Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

Abkürzung	Normal
B2B	**Business To Business**
B2C	**Business To Customer**
BB-Code	**Bulletin Board Code**
CAPI	**Cartridge Application Programming Interface**
DBS	**Datenbanksystem**
ERM	**Entity-Relationship-Modell**
JSP	**Java Server Pages**
Java EE	**Java Platform Enterprise Edition (früher J2EE)**
MVC	**Model View Controller (Web Presentation Design Pattern)**
ORM	**Object Relational Mapping**
SEO	**Search Engine Optimization**
UML	**Unified Modeling Language**

Glossar

Begriff	Beschreibung
Ajax	Asynchronous JavaScript and XML (Ajax) ist ein Konzept zur asynchronen Datenübertragung zwischen Server und Client (Browser). Es können Teilbereiche der Seite aktualisiert werden, ohne die Seite vollständig laden zu müssen.
Intershop Communications AG	Die Intershop Communications AG (1992 gegründet) ist ein führender Anbieter von umfassenden E-Commerce Lösungen. Weltweit setzen über 300 große und mittelständische Unternehmen auf Lösungen von Intershop. Zu ihnen zählen unter anderem Otto, Quelle, Tchibo, T-Home, Sun Microsystems, Hewlett Packard und Bosch. Weitere Informationen unter http://www.intershop.de/.
Microsoft Visio	Microsoft Visio ist eine Visualisierungs-Software zur Darstellung von Informationen und Diagrammen. Mehr Informationen unter http://office.microsoft.com/de-de/visio/FX100487861031.aspx.
Oracle Corporation	Oracle Corporation (1977 gegründet) ist einer der weltweit größten Softwarehersteller. Das bekannteste Produkt des Unternehmens ist das DBMS Oracle Database. Weitere Informationen unter Oracle Deutschland http://www.oracle.com/global/de/index.html.
Sun Microsystems	Sun Microsystems (1982 gegründet) ist ein führender Anbieter von Serversystemen und Software. Mit Java schuf Sun 1995 eine objektorientierte, systemunabhängige Programmierplattform. Weitere Informationen unter http://de.sun.com/.

1 Einführung

1.1 Zielstellung

Ziel dieser Arbeit ist es, dem Leser eine Einführung in die Web-Entwicklung mit Intershop Enfinity Studio zu geben. Am konkreten Beispiel des „SCOOBOX“-Internetforums sollen die einzelnen Entwicklungsschritte aufgezeigt und erläutert werden.

1.2 Motivation

Das Thema Softwareentwicklung, mein Interesse an Webtechnologien und die Unterstützung durch das Entwicklungsteam motivierten mich zur Umsetzung der Thematik. Das Forum stellt eines der komplexesten Bestandteile eines Shops dar, wodurch viele verschiedene Faktoren bei der Implementierung zu berücksichtigen sind.

1.3 Vorgehen

Aufgrund der Komplexität der Entwicklungsumgebung ist es notwendig eine theoretische Vorbetrachtung zum Thema Intershop Enfinity Suite anzustellen, bevor im Hauptteil der detaillierte Entwurf und seine Umsetzung erläutert werden.
Bedingt durch die umfangreiche Funktionalität eines Internetforums konzentriert sich diese Arbeit hauptsächlich auf den Entwurf desselben. Die Implementierung wird an ausgewählten Beispielen beschrieben.
Abschließend werden sog. „Lessons Learned“ formuliert, in denen ich Rückschlüsse aus Entwurf und Implementierung ziehe.

2 Grundlagen

2.1 Intershop Enfinity Suite

„Enfinity Suite 6 ist eine führende Software für professionellen Online-Handel. […] Unterschiedliche Geschäftskanäle werden auf einer Plattform zusammengeführt und zentral gesteuert.“[1]

Enfinity Suite wird seit 1994 von der Intershop AG aktiv weiterentwickelt und stellt dabei, laut eigenen Angaben, das erste voll funktionierende E-Commerce System dar. Durch den vollständig modularen Aufbau des Systems ist es möglich, sämtliche Kanäle, wie beispielsweise den Consumer Channel (B2C) oder den Business Channel (B2B), eines Shops abzudecken.

Weiterhin umfasst es Themen wie Internationalisierung, Auffindbarkeit (SEO), zentrales Produktmanagement, Analyse, Vertrieb und Marketing, Content Mangement sowie Kunden- und Bestelldatenverwaltung.

Im folgenden Abschnitt wird Enfinity Suite 6 kurz vorgestellt, das Hauptaugenmerk liegt hierbei auf den Aspekten Technik und Architektur.

2.1.1 Server Architektur

Intershop Enfinity Suite 6 wurde entwickelt um mehrere Web-Applikationen sowie Webservices auf einer Serverumgebung parallel betreiben zu können. Der Enfinity Suite 6 Application Server nutzt dafür von Sun Microsystems entwickelte Java EE[2] Technologien wie Servlets[3] und JSPs[4].

An in dieser Stelle wird der Aufbau des Application Servers anhand Abbildung 1 beschrieben.

---

[1] [IS_WEB]

[2] Java EE - http://java.sun.com/javaee/ - 07.09.2009

[3] Java Servlet Technology - http://java.sun.com/products/servlet/ - 24.08.2009

[4] Java Server Pages Technology - http://java.sun.com/products/jsp/ - 24.08.2009

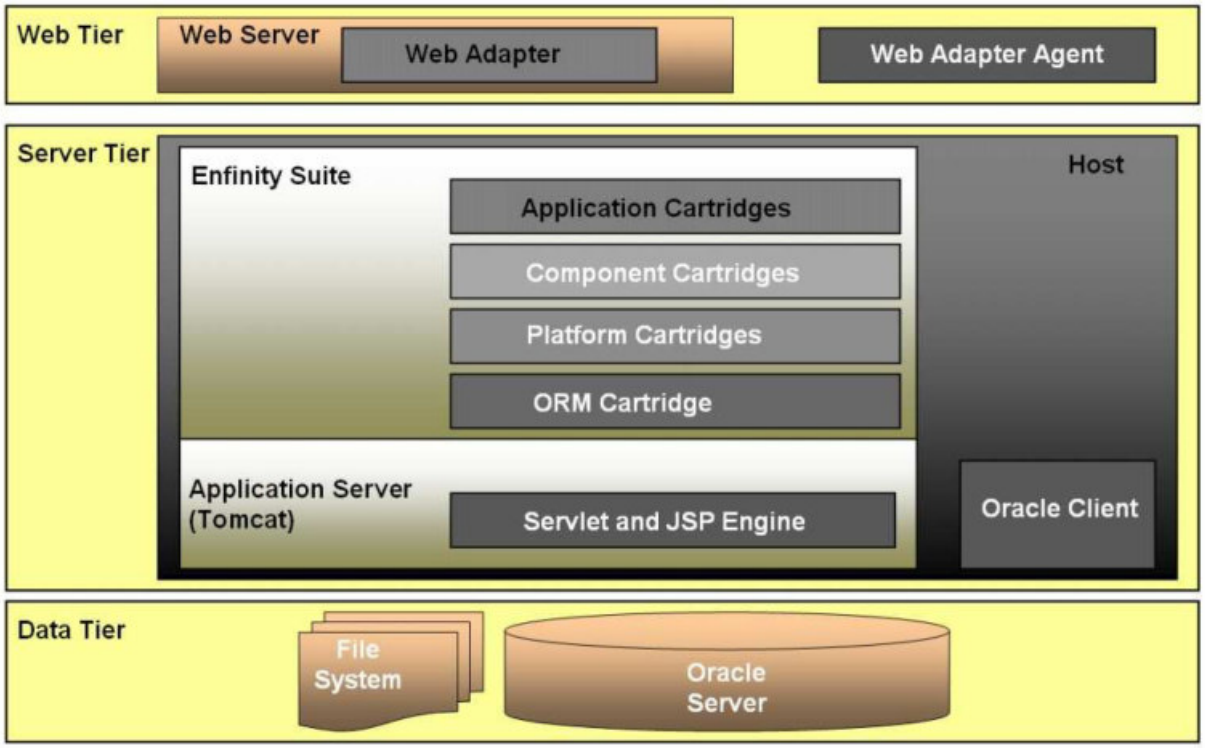

Abbildung 1: Enfinity Suite Application Server Komponenten[5]

Die Web-Schicht (Web Tier) beinhaltet Webserver und Webadapter. Der Webserver, ein Apache HTTP[6] Server, ist für das Entgegennehmen von HTTP- Requests, die Lastverteilung (Loadbalancing) auf die Applikationsserver und die Darstellung des HTTP-Response verantwortlich. Der Webadapter, eine von Intershop entwickelte Erweiterung für den Apache 2 HTTP Server, verwaltet hauptsächlich den Cache.
Die Serverschicht (Server Tier) beinhaltet die eigentliche Enfinity Suite, bestehend aus mehreren voneinander abhängigen Modulen (sog. Cartridges). In dieser Schicht läuft der mittels HTTP-Request gestartete Prozess ab, er wird in einem Apache Jakarta Tomcat 5.0[7] Servlet Container ausgeführt. Dieser generiert den HTTP-Response, welcher durch die Web-Schicht ausgeliefert wird.
Zur Datenverwaltung (Data Tier) nutzt Enfinity Suite 6 einen Oracle 10g Server. Die Persistierung der (Objekt-)Daten erfolgt hierbei durch eine von Intershop entwickelte ORM-Schicht. Statische Daten werden im Dateisystem ausgelagert.

---

[5] Vgl. [IS07], S.11

[6] The Apache Software Foundation – Apache HTTP Server Project http://httpd.apache.org/ - 24.08.2009

[7] The Apache Software Foundation – Apache Tomcat http://tomcat.apache.org/ - 24.08.2009

2.1.2 Implementierungsarchitektur

Enfinity Suite differenziert sich in vier verschiedene Implementierungsschichten.

