangen und umgesetzt worden. Da sich zu diesem Zeitpunkt die Anwendung bereits im Produktiveinsatz in der novomind AG befindet, kann man zudem sagen, dass das gesamte Projekt seinen Zweck erfüllt hat.

Wie in fast jedem Projekt gibt es, trotz des Erfolges, einige Aspekte, die etwas vernachlässigt wurden. Weder in der Arbeit von Martin Sadowski (Sadowski, 2008) und André Schmer (Schmer, 2008), noch in dieser Arbeit wurden Anforderungen außerhalb der novomind AG erfasst. Wie Martin Sadowski bereits in seiner Arbeit geschildert hat, fehlen einige Funktionen die für eine Open Source Software unabdingbar sind (vgl. Sadowski, 2008, S.61). Zusätzlich sollte man sich bei einem Open Source Projekt fragen, wer die zukünftigen Benutzer sein werden. Abhilfe könnten hier unter anderem Unternehmensumfragen und deren Auswertungen schaffen.

Dessen ungeachtet war die prototypische Vorgehensweise in diesem Projekt ein voller Erfolg. Die zukünftigen Benutzer haben quasi an der Entwicklung teilgenommen und die Anwendung mit den Entwicklern gemeinsam entworfen. Dementsprechend groß ist auch die Akzeptanz des Gesamtsystems. Aus Entwicklersicht wird man durch das Feedback der Benutzer motiviert, so dass man in jeder Phase des Projektes Spaß hat.

A Die zugehörige CD

Die CD zu der vorliegenden Arbeit enthält zusätzliche Dokumente und Dateien. Neben einer digitalien Version dieser Bachelorarbeit beinhaltet sie den vollständigen Quelltext der Web-Anwendung. Darüber hinaus sind einige ergänzende Literaturen der Internet-Links auf der CD vorhanden.

Glossar

Ad-hoc
Vorgänge, die spontan ohne Planung ausgeführt werden.

AJAX
Konzept für eine asynchrone Datenübertragung zwischen Server und Client/Browser

Blackbox-Testing
Testen von Software ohne Kenntnis über die innere Funktionsweise (z. B. Programmcode). Die Tests werden in der Regel über das User Interface oder über Schnittstellen durchgeführt.

Checkstyle
Eclipse-Plugin für eine automatische Überprüfung von Coding Conventions bei der Erstellung von Java Code.

Cross-Selling
Verkauf von ergänzenden Produkten/Dienstleistungen.

Interopabilität
Fähigkeit der Zusammenarbeit verschiedener Systeme. Meist durch die Einhaltung von Standards.

New Economy
Entstehung einer neuen Wirtschaftsform durch die digitale Revolution.

Permission Marketing
Marketingmethoden, die durch Erlaubnis des Kunden herbeigeführt werden.

Up-Selling
Verkauf von höherwertigen Produkten/Dienstleistungen.

Whitebox-Testing
Software-Tests direkt am Quellcode.

Abkürzungsverzeichnis

AG	Aktiengesellschaft
API	Application Programming Interface
AJAX	Asynchronus JavaScript and XML
CRM	Customer Relationship Management
DWH	Data Warehouse
EJB	Enterprise Java Beans
RM	Relationship Marketing
JDBC	Java Database Connector
JNDI	Java Naming and Directory Interface
JPA	Java Persistence API
JPQL	Java Persistence Query Language
JSP	Java Server Paces
JSF	Java Server Faces
HTTP	Hyper Text Transfer Protocol
SSL	Secure Sockets Layer
EJB	Enterprise Java Beans
UI	User Interface
UML	Unified Modeling Language
HTML	Hyper Text Markup Language
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
CSV	Comma Separated Values
SQL	Structured Query Language
PL/SQL	Procedural Language/SQL

Literaturverzeichnis

[Adelchi 2004] ADELCHI, Missio: *JasperReports*. Aargau, Schweiz, Fachhochschule Nordwestschweiz, Seminarbericht, Juli 2004. – URL `http://www.gruntz.ch/courses/sem/ss04/jasperreports.pdf`. – (Stand 05.09.2008)

