Ludwig-Maximilians-Universität München
Institut für Informatik

Diplomarbeit

Modellierung und Generierung von Web 2.0 Webanwendungen

Aufgabensteller: Prof. Dr. Alexander Knapp

Betreuer: Dr. Nora Koch
Matthias Pigerl (S.CO LifeScience GmbH)

Bearbeiter: Tatiana Morozova

Abgabedatum: 28. November 2008

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

München, den 27. November 2008
Tatiana Morozova .....

Vorwort

„Keine Schuld ist dringender, als die, Dank *zu sagen.“*
Marcus Tullius Cicero (106-43)

„Kein Bedürfnis ist dringender, als das, Dank zu sagen.“
Meine Wenigkeit

Ich möchte mich bei dem PST Lehrstuhl an der LMU bedanken, dafür dass Sie sich ununterbrochen Fragen stellen und mit der gesunden Neugierde an alles Neue voran gehen. Somit entstehen solche interessante Projekte wie UWE, die eine Menge Stoff für Inspiration und Forschung bieten.

Ich bedanke mich sehr bei meiner Betreuerin an der LMU Dr. Nora Koch. Ohne Sie hätte diese Arbeit nie das Licht erblickt. Danke dass Sie mir in den richtigen Momenten den Weg gewiesen haben und in den richtigen Momenten genug Spielraum gegeben haben. So dass ich die Antworten auf meine Fragen gesucht habe und nicht nur starr gestellte Aufgaben erledigte. Außerdem danke ich Ihnen, dass Sie es möglich gemacht haben, dass ich bei der S.CO LifeScience GmbH eine kurze aber wunderbare Zeit mitarbeiten durfte.

Ich danke dem gesamten S.CO LifeScience GmbH Team, das mich so schnell und liebevoll aufgenommen und unterstützt hat. Ihr seid wie eine kleine Familie, die gemeinsam an alle Probleme heran geht und sich gegenseitig unterstützt. So ein Teamgeist habe ich selten gesehen und bin sehr froh, dass es so was gibt. Bleibt wie ihr seid und nicht ein Problem wird es wagen sich in eurer Nähe aufzuhalten. Mein besonderer Dank gilt natürlich Matthias Pigerl. Du hast immer mitgedacht und mir mit Rat und Tat geholfen. Du hast deine Erfahrung und dein Talent so unaufdringlich und sinnvoll eingesetzt, dass ich nur staunen konnte, wie interessant und aufregend die Arbeit sein kann. Mir bleibt nur hoffen, dass ich in meinem späteren Berufsleben im professionellen Sinne an dich heran wachsen kann.

Ich danke meiner Neugier, dass sie mich immer weiter im Leben begleitet und vorantreibt. Dank dir bin ich das geworden, was ich bin, sei immer mit mir.

Zusammenfassung

Die Firma S.CO LifeScience GmbH bietet den Kunden mittels ihres Produktes „S.CORE“ automatische Bildanalysen an. Im Unterschied zu anderen Firmen in diesem Bereich wird bei S.CO LifeScience GmbH der angebotene Service komplett über das Web abgewickelt. In Zusammenarbeit mit dem Lehrstuhl Programmierung und Softwaretechnik (LMU) wurde der objektorientierte Ansatz für systematische Entwicklung von Webanwendungen UML-based Web Engineering (UWE) zur Modellierung von der webbasierten Schnittstelle zum S.CORE in die Praxis umgesetzt.

Um S.CORE noch benutzerfreundlicher zu machen, kommen neuste Web 2.0 Technologien wie AJAX im neuen Release zum Einsatz. Dabei ist die S.CORE-Entwicklungsgruppe an die Grenzen von UWE angelangt, weil es bei der Modellierung von Web 2.0 Konzepten (wie z.B. von AJAX basierten Elementen) für die Benutzeroberfläche an aussagekräftigen Konstrukten mangelte. In der vorliegenden Diplomarbeit wurde UWE Web 2.0 fähig gemacht. Dabei wurde UWE um die nötigen Konzepte zur Modellierung und Generierung von Web 2.0 Webanwendungen erweitert. Die Erweiterung basiert auf Untersuchungen der heutigen Trends auf dem Gebiet von modernen Technologien für die Entwicklung und Implementierung von Webapplikationen und wurde mit der Webanwendung S.CORE validiert.

Inhaltsverzeichnis

Kapitel 1 Einleitung

Das Internet ist ein Teil unseres Lebens geworden. Wir ziehen es zu Rate, falls wir Informationen brauchen; wir nutzen Internet um uns zu unterhalten sowie zur Kommunikation; wir speichern im Internet unsere Daten und besuchen Kurse; wir kaufen im Internet ein und finden alte Freunde. Mit wachsender Bedeutung des Internets wachsen auch die Anforderungen an die Anwendungen, die im Internet dem Nutzer zur Verfügung gestellt werden. Aufgrund dessen haben sich in den letzten Jahren Techniken und Methoden für die Implementierung von Webanwendungen rasant entwickelt. Infolgedessen sind mehrere Ansätze entstanden, die eine methodische Vorgehensweise bei der Entwicklung von Webanwendungen beschreiben. Dennoch bietet der rapide Fortschritt auf diesem Gebiet sehr viel Stoff für Forschung und Weiterentwicklung an.

In diesem Kapitel wird die Notwendigkeit der vorliegenden Arbeit erörtert, es wird eine detaillierte Problembeschreibung vorgestellt und einen kurzen Überblick über die gesamte Arbeit gegeben.

1.1 Motivation

Da die Komplexität von Webanwendungen stark gestiegen ist, stieg folglich auch der Aufwand bei deren Entwicklung. Der höhere Aufwand entstand zum einen durch immer mächtigere Funktionalitäten und aufwendigere graphische Oberflächen, welche moderne Webanwendungen bieten sollen und zum anderen durch immer häufiger geforderte extreme Anpassungs- und Erweiterungsfähigkeiten moderner Webanwendungen. Beides ist notwendig, damit die Anwendung auf dem heutigen Markt konkurrenzfähig bleibt. Aus diesen Gründen sollte die Entwicklung durch etablierte Technologien und wirksame Methoden unterstützt werden. Die Methoden einer schon längst ausgereiften Wissenschaft, wie „Software Engineering“ („*zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen.*“ ([10], S.36)), wurden zur Hilfe gezogen. Die langjährige Erfahrung bei Entwicklung von komplexen Anwendungen wurde für den Entwurf von Webanwendungen zur Nutze gemacht. Dennoch reichten die herkömmlichen Methoden für die Entwicklung und Evolution web-basierter Systemen nicht aus. Denn bei der Entwicklung einer Webanwendung noch folgende Aspekte, wie selbsterklärende Benutzeroberfläche, Modellierung der Navigationsstruktur innerhalb der Anwendung und viele andere, in Betracht gezogen werden sollen. Dabei ist ein neuer Wissenschaftszweig „Web Engineering“ als eine Erweiterung des herkömmlichen Software Engineerings entstanden. Der Kernansatz im Software Engineering ist das Teilen von komplexem Softwareentwicklungsprozess in überschaubare, zeitlich und inhaltlich begrenzte Schritte. Basierend auf einem der bekannten Beispielen für die Vorgehensmodelle in der Software Entwicklung- dem Wasserfallmodell, kann man folgende Schritte im Lebenszyklus einer Software nennen: Anforderungsspezifikation, Anforderungsanalyse, Entwurf, Implementierung und Validierung. Zu jedem dieser Schritte haben sich bestimmte Techniken und Methoden etabliert, die wichtigsten im Rahmen der vorliegenden Arbeit sind in der Tabelle 1 aufgelistet:

Schritte	Methoden und Techniken
Anforderungsspezifikation	Interview, Lesen von Hintergrund Material, u.a.
Anforderungsanalyse	Anforderungsbeschreibung, erstellen von Use Case Diagrammen (UML)
Entwurf	Modellierung des Systems (UML)
Implementierung	Kodierung des Systems (diverse Programmiersprachen)
Validierung und Verifikation	Testen

Tabelle 1. Lebenszyklus einer Software

Web Engineering baut auf diesen Methoden auf. Einer von mehreren auf diesem Feld entstandenen Ansätzen ist UML-based Web Engineering (UWE siehe [8], [20], [24]), der im Wesentlichen eine Erweiterung von Unified Modeling Language (UML siehe [9]) darstellt und die Modellierung von allen wesentlichen Aspekten einer Webanwendung, nämlich Inhalt, Navigation, Prozess und Präsentation, ermöglicht. Dennoch hat die rasante Entwicklung von Webtechnologien einige Lücken in dem gesamten Konzept von UWE aufgedeckt, und zwar in der letzten Zeit sind die Webanwendungen sehr interaktiv geworden, es fehlte aber ein Konstrukt um auf der Modellierungsebene diese Interaktivität darzustellen. Die neueste Strömung in der Welt der Internettechnologien ist Web 2.0. Web 2.0 Anwendungen haben die Benutzung vom Internet verändert. Die innovativen Techniken wie AJAX für die Implementierung von Web 2.0 Anwendungen stellen auch das Web Engineering vor neuen Aufgaben. Die bekannteste Regel für die Folgekosten eines Fehlers im Software Engineering lautet: *„Je später im "Software Lifecycle" ein Fehler aufgedeckt wird, desto höher sind die Kosten zu derer Behebung!“* Somit ist es sehr wichtig funktionsfähige Konzepte zur Modellierung von Web 2.0 Anwendungen zu finden.

Eins von den Zielen der vorliegenden Arbeit war unter anderem fehlende Konstrukte und Technologien basierend auf praktischen Erfahrungen aufzuspüren und anschließend ein Konzept zur der Vervollständigung von UWE in diesen Hinsichten auszuarbeiten und auf konkreten Beispielen darzustellen.

1.2 Problembeschreibung

Firma S.CO LifeScience GmbH [w6] entwickelt eine neue Webanwendung namens S.CORE [w7]. S.CORE ist ein webbasiertes Analysesystem für automatische Bildanalyse. Im Rahmen des S.CORE Projektes werden „Standard-Analysemodule“ und an den Kunden angepasste „Customized Analysemodule“ angeboten. Im Unterschied zu anderen Firmen, die ähnliche Produkte anbieten, wird das bei der S.CO LifeScience GmbH angebotene Service komplett über das Web abgewickelt. Neuste Web 2.0 Technologien wie AJAX kommen im neuen Release von S.CORE zum Einsatz. Dadurch hat sich die Kundennutzbarkeit der Anwendung erhöht.

S.CORE wird in enger Zusammenarbeit mit der UWE (UML based web engeneering) Gruppe der Ludwig- Maximilians- Universität (LMU) München entwicklelt. Besagte Forschungsgruppe entwickelte UWE und nutzt S.CORE oft als erstes Testprojekt für UWE. Während der Entwicklung von S.CORE ist die Entwicklungsgruppe an die Grenzen von

UWE gelangt, in dem es bei der Modellierung von dynamischen Konzepten (wie z.B. von AJAX basierten Elementen) auf der Benutzeroberfläche an den aussagekräftigen Konstrukten mangelte.

In dieser Diplomarbeit wurden im Rahmen einer Erweiterung von S.CORE die fehlenden Konstrukte aufgedeckt und untersucht. Nämlich, es bestand noch der Bedarf an den praxistauglichen Konstrukten im Präsentation Modell, zum Beschreiben von modernen Interaktiven Elementen. Ein anschauliches Beispiel dafür ist ein modernen Formular, der nahezu in jeder Webanwendung vorkommt. Der wesentliche Unterschied zu dem herkömmlichen Formular in einer konventionellen Webanwendung liegt zu einem in der Überlagerung manchen Funktionalitäten direkt in Client (d.h. direkt in Browser) und zu anderem in der modernen AJAX-Technologie, die es möglich gemacht hat manche Anfragen zum Server im Hintergrund abzuwickeln, was dem Benutzer ein Gefühl der nahezu desktopähnlichen Anwendung vermittelt und sehr viel Zeit spart. Abgesehen von der Tatsache, dass solche Elemente noch während der Designerphase dargestellt werden sollen, soll das ausgearbeitete Konstrukt flexibel genug sein, um dem Entwickler genug Freiraum zu lassen, damit er die Entscheidung treffen kann, ob manche Elemente überhaupt die interaktive Fähigkeiten besitzen sollen oder nicht.

Zum Weiteren sollten möglichst viele moderne Elemente untersucht werden um die Flexibilität und Verwendbarkeit sowohl von den entstandenen Erweiterungen als auch von der Modellierungssprache UWE selbst zu beweisen. Um dieses Ziel zu verfolgen wurden in der vorliegenden Arbeit die Erweiterungen von UWE in der Praxis anhand von S.CORE Webanwendung bei S.CO LifeSience validiert.

1.3 Lösungsansatz

Um den oben beschriebenen Thematik (siehe Abschnitt 1.1) entgegen zu wirken, wurde im Rahmen dieser Arbeit eine gründliche Untersuchung von heutigen Trends auf dem Gebiet von modernen Technologien für die Entwicklung und Implementierung von zeitgerechten Webapplikationen durchgeführt. Dabei wurde eine praxisorientierte Lösung für die Modellierung der innovativen Elemente entwickelt und im Rahmen einer Erweiterung von S.CORE bei S.CO LifeScience GmbH validiert. Dafür wurde zunächst die Modellierungssprache UWE aufbauend auf den vorangegangenen Arbeiten (siehe [25], [26]) überarbeitet und erweitert, wobei die gravierendste Erweiterung hat für das Präsentation Modell im Modellierungsansatz stattgefunden.

Die Idee von Modellierung steht für hohe Abstraktion von den konkreten Anwendungen. Dennoch wird in neuesten Forschungsprojekten ein Versuch gemacht zur automatischen Generierung bestimmten Teilen von der späteren Applikation, was detailliertere Informationen über die wesentliche Aspekte der Anwendung verlangt. Diesem Gedanke folgend wurden für die detailliertere Modellierung von Benutzerinteraktionen auf der Benutzeroberfläche die Zustandsdiagramme (ein Standardkonstrukt von UML) als Lösung vorgeschlagen. Die jeweiligen Zustandsdiagramme beschreiben das gesamte Verhalten des User Interface (UI) Elements und werden als „*Elements Behavior*“ dem entsprechenden Element im Präsentationsmodell hinzugefügt. Diese Lösung bietet zu einem die wertvolle Unterstützung in der Modellierung von interaktiven Elementen, zu anderem volle Flexibilität in der Hinsicht auf die Ausführlichkeit mit der das Modell erstellt werden kann. In einem Zustandsdiagramm ist schon ein Konstrukt für die Beschreibung von Auslösungsereignissen für eine Systemaktivität eingebaut, dennoch wurde in dieser Arbeit eine explizite Sprache

(Event Language siehe Abschnitt 4.3.1) für die Beschreibung von möglichen Ereignissen, die eine Systemaktivität auslösen können, entwickelt. So ein präzises Sprachkonstrukt war nötig wegen der Aussicht, dass es eine automatische Generierung von solchen Webanwendungen später möglich gemacht werden soll. Als Vorlage für Event Language hat die Beschreibung von möglichen Events in JavaScript Programmierungssprache gedient. JavaScript wird in der Regel für die Implementierung von interaktiven Elementen auf der Benutzeroberfläche von einer Webanwendung verwendet, deswegen bietet sie den nötigen Stoff für die entstandene Event Language. Dennoch wurde die Ereignissprache mit dem gewissen Abstraktionsgrad von jeglicher konkreter Technologie ausgearbeitet, was sie zu einem flexiblen und breitverwendbaren Werkzeug macht.