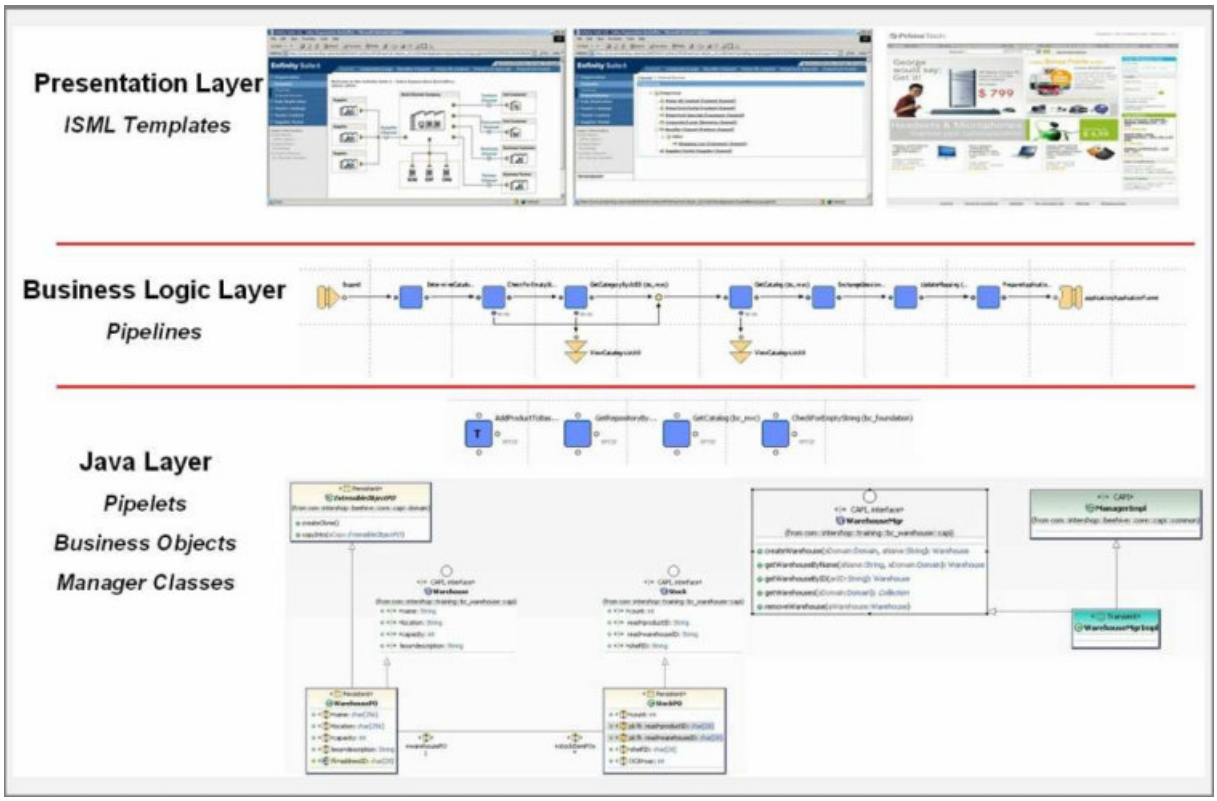

Abbildung 2: Enfinity Suite Implementierungs Schichten[8]

2.1.2.1 Java Layer: Business Objects und Manager

Diese Schicht beinhaltet Entitäten (Java Klassen), welche Business-Objekte oder Manager repräsentieren. Die Business-Objekte implementieren das Core J2EE Pattern *Business Object*[9] und *Transferobject*[10]. Die generierten Klassen *POFactory* implementieren das Core J2EE Pattern *Data Access Object*[11] und dienen als Schnittstelle zu einem persistenten Speichermedium (Datenbank). Jedes Business-Objekt ist mit einem Manager verbunden. Jene Manager stellen Methoden zum

[8] Vgl. [IS07], S.14

[9] Business Object - http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html - 20.09.2009

[10] Transferobject - http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html - 14.09.2009

[11] DAO - http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html - 07.09.2009

Erzeugen, Bearbeiten und Löschen eines Business-Objektes, sowie Methoden für einfache Business-Logik bereit.
Der Zugriff auf Business-Objekte und Manager erfolgt über klar definierte Schnittstellen (Interfaces). Dies hat den Vorteil, dass die tatsächliche Implementierung der Klasse dem Nutzer verborgen bleibt und die Implementierungsabhängigkeiten reduziert werden[12].

2.1.2.2 Java Layer: Pipelets

Pipelets stellen Business-Funktionalität basierend auf Business-Objekten und Managern bereit.
Realisiert werden Pipelets durch Java Klassen, welche die Funktionalität des Business Object Layer in kleine, wiederverwendbare Elemente aufgliedern.[13]
Die durch Pipelets zur Verfügung gestellte Business-Funktionalität kann mehrfach in verschiedenen Pipelines (siehe 2.1.2.3) verwendet werden. Zur Kommunikation zwischen Pipelets wird ein Pipeline Dictionary verwendet, das ähnlich einer Hashmap[14] aufgebaut ist.

2.1.2.3 Business Logic Layer (Pipeline Layer)

Eine Pipeline stellt die logische Abbildung eines speziellen Businessprozesses dar. Sie verbindet mehrere Pipelets (Business Funktionalitäten) mit dem Ziel komplexe Abläufe zu beschreiben. Je nach Aufgabenstellung unterscheidet man Prozess- und Viewpipelines. Prozesspipelines implementieren wiederverwertbare Teile der Business Logik, sie besitzen definierte Ein- und Ausgabeparameter, wodurch sie als Blackbox verwendet werden können. Viewpipelines dienen der Visualisierung von Daten für den Nutzer.

2.1.2.4 Presentation Layer

Der Presentation Layer wandelt die Ergebnisse eines Businessprozesses auf Basis von Templates in einen HTTP-Response um, welcher an den Nutzer gesendet wird.

---

[12] Vgl. [Gam09], S. 25

[13] Vgl. [IS09] S.65

[14] HashMap in der Javadoc 6 - http://java.sun.com/javase/6/docs/api/java/util/HashMap.html - 08.09.2009

Das Template bildet dabei ein Muster, welches die Struktur, den Style sowie die statischen Inhalte vorgibt. Über Variablen kann im Template auf die Inhalte des Resultats eines Businessprozesses zugegriffen werden. Hierbei findet die von Intershop entwickelte Intershop Markup Language (ISML) Verwendung.

2.1.3 Intershop Enfinity Studio

Enfinity Studio ist eine auf Eclipse[15] basierende Entwicklungsumgebung für die Enfinity Suite. Alle Bestandteile der Enfinity Suite, d.h. unter anderem Templates, Pipelets, Pipelines und Java Quellcode Dateien, sind vom Enfinity Studio aus zugänglich und editierbar. Die Kopplung des Studios an den Application Server erlaubt es im laufenden Betrieb Änderungen am System vorzunehmen.

Für die Modellierung von Business-Objekten und Managern wurde ein UML2 Editor integriert, der es ermöglicht aus einem validen Modell den entsprechenden Java Quellcode für Business-Objekte und Managerklassen sowie die jeweiligen Interfaces zu generieren. Zusätzlich lassen sich auch DDL[16] Files sowie Skripte zur Erstellung von Constraints und Indizes generieren.
Abbildung 3 zeigt einen Ausschnitt aus der Modellierung von Interfaces der Business-Objekte des Forums. Mittels einer Palette (rechts in der Abbildung) hat man Zugriff auf die UML-typischen Funktionen.

[15] Eclipse ist eine auf Java basierende, quelloffene Entwicklungsumgebung (IDE). Weitere Informationen unter http://www.eclipse.org/ (07.09.2009).

[16] Data Definition Language ist eine Datenbeschreibungssprache einer Datenbank.

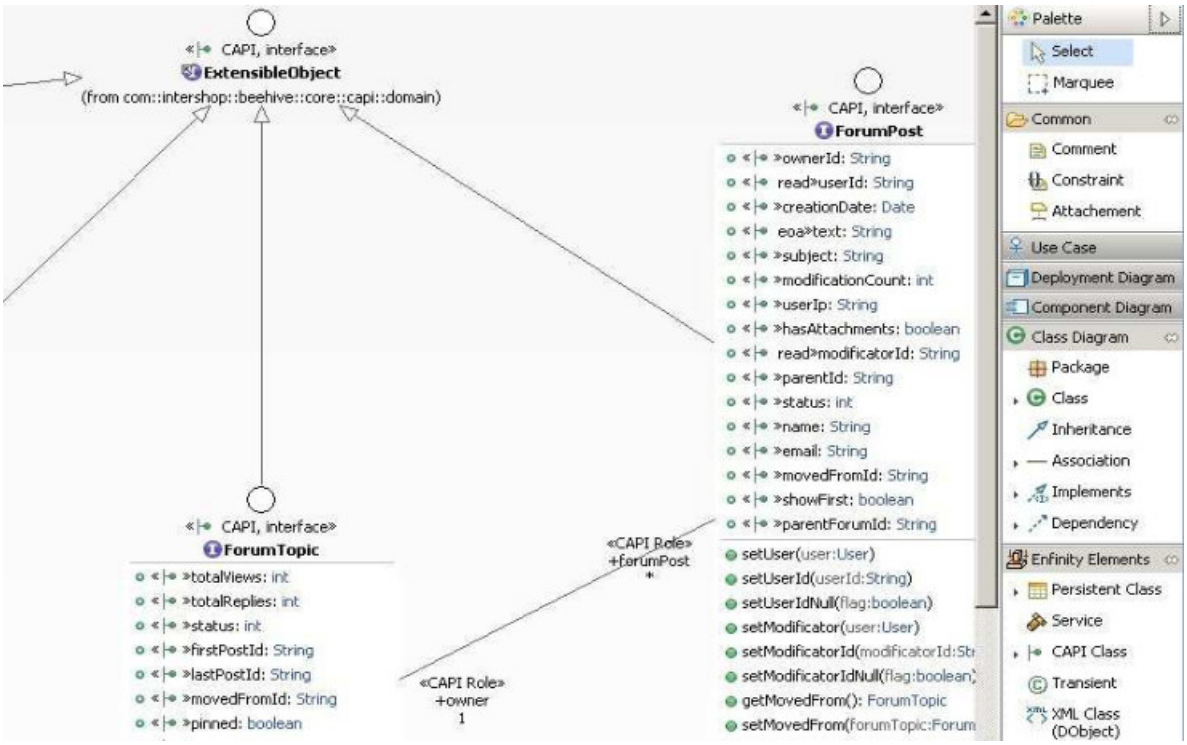

Abbildung 3: UML Editor zur Modellierung von Business-Objekten und Managern[17]

Die generierten Klassenskelette werden anschließend über einen Java Editor an die jeweiligen Anforderungen angepasst.

Die Managerklassen werden ebenfalls mit dem in Abbildung 3 gezeigten Editor modelliert und die dazugehörige Businesslogik anschließend in den generierten Manager-Klassen implementiert. Durch sog. Custom Code Sections werden die Inhalte beim erneuten Generieren des Quellcodes nicht überschrieben. Damit ist es möglich, die Manager um bestimmte Methoden zu erweitern, ohne den kompletten Quellcode neu zu schreiben.

Durch die ebenfalls generierten DDL- und Index-Files werden beim Initialisieren der Cartridge die erforderlichen Tabellen erzeugt und Felder mit definierten Indizes und Fremdschlüsseln versehen.

[17] Eigene Abbildung

```
33 public class ForumPO
34     extends ExtensibleObjectPO
35     implements Forum
36     //{{ parents
37     // insert your implemented interfaces here
38     //}} parents
39 {
40     /*----------------------------------------------------
41                                 Custom method section
42     ----------------------------------------------------*/
43
44     //{{ methods
45     public ForumPost getLastPost()
46     {
47         return getLastPostPO();
48     }
49
50     public void setLastPost(ForumPost forumPost)
51     {
52         setLastPostPO((ForumPostPO) forumPost);
53     }
```

Abbildung 4: Java Editor[18]

Der Zugriff auf Business-Objekte und Manager wird, wie in 2.1.2.2 beschrieben, durch Pipelets realisiert. Diese werden unter Verwendung eines Assistenten erstellt und anschließend mit dem in Abbildung 4 gezeigten Java Editor implementiert. Mit Hilfe des Assistenten wird festgelegt, welche Ein- bzw. Ausgabeparameter ein Pipelet besitzt. Diese Parameter werden im Pipeline Dictionary abgelegt und dienen der Kommunikation zwischen den Pipelets sowie der Flusskontrolle im Businessprozess. Abbildung 5 zeigt das Pipelet *GetForumsByType* und die dazugehörigen Ein- und Ausgabeparameter. Aufgabe des Pipelets ist es, eine Liste von Foren anhand ihres Typs und ihrer Domainzugehörigkeit in der Datenbank zu selektieren und das Ergebnis als Ausgabeparameter zurückzugeben. Der Ausgabeparameter *Forums* wird in diesem Fall mit dem Bezeichner *Categories* überschrieben, da das Pipelet an dieser Stelle zur Selektierung von Foren des Typs Kategorie genutzt wird. Dieser Bezeichner kann anderen Pipelets als Eingabeparameter dienen oder im Presentation Layer zur Darstellung der Kategorien in Templates genutzt werden.