[Bauer und King 2005] BAUER, Christian ; KING, Gavin: *Hibernate in action*. Greenwich : Manning Publications Co., 2005

[Blume 2007] BLUME, Katrin: *Ein integriertes Data Warehouse System und eine innovative Software für das Reporting - grundlegend für den zukünftigen Erfolg eines Unternehmens?* Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, Juni 2007

[Bruhn 2007] BRUHN, Prof. Dr. M.: *Kundenorientierung*. Bd. 3. Auflage. München : Deutscher Taschenbuch Verlag, 2007

[Buth 2008] BUTH, Bettina: *Software Engineering 2*. Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Vorlesungsfolien, 2008. – URL `https://users.informatik.haw-hamburg.de/home/pub/prof/buth/SE-alt/SE2-SoSe08/slides/SE2-slides-sose08-v1.pdf`. – (Stand 23.10.2008)

[Chamoni und Gluchowski 2005] CHAMONI, Peter ; GLUCHOWSKI, Peter: Einordnung und Überblick. In: *Analytische Informationssysteme*. Berlin : Springer, November 2005, S. S.3–25

[CMU 2005] CMU, Carnegie Mellon U.: *How Do You Define Software Architecture?*, Software Engineering Institute (SEI), Umfrage, 2005. – URL `http://www.sei.cmu.edu/architecture/definitions.html`. – (Stand 12.09.2008)

[DataVision 2008] DATAVISION: *DataVision Website*. Juli 2008. – URL `http://datavision.sourceforge.net/index.html#features`. – (Stand 05.09.2008)

[Diller 2007] DILLER, Hermann: Die Bedeutung von Beziehungsmarketing für den Unternehmenserfolg. In: HIPPNER, Hajo (Hrsg.) ; WILDE, Klaus D. (Hrsg.): *Grundlagen des CRM*. Wiesbaden : Betriebswirtschaftlicher Verlag Gabler, Juli 2007, S. 97–120

[Glassbox 2008] GLASSBOX: *Glassbox Website*. 2008. – URL `http://www.glassbox.com/`. – (Stand 30.09.2008)

[Gluchowski 2005] GLUCHOWSKI, Peter: Techniken und Werkzeuge zum Aufbau betrieblicher Betriebssysteme. In: *Analytische Informationssysteme*. Berlin : Springer, November 2005, S. S.179–200

[Hannemann 2002] HANNEMANN, Stefan: *Vertrauensentstehung in Netzwerken*. Hagen, Institut für Psychologie der FernUniversität Hagen, Literaturarbeit, Mai 2002. – URL `http://psychologie.fernuni-hagen.de/pdf/Vertrauen.pdf`. – (Stand 28.08.2008)

[Heider 2007] HEIDER, Sabine: Basics of the Java Persistence API - Understanding the Entity Manager / SAP Developer Network. URL `https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/60216821-e789-2910-109b-b6c6e02dce87`, Januar 2007. – Techreport. (Stand 02.09.2008)

[Heider u. a. 2006] HEIDER, Sabine ; GÖRLER, Adrian ; REISBICH, Julia: Getting Started with Java Persistence API and SAP JPA 1.0 / SAP Developer Network. URL `https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/40ff8a3d-065a-2910-2f84-a222e03d1f43`, November 2006. – Techreport. (stand 29.08.2008)

[Hippner und Wilde 2007] HIPPNER, Hajo ; WILDE, Klaus D.: *Grundlagen des CRM*. Bd. 2. Auflage Nachdruck. Wiesbaden : Betriebswirtschaftlicher Verlag Gabler, Juli 2007

[Höschel 2007] HÖSCHEL, Hans-Peter: *Data-Mining für Marketing und Vertrieb - kurz und knapp*. ScoreXpert. 2007

[Infobright 2008] INFOBRIGHT: *Infobright Community Edition*. 2008. – URL `http://www.infobright.org/Open-Source/Home`. – (Stand 23.09.2008)