Die oben angedeutete Flexibilität von dem Entwickler bei der Modellierung von interaktiven Elementen wird auch dadurch erreicht, dass der Entwickler selbst entscheiden kann, ob er die auf dem Zustandsdiagramm vorkommende Aktivitäten weiter ausführlich modellieren will, oder es nur beim Erwähnen von deren Namen bleiben wird. Ein weiterer Aspekt hinsichtlich der Flexibilität des Entwicklers bei der Modellierung stellt sich aus der Tatsache heraus, dass das größte Teil der Erweiterung von UWE anhand von Hinzunahme einiger neuen Attributen zu den schon existierenden Elementen stattgefunden hat (näheres dazu siehe in Abschnitten 2.6.1 und 4.3.2). Dadurch beim Anwenden von bestimmten Stereotypen im Präsentationsmodell kann davon verzichtet werden, dem entsprechenden Metaattribut ein Wert zu vergeben. In diesem Fall hat man das herkömmliche Element von UWE zur Verfügung.

1.4 Aufbau der Arbeit

Die vorliegende Arbeit enthält fünf Kapitel. Im ersten Kapitel wurden die einleitende Bemerkungen und Gedankenschlussfolgerungen für die gesamte Arbeit vorgestellt. Im Kapitel 2 wurde eine kleine Einführung in die moderne Webtechnologien gemacht und die dazugehörige Begriffe erklärt und definiert. Das Kapitel 3 dient der Beschreibung von S.CORE und der im Rahmen dieser Arbeit entstandenen Erweiterung von S.CORE.

Aufbauend auf den Kapiteln 2 und 3 wird im Kapitel 4 den Ansatz zur Modellierung von interaktiven Elementen der modernen Webanwendungen mit UWE vorgestellt. Dabei wurde im Abschnitt 4.2 eine Liste von interaktiven Elementen, die in zeitgenossischen Webanwendungen sehr oft zur Verwendung kommen, vorgestellt. Anlehnend an diese Liste wurden im Abschnitt 4.3 die konkreten Vorschläge zu der Erweiterung von UWE auf den jeweiligen Beispielen beschrieben und geschildert.

Das Kapitel 5 stellt eine Zusammenfassung für die gesamte Arbeit und beschreibt anschließende Möglichkeiten zur weiteren Forschung auf diesem Gebiet.

Kapitel 2 Grundlagen

Im folgenden Kapitel wird eine Grundlage für das Verständnis der darauffolgenden Ausarbeitung geschaffen. Dafür werden wichtigste terminologische Begriffe vorgestellt. In manchen Fällen werden erklärende Definitionen eingeführt.

2.1 Das Web

Das Internet wurde 1969 als ein Projekt *Advanced Research Project Agency* (ARPA) des US-Verteidigungsministeriums geboren. Das ist eine Abbreviatur vom englischen: „*interconnected Networks*“. Die genaue Beschreibung liefert folgende Definition:

„Internet ist ein weltweites Netzwerk bestehend aus vielen Rechnernetzwerken, durch das weltweit Daten ausgetauscht werden. Es ermöglicht die Nutzung der Internetdienste wie Telefonie, WWW, E-Mail und Radio.“

Wikipedia, Stand vom 25.08.08

Ein häufiger Fehler, der zurzeit immer wieder gemacht wird ist das Verwechseln des Internets mit WWW. Der Unterschied macht folgende Definition von WWW deutlich:

„Das World Wide Web (kurz Web, WWW oder deutsch: Weltweites Netz; wörtlich: web „Gewebe, Netz“) ist ein über das Internet abrufbares Hypertext-System.“

Wikipedia, Stand vom 25.08.08

Einen großen Beitrag in dieses Durcheinander hat die Tatsache beigetragen, dass die neuesten Webbrowser (Programme mit Hilfe dessen man die Websites von WWW betrachten kann) nicht nur http- sondern auch ftp- Protokoll und somit fast alle mögliche Internet Dienste, wie E-Mail, File Transfer, und viele andere unterstützen. Aus diesem Grund entstand die Gleichbedeutung von beiden Begriffen. Ein letztes Tropfen in der Vereinigung von der Bedeutung des WWWs und Internets, abgesehen von den neuesten Entwicklungen, die die Webbrowser durchgemacht haben, hat folgender Faktor gespielt. Das Web hat sich von einem starren abrufbarem Hypertext- System (siehe Definition von WWW) zu einer Menge dynamischen Anwendungen (sogenannten Webservices), die im Internet zur Verfügung gestellt werden, entwickelt.

Der große Fortschritt auf dem Gebiet der Webtechnologien hat Dale Dougherty (O'Reilly-Verlag) und Craig Cline (MediaLive) veranlasst, bei der Vorbereitung zu einer Konferenz, eine Liste von Anwendungen und neuesten Techniken zusammenzustellen. Diese Liste stellte, ihrer Meinung nach, die Entwicklung des Webs am besten dar. In diesem Zusammenhang hat auch der Begriff Web 2.0 zum ersten Mal das Licht erblickt. Seitdem sorgt das Wort für viel Wirbel und Aufregung.

AJAX, Flash, JavaScript, Weblogs, DHTML, RIAs, RSS, Google AdSense, Flickr, Napster, Wikipedia, Atom, Trackback, iPod und Podcasting… Alle diese Begriffe werden in Verbindung mit Web 2.0 gebracht. Das ist eine lange Liste und die ist noch ganz und gar nicht vollständig. Der Begriff „Web 2.0“ hat sich zu einem Schlagwort entwickelt, das praktisch für alles verwendbar ist. In den meisten Fällen dient es leider nur den Werbezielen mit der durchschlagenden Wirkung. In der Literatur kann man in der jetzigen Zeit die Erfahrungen machen, dass viele Begriffe in diesem Gebiet durcheinander gebracht werden,

was nicht immer konzeptuell sinnvoll ist und sogar falsche Schlussfolgerungen mit sich ziehen kann.

Zusammenfassend kann man betonen, dass es in der Literatur an konkreten Definitionen für alle Begriffe um das Wort Web 2.0 mangelt. Im Begriffsdschungel sollte die Struktur und Klarheit geschaffen werden. Somit ist das Ziel und Maßstäbe für diesen Kapitel gesetzt.

2.2 Web 2.0

„I think Web 2.0 is of course a piece of jargon, nobody even knows what it means. “
Tim Berners-Lee, der Begründer des WWW

Zur Zeit der Verfassung vorliegender Arbeit lieferte Wikipedia folgende Definition für Web 2.0:

*„Web 2.0 steht für eine Reihe **interaktiver** und **kollaborativer** Elemente des **Internets**, speziell des* WWWs*.“*

Wikipedia, Stand vom 15.08.08

Diese Definition ist sehr gut, aber nicht genau zutreffend. Wie es schon im Abschnitt 2.1 angedeutet wurde, steht dieser Begriff mehr für einen Trend in der Benutzung vom Internet. Sogar der Name „Web 2.0“ spricht für sich. Es ist nichts prinzipiell Neues, es ist eine Entwicklung, die das Web an sich durchgemacht hat und immer noch durchmacht, somit kann der Begriff keine eindeutige Definition haben. Wobei die Adjektive „interaktive“ und „kollaborative“ in der Definition auf wichtigste **neue Funktionalitäten** hinweisen, die im modernen Web dem Nutzer zur Verfügung gestellt werden.

Aus diesem Betrachtungswinkel sollte noch auf ein kurzes Vergleich mit dem Web 1.0 eingegangen werden, damit es verständlich wird was für Innovationen das Web 2.0 ausmachen. Dies wird in der Tabelle 2 deutlich.

	Web1.0	**Web2.0**
Kollaboration	Inhaltszentriert (Ziel: Benutzer informieren). Inhalte werden von Programmierern geändert.	Benutzerzentriert (Ziel: Ein Dienst zur Verfügung stellen). Inhalte werden oft von Benutzern selbst geändert.
Interaktion	Thin-Client mit Push-up Interaktion (User fordert, Server antwortet).	Tendiert zum Fat-Client mit Pull-down Interaktion (User macht, Server reagiert).

Tabelle 2. Vergleich zwischen Web 2.0 und Web 1.0

Dazu kommen noch prinzipiell **neue Techniken** für die Entwicklung einer Webanwendung zur Geltung, wie AJAX-Technologie, die es unter anderem zum Beispiel möglich macht, „Pull-down“ Prinzip für die Client-Server Interaktion zu simulieren. D.h. ohne explizite Anfrage vom Benutzer werden Anfragen zum Server gestartet und durchgeführt. Dies gibt

dem Benutzer das Gefühl von der sofortigen Interaktion, was bis jetzt nur aus den Erfahrungen mit den Desktop Anwendungen bekannt gewesen ist.

Um unter dem Ganzen ein Strich zu ziehen, und dem Ziel von diesem Kapitel ein Stück näher zu kommen, wird jetzt die erste Definition eingeführt Somit wird im Rahmen dieser Diplomarbeit unter Web 2.0 Folgendes vorgestellt:

Web 2.0 *steht für die Entwicklung des Webs zu einer Plattform von* ***interaktiven*** *und* ***kollaborativen*** *Anwendungen und Diensten.*

Da diese Definition auf die Entwicklung des Webs im Focus neuer Funktionen wie Kollaboration und Interaktion eingeht und neue Technologien beschreibt, die es ermöglichen solche Anwendungen zu entwickeln, können solche unterschiedliche Aspekte wie AJAX (Technik) und Webloggin (Funktion) mit dem Begriff Web 2.0 unter einen Hut gebracht werden.

Den kollaborativen Aspekt von Web 2.0 Anwendungen realisieren folgende Beispiele, wie:

- verschiedene Videoportals, wie YouTube [w1], die von vielen User gemeinsam erstellt werden, indem alle Benutzer, eigene Bilder und Videos auf das zur Verfügung gestelltes Server herunterladen können;
- unterschiedliche wikis, die von den Usern entwickelt werden und eine Rolle von öffentlicher Bibliothek spielen(prominenteste Beispiel Wikipedia [w2]);
- Weblogs, die so was wie unendliche Journals darstellen, die auf einer Website geführt werden, und auf denen wiederum alle Benutzer eigene Einträge hinterlassen können.

Auf den interaktiven Aspekt von Web 2.0 Anwendungen wird in dieser Arbeit viel intensiver eingegangen. Zum einen weil die im Rahmen dieser Arbeit erweiterte Webanwendung von S.CO LifeScience GmbH diesen Aspekt von Web 2.0 Anwendungen realisiert. Zum anderen um einen gewissen Rahmen für diese Arbeit zu setzen.

Den interaktiven Aspekt von Web 2.0 Anwendungen stellen RIAs dar. Um dieses Begriff verständlich zu machen, wurde den nachfolgenden Abschnitt verfasst.

2.3 RIAs

„ Gewisse Bücher scheinen geschrieben zu sein, nicht damit man daraus lerne, sondern damit man wisse, dass der Verfasser etwas gewusst hat.“
Johann Wolfgang von Goethe

Analog zu obiger Thematik stellte es sich auch als schwierig heraus, eine Eindeutigkeit in den Definitionen für RIAs festzustellen. Im Folgenden werden ein Paar Beispiele vorgeführt, die zurzeit in der Literatur zu finden sind:

„Der Begriff Rich Internet Application (RIA, deutsch: reichhaltige Internet Anwendung) beschreibt eine Anwendung, die Internet-Techniken benutzt und eine intuitive Benutzeroberfläche bietet.“

Wikipedia, Stand vom 15.08.08

„Rich Internet Applications (RIA) sind Internet Anwendungen, welche über eine Web-GUI verfügen.“

Neogrid, das innovative EDV Lexikon

„Der Begriff RIA wird aktuell unabhängig von bestimmten Techniken oder Firmen verwendet und ist eine allgemeine Bezeichnung für moderne Anwendungen, die im Webbrowser laufen.“

PHP Magazin, Kristian Wenz und Tobias Hausen

Hier kristallisiert sich die Problematik, dass die Begriffe sehr verschwommen dargestellt werden und meistens kann es alles Mögliche hinein interpretiert werden. Eins von den Zielen der vorliegenden Arbeit ist aber so eine Definition zu finden, die möglichst zutreffend den Begriff beschreibt.

Aufbauend auf den oben eingeführten Definitionen und auf dem vorherigen Abschnitt über Web 2.0 wird folgende Definition für RIA vorgeschlagen:

*Als **RIA** (Rich Internet Application) bezeichnet man **eine Webanwendung**, die sich durch die **hohe Nutzerinteraktion** von den herkömmlichen Webanwendungen unterscheidet. Die hohe Nutzerinteraktion bringt solche Webanwendung näher den Standards einer Desktopanwendung.*

Diese Definition ist viel konkreter in der Hinsicht, dass RIAs ein Teil von Web 2.0 Anwendungen darstellen und zwar, die interaktiven Web 2.0 Anwendungen.

Einige sehr gute Beispiele was eine RIA sein kann liefern z.B. die Webmail- Dienste wie die von Google [w3], Yahoo [w4] und andere.

2.4 Technologien zur Implementierung von RIAs

Das Ziel von diesem Kapitel ist nicht die komplette Liste von Technologien darzustellen, mit denen man RIAs erfolgreich implementieren kann, sondern die gängigsten Methoden zu nennen und deren Unterschiede zu beleuchten.

2.4.1 Flash Technologie

Flash Technologie ermöglicht auf der Vektorgrafik basierendes Erstellen von multimedialen Inhalten (Flash-Filmen).

Technik		**Beispiele**
Konzepte für Programmierung	***Tools***	
• Benutzt **ActionScript** Scriptsprache (sehr ähnlich zu JavaScript) und braucht ein **Flash-Plugin** auf der Clientseite (d.h. im Browser)	• **Adobe Flash** (das mächtigste Tool) • **Ming-Modul von PHP** (zur Generierung von Flash- Dateien)	• Erstellung von **Online Spielen** • Macht möglich dynamisches Erstellten von **Diagrammen** • Erstellung von **Werbebanner** • Erstellung von **Multimedialen Tutorials** • Erstellung von **interaktiven Animationen** • Erstellung von Teilen der Steuerungsmenus

Tabelle 3. Flash Technologie

Vorteile:

- das Plugin für Flash ist in fast 99% der Clients vorhanden;

Nachteile:

- die aktuellste Version vom Plugin für Flash ist für Linux, Microsoft Windows und Mac OS X (aber nicht z.B. für Solaris (Sun Microsystems), oder für AIX (IBM), odr für BeOS von Be Inc., oder für Zeta und andere) verfügbar, somit sind die Flash- Dateien auf diese Betriebsysteme angewiesen;
- Flash Technologie ist kein offener Standard und dadurch sind die Entwicklungstools teuer und man ist bei der Entwicklung an den Anbieter angewiesen.

2.4.2 Silverlight

Silverlight Technologie wurde von Microsoft entwickelt und dient zu der Darstellung und Animation von Oberflächen aus grafischen Elementen und multimedialen Daten in Browsern.