[18] Eigene Abbildung

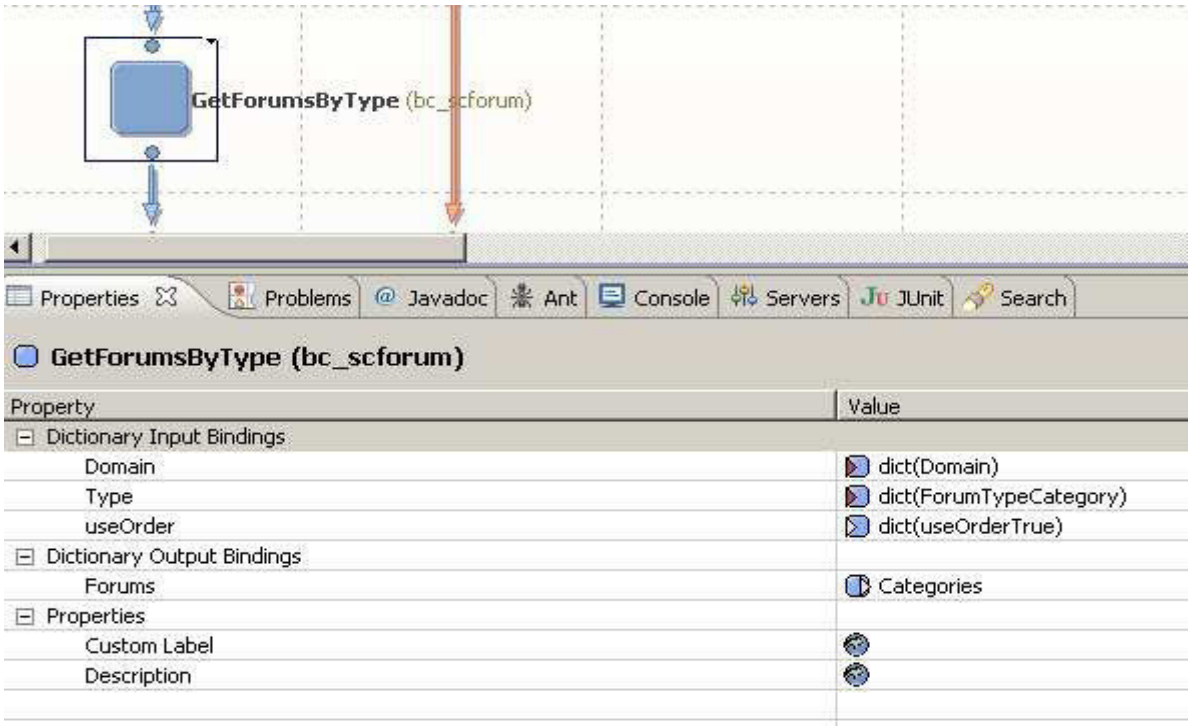

Abbildung 5: Pipelet *GetForumsByType* mit Parametern und Einstellungen[19]

Pipelines verbinden Pipelets zu einem logischen Businessprozess und werden ebenfalls über einen entsprechenden Assistenten erstellt. Anschließend werden sie mit Hilfe des in Abbildung 6 gezeigten Pipeline Editors modelliert. Bei der Erstellung hat der Entwickler unter anderem die Wahl zwischen View- und Prozesspipeline. Jedes Element einer Pipeline wird als Node (engl. Knoten) bezeichnet, dabei muss jede Pipeline mindestens einen Start- und einen Stop- bzw. Exit-Node beinhalten. Nodes werden mit Transitions (engl. Übergänge) verbunden, welche auch als Transaktionssteuerung Verwendung finden. Weiterhin existieren Knotenelemente, welche es dem Entwickler ermöglichen, Schleifen und Entscheidungen zu realisieren. Darüber hinaus besteht die Option, innerhalb der Pipeline andere Pipelines über Call-Nodes aufzurufen. Prozesspipelines werden generell nur innerhalb von Viewpipelines aufgerufen.

[19] Eigene Abbildung

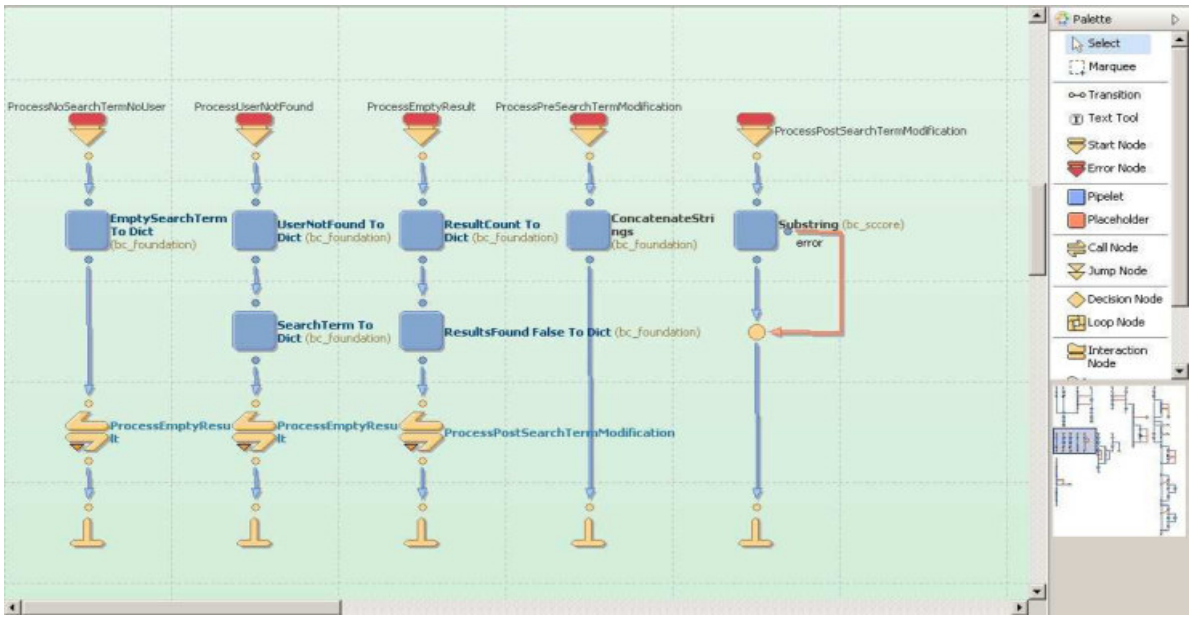

Abbildung 6: Pipeline Editor[20]

2.1.4 Cartridge Konzept

Eine Cartridge bezeichnet ein installierbares Softwaremodul, welches dem Enfinity Suite Application Server eine bestimmte Teilfunktionalität zur Verfügung stellt. Sie bündelt dabei alle Komponenten, die notwendig sind, um diese Funktionalität zu gewährleisten. Im Detail umfasst dies Templates, Java Quellcode (Business-Objekte, Pipelets), Pipelines, Bilder und statische HTML Dateien, Ressource Dateien und eine Konfigurationsdatei.

Beim initialen Einrichten der Datenbank wird durch eine Konfigurationsdatei festgelegt, welche Cartridges in welcher Reihenfolge dem Application Server zur Verfügung gestellt werden. Die Hierarchie definiert, in welcher Abfolge Cartridges nach Ressourcen (z.B. Templates, Pipelets) durchsucht werden („lookup“). Funktionalität aus den vier Implementierungsschichten kann somit durch Änderung der Hierarchie überschrieben werden. Die zuerst gefundene Ressource wird geladen, wobei das Unterscheidungsmerkmal der Bezeichner ist.

[20] Eigene Abbildung

3 Entwurf des Forums

3.1 Anforderungen

Der erste Schritt bei der Entwicklung einer neuen Komponente ist das Ausarbeiten von Anforderungen, sog. Requirements[21]. Diese stellen die Grundlage für das anschließende High Level Design und die Modellierung der Komponente dar. Requirements werden in der dotSource GmbH durch das entsprechende Team erstellt. Anforderungsanalysen, die in Absprache mit dem Management durchgeführt werden, ermöglichen es präzise Vorstellungen hinsichtlich Funktionalität und Erscheinungsbild der Komponente zu entwickeln. Das Requirement enthält dabei keinerlei Vorgaben inwiefern eine Funktionalität technisch umgesetzt werden soll.
Nachdem die Requirements aufgestellt sind, ist es Aufgabe des Entwicklers diese zu studieren und kritische Anmerkungen zu äußern. Erst nachdem alle Unklarheiten beseitigt sind, folgt der nächste Schritt, das High Level Design. Die Bezeichnung der Requirements setzt sich aus dem Kürzel *Sc* für Social Commerce, einem Index für die zu entwickelnde Komponente und einem Buchstaben zur Unterteilung der Gesamtaufgabe in kleinere Teilaufgaben zusammen. Für das Forum existieren drei Requirements: *Sc15A*, *Sc15B* und *Sc15C*.

Sc15A befasst sich mit der Thematik „Darstellen und Suchen“. Im Detail beinhaltet das die Darstellung des Forums für den Nutzer im Frontend des Shops. Jede Seite, welche der Nutzer zu sehen bekommt, ist im Requirement schematisch als sog. Mock-up[22] dargestellt. Dies hat zum Ziel, dem Entwickler eine Vorlage zu geben wie die Elemente einer bestimmten Seite anzuordnen sind. Teilweise enthalten Mock-ups (siehe Abbildung 7) auch Randnotizen in Form von Sprechblasen, welche ebenfalls Informationen über das referenzierte Element enthalten.

---

[21] Matthias Maak widmet sich in seiner Praxisarbeit „Softwareentwicklungsprozess der dotSource GmbH“ ausführlich dem Thema Requirement.

[22] Mook-Ups stellen in der Softwareentwicklung Prototypen einer Benutzeroberfläche dar um Anforderungen besser ermitteln zu können.

Desweiteren ist im Requirement jeder Prozess detailliert dokumentiert, der durch das Aktivieren eines bestimmten Elementes (bspw. einer Schaltfläche oder einer Verknüpfung) ausgelöst wird.