[ISO 2008] ISO: *ISO and IEC members give go ahead on ISO/IEC DIS 29500*. August 2008. – URL `http://www.iso.org/iso/pressrelease.htm?refid=Ref1151`. – Stand 17.10.2008

[JasperSoft 2006] JASPERSOFT: *JasperReports Datasheet*. 2006. – URL `http://www.jaspersoft.com/downloads/Datasheet/jasperreports-0206.pdf`. – (Stand 05. 09.2008)

[Keller 2002] KELLER, Wolfgang: *Software Engineering für große Informationssysteme - Software Architektur I*. Wien, Technische Universität Wien, Vorlesungsunterlagen, 2002. – URL `http://www.inf.fu-berlin.de/inst/ag-se/teaching/V-J2EE-2003/22_Architektur.pdf`. – (stand 12.09.2008)

[Kurz 1999] KURZ, Andreas: *Data Warehousing. Enabling Technology*. MITP-Verlag, 1999

[Lilienthal u. a. 2007] LILIENTHAL, Carola ; RAASCH, Jörg ; ZUKUNFT, Olaf: *Architekturen von Informationssystemen*. Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Vorlesungsfolien, April 2007. – URL `https://users.informatik.haw-hamburg.de/home/pub/prof/lilienth/2007SS/Unterlagen20070412/2-ai-ss07-kap2-Architektur-SE.pdf`. – (Stand 22.10.2008)

[McLaughlin 2002] MCLAUGHLIN, Brett: *Java & XML*. O'Reilly, 2002

[Meyer 1988] MEYER, Bertrand: *Object-oriented Software Construction*. Prentice Hall, 1988

[Murray 2006] MURRAY, Katherine: *Microsoft Office 2007 - Die Neuerungen im Überblick*. Unterschleißheim : Microsoft Press Deutschland, 2006

[NetworkWorkingGroup 2005] NETWORKWORKINGGROUP: *RFC4180: Common Format and MIME Type for Comma-Separated Values (CSV) Files*. 2005. – URL `http://tools.ietf.org/html/rfc4180`. – (Stand 20.10.2008)

[Offe 1999] OFFE, Lars: *Produktvergleich von Reporting-Tools im Data Warehouse-Umfeld*. Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, September 1999

[Palloks-Kahlen 2001] PALLOKS-KAHLEN, Monika: Kennzahlengestütztes Controlling im kundenorientierten Vertriebsmanagement. In: *Handbuch Marketingcontrolling: Marketing als Motor von Wachstum und Erfolg*. Frankfurt/Wien : Wirtschaftsverlag Carl Ueberreuter, 2001, S. S.520–543

[Pentaho 2007] PENTAHO: *Pentaho Reporting Datasheet*. Oktober 2007. – URL `http://www.pentaho.com/docs/pentaho_reporting.pdf`. – (Stand 05.09.2008)

[Pentaho 2008] PENTAHO: *Business Intelligence Suite*. 2008. – URL `http://www.pentaho.com/products/bi_platform/`. – (Stand 23.09.2008)

[Raasch 2007] RAASCH, Jörg: *Usability, Workflow und Software-Architektur.* Hamburg, World Usability Day, Vortrag, November 2007. – URL `http://www.worldusabilityday.de/2007/hamburg/vortraege/Raasch_Workflow_Softwarearchitektur.pdf`. – (Stand 09.09.2008)

[Raasch u. a. 2007] RAASCH, Jörg ; ZUKUNFT, Olaf ; KAHLBRANDT, Bernd: *Requirements Engineering*. Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Vorlesungsfolien, November 2007. – URL `https://users.informatik.haw-hamburg.de/home/pub/prof/raasch/SE1/026v-Requirements-Engineering.pdf`. – (Stand 04.09.2008)

[RedHat 2008] REDHAT: *JBoss Product Datasheet*. 2008. – URL `http://www.jboss.com/pdf/JBossASBrochure-Jun2005.pdf`. – (Stand 01.09.2008)