Technik		**Beispiele**
Konzepte für Programmierung	***Tools***	
• Basiert auf **XMAL** (e**X**tensible **A**pplication **M**arkup **L**anguage eine Sprache zur Beschreibung und Erstellung von Oberflächen, die in XML formuliert ist) • Für die Implementierung vom dynamischen Verhalten kann **JavaScript** eingesetzt werden	Analog zu Flash Technologie: • **Adobe Flash** (das mächtigste Tool) • **Ming-Modul von PHP** (zur Generierung von Flash-Dateien)	Steht in Konkurrenz mit der Adobe Flash Technologie, d.h. dass alle Beispiele die für Flash Technologie vorgeführt wurden können als Beispiele für Silverlight Technolie dienen (siehe Tabelle 3); Ein reales und sehr interessantes Beispiel ist die von Microsoft erstellte Suchmaschine „Tafiti“ (siehe [w5]), die komplett auf der Silverlight Technologie basiert.

Tabelle 4. Silverlight Technologie

Die Vor- und Nachteile spiegeln sich in den Vor- und Nachteilen von Flash Technologie ab. Dennoch es gibt zurzeit noch ein Nachteilt, der bei der Beschreibung von Silverlight zu berücksichtigen ist. Zurzeit gibt es einige Plugins zum Abspielen von mit der Silverlight Technologie erstellten Anwendungen. Das Problem liegt daran, dass diese Plugins noch nicht für alle gängigen Webbrowser existieren. Als Beispiel kann man Opera und Safari Webbrowser nennen. Für diese Browser existieren noch keine Plugins zum Abspielen von Silverlight Anwendungen. Das wird sich aber schon bald ändern.

2.4.3 AJAX

Eine innovative Technologie auf dem Gebiet von der Entwicklung der Webanwendungen stellt AJAX dar. Das Wort „AJAX“ repräsentiert eine Abbreviatur von „**A**synchronous **J**avaScript **a**nd **X**ML“. Diese Technologie hat die asynchrone Datenübertragung bei der Client- Serverseitigen Kommunikation möglich gemacht. Es ist schwer die Vorteile, die diese Technologie mit sich bringt, zu überschätzen. Sie hat die Techniken im Webengineering revolutioniert. Sie hat ermöglicht, dass moderne Webanwendungen das Gefühlt einer Desktopanwendung in der Hinsicht auf die Reaktionszeiten erwecken. Sie spart Zeit und macht die Webanwendungen interaktiv.

Technik		Beispiele
Konzepte für Programmierung	***Tools***	
• **JavaScript** ermöglicht die Erstellung der clientseitigen Logik (aufbauend auf dem Document Object Model (DOM) bei der Bearbeitung von Teilen einer HTML-Seite) • **XMLHttpRequest-**Objekt ermöglicht dem JavaScript asynchron auf den Server zu zugreifen, während der Benutzer noch mit der Anwendung arbeitet.	• Eclipse • Andere geeignete Entwicklungsumgebungen	• Eine unmittelbare Formularüberprüfung • Erstellung von Online-Chats • Implementierung einer Echtzeit-Lösung zur Erstellung von Diagrammen basierend auf Scalable Vector Graphic(SVG) • Erstellen von Drag&Drop-Listen • Programmierung von Anwendungen, die verteilte Aktualisierungen in Echtzeit benötigen.

Tabelle 5. AJAX-Technologie

Vorteile:

- Im Vergleich zu den schon genannten Methoden zählt AJAX-Technologie zu den offenen Standards, was sie zugänglicher und verbreiteter macht.
- Sie baut auf den vorhandenen Technologien auf, was relativ kurze Einarbeitungszeiten für den Entwickler beim Einsatz von dieser Technologie erzielt.

Nachteile:

- Performance von JavaScript im Vergleich zu der Performance von hohen Programmiersprachen wie C++ oder Java ist noch nicht ausreichend. Somit soll es so viel wie nötig und so wenig wie möglich JavaScript beim Programmieren verwendet werden. Denn „zu viel" JavaScript kann zu den negativen Effekten wie Flattern des Bildschirms führen.
- Es fehlen immer noch die richtigen Debugger für JavaScript, was die Entwicklung sehr erschwert.
- Unter Umständen können nicht alle Teile einer Site, die mit AJAX-Technologien erstellt wurde, von Suchmaschinen indiziert werden.

2.5 Vor- und Nachteile von RIAs

RIA ist eine moderne Webanwendung, mit benutzerfreundlicher Oberfläche. Es heißt jedoch nicht, dass die so genannten Web 1.0 Webanwendungen nur als veraltet und überholt gelten sollen. Bei der Entwicklung einer modernen Webanwendung soll natürlich auf die moderne Technologien eingesetzt werden, dennoch jede Innovation hat ihr Preis. Aus diesem Grund muss beim Erstellen von Webanwendungen immer die Vor- und Nachteile jeder Technologie abgewogen werden. In diesem Abschnitt wird auf die Vor- und Nachteile, die eine RIA ausmachen, eingegangen.

Vorteile:

Mit der Entwicklung von modernen Technologien wurde ein Anfang gemacht die Vorteile von Webanwendungen und Desktopanwendungen zu verbinden. In der nachfolgenden Tabelle (siehe Tabelle 6) werden alle Nachteile und Vorteile bezüglich bestimmten Eigenschaften aufgelistet. Dabei tritt deutlich hervor, dass eine RIA im Allgemeinen sowohl einer herkömmlichen Webanwendung als auch einer Desktopanwendung zu überlegen ist. Mit einem Plus sind dann jeweils die Vorteile und mit einem Minus die Nachteile markiert.

	Herkömmliche Webanwendung (WA)	**Desktopanwendung (DA)**	**Rich Internet Application (RIA)**
SW-Installation	nein (+)	ja (-)	nein (+)
Datenhaltung	zentral (+)	meist verteilt (-)	zentral (+)
Erreichbarkeit	von überall (+)	an best. Ort gebunden(-)	von überall (+)
Aktualisierung und support	sehr einfach und billig (+)	meist teuer (-)	sehr einfach und billig (+)
Bedienelemente wie Tabs und Trees	keine oder sehr beschränkt (-)	ja (+)	ja (+)
Tastaturbedienung	keine oder sehr beschränkt (-)	ja (+)	ja (+)
Reaktionszeiten auf Benutzeraktion	meist lange Wartezeiten, wegen Server- Anfragen (-)	sehr schnelle Reaktionszeiten (+)	viel schnellere Reaktionszeiten, als bei herkömmlichen Webanwendung (±)
Veränderungen auf dem Bildschirm	Seite wird komplett neue geladen (-)	nur Differenzen zum vorherigen Zustand werden auf dem Bildschirm geändert (Dynamischen UserInterface) (+)	meist werden nur Differenzen zum vorherigen Zustand auf dem Bildschirm geändert (Fast Dynamischen UserInterface) (+)

Tabelle 6. Vorteile von RIA im Vergleich zu DA und WA

Nachteile

Die wichtigsten Nachteile von RIAs zählen sich zusammen aus den Nachteilen von den Technologien mit denen sie erstellt werden.

- RIAs, die auf den nicht offenen Standards basieren (Wie z.B. Flash und Silverlight), sind an den Hersteller gebunden und können nicht überall angesetzt werden, abgesehen davon haben hohe Entwicklungskosten.
- Falls im Webbrowser JavaScript deaktiviert wird, funktioniert die Anwendung nicht mehr. Dadurch muss immer eine alternative Anwendung mit herkömmlichen Funktionalitäten zur Verfügung gestellt werden, was bedeutet, dass in vielen Fällen **doppelte Arbeit bei der Entwicklung und Implementierung** geleistet werden muss.
- Die Anwendungen lassen sich nicht mehr so leicht mit den Bookmarks versehen.

2.6 Modellgetriebene Ansätze im Web Engineering

Wie schon oben angedeutet wurde, waren die Ideen des konventionellen Software Engineerings sehr hilfreich bei der Entwicklung der Webanwendungen, jedoch nicht ausreichend. In der letzten Zeit ist eine Reihe methodischen Vorgehensweisen für die Entwicklung von Webanwendungen entstanden. Diese Ansätze haben zur Grunde Unified Modeling Language (UML) bzw. UML ähnliche Konstrukte, die sie um einige Elemente erweitern. Zu den prominentesten Beispielen von solchen Ansätzen werden heute unter anderen UML-based Web Engineering (UWE, näheres dazu siehe im Abschnitt 2.6.1), Object-Oriented Hypermedia Design Method (OOHDM näheres dazu siehe im Abschnitt 2.6.4), WebML (WebML, näheres dazu siehe im Abschnitt 2.6.2) und Objekt-Oriented Hypermedia (OO-H, siehe Abschnitt 2.6.3) gezählt. Im Folgenden werden einige Methoden kurz erörtert.

2.6.1 UWE

UWE ist ein objektorientierter Ansatz für systematische Entwicklung von Webanwendungen. Diese Methode baut auf der Unified Modeling Language (UML) auf und stellt im Wesentlichen eine Erweiterung von UML dar. Der Modellierungsprozess anhand UWE Ansatzes wird in vier folgende Phasen unterteilt:

- Anforderungsanalyse
- konzeptueller Entwurf
- Navigationsentwurf
- Präsentationsdesign

Aus einzelnen Entwurfsschritten erfolgen entsprechend der obigen Reihenfolge folgende Entwurfsmodelle:

- Anforderungsmodell (Requirements)
- konzeptuelles Modell (Content Model)

- Navigationsmodell (Navigation Model)
- Prozessmodell (Process Model)
- Präsentationsmodell (Presentation Model)

Einzelne Modelle bauen aufeinander auf (siehe Abbildung 1). Der größte Vorteil von solcher Aufteilung des Entwicklungsprozesses liegt in der Möglichkeit alle Teilaspekte unabhängig von einander zu modellieren.

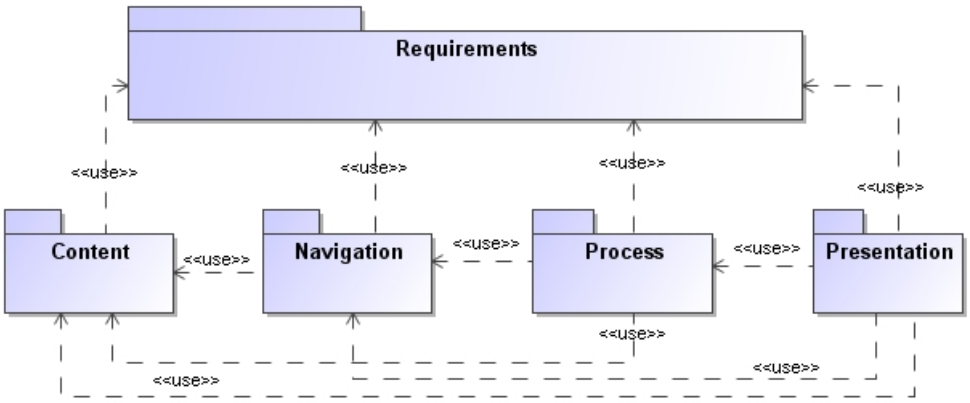


Abbildung 1. UWE Metamodell (Überblick)

Während der Anforderungsanalysephase werden anhand von UML-Anwendungsfalldiagrammen Anforderungen an die Webanwendung gesammelt und beschrieben. Danach wird das konzeptuelle Modell (Inhaltsmodell) aus den Anwendungsfällen abgeleitet. Im Inhaltsmodell werden alle Daten dargestellt, die eine Webanwendung zu verwalten hat. Das Inhaltsmodell wird in der Form von UML-Klassendiagramm realisiert. Im Anforderungsanalysemodell enthaltene Objekte liefern wertvolle Hinweise zu den möglichen Entitäten im Inhaltsmodell. Dabei beschreiben Klassen alle verfügbaren Entitäten in der Webapplikation, die zumindest zum Teil aus dem Anforderungsanalysemodell gewonnen werden. Aufbauend auf dem Inhaltsmodell wird das Navigationsmodell entwickelt. Dieses Modell stellt abstrakte Navigationsmöglichkeiten in der Webanwendung dar. UWE-Navigationsmodell basiert ebenfalls auf den UML-Klassendiagrammen. Das Navigationsmodell wird durch einen gerichteten Graph mit so genannten Navigationsknoten und Links als Kanten repräsentiert. Ein Navigationsknoten beschreibt dabei keine HTML-Seite oder ähnlich konkretes Konzept, sondern steht im Allgemeinen für eine navigierbare Einheit an Information oder Funktionalität in der Webapplikation (näheres dazu siehe in [24]). Im Prozessmodell werden alle Systemfunktionalitäten basierend auf den Inhalts- und Navigationsmodellen detailliert dargestellt. Dabei werden die UML-Aktivitätsdiagramme als Realisierungswerkzeug empfohlen. Anschließend wird das Präsentationsmodell als eine Skizze zur Benutzerschnittstelle von der Webanwendung erstellt. Im Rahmen dieses Modells wird es zwar von dem konkreten Layout auf der Benutzeroberfläche abstrahiert. Dennoch werden alle für die Präsentation wesentliche Elemente, wie Texte, Bilder, Verweise und andere, dargestellt.

Aktuelle Version von UWE (zurzeit 1.7) stellt eine konservative Erweiterung von UML 2.1 dar. Dabei ist die Erweiterung von UML in dem Sinne konservativ, dass die Semantik von UML in keiner Weise verändert wurde. Alle UML-Elemente behalten ihre Semantik bei und

können weiter verwendet werden. Zurzeit wird UWE als ein Plug-In für das kommerzielle Werkzeug „MagicDraw“ (siehe [40]) angeboten. Dieses Plug-In stellt eine leichtgewichtige Erweiterung von UML in Form eines Profils dar.

Ein Profil umfasst eine Menge von elementaren Erweiterungen des UML 2.x Metamodells - Stereotypen und spezialisiert UML-Element - „*Paket*“ (siehe [6], [9]). Ein Stereotyp gibt *in der Praxis vor allem die möglichen Verwendungszusammenhänge (Verwendungskontext) einer Klasse, einer Beziehung oder eines Paketes* an. (Wikipedia, Stand vom 01.11.2008) Basierend auf der Unterteilung des gesamten Modells in Inhalts-, Navigations-, Prozess- und Präsentationsmodelle werden im UWE Profil jeweilige Stereotypen definiert, die in entsprechenden Projektteilen auf die UML-Elemente angewendet werden können. Dabei bekommt dieses UML-Element eine Bedeutung im Kontext des auf ihn angewandten Stereotyps.

Basieren auf der oben beschriebenen Idee von Stereotypen wurden in UWE 1.7 folgende Stereotypen definiert. Für das Navigationsmodell wurden Stereotypen <<node>> und <<link>> definiert (siehe Abbildung 2). Von diesen Stereotypen wurden anhand Generalisierung weitere Stereotypen wie <<menu>>, <<navigationClass>>, <<accessPrimitive>>, <<navigationLink>> u.a., abgeleitet.