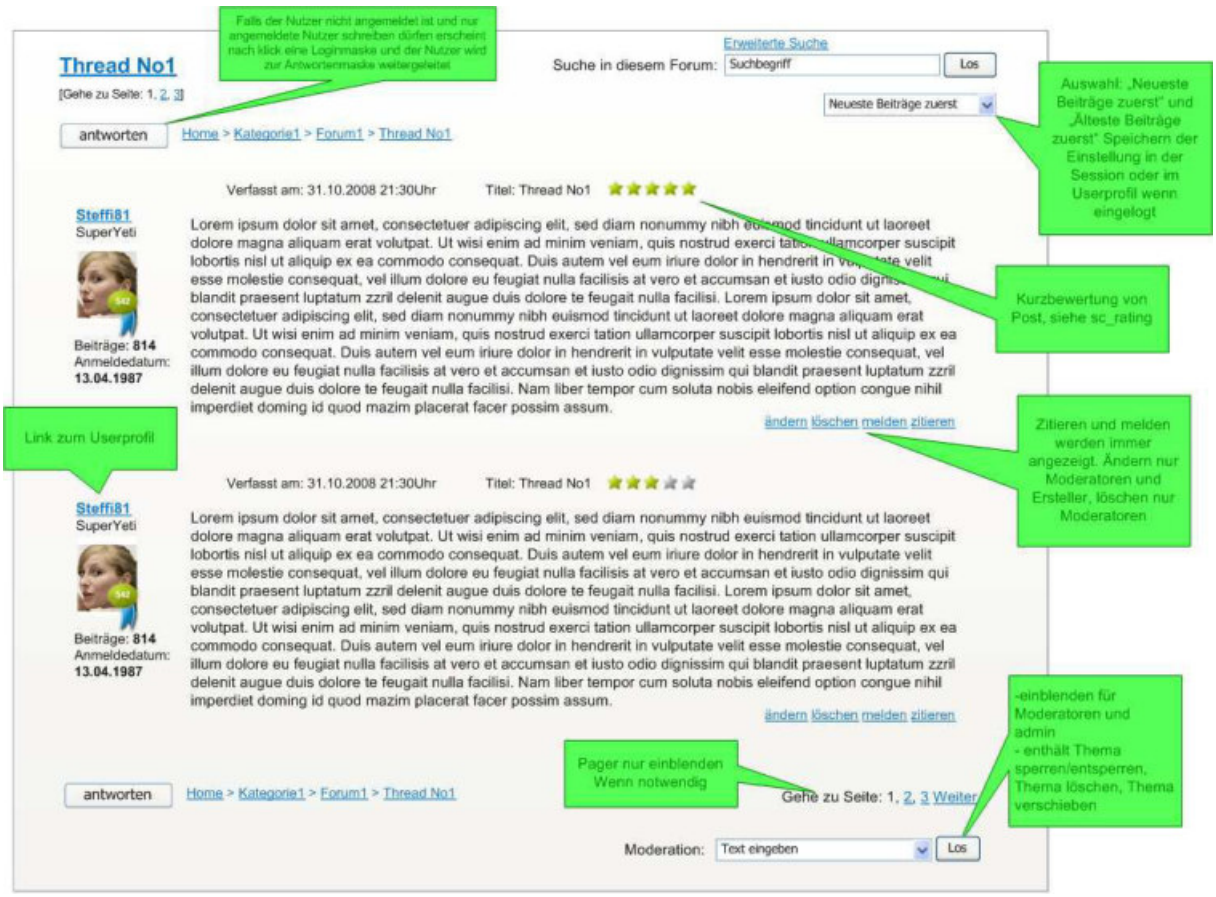

Abbildung 7: Mock-Up aus Sc15A inkl. Anmerkungen[23]

Neben der Darstellung im Frontend gehört auch das Thema „Suchen" zum Requirement *Sc15A*. Dieses beschreibt die Funktionalität, das gesamte Forum nach bestimmten Schlagwörtern zu durchsuchen. In einer erweiterten Suche hat der Nutzer die Möglichkeit Suchergebnisse durch definierte Parameter zu filtern.

Das Requirement *Sc15B* definiert die Funktionalität „Erstellen von Beiträgen". Hierbei handelt es sich um ein weniger umfangreiches Requirement, welches sich ausschließlich auf die Masken zur Erstellung neuer Themen und Beiträge beschränkt. Auch hier wurden Mock-ups verwendet, um dem Entwickler Vorgaben

[23] Vgl. [http://wiki.dotsource.de/pub/SCOOB/Sc15A/Threadansicht.png]

hinsichtlich der Anordnung der Elemente und deren Funktionalität zu geben. Als Besonderheit sei an dieser Stelle die Nutzung des BB-Codes[24] zu erwähnen. Dieser dient der Formatierung von Nutzerbeiträgen.

Das Requirement *Sc15C* („Backend“) beschäftigt sich mit der Administration des Forums. Dem Shopbetreiber wird hier die Möglichkeit eingeräumt, Basiseinstellungen für die gesamte Komponente vorzunehmen. Hierzu zählen zum Beispiel die Möglichkeit nicht registrierte Nutzer vom Forum auszuschließen, das Forum auf Grund von Wartung zu deaktiveren oder die automatische Archivierungsfunktion zu steuern.
Zusätzlich zu den Basiseinstellungen kann der Shopbetreiber Foren anlegen oder löschen, sie bearbeiten, ihre Hierarchie ändern und ihnen Moderatoren zuweisen.
Das Requirement sieht hierbei vor, welche Einstellungen getroffen werden können und wie die entsprechenden Eingabemasken dargestellt werden.

3.2 High Level Design

Im Requirement wurde festgelegt, welche Prozesse durch Interaktion des Nutzers mit bestimmten Elementen ausgelöst werden. Diese Prozesse werden im High Level Design (HLD) in Form von Ablaufmodellen visualisiert. Das Erstellen des HLD dient dem Entwickler dazu, sich ein klares Bild über die Funktionalität zu machen. Spätestens an dieser Stelle des Entwurfs wird deutlich, ob die Aufgabe verstanden wurde oder ob noch Rücksprachen mit dem Requirement Team erforderlich sind.
Ein weiterer Vorteil des HLD ist die Überprüfung des Requirements. Treten beim Modellieren der Prozesse Unklarheiten auf, so können diese bereits während der Entwurfsphase diskutiert und eventuell auftretende Probleme im Vorfeld vermieden werden.
Ein dritter entscheidender Vorteil des HLD ist der Dokumentationscharakter. Wird eine Komponente an einen anderen Entwickler übergeben oder findet eine

[24] Bulletin Board Code ist eine an HTML angelehnte Auszeichnungssprache, mit der es möglich ist Nachrichten zu formatieren. Weiter Informationen unter http://www.vbulletin-germany.com/forum/misc.php?do=bbcode (10.09.09)

Weiterentwicklung der Komponente statt, so stellt das HLD eine Übersicht über die vorhandene Implementierung der Prozesse dar.

Das High Level Design wird von jenem Entwickler erstellt, welcher für die Umsetzung der Anforderung zuständig ist. Für eventuell auftretende Nachfragen ist der Projektleiter erster Ansprechpartner. Erst wenn dieser das HLD abgesegnet hat, wird die technische Spezifikation erstellt und eine Cartridge Struktur für die neue Komponente entworfen.

High Level Designs werden bei der dotSource GmbH mit der Software Microsoft Visio angefertigt. Dem Entwickler stehen mehrere Symbole zur Verfügung, anhand derer er die Prozesse modellieren kann.

Element	Symbol	Beschreibung
Aktion		Ein durch Nutzer ausgelöster Prozess (z.B. Klicken eines Links)
Interaktion		Eine HTML-Seite, auf die ein Nutzer geleitet wird
Ajax Interaktion		Asynchrones Nachladen eines Bereiches der Seite
Mock-Up		Ein Verweis auf das entsprechende Mock-Up im Requirement
Fallunterscheidung		Mehrfachauswahl

Funktion		Eine elementare Systemfunktion
Prozess		Ein Prozess, welcher meist ein eigenes Requirement, auf jeden Fall aber ein eigenes HLD hat
Daten		Daten, welche bei diesem Schritt benötigt oder erzeugt werden
Pipeline		Verweis, welche Pipeline durch die referenzierte Aktion ausgeführt wird
Template		Template, welches als Basis für den Response genutzt wird

Tabelle 1: Elemente des High Level Designs[25]

Werden Formulare bei der Umsetzung der Anforderung benötigt, so beinhaltet das HLD zu jedem Formular eine Datentabelle (Abbildung 8) in der alle Felder, der jeweilige Datentyp, ein Required Flag und ein Feld für Bemerkungen vorhanden sind.

Fieldname	Type	Required	Miscellaneous
Attachment	Upload		2MB Max File Size
Forum ID	Text		The UUID of the parent Forum (hidden)
Pinned	Checkbox		Topic is pinned („Important" Topic)
ShowFirst	Checkbox		Always first post, independent from order
Subject	Text	x	The Subject of the Topic (limited by forum pref)
Text	Text	x	The Content of the Topic (limited by forum pref)

Abbildung 8: Datentabelle des Formulars zur Erstellung eines neuen Themas[26]

[25] Vgl. [DS_TW], Unterseite: http://wiki.dotsource.de/bin/view/Main/StyleGuideHld

[26] Eigene Abbildung

Stellvertretend für alle Prozesse wird im Folgenden ein Ausschnitt aus dem Requirement *Sc15B* „Erstellen von Beiträgen“ detailliert erläutert. Der in Abbildung 9 dargestellte Prozess beschreibt das Antworten auf ein bestehendes Thema im Forum. Dem Nutzer stehen hier mehrere Möglichkeiten der Interaktion zur Verfügung, welche durch das HLD abgedeckt werden müssen. Der Ausschnitt soll dem Leser einen Einblick vermitteln, wie einzelne Elemente eines HLD miteinander verbunden werden.

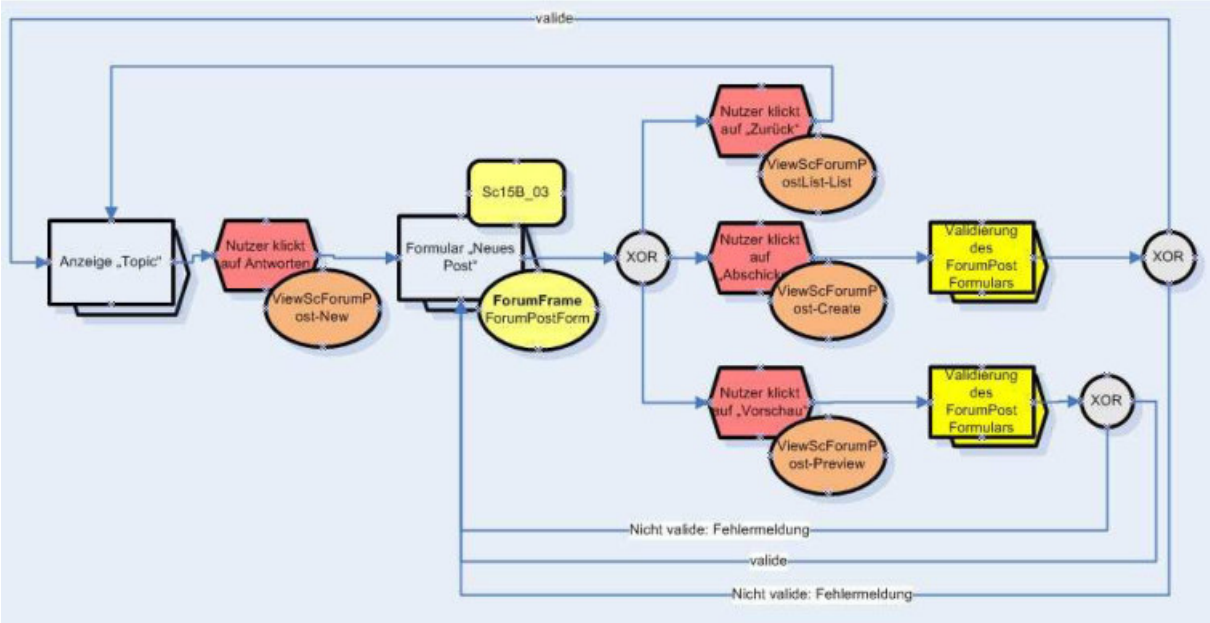

Abbildung 9: Auszug HLD Sc15A für das Erstellen eines Beitrags[27]

Ausgangspunkt des Prozesses ist die Anzeige eines Topic (engl. Thema). Der Nutzer sieht eine bestimmte Anzahl von Posts (engl. Beiträge) zu diesem Thema und hat die Möglichkeit über die Schaltfläche „Antworten“ einen eigenen Beitrag zu verfassen. Die Aktion des Nutzers ist demzufolge das Klicken dieser Schaltfläche. Dadurch wird die Pipeline *ViewScForumPost-New* gestartet. Diese Information befindet sich rechts unten am Aktions-Element. Das Ergebnis dieser Pipeline ist die Weiterleitung des Nutzers auf jene Seite, auf der sich das Formular zum Erstellen eines „Neuen Posts“ befindet. Das Design dieses Formulars orientiert sich dabei am Mock-Up *Sc15B_03*. Das Template, welches durch das Anfordern des Formulars

[27] Eigene Abbildung

geladen wird, trägt die Bezeichnung *ForumPostForm*. Diese beiden Informationen werden an das Interaktions-Element angehangen.
Der Nutzer hat nun die Möglichkeit drei verschiedene Schaltflächen zu betätigen. Die Schaltfläche „Zurück“ stößt die Pipeline *ViewScForumPostList-List* an, welche die Ausgangssituation als Resultat anzeigt. Die Schaltfläche „Abschicken“ startet die Pipeline *ViewScForumPost-Create*, welche einen neuen Beitrag erzeugt, nachdem die Formulardaten validiert wurden. Sind die Daten nicht valide, so wird das Formular mit entsprechenden Fehlermeldungen erneut geladen. Die dritte Option, die dem Nutzer an dieser Stelle zur Verfügung steht, ist die Anzeige einer Vorschau. Betätigt er die Schaltfläche „Vorschau“, so werden ebenfalls die Formulardaten validiert, jedoch wird sein Beitrag nicht persistiert. In einem entsprechenden Template wird dem Nutzer lediglich präsentiert, wie der Beitrag nach dem Speichern aussehen wird. Er hat an dieser Stelle die Wahl den Beitrag abzuspeichern oder zum Formular zurückzukehren, um weitere Anpassungen vorzunehmen.