[Reichheld und Schefter 2000] REICHHELD ; SCHEFTER: E-Loyalty: Your Secret Weapon on the Web. In: *Harvard Business Review* Vol. 78 (2000), S. 105–113

[Reinecke u. a. 2001a] REINECKE, Sven ; TOMCZAK, Torsten ; GEIS, Gerold: Einsatz von Instrumenten und Verfahren des Marketingcontrollings in der Praxis. In: *Handbuch Marketingcontrolling: Marketing als Motor von Wachstum und Erfolg*. Frankfurt/Wien : Wirtschaftsverlag Carl Ueberreuter, 2001, S. S.76–88

[Reinecke u. a. 2001b] REINECKE, Sven ; TOMCZAK, Torsten ; GEIS, Gerold: Marketingkennzahlensysteme: Notwendigkeit, Gütekriterien und Konstruktionsprinzipien. In: *Handbuch Marketingcontrolling: Marketing als Motor von Wachstum und Erfolg*. Frankfurt/Wien : Wirtschaftsverlag Carl Ueberreuter, 2001, S. S.690–719

[Sadowski 2008] SADOWSKI, Martin: *Konzeption und prototypische Entwicklung eines Open-Source Vertriebssystems*. Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, Juli 2008

[Schmer 2008] SCHMER, André: *Weiterentwicklung, Evaluation und Einführung eines prototypischen Open Source Vertriebssystems*. Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, Juli 2008

[Shannon 2006] SHANNON, Bill: Java Platform, Enterprise Edition (Java EE) Specification, v5 / Sun Microsystems, Inc. URL `http://www.jcp.org/en/jsr/detail?id=244`, Juni 2006. – Java Specification Requests (JSR 244). (Stand 29.08.2008)

[Siedersleben 2003] SIEDERSLEBEN, Johannes: *Softwaretechnik*. München : Carl Hanser / sd&m, 2003

[Snyder 2003] SNYDER, Carolyn: *Paper Prototyping*. Morgan Kaufmann Publishers, 2003

[Spillner und Linz 2005] SPILLNER, Andreas ; LINZ, Tilo: *Basiswissen Softwaretest - Aus- und Weiterbildung zum Certified Tester*. Heidelberg : dpunkt.verlag GmbH, 2005

[Stark 2005] STARK, Thomas: *J2EE - Einstieg für Anspruchsvolle*. München : Addison-Wesley AND Pearson Studium, 2005

[Sun 2002] SUN: *Core J2EE Patterns - Service Locator*. 2002. – URL http://java.sun.com/blueprints/corej2eepatterns/Patterns/ServiceLocator.html. – (Stand 25.09.2008)

[Sun 2004] SUN: *Java EE 1.4 Compatible Implementations*. 2004. – URL http://java.sun.com/j2ee/compatibility_1.4.html. – (Stand 30.08.2008)

[Sun 2007] SUN: *Java EE 5 Tutorial*. Santa Clara, U.S.A.: , September 2007. – URL http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf. – (Stand 02.09.2008)

[UsabilityNet 2006] USABILITYNET: *International standards for HCI and usability*. 2006. – URL http://www.usabilitynet.org/tools/r_international.htm#9126-1. – (Stand 05.09.2008)

[Wikipedia 2008a] WIKIPEDIA: *Data-Warehouse-Anwendungen*. 2008. – URL http://de.wikipedia.org/wiki/Data-Warehouse#Data-Warehouse-Anwendungen. – (Stand 27.10.2008)

[Wikipedia 2008b] WIKIPEDIA: *Windows-1252*. 2008. – URL http://en.wikipedia.org/wiki/Windows-1252. – (Stand 31.10.2008)

[Züllighoven u. a. 1998] ZÜLLIGHOVEN, Heinz ; BÄUMER, Dirk ; BLEEK, Wolf-Gideon ; LILIENTHAL, Carola: *Das objektorientierte Konstruktionshandbuch*. Dpunkt Verlag, 1998

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 14. November 2008	
Ort, Datum	Unterschrift