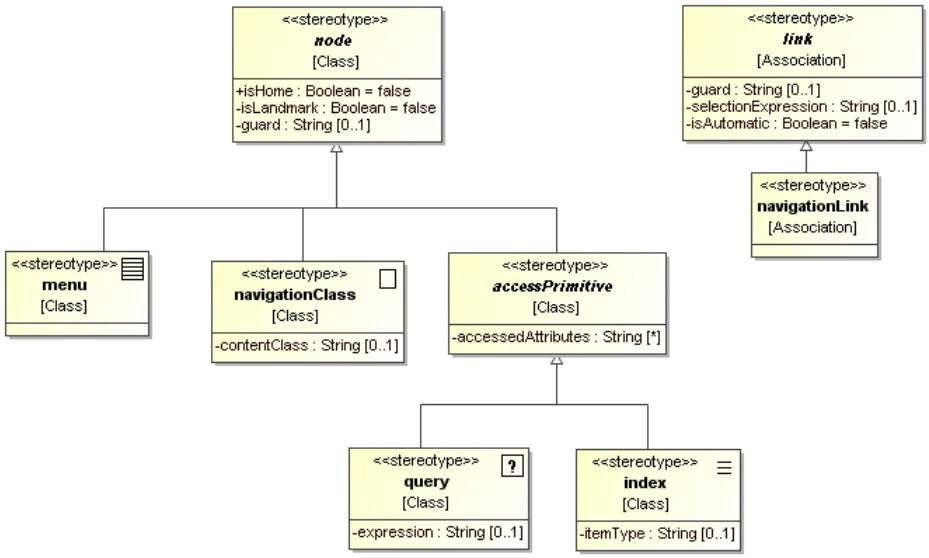


Abbildung 2. UWE Profile: Navigationsstereotypen

Für das Prozessmodell (siehe Abbildung 3) wurden analog <<processClass>> und <<processLink>> Stereotypen konzipiert, die eine Erweiterung von <<node>> und <<link>> Navigationsstereotypen darstellen. Der Stereotyp <<userAction>> kann auf eine Aktivität (Action) im Aktivitätsdiagramm angewendet werden. Dieser Stereotype entstand speziell für die Bezeichnung des Interaktionsprozesses zwischen dem Benutzer und dem System im Aktivitätsdiagramm. Dabei stellt er eine Blackboxsicht auf diese Interaktion dar, d.h. es wurde nicht weiter beschrieben wie diese Interaktion abläuft, sie wurde nur im Aktivitätsdiagramm angedeutet. In manchen Fällen ist es aber vom größten Interesse die Interaktionsabläufe genauer zu modellieren, und die Auslöseereignisse für bestimmte Prozesse genauer zu beschreiben bzw. von einander zu unterscheiden. Diese Mängel wurden in dieser Arbeit in Betracht gezogen. Dabei wurde als Lösung, zu der Beschreibung von komplexeren Interaktionen, Zustandsdiagramme vorgeschlagen (siehe Abschnitt 4.3).

Zusätzlich wurde „Event Language“ eingeführt (siehe Abschnitt 4.3.1), die eine Menge von Auslöseereignissen beschreibt, die während der Interaktion stattfinden können.

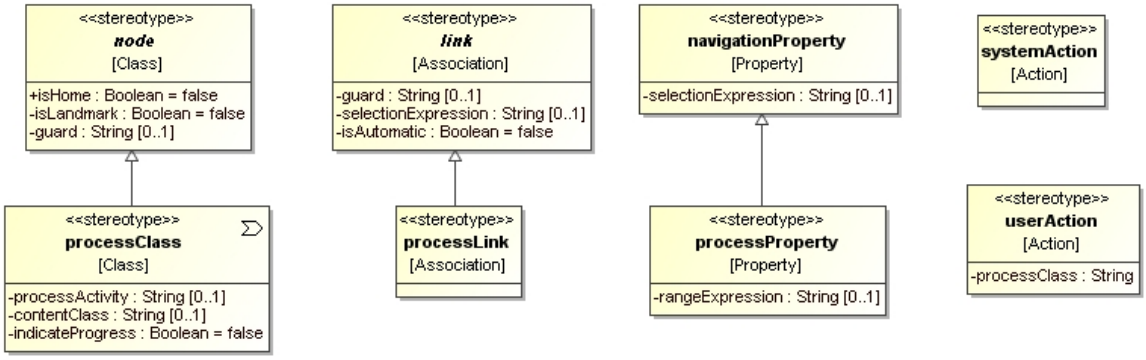


Abbildung 3. UWE Profil: Prozessstereotypen

Wie genau ein Stereotyp angewendet wird, sieht man deutlich auf folgendem Ausschnitt aus dem Navigationsmodell von S.CORE (siehe Abbildung 4).

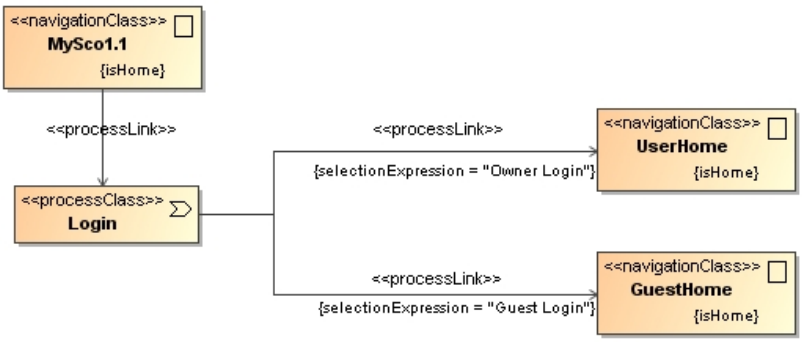


Abbildung 4. Login Verfahren bei S.CORE

Im Beispiel (siehe Abbildung 4) wurde auf die Klassen: MySco1.1, UserHome, GuestHome der Stereotyp <<navigationClass>> aus dem Navigationsteil des UWE Profils angewandt. Im Kontext des Ansatzes, der durch UWE Profil beschrieben wird, bedeutet dies, dass diese Klassen diverse Navigationsknoten in der entwickelten Applikation repräsentieren. Die Klasse Login trägt den Stereotyp <<processClass>> und stellt dabei eine Prozesseinheit in der Webanwendung dar.

Um die Bedeutung von manchen Stereotypen noch breiter und flexibler zu gestalten, können dazu noch einige Attribute als Metaattribute definiert werden. In diesem Fall sollen bei der Anwendung eines Stereotyps die eventuell definierte Metaattribute mit Werten belegt werden, dann spricht man über die so genannten „tagged values“. „*Tagged values – Metaattribut/Wertpaar ist eine Belegung der Attribute eines Stereotyps mit Werten.*“ (siehe [6], [9]). Im obigen Beispiel (siehe Abbildung 4) steht zum Beispiel tagged value {isHome} für die Belegung des Metaattributes „isHome“ des Stereotyps <<navigationClass>> mit dem Wert „true“.

Eine weitere Möglichkeit die Semantik von verschiedenen Modellelementen zu verfeinern, bietet noch ein weiteres UML Erweiterungskonstrukt - „Constraints“. Ein Constraint beschreibt die Semantik des Modelelements anhand einer Bedingung oder einer Beschränkung, die das entsprechende Element erfüllen soll. Diese Beschränkungen oder Bedingungen werden in der Form einer textuellen Anweisung beschrieben. Für Definition von Constraints im Modell können unterschiedliche Sprachen, wie zum Beispiel natürliche Sprachen (wie Englisch oder Deutsch), mathematische Notationen oder gar Object Constraint Language (OCL) (siehe [9]), verwendet werden. Ein Constraint kann ebenfalls im Profil definiert werden. Als Beispiel dazu kann man das Metaattribut „selectionExpression“ des Stereotyps <<link>> nennen (siehe Abbildung 2). Dieses Metaattribut beschreibt eine Bedingung während der Navigation innerhalb der Webanwendung. Die Belegung dieses Metaattributs jeweils mit den Werten „Owner Login“ und „Guest Login“ beschreibt die Auswahl bei der entsprechenden Navigation auf das UserHome bzw. GuestHome Navigationsknoten im S.CORE.

Da die wichtigsten Erweiterungen, die im Kapitel 4 vorgestellt sind, hauptsächlich das Präsentationsmodell betreffen, stellen im Rahmen dieser Diplomarbeit die Präsentationsstereotypen die interessanteste Erweiterung von UML dar (siehe Abbildung 5).

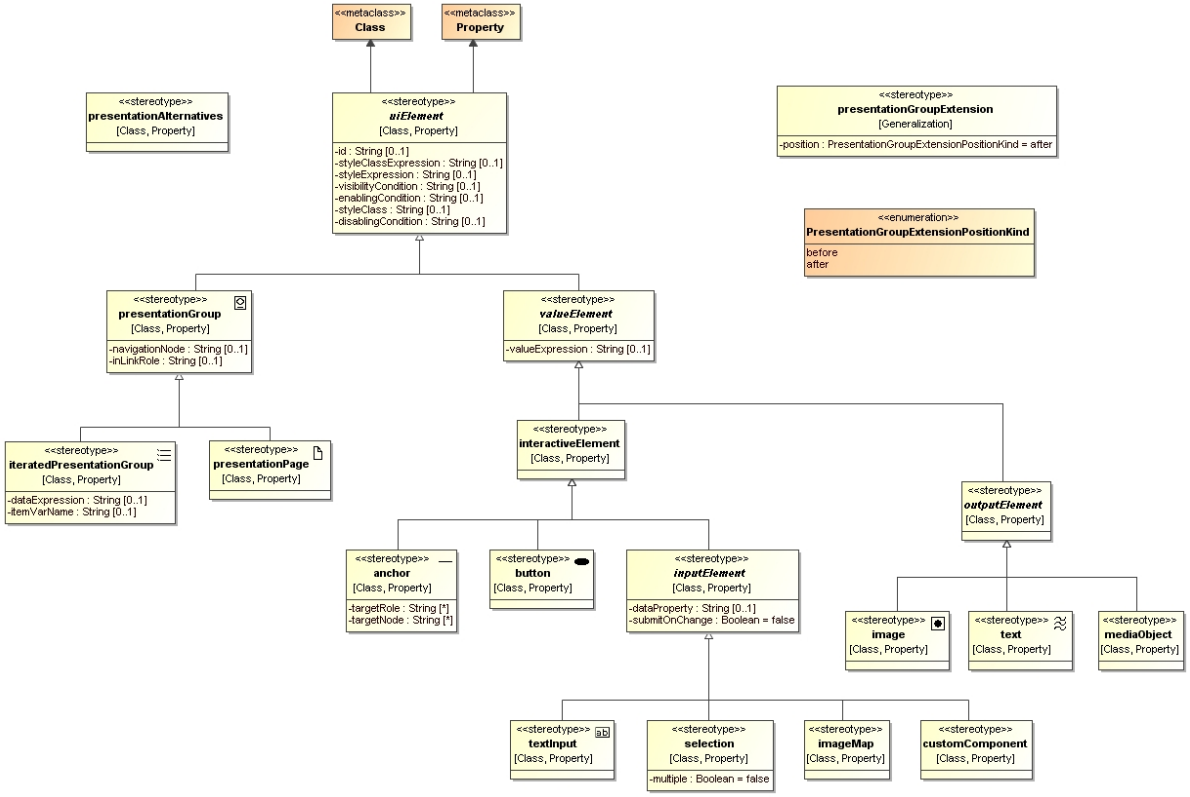


Abbildung 5. UWE Profil: Präsentationsstereotypen

In folgender Tabelle werden alle wichtigsten Präsentationsstereotypen des UWE Profils beschrieben. Eine detailliertere Beschreibung von UWE-Präsentaionselementen findet man in [26].

Stereotyp	Beschreibung
uiElement (user interface element)	Steht für die Bezeichnung eines Elements auf der Benutzeroberfläche im Modell.
presentationAlternatives	Bezeichnet ein Containerelement. Dieses Element hat keine visuelle Darstellung auf der Benutzeroberfläche in der Applikation. Beim Navigieren innerhalb der Webapplikation wird genau eins der Elemente aus diesem Container tatsächlich visualisiert, alle anderen stellen mögliche Alternativen dar.
presentationGroup	Bezeichnet eine Gruppe von Elementen auf der Benutzeroberfläche.
iteratedPresentationGroup	Steht für eine iterative Menge von Elementen auf der Benutzeroberfläche. Dieses Stereotype wird oft für die Darstellung von Indexen eingesetzt.
presentationPage	Bezeichnet die gesamte Menge von Elementen auf der Benutzeroberfläche von der Webapplikation. Dieses Element wird auf der oberste Ebene im Präsentationsmodell verwendet und darf nicht in einem anderen Container enthalten sein.
valueElement	Dient der Beschreibung eines Elementes auf der Benutzeroberfläche, das ein Wert beinhalten kann.
interactiveElement	Dient der Beschreibung eines Elementes auf der Benutzeroberfläche, mit dem der Benutzer interagieren kann.
anchor	Wird für die Darstellung eines Links der Webapplikation im Modell verwendet
button	Dient der Darstellung eines Buttons in der Webapplikation.
inputElement	Dient der Darstellung reiner Eingabeelementen auf der Benutzeroberfläche.
textInput	Repräsentiert ein Eingabetextfeld.
selection	Repräsentiert ein selektives Element auf der Benutzeroberfläche. Wird oft für die Darstellung von Checkboxen, oder Radiobuttons, oder ähnlichen Elementen verwendet.
imageMap	Dient der Darstellung von Bilder/Grafiken auf der Benutzeroberfläche über die eine Koordinatenmappe (Map) gelegt wurde. Mittels dieser Koordinatenmappe können nun auf dem Bild/ der Grafik verschiedene Bereiche markiert werden.
customComponent	Dient der Darstellung eines benutzerdefinierten Elements auf der Benutzeroberfläche. Meist sind es zusammengesetzte Elemente die ein Entwickler zu einer Einheit zusammengebracht hat.
outputElement	Dient der Darstellung von Elementen die reiner Datenausgabe auf der Benutzeroberfläche dienen.
image	Wird für die Darstellung eines Bildes auf der Benutzeroberfläche verwendet.
text	Repräsentiert ein Textfeld.

mediaObject	Repräsentiert ein mediales Objekt auf der Benutzeroberfläche.

Tabelle 7. Überblick über die Präsentationsstereotypen

Um die Gesamtstruktur von UWE anhand von Beziehungen zwischen den einzelnen Elementen zu beschreiben, wird in dem Zusammenhang mit UWE Profil das UWE Metamodell vorgestellt. Alle Assoziationen zwischen den UWE-Elementen werden im UWE Metamodell beschrieben und dargestellt. Auf der Abbildung 6 ist ein Ausschnitt aus dem UWE Metamodell vorgestellt. Dieser Ausschnitt betrifft das Präsentationsmodell und beschreibt alle Beziehungen zwischen den Elementen, die im UWE Profil vorgestellt wurden.