Das High Level Design sieht für jeden Prozess ein entsprechendes Ablaufdiagramm vor. Während der Entwicklung werden diese Diagramme oft ergänzt, da sich der ganze Umfang des Prozesses nicht von Beginn an erfassen lässt. Weiterhin kommt es vor, dass Requirements nachträglich geändert werden. In solchen Fällen muss ebenfalls das HLD angepasst werden.

3.3 Technische Spezifikation

Nachdem das High Level Design vom Projektleiter abgesegnet wurde, erfolgt der Entwurf der Cartridge Struktur.

Aufgrund der Tatsache, dass das Forum mit persistenten Objekten operieren muss, ist es notwendig, eine Cartridge für die gesamte Business Funktionalität zu erstellen. Weiterhin ist das Forum von Endnutzern über das Shop-Frontend, sowie vom Shopbetreiber über das Shop-Backend, verwendbar. Aus diesem Umstand resultiert die Verwendung von zwei getrennten Cartridges zur Verwaltung der Front- sowie der Backendfunktionalität. Weiterhin existieren Features, die sowohl im Frontend als auch im Backend genutzt werden können. Die beiden Cartridges müssen

demzufolge auf eine gemeinsame Basis zugreifen, um diese Schnittmenge an Funktionalität nicht duplizieren zu müssen.
Die Forumskomponente besteht folglich aus vier Cartridges, welche in Abbildung 10 hierarchisch aufgelistet sind. Die Abhängigkeit untereinander entspricht dabei der Pfeilrichtung. Die Cartridges des Application Layer greifen auf Funktionalität des Business Component Layer zu, jedoch nicht umgekehrt.

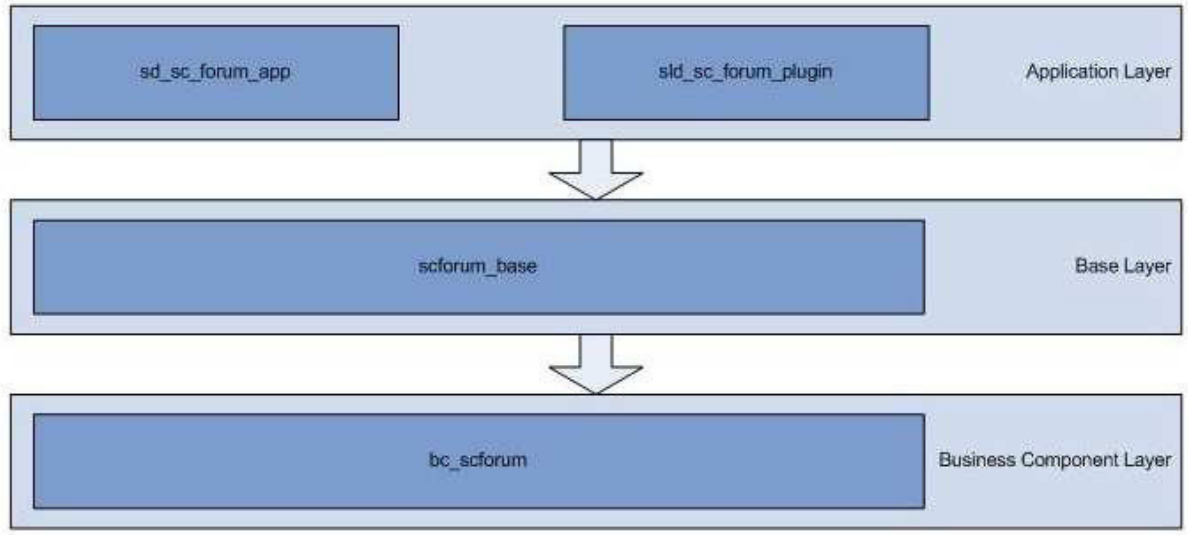

Abbildung 10: Cartridges der Forumskomponente[28]

Die grundlegende Cartridge ist *bc_scforum*. Sie beinhaltet sämtliche Business-Funktionalität der Forumskomponente. Dies umfasst die Java Klassen für Business-Objekte und Manager, Pipelets zum Verwalten von Business-Objekten, sowie Prozesspipelines, welche diese Pipelets verwenden, um Businessfunktionalität nach außen anzubieten. Weiterhin beinhaltet *bc_scforum* das Datenbankschema der Komponente, sowie die dazugehörigen Constraints und Indizes. Die Cartridge enthält weder allgemeine noch shopspezifische Templates. Sie ist für den Einsatz in mehreren Projekten konzipiert und wurde unter dem Aspekt der Wiederverwendbarkeit entworfen.

In der aufsetzenden Schicht *scforum_base* befindet sich Funktionalität, welche von beiden Cartridges des Application Layer genutzt wird. Base Cartridges werden seit

[28] Eigene Abbildung

Enfinity Suite 6.2 eingesetzt um das Kopieren identischer Ressourcen in mehrere Cartridges zu vermeiden[29]. Im Fall des Forums enthält sie Pipelines und Templates zur Verwaltung von Dateianhängen eines Forumsbeitrags.

In der obersten Schicht, dem Application Layer, wurden zwei Cartridges erstellt, um die verschiedenen Nutzeransichten für das Forum zu implementieren. Die Cartridge *sld_sc_forum_app* stellt dabei das Forum im Storefront für den Nutzer dar. Demgegenüber beinhaltet *sld_sc_forum_plugin* alle Bestandteile für die Darstellung der Forumsfunktionalität im Backend, sowohl für den Administrator, als auch für den Shopbetreiber.
Application Layer Cartridges bündeln Pipelines, Templates und statischen Content, selten Pipelets und niemals Business-Objekte[30]. Diese Vorgabe wurde auch beim Forum umgesetzt. Die Cartridges beinhalten ausschließlich Viewpipelines und Templates. Sie nutzen den Funktionsumfang, welcher durch die Komponenten der Business Component Schicht und der Plattform Schicht[31] bereitgestellt wird.

3.4 Datenhaltung

Nach Festlegung der Cartridges und deren Hierarchie wurde im nächsten Schritt das Modell der Business-Objekte erstellt. Hierbei war es zunächst erforderlich, die Business-Objekte zu definieren und deren Abhängigkeitsverhältnis festzulegen.
Das Forum sollte sich strukturell an einem PHPBB-Forum[32] orientieren. Anhand dieser Zielstellung wurden zunächst drei Business-Objekte entworfen: Forum, ForumTopic (Thema) und ForumPost (Beitrag). Zusätzlich wurde noch das Objekt ForumModeratorAssignment eingeführt, welches die Zuordnung von Nutzern (Moderatoren) zu einem Forum ermöglicht.

---

[29] Vgl. [IS_APG_07], S.32

[30] Vgl. [IS_APG_07], S.32

[31] Die Plattform Schichten wurden von Intershop entwickelt und bieten Systemfunktionalität in Form von Pipelets und Pipelines an. Ein Beispiel dafür ist die Cartridge *core*.

[32] PHPBB ist eine freie und kostenlose Forensoftware. Weitere Informationen unter http://www.phpbb.de/ (2009-09-14).

Ein Forum kann als Kategorie deklariert werden. Kategorien stehen an oberster Stelle in der Hierarchie und beinhalten ausschließlich Foren. Diese können wiederum Unterforen oder ForumTopics besitzen, welche ausschließlich ForumPosts beinhalten. Jedes Objekt kennt jeweils sein übergeordnetes Element (Owner). Daraus ergibt sich die in Abbildung 11 gezeigte Baumstruktur.

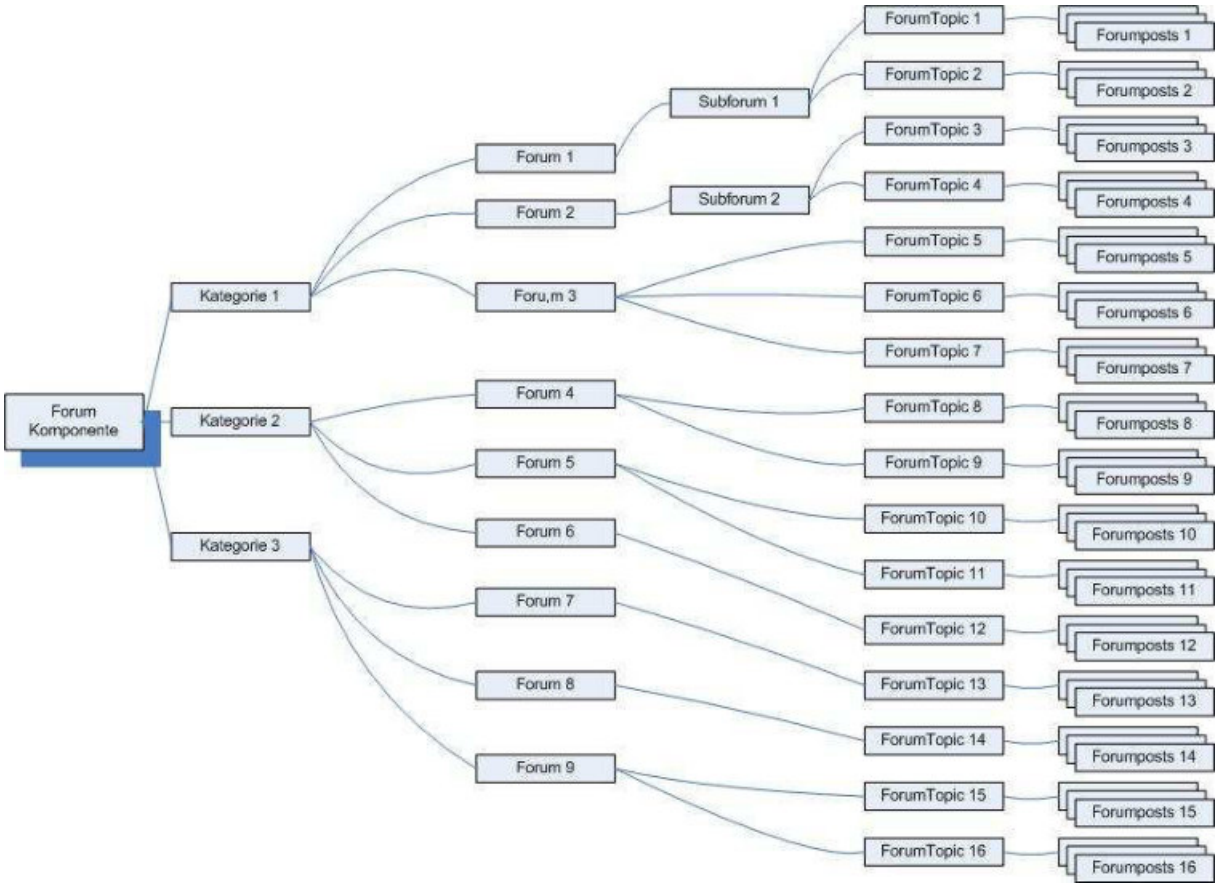

Abbildung 11: Baumstruktur der Forumskomponente[33]

Aus der Baumstruktur ergeben sich die in Tabelle 2 abgebildeten Kardinalitäten für die Beziehungen zwischen Objekten.

Beziehung	Kardinalität
Forum (Kategorie) – Forum	1:n
Forum – ForumTopic	1:n
ForumTopic – ForumPost	1:n

[33] Eigene Abbildung

Forum – ForumModeratorAssignment	1:n
User – ForumModeratorAssignment	1:n
Forum – User	n:m

Tabelle 2: Kardinalitäten zwischen Forumsobjekten

In Abbildung 12 ist das ERM der Forumskomponente dargestellt. Im Folgenden wird auf Attribute der einzelnen Objekte eingegangen. Hervorgehobene Attribute stellen dabei Pflichtattribute dar, die zur Erzeugung eines Objektes benötigt werden.

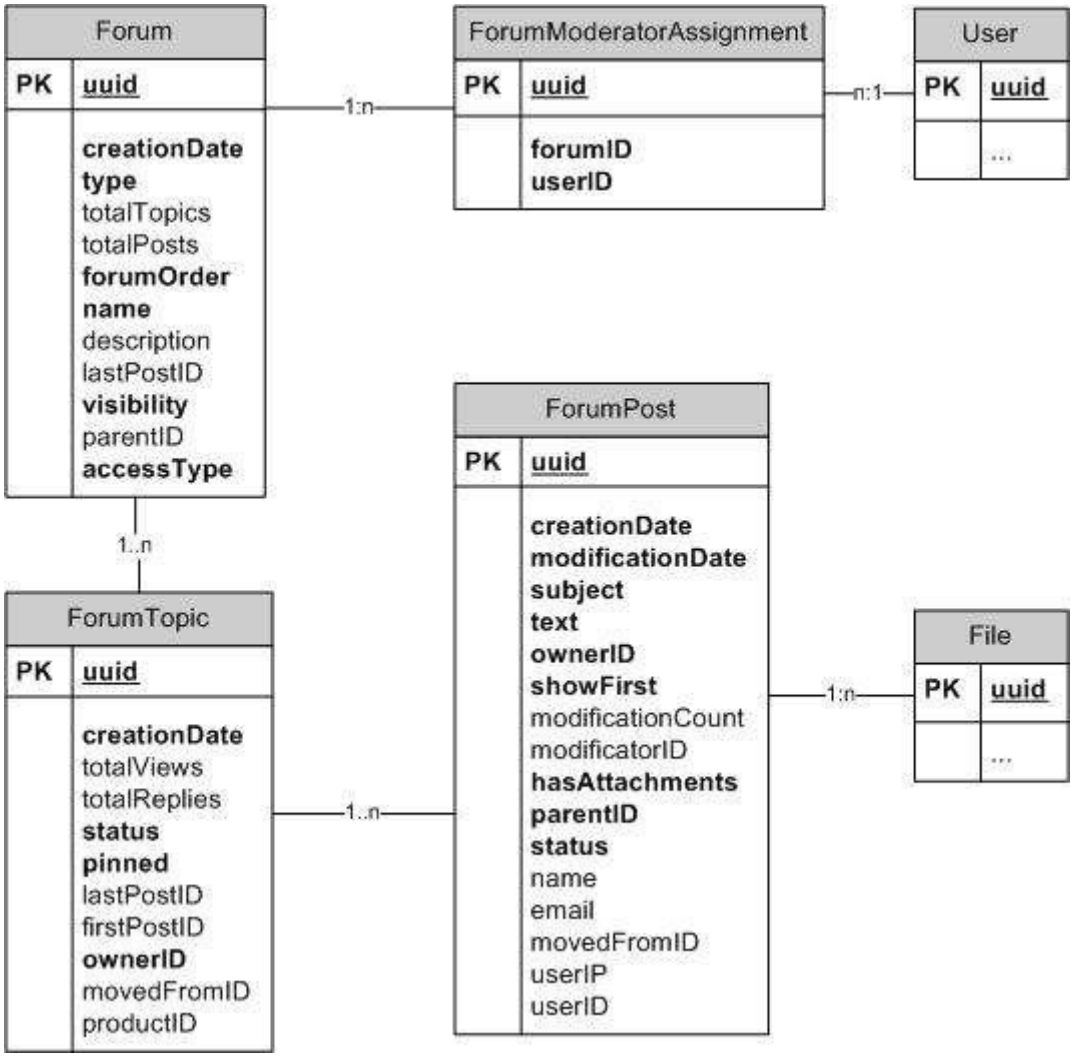

Abbildung 12: Enitity Relation Model (ERM) Forum[34]

[34] Eigene Abbildung

Jedes Objekt besitzt standardmäßig vier Attribute, welche von Intershop durch das Objekt *ExtensibleObjectPO* vorgegeben werden. Alle persistenten Objekte erben von dieser Klasse und erhalten die nachfolgend aufgeführten Attribute.

Die *uuid* (Universally Unique Identifier) stellt den Primärschlüssel für die Entitäten dar und identifiziert diese. Weiterhin existiert das Attribut *lastModified*, welches den Zeitpunkt der letzten Änderung des Objektes repräsentiert. Darüber hinaus werden die Attribute *domainID* und *ocaID* vorgegeben, welche im Modell nicht aufgelistet sind, aber zur internen Zuordnung innerhalb der Enfinity Suite genutzt werden.

Das Forum besitzt zusätzlich das Attribute *type*, welches eine Unterscheidung zwischen Forum und Kategorie zulässt. Der Datentyp dieses Attributes ist Integer und dieses Feld ist nicht optional.
Die Zählvariablen *totalPosts* und *totalTopics* stehen für die Anzahl der im jeweiligen Forum enthaltenen Beiträge bzw. Themen. Dies dient der Anzeige im Frontend und erspart zeitintensive count – Queries beim Anzeigen der Forenübersicht. Der Datentyp beider Felder ist Integer.
Das Integer-Attribut *forumOrder* wird genutzt um Hierarchien von Kategorien und Foren zu erstellen.
Das Pflichtattribut *name* stellt die Bezeichnung eines Forums dar und ist vom Typ Varchar.
Das optionale Attribut *description* kann genutzt werden um eine Beschreibung des Forums abzulegen, die ihrerseits auf Wunsch im Frontend dargestellt werden kann.
Das Feld *lastPostID* referenziert auf einen ForumPost und dient, der im Requirement geforderten, Darstellung des „letzten Beitrags“. Ebenso wie *uuid* ist dieses Attribut vom Typ Varchar und dient der Einsparung von Queries und der damit verbundenen Performancesteigerung.
Das Feld *visibility* gibt dem Shopbetreiber die Möglichkeit Einstellungen bzgl. der Sichtbarkeit von Foren für ausgewählte Nutzergruppen festzulegen. Das Feld des Typs Integer ist nicht optional.
Um eine Baumstruktur von Foren zu erzeugen, wird das Feld *parentID* vom Typ Varchar genutzt. Es verweist auf ein Forum und ermöglicht eine beliebig tiefe Schachtelung.

Ein weiteres Attribut ist der *accessType.* Dieser legt fest, welche Nutzer im Forum schreiben dürfen. Es ist vom Typ Integer und ebenfalls optional.

Ein ForumTopic ist ein Thema innerhalb eines Forums. Neben den Standard-Attributen enthält es die optionalen Zählvariablen *totalPosts* und *totalViews* vom Typ Integer, welche die Anzahl der im ForumTopic enthaltenen Beiträge bzw. die Anzahl der Klicks auf das ForumTopic speichern.
Das *status*-Attribut ist ein Pflichtfeld und gibt an, ob das Thema gesperrt ist. Ist dies der Fall, so können keine neuen Beiträge mehr zu diesem Thema geschrieben werden. Der Datentyp dieses Feldes ist Integer.
Ein weiteres Pflichtfeld ist *pinned*, welches das Thema als „wichtig“ markiert und dementsprechend in einer Auflistung an vorderster Stelle erscheinen lässt. Das Attribut ist vom Typ Integer.
Die Referenzen *lastPostID* und *firstPostID* sind wie *uuid* vom Typ Varchar und verweisen auf ForumPosts. Der „First Post“ ist jener Beitrag, der gleichzeitig mit dem Erstellen des Themas angelegt wird und dieses repräsentiert. Der „Last Post“ dient der Erfüllung der Anforderung, dass aus einer Themenübersicht heraus direkt auf den letzten Beitrag zugegriffen werden kann.
Das Pflichtfeld *ownerID* ist vom Typ Varchar und legt fest, welchem Forum das Thema zugeordnet ist.
Das Attribut *movedFromID* verweist ebenfalls auf ein Forum und wurde mit der Zielsetzung erstellt, dass Themen gelöscht, aber auch wiederhergestellt werden können.
Die *productID* ist vom gleichen Typ wie *movedFromID*, verweist jedoch auf ein Produkt, so dass spezielle Produktforen möglich werden. Dieses Feld ist optional.

Der ForumPost ist das umfangreichste der Objekte, da viele Nutzerinformationen abgelegt werden müssen. Neben den Standard-Attributen besitzt es die Pflichtfelder *subject* (Betreff) und *text* (Inhalt), welche den Inhalt des Beitrages referieren.

Erstgenanntes ist vom Typ Varchar, letzteres vom Typ Text. Beide Felder wurden später mit Oracle Text[35] indiziert um die geforderte Volltextsuche zu gewährleisten.
Das Pflichtfeld *ownerID* referenziert auf das übergeordnete ForumTopic und ist vom Typ Varchar.
ShowFirst ist vom Typ Boolean und wird für die Festlegung genutzt, einen Beitrag in einer Liste unabhängig von der Sortierung zu verorten.
Das Integer-Attribut *modificationCount* dient als Nachweis der Häufigkeit von Beitragsänderungen. Um festzustellen, welcher Nutzer den Beitrag zuletzt editiert hat, wurde das Attribut *modificatorID* als Referenz eingeführt.
Das optionale Feld *hasAttachments* gibt an, ob der Beitrag Dateianhänge vom Typ File besitzt.
Um in einer Weiterentwicklung des Forums eine Baumstruktur im Frontend anzeigen zu können, wurde das Attribut *parentID* vom Typ Varchar eingeführt. Es verweist auf einen ForumPost und dient der Darstellung von Antworten auf einen Beitrag.
Ein Beitrag kann entweder im Frontend zu sehen sein oder er befindet sich in der Verwaltung und wartet auf eine Freigabe durch den Shopbetreiber. Dieser Zustand wird durch das Integer-Attribut *status* realisiert.
Wird der Beitrag von einem registrierten Nutzer geschrieben, so wird seine *uuid* im Feld *userID* hinterlegt. Für anonyme Nutzer werden die Felder *name* und *email* vom Typ Varchar verwendet. In beiden Fällen wird die IP des Nutzers im Attribut *userIP* gespeichert, um im Falle einer Notwendigkeit Rückschlüsse auf den Urheber zu ermöglichen.
ForumPosts können innerhalb von Themen verschoben werden. Um dem Moderator die Option zu geben diesen Vorgang rückgängig zu machen, wurde das Feld *movedFromID* vom Typ Varchar eingeführt, in dem die *uuid* des originalen ForumTopics hinterlegt wird.