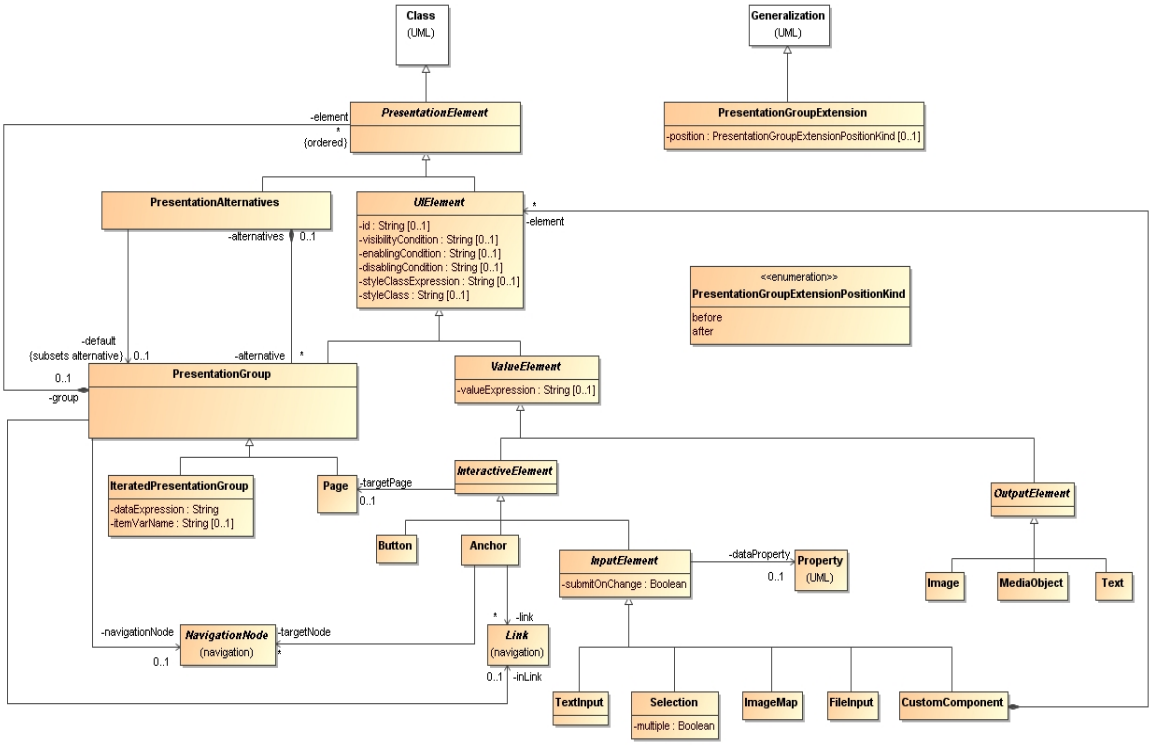


Abbildung 6. UWE Metamodell: Präsentationspaket

Die vorgestellte Vielfältigkeit von Modellelementen macht UWE zu einer mächtigen und flexiblen Modellierungssprache, die sowohl eine robuste Stütze während der gesamten Designphase für den Entwickler bietet als auch eine reale Basis für die automatische Generierung des Quellkodes aus den Modellen (näheres dazu siehe in [20], [25]) darstellt.

2.6.2 WebML

WebML (siehe [15]) stellt eine ausgereifte und sich in der Praxis bestätigte Methode vom Webengineering dar. Die Spezifikationssprache WebML nimmt ihr Ursprung im akademischen Bereich und ist durch Politecnico di Milano patentiert worden. Die Basis von diesem Ansatz wurde hauptsächlich von Stefano Ceri, Piero Fraternali und Aldo Bongio im Jahr 2000 ausgearbeitet.

Im Rahmen dieser Methodik werden analog zu UWE drei Aspekte (Strukturmodell, Hypertextmodell, Präsentationsmodell) im gesamten Entwicklungsprozess ausgegliedert. Die ersten zwei davon sind sogar ziemlich ähnlich in der Semantik und zwar das Strukturmodell (das dem Inhaltsmodell in UWE ähnelt) und das Hypertextmodell (das eben dem Navigationsmodell in UWE zu beschreiben ist). Im Strukturmodell werden alle für die Webapplikation relevanten Entitäten und deren Beziehungen definiert. Im Hypertextmodell, das seinerseits aus dem Kompositionsmodel und dem Navigationsmodell besteht, werden Sichten (Views) auf die Website spezifiziert. Das Kompositionsmodell teilt den gesamten Navigationsraum in Seiten (Pages) auf. Diese Seiten werden anschließend durch die Kompositionseinheiten (Content Units) beschrieben. Das Navigationsmodell beschreibt alle möglichen Beziehungen zwischen den Seiten (Pages). Das weitere Modell (Präsentationsmodell) wird sehr übersichtlich gehalten und der größte Anteil von der Komplexität wird dabei in die Implementierungsphase verlagert. Nicht desto trotz wird anhand des abstrakten XML- Syntaxen das graphische Erscheinungsbild der Seiten beschrieben. WebML-Seiten werden dabei gemäß eines Style Sheets erstellt.

Zur Zeit der Verfassung dieser Arbeit wird ein kommerzielles Case Tool – „WebRatio“ angeboten, das die gesamte Entwicklung einer Webanwendung anhand WebML unterstützt. Diese Kombination vom WebML und WebRatio bietet wenige Möglichkeiten zu der kompatiblen Wiederverwendbarkeit der Resultate in anderen Ansätzen auf diesem Gebiet. Dennoch wurde vor kurzem analog zu UWE ein Metamodell für WebML (siehe [16]) definiert, was die Interoperabilität des WebMLs mit anderen Ansätzen ermöglichen soll.

2.6.3 OO-H

Der Object- Oriented Hypertext (OO-H) Ansatz für Entwicklung und Implementierung von Webanwendungen ist ebenfalls eine ausgereifte Methode, die sehr ähnlich zu UWE und WebML ist. Analog zu den oben beschriebenen Methoden findet man die Unterteilung des Entwicklungsprozesses in mehrere Etappen (siehe [12], [13]). Das Inhaltsmodell wird dabei anhand von klassischen Klassendiagrammen von UML beschrieben. Für die Beschreibung des Navigationsraums in der Anwendung werden sogenannte Navigation Access Diagrams (NADs) erstellt. NADs sind an die UML- Klassendiagramme angelehnt aber die Semantik von Modellelementen entspricht nicht der UML-Semantik. Abgesehen davon, NADs besitzen zusätzliche Navigations- und Interaktionseigenschaften, die durch eine Object Constraint Language (OCL) ähnliche Sprache definiert werden. Mittels Abstract Presentation Diagrams (APDs) und Composite Layout Diagrams (CLDs) werden jeweils alle Konzepte bezüglich der abstrakten Struktur der Applikation und spezifischen Präsentationsdetails dargestellt. Die APDs werden durch ein auf XML-basiertes Template-Mechanismus erstellt. In OO-H wurde dafür eine Menge von Templatestypen definiert, die anhand XML beschrieben werden.

Eine Besonderheit, die der OO-H Ansatz ausmacht, ist das Erstellen eines Musterkatalogs, das immer weiter entwickelt und gepflegt wird. In dem Katalog enthaltene Musterlösungen stellen eine Abhilfe für viele bekannte Probleme im Webengineering bereit und berücksichtigen unterschiedliche Sichtwiesen des Benutzers. Dieser Musterkatalog kann sowohl bei der Erstellung NADs als auch bei der Konzipierung von APDs verwendet werden.

Das Interessante an dieser Methode ist, dass im Rahmen der Erweiterung von OO-H eine Herangehensweise (näheres dazu siehe in [17], [39]) zur Modellierung einer Rich Internet Application (RIA, näheres dazu siehe im Abschnitt 2.3) entwickelt wurde. Diese Methode beschreibt einen modernen Ansatz für Transformation einer herkömmlichen Webanwendung

zu einer RIA und hat ihr Ursprung im modelbasierten Refactoring. Das modelbasierte Refactoring ist *„modelbasierte Strukturverbesserung von Programm-Quelltexten unter Beibehaltung des bestehenden Programm- Verhaltens.“* (Wikipedia, Stand vom 10.10.08) Dieser Ansatz wird durch Spezifikation der zu erstellenden Benutzeroberfläche anhand sogenannter abstrakter Sichten auf das Datenmodell (Abstract Data Views, ADVs) formalisiert. Dabei wird Struktur der Modelle auf der Präsentationsebene anhand interaktiver Benutzerschnittstellen verbessert (siehe [17]).

Die Methode ist sehr interessant und hilfreich. Zu bemängeln wäre aber die Tatsache, dass es erst eine herkömmliche Webanwendung modelliert werden soll und danach anhand vom modelbasierten Refactoring eine Verbesserung des Modells durchgeführt werden soll. Dabei mutieren mögliche Erweiterungen des Systems zu einer aufwendigen und Zeit raubenden Angelegenheit, da zum einen das herkömmliche Modell erweitert werden soll und danach das Refactoring durchgeführt werden soll.

2.6.4 OOHDM

Object-Oriented Hypermedia Design Method (OOHDM) ist eine weitere Methode zu dem Entwurf von Webanwendungen (siehe [14]). Diese Methode basiert auf dem Hypertext Design Model (HDM), das eine Strukturierungssprache auf Basis der Entity-Relationship (E-R) Methode darstellt, wo der Entwicklungsprozess anhand eines zweistufigen Modells (Hyperbasis (hyperbase layer) und Zugriffstruktur (access layer)) beschrieben wird.

Die Besonderheit und das eigentliche Unterschied der OOHDM von HDM liegen zum einen in der Objektorientierung. Zum anderen besteht in der OOHM die Möglichkeit mögliche Benutzerschnittstelleninteraktionen anhand Abstract Data Views (ADVs) explizit zu modellieren. Während der Entwicklung einer Webanwendung wird in OOHDM vier Phasen unterschieden, und zwar konzeptueller Entwurf (conceptual design), Navigationsentwurf (navigational design), abstrakter Schnittstellenentwurf (abstract interface design) und Implementierung (implementation).

Während der Phase des konzeptuellen Entwurfes wird ein Domänenmodell erstellt, wo möglichst alle Objekte der Anwendung und deren statische Beziehungen zu einander beschrieben werden. Daraufhin wird das Navigationsmodell, als kontextabhängige Sicht auf das konzeptuelle Modell, entwickelt. Daraufhin können mehrere Navigationsmodelle, somit mehrere Sichten, zu einem Domänenmodell existieren. Im Schnittstellenentwurf wird die Benutzeroberfläche mittels ADVs erstellt, dadurch wird festgelegt, wie die navigierbaren Objekte innerhalb eines bestimmten Navigationskontextes dargestellt werden. Dennoch tragen die ADVs für die Implementierungsphase der Entwicklung nur informellen Charakter, da es keine Abbildungsvorschriften in OOHDM existieren.

Als ein Schwachpunkt der Methode hat sich die Erweiterbarkeit der Modellierungsgrundlage erwiesen, da alle Phasen der Entwicklung erneuert durchlaufen werden müssen, was die Pflege des gesamten Systemmodells sich als äußerst aufwändig herausstellt.

Kapitel 3 S.CORE

Die S.CO LifeScience GmbH ist als „spin-off“- Gesellschaft des Institut für medizinische Technik (ImedTUM ehemals ZIMT) der Technischen Universität München entstanden und wurde im Oktober 2004 offiziell gegründet. Die S.CO LifeScience GmbH beschäftigt sich mit der Entwicklung und Vermarktung innovativer bildanalytischer Lösungen mit dem Hauptfokus auf der Zellbiologie, Medizin und Pharmazie. Im Rahmen der Firmentätigkeit wurde ein System für die automatische Bildanalyse entwickelt, das auf der revolutionären Bilderkennungstechnologie „eCognition“ der „Definiens AG“ (Partnergesellschaft) basiert. Die eCognition Language ist eine Metasprache, welche ermöglicht bildanalytische Prozesse sequentiell ablaufen zu lassen. Desweiteren greift S.CORE auf Algorithmen des Open Source Projektes „ImageJ“ zurück, vor allem um Bilder vor der Analyse vorzubearbeiten.

S.CORE steuert den gesamten Bildanalyseablauf von der Entgegennahme der Bilder, über die eigentliche Analyse bis hin zur Erzeugung statischer Daten und kann dann die erzeugten Daten entsprechend der Kundenwünsche statistisch nachbearbeiten.

Eine der Mensch-Maschine Schnittstellen von S.CORE ist ein webbasiertes Interface welches über das Internet erreichbar ist. Dies bietet für den Kunden den Vorteil, dass er keinerlei Softwareinstallation bei sich vornehmen muß.

Die intuitive Benutzerschnittstelle im Web kann mit einem Standardwebbrowser bedient werden (es werden folgende Webbrowser empfohlen: Microsoft Internet Explorer 7, Mozilla Firefox 2 und 3, Apple Safari).

Im Folgenden wird auf die Möglichkeiten, die S.CORE einem Benutzer bietet, detailliert eingegangen, um sich ein Überblick über die Funktionalitäten von S.CORE zu verschaffen

3.1 Funktionalitäten

Wie schon oben angedeutet wurde, bietet S.CORE (siehe [w6]) eine Benutzerschnittstelle im World Wide Web für die automatische Bildanalyse. Die meisten Analysen dienen Anwendungen in der Medizin, wie z.B. das Zählen von bestimmten Zellen in einem Gewebe oder die Klassifizierung einer bestimmten Zellart. Die verwendeten Bilder stammen meist aus Konfokalmikroskopen. (*„Ein **Konfokalmikroskop** ist eine Variante des Lichtmikroskopes, hauptsächlich des Fluoreszenzmikroskops, mit dem virtuelle optische Schnitte durch ein Objekt erzeugt werden können. Diese Schnittbilder können anschließend durch geeignete Software zu einer räumlichen Darstellung zusammengesetzt werden.“* Wikipedia, Stand vom 17.11.08) Allerdings sind auch viele weitere Mikroskoptypen denkbar und werden gelegentlich auch verwendet. Die größte Innovation und der entscheidende Unterschied zu anderen existierenden Systemen liegen in der Webbasierung. Dies macht das System zu einem flexiblen und mächtigen Werkzeug auf diesem Gebiet.

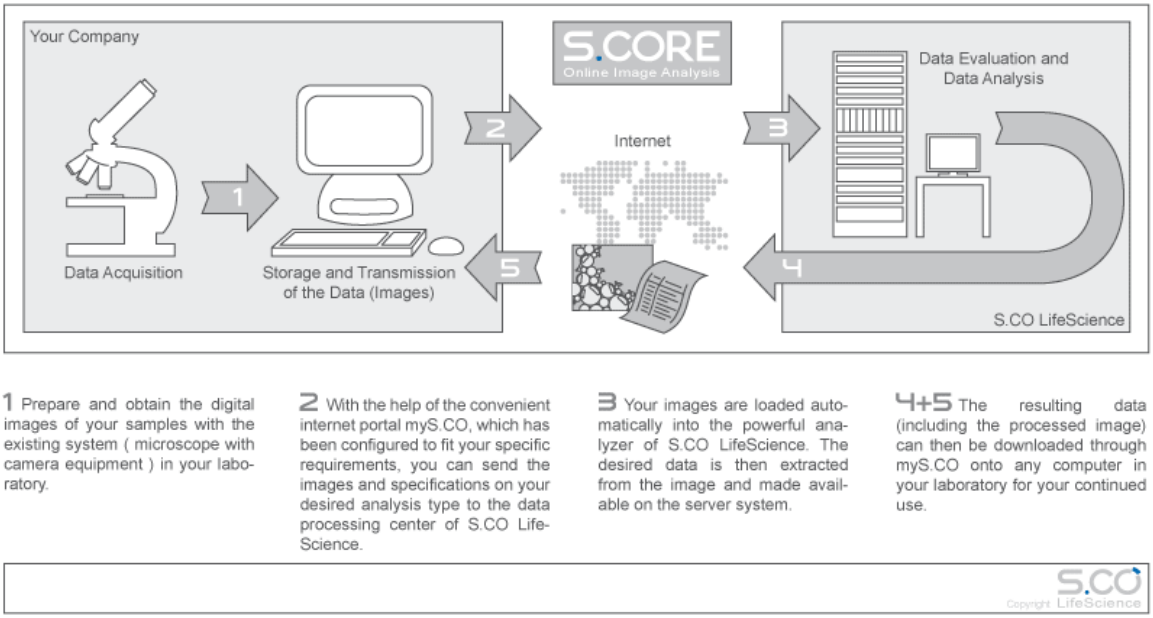


Abbildung 7. Bildanalyse durch S.CORE System (Bildquelle [w6])

Die wichtigsten Schritte im Analyse Prozess sind auf der Abbildung 7 angezeigt. (Die folgende Schrittbeschreibung 1.- 5. bezieht sich auf die Schrittnummerierung auf der Abbildung 7.)