Ist das ERM erstellt und vom Projektleiter bewilligt, wird die Struktur in Enfinity Studio unter Verwendung des UML2 Editors abgebildet. Hierbei werden zunächst die Interfaces für Business-Objekte und Manager modelliert, welche Basisfunktionalität

[35] Oracle Text ist der in Oracle 10g integrierte Standard zur Erstellung von Volltextindizes. Es ermöglicht eine effiziente und detaillierte Suche in Texten und Dokumenten. Weiter Informationen unter http://www.oracle.com/technology/products/text/index.html (2009-09-14).

von der Schnittstelle *ExtensibleObject* erben. Abbildung 13 zeigt die Objektschnittstellen und ihre Relationen zueinander. Die Interfaces besitzen Attribute gemäß dem ERM, ihre Datentypen sind Java Datentypen. Die später benötigten Getter- und Setter-Methoden werden an dieser Stelle ebenfalls angelegt.
Über die Interfaces erfolgt der Zugriff auf die persistenten Objekte in der Datenbank. Attribute und Methoden werden über das Kontextmenü angelegt, zur Modellierung von Relationen stehen in einer Palette alle benötigten Werkzeuge bereit.

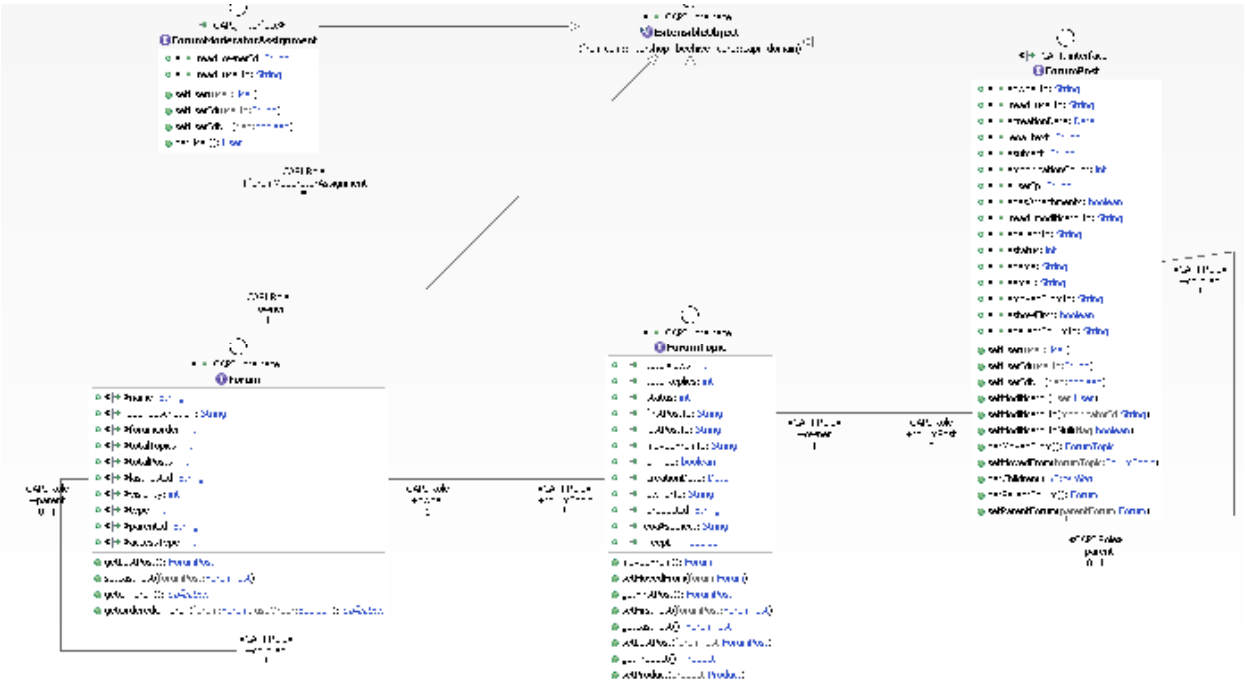

Abbildung 13: Schnittstellen der Business-Objekte[36]

Auch die Manager werden nach diesem Schema modelliert. Im Unterschied zu den Business-Objekten erben sie von der Schnittstelle *Manager*.
Manager besitzen keine Attribute, sondern nur Methoden, welche die Business Logik zur Verfügung stellen. Da die Kommunikation mit Managern nur über deren Schnittstellen erfolgt, müssen die Methoden im Modell angegeben werden. Besteht zu einem späteren Zeitpunkt die Notwendigkeit einer Ergänzung, so ist es jederzeit möglich das Modell zu erweitern und den Java Quellcode neu generieren zu lassen.

[36] Eigene Abbildung

Nach den Schnittstellen werden die persistenten Objekte modelliert, welche die Entitäten in der Datenbank abbilden. Jedes persistente Objekt implementiert sein zugehöriges Interface und erbt von *ExtensibleObjectPO*. Im Gegensatz zur Modellierung der Schnittstellen werden an dieser Stelle SQL Datentypen für die Attribute verwendet. Weiterhin werden alle Fremdschlüsselbeziehungen schematisch dargestellt. Aus diesem Modell werden, wie bereits angesprochen, DDL-Files sowie Skripte zur Erzeugung der Indizes und Constraints generiert. In Abbildung 14 ist das komplette Modell abgebildet. Auch die Relationen zu Intershop Objekten wie Product oder User werden dargestellt.

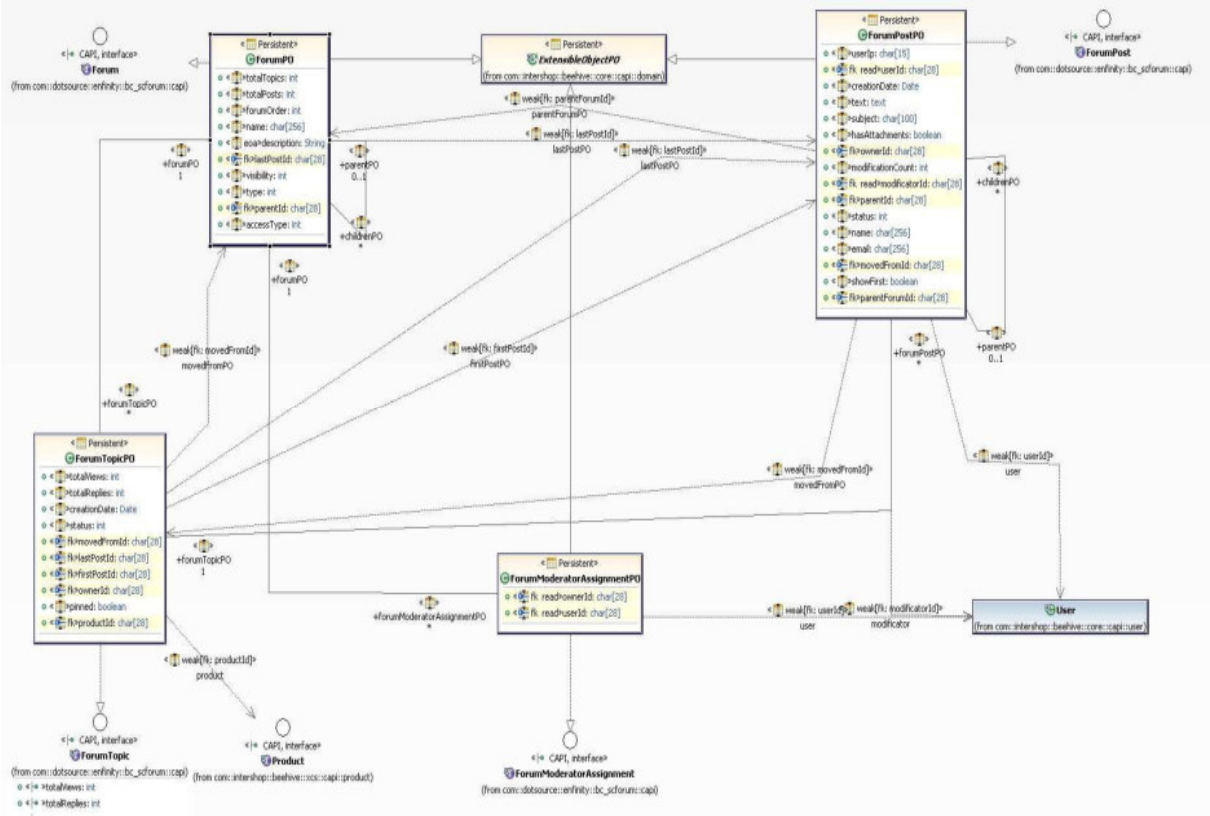

Abbildung 14: ERM der persistenten Objekte[37]

Nach Abschluss der Modellierung wird über das Kontextmenü des UML Editors die Generierung sowohl des Quellcodese als auch der Datenbank-Dateien gestartet.
Die generierten Elemente dienen als Basis für den nächsten Schritt: die Implementierung.

[37] Eigene Abbildung

4 Implementierung des Forums

Die Implementierung ist der umfangreichste Abschnitt der Entwicklung einer Komponente mit Enfinity Studio. Jede Funktionalität, die durch das Requirement vorgegeben wurde, muss mittels Pipelets, Pipelines und Templates umgesetzt werden. Im Fall des Forums nahm die Implementierung etwa drei Viertel der gesamten Entwicklungszeit in Anspruch. Dies begründet sich vorrangig im Funktionalitätsumfang des Forums.

Stellvertretend für alle Prozesse wird auf den folgenden Seiten der Prozess „Darstellung des Forums“ in der Absicht vorgestellt, einen Eindruck des Zusammenwirkens der einzelnen Implementierungsschichten zu vermitteln.

Das Ziel des repräsentativ erläuterten Prozesses ist es, dem Nutzer eine Auflistung aller Kategorien und enthaltener Foren zu präsentieren.