1. Der erste Schritt beschreibt die Vorbereitung eines digitalen Bildes zur weiteren Analyse. Das Bild kann z.B. durch ein Mikroskop mit einer integrierten Kamera erstellt werden und danach auf einen Computer übertragen werden.
2. Im zweiten und dritten Schritt wird der Benutzer durch S.CORE bei dem Hochladen und Verschicken des Bildes unterstützt.
3. In dem vierten und fünften Schritt wird die Analyse durch die Analyse Engine in dem Rechenzentrum der S.CO LifeScience GmbH durchgeführt und die Resultate im Internet zur Verfügung gestellt. Diese Resultate kann der Benutzer entweder direkt im Internet betrachten, oder auf seinen Computer herunterladen.

Die Registrierung zur Mitbenutzung erfolgt ebenfalls im Netz. Dafür sollte ein im Internet angebotenen Formular ausgefüllt und verschickt werden. Jedem neuen Benutzer wird sofort das Analysemodul („Feasibility Study“) zum Ausprobieren von Bildanalyse zugeteilt. Danach können, nach der Absprache mit dem Personal der S.CO LifeScience GmbH, weitere Module gekauft werden.

Zur Zeit der Verfassung der vorliegenden Arbeit ist die Version 1.1 von S.CORE im Einsatz. Ein Überblick über die wichtigsten Funktionalitäten, die in dieser Version realisiert sind, bietet folgendes Use Case Diagramm (siehe Abbildung 8)

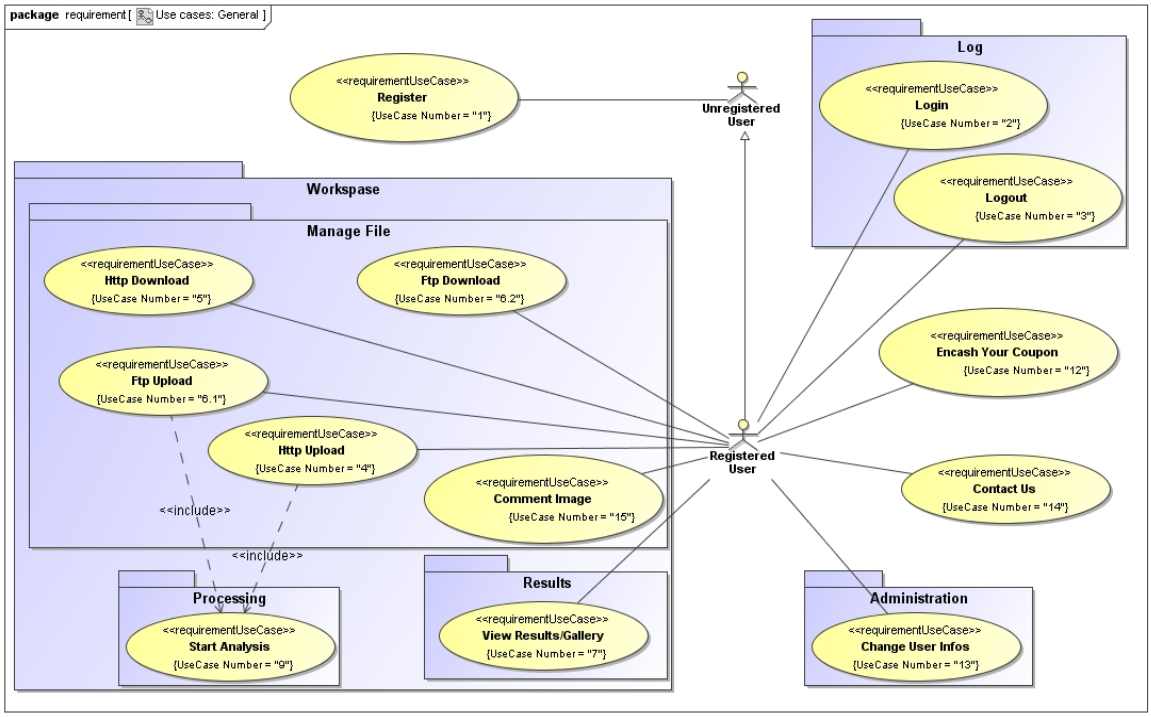


Abbildung 8. S.CORE 1.1 Use Case Diagramm

Dem Diagramm folgend bekommt ein angemeldeter Benutzer ein virtuelles Workspace in S.CORE zugeteilt. Direkt nach der Anmeldung gelangt der Benutzer zu seiner Workspaceoberfläche (siehe Abbildung 9). Im Workspace sind Informationen über die Analysen, die dem Nutzer zur Verfügung stehen, vorhanden. Diese Informationen sind im Baum rechts im Fenster „My Modules“ zu finden. Zum Beispiel sind im aktuellen Workspace auf der Abbildung 9 Analysemodule: „Scrach Assay“ und „Sprouting Assay“. Diese Analysen stehen dem Benutzer – „Tatiana Morozova“ zur Verfügung. Durch Anklicken vom bestimmten Analysemodul gelangt man zu den Bildanalysefunktionalitäten vom S.CORE. D.h. man kann Bilder, die diesem Analysemodul entsprechen, in Benutzerworkspace hochladen und dessen Analyseprozess starten. Die Art wie das Bild analysiert werden soll, bestimmt der Name des Analysemoduls in dem man sich befindet (d.h. das „Aktive Modul“). Analog dazu sind auch die Resultate zu den Bildanalysen bezüglich des aktiven Moduls in dem Benutzerworkspace zu finden (siehe „View Results/Gallery“ auf der Abbildung 8). Bezüglich der Down- bzw. Uploadart werden im S.CORE zwei Varianten angeboten. Es kann der Down- bzw. Upload mittels Hyper Text Transfer Protocol (HTTP) oder File Transfer Protocol (FTP) erfolgen. Jeder Upload startet automatisch der Bildanalyseprozess (siehe <<include>> Beziehung zwischen „Http- bzw. Ftp Upload“ Use Cases und dem „Start Analysis“ Use Case auf der Abbildung 8).

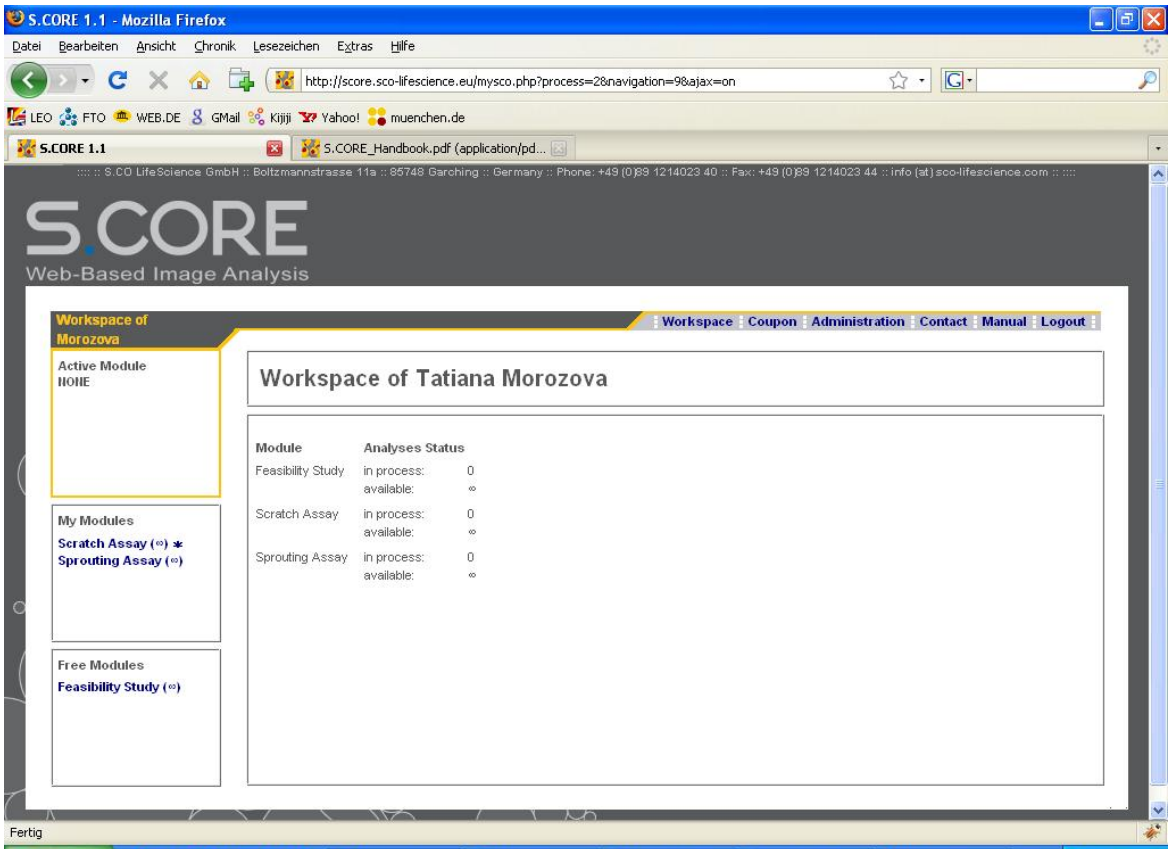


Abbildung 9. S.CORE Benutzer Workspace

Die Hauptnavigation auf der Seite erfolgt durch die Tabs (siehe oben rechts auf der Abbildung 9): Workspace, Coupon, Administration, Contact, Manual und Logout. Das Tab „Coupon“ entspricht dem Use Case „Encash Your Coupon“ und bietet dem Benutzer eine Möglichkeit ein vom S.CORE erstellte Coupon einzulösen. Durch das Einlösen vom Coupon wird es dem System mitgeteilt, wie viele Bilder zur Analyse diesem Benutzer freistehen und in welchen Analysemodulen. Diese Bilderanzahl steht dann unmittelbar in Klammern neben den Namen von dem jeweiligen Analysemodul. Im Beispiel auf der Abbildung 9 hat der Benutzer Tatiana Morozova das Recht auf die unbegrenzte Bilderanzahl in allen Analysemodulen (repräsentiert durch ∞ Symbol in Klammern neben dem Modulnamen, siehe z.B. Scrach Assay (∞) auf der Abbildung 9). Das Navigationtab – „Administration“ verwirklicht Use Case - „Change User Info“ auf dem Use Case Diagramm (siehe Abbildung 8). In diesem Bereich kann der Benutzer sein Profil pflegen, in dem er z.B. seine Adresse angibt oder ändert, oder seine Telefonnummer oder gar sein Password bzw. sein Benutzername austauscht. Im Navigationsbereich - „Contact“ (entspricht dem Use Case „Contact Us“ auf der Abbildung 8) findet man nötige Informationen zu seiner Kontaktperson und man kann direkt im S.CORE eine E-Mail-Nachricht für diese Person verfassen und verschicken. Die Tabs Manual und Logout sprechen für sich. Im Manual findet man eine ausführliche Beschreibung zur Systembedingung und durch Logout meldet man sich beim System ab.

3.2 Technische Grundlagen

S.CORE ist in PHP, Java und JavaScript implementiert und läuft auf einem Server mit AMD-Prozessoren, 8 GB Arbeitsspeicher, 4 Festplatten von insgesamt 1 Terabyte Speicherkapazität und 2 Netzwerkkarten (im weiteren „*Helios*“). Die eigentliche Bildanalyse läuft auf einem anderen Server mit zwei Intel Premium- 4 Kern Prozessoren, und 2 Netzwerkkarten einem 16 GB Arbeitsspeicher (im weiteren „*Eos*“). Die Zusammensetzung von der gesamten Infrastruktur findet man auf der Abbildung 10.

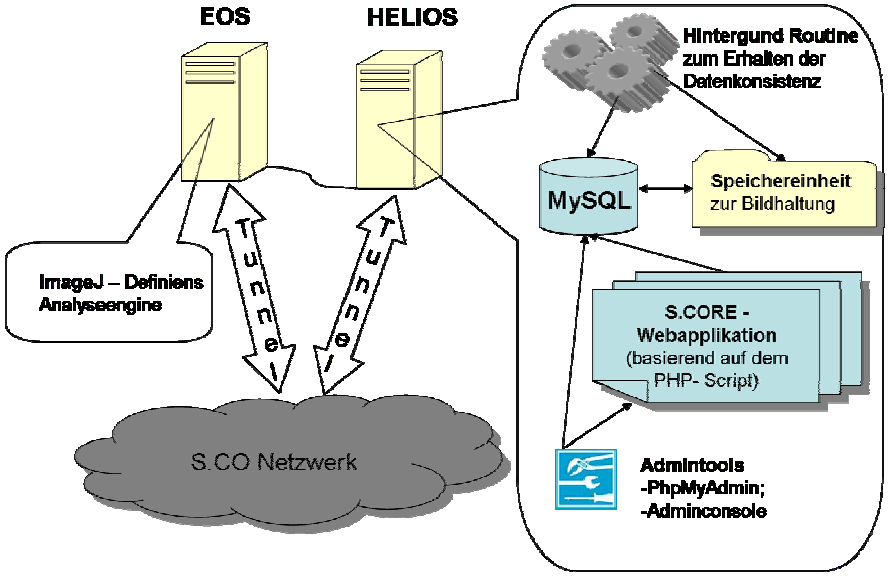


Abbildung 10. Infrastruktur bei S.CO LifeScience GmbH

Die zwei Server „Eos“ und „Helios“ sind mit einander mit einem LAN- Netzwerkkabel verbunden. Dabei laufen auf „Eos“ die Bildanalyseengines „ImageJ“-Algorithmen und die eCongnition Engine von Definiens auf „Helios“ laufen alle dazugehörigen Dienste, unter anderen S.CORE. Wie es auf dem Bild angedeutet ist, werden die Bilder nicht direkt in der Datenbank gespeichert, sondern in einer Speichereinheit. Diese Trennung ist aus Performancegründen notwendig. Denn aufgrund von enormen Bildergrößen wären sonst die Anfragen an die Datenbank sehr langsam und träge, was die Performance des gesamten Systems stark beeinträchtigen würde.

Eine weitere Besonderheit bezüglich der oben beschriebenen Trennung ist die **Hintergrund Routine** zum Erhalten der Datenkonsistenz, die auch auf der Abbildung 10 angezeigt ist. Die Notwendigkeit dieser Routine kann man durch eine Besonderheit vom PHP erklären, und zwar im PHP gibt es kein Konstrukt zur Implementierung von Threads. Somit soll die Datenstandüberprüfung in einen externen Prozess verlagert werden. Denn falls in der Webapplikation neue Bilder zur Analyse hochgeladen wurden, sollen die sich auch in der Speichereinheit befinden, und umgekehrt sollen dessen Namen auch in die Datenbank eingetragen werden.

Die vereinfachte Systemarchitektur von S.CORE sieht man auf dem Klassen Diagramm (siehe Abbildung 11 unten). Die wichtigsten Pakete im System sind **Infrastructure**, **ContentUser** und **Html_templates**.

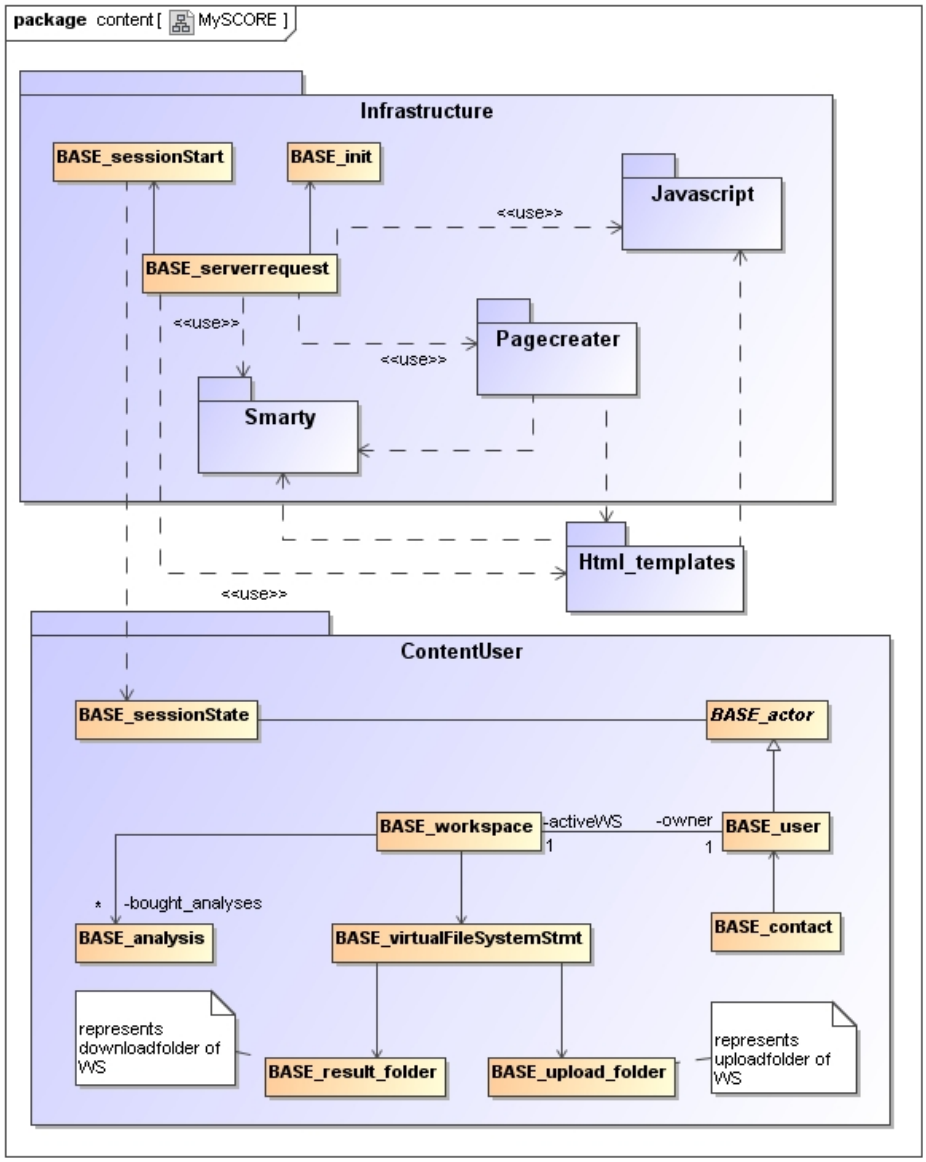


Abbildung 11. S.CORE Klassendiagramm Überblick

Im Paket – „**ContentUser“** sind alle wichtigsten Klassen zusammengestellt, die einen Benutzer charakterisieren. Dabei sind sowohl persistierbare Daten als auch nicht persistierbare Daten in diesem Paket enthalten. Dabei persistierbare Daten sind Daten deren Lebensdauer länger sein kann als eine Sitzung. Zu solchen Daten kann man beispielsweise gekaufte Analysismodule, Anzahl der Bilder, die freigegeben zur Analyse wurden, oder die gesamte Informationen über den Benutzer, usw. zählen. Währenddessen nicht persistierbare Daten sind solche, die sich auf eine Session beschränken. Dazu zählen Informationen bezüglich der Position des Benutzers im Navigationsraum der Webseite und viele andere. Die persistierbare Daten werden in Klassen, wie **BASE_user, BASE_workspace, BASE_analysis, BASE_virtualFileSystemStmt** oder **BASE_contact** dargestellt (siehe Abbildung 11). Die Klasse **BASE_user** auf der Abbildung 11 repräsentiert den Benutzer selbst mit allen seinen Eigenschaften (wie Name, Adresse usw.). Die **BASE_workspace** Klasse steht für Workspace, das jedem Benuzter nach der Registrierung zugeteilt wird. Dies

wird durch die Beziehung zwischen den **BASE_user** und **BASE_workspace** Klassen repräsentiert (siehe Abbildung 11). Die eins zu eins Multiplizität besagt dabei, dass es für jeden Benuzter genau ein Workspace im System existiert und umgekehrt für jedes Workspace genau einen Benutzer existieren kann. Klasse **BASE_analysis** repräsentiert ein Analysismodul, das einem Benutzer in seiner Workspace zugeteilt werden kann (repräsentiert durch **-bought_analysis** Rolle in der gerichteten Assoziation zwischen den **BASE_workspace** und **BASE_analysis** Klassen. Die Multiplizität (*) bei dieser Assoziation deutet darauf hin, dass es von 0 bis unbegrenzter Anzahl Analysismodulen jedem Benutzer zugewiesen werden kann). Klasse **BASE_virtualFileSystemStmt** repräsentiert ein virtuelles File System im Workspace, wo Klassen **BASE_result_folder** und **BASE_upload_folder** dieser Reihenfolge nach Verzeichnisse für Resultate und für hochgeladene Bilder darstellen. Die Klasse **BASE_contact** repräsentiert eine Kontaktperson, die einem Benutzer zugewiesen wird. Die nicht persistierbare Daten bezüglich einer Session werden durch die Klasse **BASE_sessionState** dargestellt.

Die sogenannte Verwaltungsfunktion über dem gesamten System übernimmt „**Infrastructure“-** Package. Dabei spielt die Klasse **BASE_serverrequest** eine übergreifende Rolle im gesamten System. Sie greift auf JavaScript-Funktionen (auf der Abbildung 11 **Javascript** Package) und regelt die gesamte Dynamik beim Erstellen von HTML-Seiten während einer Benutzersession (Spinnefunktion). Ein weiteres wichtiges Konstrukt das in S.CORE integriert wurde, ist Smarty Engine. Dies ist: „*eine quelloffene Template Engine, die als (LGPL) PHP-Bibliothek vorliegt. Sie wurde mit dem Ziel entworfen, bei der Entwicklung von Webapplikationen die **Trennung von Code und Ausgabe** zu ermöglichen. Die Ausgabe erfolgt meist in HTML, möglich ist jedes textbasierte Dateiformat, zum Beispiel auch XML.*“ (Wikipedia, Stand vom 10.10.2008).

Diese Trennung der Ausgabe vom Code findet im **Pagecreater** Package dadurch statt, dass bestimmte *„Platzhaltervariablen“*, die im HTML-Dokument aufbauend auf dem **Html_templates** Package deklariert werden. Diese Platzhaltevariablen werden im **Pagecreater** mit entsprechenden Werten belegt. Im folgenden Beispiel wird dies deutlicher.

Ein Auszug aus der Source Datei Html_templates :: add_new_guest_ajax.tpl

```
<p>
        {$note}
</p>
```

Variable „note“ in geschweiften Klammern ist eine Smarty Variable und repräsentiert die oben beschriebene Platzhaltervariable. Das „$“ Zeichen davor ist ein üblicher Zeichen für Variablendeklaration in PHP. Diese Variable wird erst in der build_add_guest_page_ajax.inc Datei aus dem **Pagecreater** Package mit dem Wert L_guest_name_description belegt (siehe Auszug unten).

Auszug aus dem Quellcode in der Pagecreater:: build_add_guest_page_ajax.inc

```
$smarty->assign("note", language::L_guest_name_description);
```

Und schon hier wird dem PHP Compiler mitgeteilt, wo die Variable L_guest_name_description zu finden ist, und zwar im Package – „language“. Nach ein Paar weiteren Schritten wird auch diese Variable gefunden und aufgelöst und zwar:

```
const L_guest_name_description = "A guest name should be a valid email adress";
```

Diese auf ersten Blick aufwendigen Schritte lohnen sich in der Hinsicht auf das Trennen des Codes von der Ausgabe und in der Hinsicht, dass die gesamte Menge von allen Möglichen Labels und Error- Messages u.a. in einer Datei gesammelt wird. Dies zieht mit sich eine enorme Erleichterung bei häufigen Änderungen von Meldungstexten und macht es möglich, dass der Programmierer, sich gar nicht um die Texte kümmern muss. Der Entwickler skizziert nur die Überschriftstexte, um die genauen Formulierungen kann sich eine weitere Person kümmern. Programmierkenntnisse sind dabei nicht erforderlich. Daraus kommt noch ein weiteres Vorteil zum Vorschein, nämlich eine Integration von Designers, Übersetzers und weiteren Personen in die Entwicklung von Webanwendung.

Nach dem Anmelden eines Benutzers bei S.CORE wird in der **BASE_sessiontStart** Klasse ein Objekt der **BASE_sessionState** Klasse erstellt wo alle Benutzerspezifische Informationen abgefragt werden können. Ein Benutzer hat ein Workspace (repräsentiert durch **BASE_workspace** Klasse im Klassendiagramm) in dem alle benutzerspezifische Informationen, sowie vom Benutzer gekaufte Analysis Module (siehe **BASE_analysis** Klasse) und die Pfade zu den Resultaten der analysierten Bilder (siehe **BASE_result_folder**) und vieles Anderes, zur Verfügung stehen. Detailliertere Klassendiagrammen und Modelle bzgl. S.CORE siehe im Anhang (auf CD-ROM).

3.3 Erweiterung von S.CORE 1.1

Im Rahmen dieser Arbeit wurde S.CORE um einige Funktionalitäten erweitert. Das Ziel von diesem Abschnitt ist die erfolgte Erweiterung detailliert zu beschreiben und deren Funktionalität und Entwicklung zu schildern.

3.3.1 Problembeschreibung

Bis vor kurzem wurde S.CORE für einzelne User nur als Einzelbenutzerbetrieb entwickelt und implementiert. Es hat sich aber folgende Erweiterung als erforderlich herausgestellt: für ein registrierter Benutzer (im Weiteren *Owner*) soll die Grundlage der ***Mehrbenutzersystem*** geschaffen werden, indem ein *Owner* andere Personen (im Weiteren Gäste) zum eingeschränkten Mitbenutzen seines Benutzerkontos einladen kann. Die Erweiterung von S.CORE bezüglich *Owner* bezieht sich auf den Administration Bereich der Webanwendung.

3.3.2 Anforderungsanalyse

In diesem Abschnitt erfolgt eine textuelle Beschreibung der Anforderungen die für die Systemerweiterung erforderlich sind.

1. Mögliche Nutzer der Systemerweiterung:

- alle angemeldete Benutzer *(Owner)* im S.CORE;

- alle eingeladene Gäste *(Gast)*;

(Somit wurde die Menge von möglichen Systemnutzern um einen neuen Aktor - *Gast* erweitert)

2. Anforderungen:

a) bzgl. Owner:

- im Administration Bereich neue Erweiterung: „*Guest Administration*“;
- im „*Guest Administration*“ Bereich:
  - MyGuests – Liste von allen eingeladenen Gästen;
  - AddGuest – Neuen Gast einladen, dabei wird einen neuen Gästeaccount erstellt und an den Gast wird eine Einladungs-E-Mail gesendet, mit dem Loginname, der gleich seiner E-Mail-Adresse, und Password, der gleich einem Zufallsstring ist. Mit diesen Daten kann dann der neue Gast sich bei S.CORE anmelden;
  - DeleteGuest – Den schon existierenden Gast löschen;

b) bzgl. Gast:

- soll sich beim S.CORE System anmelden können. Das Login wird mit dem *Loginname* (wird beim Erstellen vom Gästeaccount vergeben und kann nicht mehr geändert werden) und mit dem Password (beim Erstellen vom Gästeaccount wird durch Randomverfahren ein Zufallsstring generiert, und als Password in die Datenbank eingetragen. Dieses Password kann aber später im Administration Bereich vom Gästeaccount geändert werden.) durchgeführt.

- der angemeldete Gast kann ein Teil von *Owners*workspace mitnutzen, indem er die Resultate von den durchgeführten Analysen anschauen und herunterladen kann, sein Profile pflegen kann und sich in die Verbindung mit der dem *Owner* zugeteilten Kontaktperson setzen kann.

3. Use Case Diagramm:

Auf dem folgenden Diagramm sind Use Cases dargestellt, die für die beschriebene Systemerweiterung nötig sind (siehe Abbildung 12 unten). Alle Use Cases beschreiben neue Funktionalitäten, die durch die Erweiterung von S.CORE zur Verfügung gestellt wurden. Dabei stellen die Beziehungen zwischen den jeweiligen Aktoren und den Use Casen die Zuordnung zwischen den Aktoren und deren Funktionalitäten dar.

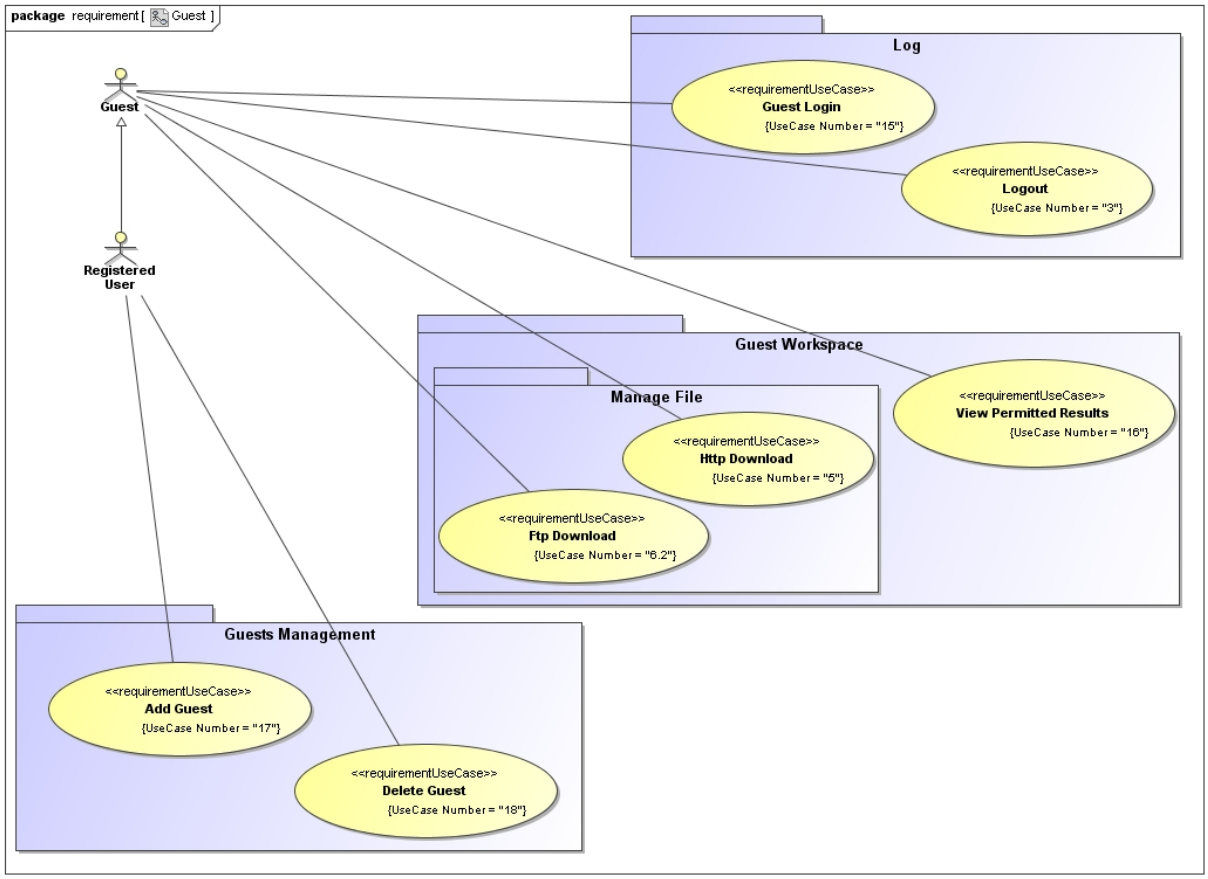


Abbildung 12. Erweiterung vom UseCase Diagramm bzgl. Gästefunktionalität.