4.1 Manager

Um den Prozess umsetzen zu können, ist es erforderlich durch den *ForumManager* eine Methode bereitzustellen, die es ermöglicht, Foren nach Typ und Domainzugehörigkeit zu selektieren. Die entsprechende Methode wird als *getForumsByType()* bezeichnet, sie wurde im Schnittstellenmodell der Manager deklariert. Im ersten Schritt muss diese Methode implementiert werden. Die Umsetzung findet in der Klasse *ForumManagerImpl* statt, welche die Schnittstelle implementiert.

```
@SuppressWarnings("unchecked")
public Collection<Forum> getForumsByType( int type, boolean useOrder, Domain domain )
{
    if( type < 0 )
    {
        throw new IllegalArgumentException( "Wrong Type to get Forums." );
    }
    String order = " order by forumOrder asc";
    return forumFactory.getObjectsBySQLWhere( "type = ? AND domainid = ?" +
                    (useOrder ? order : "") + "" ,
                    new String[]{ Integer.valueOf(type).toString(),
                    domain.getUUID() });
}
```

Abbildung 15: Quellcode *getForumsByType()*[38]

In Abbildung 15 ist die implementierte Methode dargestellt. Da für den Parameter *type* keine negativen Werte zugelassen sind, wird dies zu Beginn geprüft und im Fehlerfall eine Exception (engl. Ausnahme) ausgelöst. Die Variable *order* ist vom Typ String und Teil der späteren Datenbankabfrage. Die Boolean-Variable *useOrder* aus der Methodensignatur entscheidet über die Verwendung einer Sortierung. Der Vergleich wird durch den ternären Operator durchgeführt. Die ebenfalls generierte ForumFactory stellt Methoden für den Datenbankzugriff zur Verfügung. In diesem Fall wird die Methode *getObjectsBySQLWhere()* aufgerufen, da die Datenbankabfrage benutzerdefiniert ist und sich aus Forentyp, eventueller Sortierung und der DomainUUID[39] zusammensetzt. Das Ergebnis wird in Form einer generischen Kollektion zurückgegeben.

4.2 Pipelets

Im nächsten Schritt wird ein Pipelet benötigt, welches die im Manager implementierte Funktionalität nutzt und das Resultat der Abfrage in einer Pipeline zur Verfügung stellt. Das Pipelet trägt den Namen *GetForumsByType* und setzt als Eingangsparameter den Typ (Kategorie oder Forum) vom Datentyp Integer, die Domain und eine Boolean Variable zur Aktivierung der Sortierung voraus.

[38] Eigene Abbildung

[39] Die DomainUUID dient der Zuordnung des Objektes zur entsprechenden Site in der Enfinity Suite.

```
public int execute(PipelineDictionary dict)
throws PipeletExecutionException {
    // lookup 'useOrder' in pipeline dictionary
    Boolean useOrder = (Boolean)dict.get(DN_USE_ORDER);

    // lookup 'Type' in pipeline dictionary
    Integer type = (Integer)dict.get(DN_TYPE);
    if (null == type)
    {
        throw new PipeletExecutionException("Mandatory input parameter 'Type' not available in pipeline dictionary.");

    }

    //lookup 'Domain' in pipeline dictionary
    Domain domain = (Domain)dict.get(DN_DOMAIN);
    if(null == domain)
    {
        throw new PipeletExecutionException("Mandatory input parameter 'Domain' not available in pipeline dictionary.");
    }

    @SuppressWarnings("unchecked")
    Collection<Forum> forums = forumMgr.getForumsByType( type, useOrder == null ? false : useOrder, domain );

    dict.put(DN_FORUMS, forums);
    return PIPELET_NEXT;
}
```

Abbildung 16: *execute()* Methode des Pipelets GetForumsByType[40]

Jedes Pipelet besitzt eine Methode *execute()*, welche beim Aufruf des Pipelets abgearbeitet wird. Zu Beginn werden die Eingangsparameter aus dem Pipeline Dictionary ausgelesen und ihre Werte an funktionsinterne Variablen weitergegeben. Über den *ForumManager*, initialisiert im Konstruktur des Pipelets, wird auf die in 4.1 beschriebene Methode *getForumsByType()* mit den ausgelesenen Parametern zugegriffen. Die Kollektion der Foren als Ergebnis der Methode wird abschließend in das Pipeline Dictionary geschrieben und die Ausführung ist beendet.

[40] Eigene Abbildung

4.3 Pipelines

Die Pipeline, welche durch den Nutzer ausgeführt wird, ist im linken Bereich von Abbildung 17 dargestellt. Die Bezeichnung lautet *ViewScForum-Overview*.

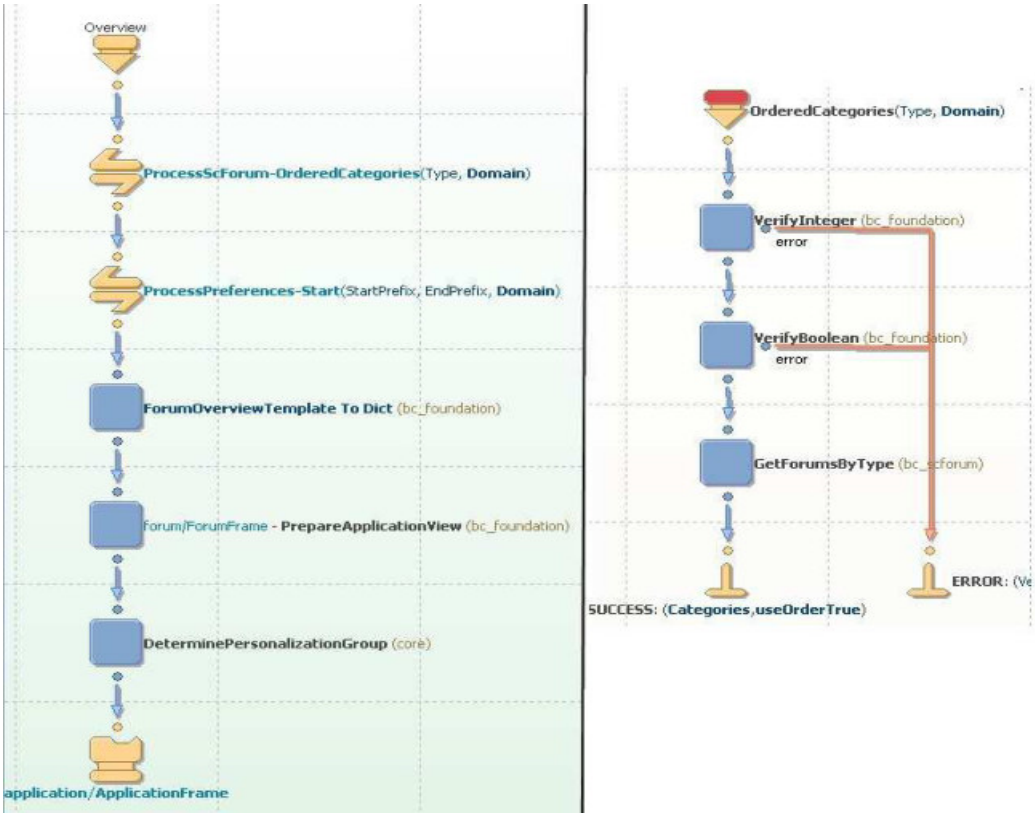

Abbildung 17: Pipelines zur Darstellung der Forenübersicht[41]

Nachdem die Pipeline gestartet wurde, erfolgt der Aufruf einer Prozesspipeline mit der Bezeichnung *ProcessScForum-OrderedCategories*, welche rechts in der Abbildung zu sehen ist. Die Pipeline ist strikt, sie nimmt Typ und Domain entgegen. Innerhalb der Prozesspipeline werden Intershop Pipelets genutzt um die Parameter zu validieren. An dritter Stelle wird das in 4.2 beschriebene Pipelet aufgerufen. Das Ergebnis, die Kollektion der Foren, kann nach Beendigung der Pipeline vom Template zur Darstellung genutzt werden. Nachdem über eine zweite Prozesspipeline forenspezifische Einstellungen geladen wurden

[41] Eigene Abbildung

(*ProcessScPreferences-Start*), wird das Template *ForumOverview* in ein Rahmentemplate (*ForumFrame*) geladen und die Pipeline ist beendet.

4.4 Templates

Die durch die Pipeline produzierten Daten können im Template unter Verwendung von ISML zur Darstellung der Forenübersicht genutzt werden. Mit dem Aufruf <isloop iterator="*Categories*" alias="Category"> wird in einer Schleife über die Kollektion mit der Bezeichnung *Categories* iteriert. In jedem Durchlauf steht das aktuelle Element unter der Bezeichnung *Category* zur Verfügung und kann im Template verwendet werden. Mit dem ISML Tag <isprint value="#*Category*:*Name*#"/> lässt sich auf das Attribut *Name* der aktuellen Kategorie zugreifen.

Abbildung 18 zeigt das Resultat aus Sicht des Nutzers.

Forum	Topics	Posts	Last Post
ODS - Community			
Outdoorseiten.net e.V.	0	0	no post available
Laberforum	0	0	no post available
Leben an Board - FAQ, Regeln, Hintergründe	0	0	no post available
Reisevorbereitung & Reiseberichte			
Reiseberichte	0	0	no post available
Reisevorbereitung - Europa	0	0	no post available
Fitness & Vorbereitung	0	0	no post available

Abbildung 18: Anzeige im Frontend[42]

[42] Eigene Abbildung

5 Ausblick / Lessons Learned

Das Forum wurde nach etwa vier Monaten Entwicklungszeit erfolgreich abgeschlossen. Bis auf kleine Details wurden dabei die Anforderungen erfüllt. Insbesondere die Tatsache, dass im Forum mehrere Komponenten wie Bewertung, Freischalten von Beiträgen und Mitgliederbelohnungssystem zusammentreffen, lässt das Forum zu einem Kernelement des Shopsystems werden.

Mich persönlich hat die Entwicklung des Forums vor allem im Umgang mit Intershop Enfinity Studio geschult und ich hoffe dieses Wissen in naher Zukunft erneut anwenden und erweitern zu können.

Rückblickend lässt sich jedoch auch feststellen, dass der Entwicklungsprozess nicht vollständig eingehalten wurde, da insbesondere High Level Designs zum Teil erst parallel zur Entwicklung angefertigt wurden oder zu einem späteren Zeitpunkt Aktualisierungen erforderlich waren. Dies lag am phasenorientierten Entwicklungsprozess selbst, da hier versucht wird, alle Anforderungen im Vorfeld zu definieren. Neu aufkommende Wünsche sind zu Beginn nicht immer vorhersehbar, was unweigerlich zu nachträglichen Anpassungen führt.[43]

Abschließend möchte ich anmerken, dass ich mit meiner Arbeit sehr zufrieden und stolz auf das Geleistete bin. Ich hoffe, dass das Forum der Firma im Rahmen der Neukundengewinnung von Nutzen ist und sich mir dadurch die Möglichkeit bietet, erworbene Kenntnisse hinsichtlich Enfinity Suite in einer kommenden Praxisphase zu vertiefen.

[43] Vgl. [Eic09]

Quellen

Kürzel	Quelle
DS_TW	dotSource Gmbh, dotsource Wiki, http://wiki.dotsource.de/bin/view Abruf: 2009.09.11
[Eic09]	Eickmann M., Tomasini A., "Scrum Teil 3: Die Rolle des Entwicklungsteams", in iX, Oktober 2009, heise Verlag, S.131-134
Gam96	Gamma E: „Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software" 1. Auflage, Addison Wesley, München, 1996
IS_TI_07	Intershop Communications AG , Enfinity Suite 6 – Technical Introduction E62-110-1EN, Juni 2007
IS_APG_07	Intershop Communitcations AG, Enfinity Suite – Application Programming Guide, Document ID: EMS-LIB-62-05-07, März 2007
IS_WEB	Intershop Communications AG. http://www.intershop.de/e_commerce_software/enfinity/ Abruf: 2009.08.17
IS09_2	Intershop Communications AG, Enfinity Suite 6 – Produktbroschüre http://www.intershop.de/uploads/media/2009-Enfinity-de.pdf, 2009

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich meine Praxisarbeit mit dem Thema

„Entwurf und Entwicklung einer Forumskomponente mit Enfinity Studio“

ohne fremde Hilfe angefertigt habe,

dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe und dass ich meine Praxisarbeit bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

_____ _____

Ort, Datum Unterschrift