4. Funktionsbeschreibungen:

Im Folgenden wird es noch detaillierter auf die neue Systemfunktionalitäten eingegangen.

a) *bzgl. Owner:*

- **addGuest()**

  Auslöser: *Owner* tätigt im „*Guest Administration*“ Bereich der „*Add new guest*“-Button.

  Standard Ablauf:

  1. Es erscheint ein Formular zum erstellen des neuen Gastes.
  2. *Owner* füllt der Formular aus, wo er unbedingt: die E-Mail-Adresse des Gastes angeben soll, und die Analysen, die später dem Gast zum Mitbenutzen stehen sollen, auswählen soll;
  3. *Owner* klickt auf „Create guest account“-Button;
  4. Formulardaten werden überprüft (prüfe E-Mail-Adresse, ob die schon in der DB ist und ob die ein gültiger E-Mail Server (bzw. Domän) enthält, prüfe ob mindestens eine Analyse dem Gast zugeteilt worden ist (d.h., ob der Analysis Checkbox den Status „checked“ besitzt));

5. Der eingeladene Gast bekommt eine Einladungs E-Mail (an die Adresse aus dem Formular, mit der Information über Logindaten);
6. DB anpassen (Zu der Gästeliste von diesem *Owner* wird ein neuer Gast hinzugefügt);

Alternative Abläufe:

I. *Owner* bricht ab:

1. Punkte 1. bis 2. der Standard Ablauf;
2. *Owner* klickt auf „Abbrechen“;
3. Abbruch.

II. Formulardaten sind nicht korrekt:

Statisches Verfahren:	Dynamisches Verfahren (JavaScript, AJAX):
1. Punkte 1. bis 4. der Standard Ablauf; 2. Falls Fehler festgestellt wurden => Fehler Meldung(en) ausgeben und zum Punkt 1. übergehen, sonst zum Punkt 3.; 3. Punkte 4. bis 6. der Standard Ablauf;	1. *Owner* füllt ein Formular aus. Beim Verlassen der Felder, mit erforderlicher Information, werden parallel zu der Benutzertätigkeit auch die nötigen Überprüfungen durchgeführt und falls nötig die Fehler Meldung(en) ausgegeben; 2. Submit-Button wird aktiviert; 3. Punkte 4. bis 6. der Standard Ablauf;

- **deleteGuest (Guest g)**

  Auslöser:

  Owner tätigt der Delete anchor neben dem Gast g.

  Standard Ablauf:

Statisches Verfahren:	Dynamisches Verfahren (JavaScript):
1. Gäste Liste in der DB anpassen;	1. Warnung anzeigen: „Wollen Sie wirklich g aus der Liste ihrer Gäste löschen?“ 2. Falls *Owner* der Vorgang bestätigt => Gäste Liste in DB anpassen, sonst nichts machen

b) *bzgl. Gast*

- **guestLogin()**

  Auslöser:

  Aus Standard Login Verfahren, stellt sich ein Gast Login heraus.

  Standard Ablauf:

1. Punkte 1. bis i. der Standard Ablauf vom Login Verfahren in S.CORE;

2. Generiere neues Session Objekt;
3. Generiere „*Guest Workspace*“ ⊂ „*Owner Workspace*“

- **setAccountSettings()**

  Das gleiche Verfahren wie im S.CORE
- **downloadResults()**

  Das gleiche Verfahren wie im S.CORE
- **contactUs()**

  Das gleiche Verfahren wie im S.CORE

3.3.3 Entwicklung

Bzgl. Owner:

Content Modell:

Neue Funktionen addGuest(Guest g, Set analyses) und deleteGuest(Guest g) im User Workspace (siehe letzte zwei Funktionen auf der Abbildung 13 unten);

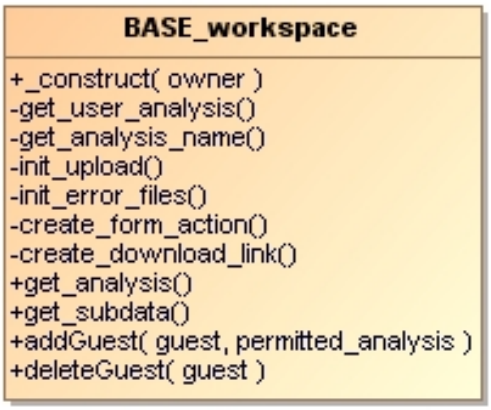


Abbildung 13. User Workspace Klassenfunktionen

Navigation Model:

Im Administration Bereich wird neue Möglichkeit „GuestsAdministration“ angeboten (siehe Abbildung 14 unten). Diese Erweiterung schließt die „addGuest()“- und „deleteGuest()“-Funktionalitäten ein.

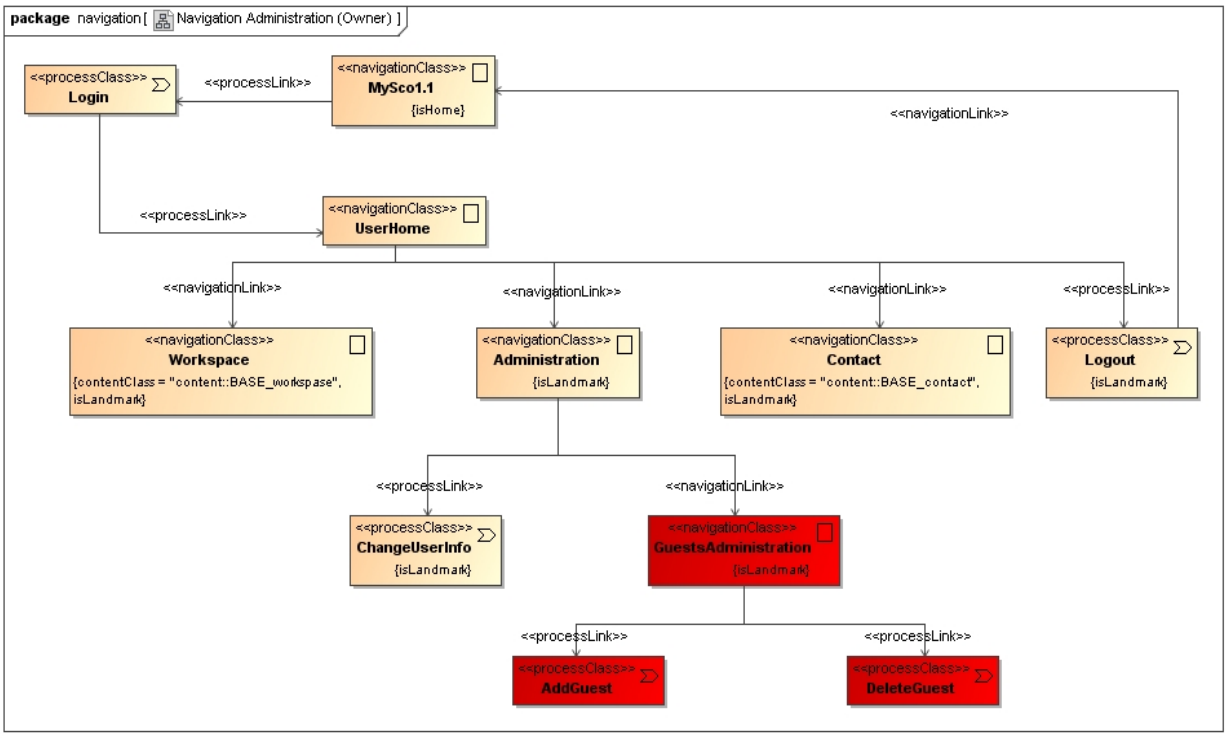


Abbildung 14. Navigationsdiagramm (Owner)

Presentation Model:

Die Erweiterungen im Präsentationsmodell beziehen sich wiederum auf den Administration Bereich (siehe Abbildung 15 unten). Wobei „Guest Administration“ Präsentationselement eine Erweiterung des Diagramms im Vergleich zu dem bisherigen Modell. Die Erweiterung bezieht sich auf die Hinzunahme des Präsentationselementes „Guest Administration“ zum Bereich - „Administration Owner“.

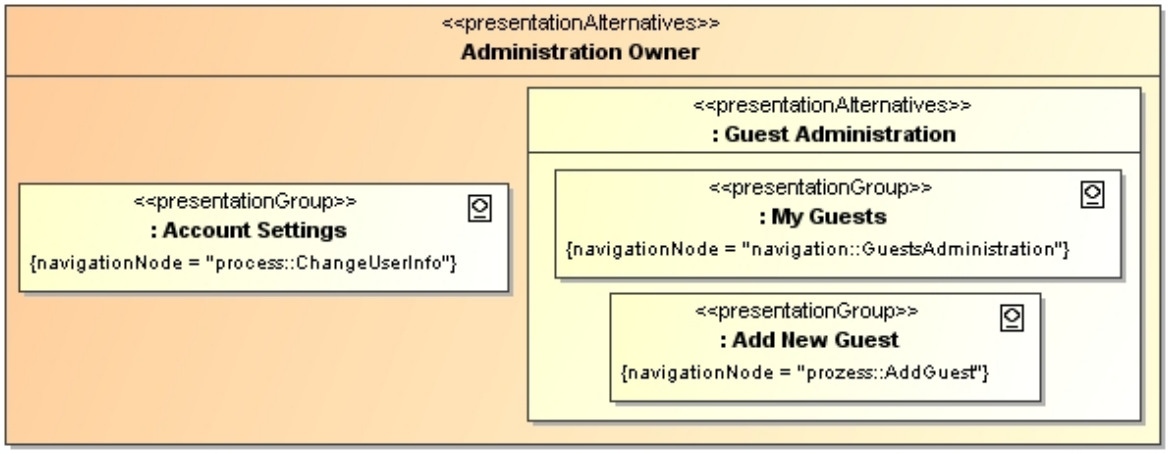


Abbildung 15. „Administration Owner“ Präsentation Group

Bzgl. Gast:

Content Model:

Die Erweiterung bezieht sich hauptsächlich auf das Paket - **ContentUser**, dem eine neue Benutzerrolle - „Guest“ hinzugefügt werden soll. Dafür wird eine neue Klasse *Guest,* als eine Erweiterung der abstrakten Klasse *BASE_actor*, erstellt. Dadurch, dass, wie es schon in der Anforderungsanalyse erwähnt wurde, „*Guest Workspace*“ eine Teilmenge des „*Owner Workspace*“ repräsentiert, stellt es sich als offensichtlich heraus eine „Teil des Ganzen“-Beziehung zwischen den Klassen Repräsentanten einzuführen. Aus diesen Überlegungen ist die Abbildung 16 als eine Erweiterung des „ContentUser“ - Paketes entstanden.

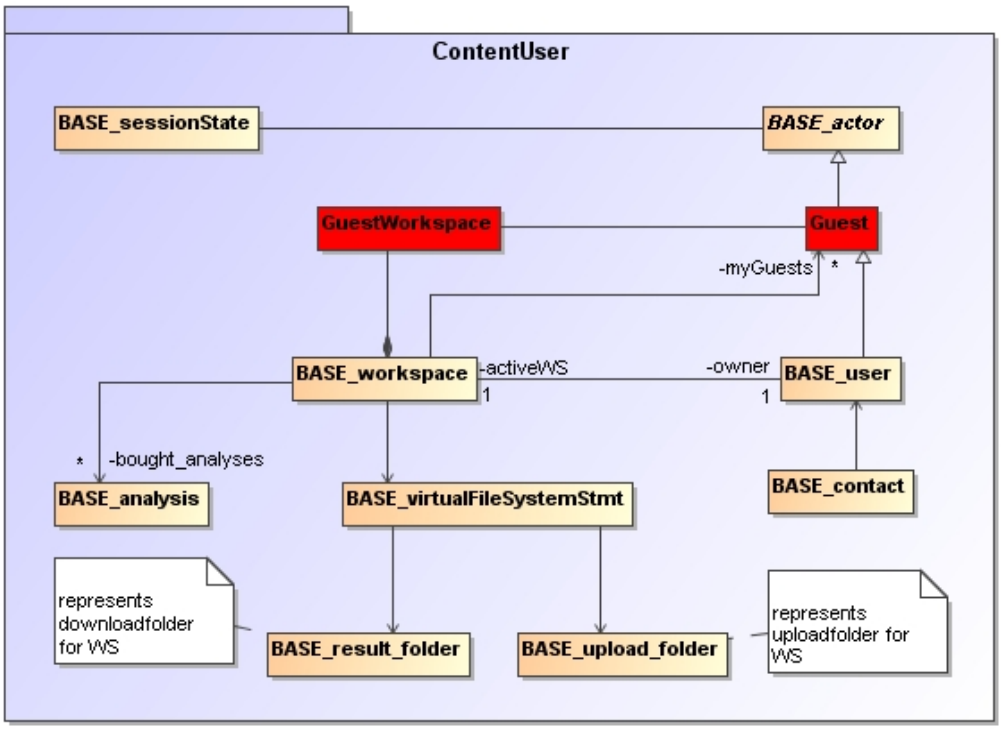


Abbildung 16. Erweiterung von Content Model

Navigation Model:

Das Navigationsmodell für einen Gast bildet eine Einschränkung gegenüber dem Navigationsmodell des Kontobesitzers (Owner) (siehe Abbildung 17 unten). So gibt es im Workspace keine Möglichkeit Bilder zur Analyse zu schicken, oder ein Coupon einzulösen. Im Administrationsbereich ist es für einen Gast unmöglich noch weitere Gäste einzuladen, somit fällt die Gästeadministration aus der Navigationsstruktur komplett aus. Dennoch folgend dem entstandenen Use-Case-Diagramm (siehe Abbildung 12) soll dem eingeladenen Benutzer eine Möglichkeit zur Verfügung gestellt werden, die Resultate von durchgeführten Analysen in den zugeteilten Analysenmodulen zu benutzen. Dadurch ist folgender Navigationsknoten wie Results(Outputs) (siehe Abbildung 17) erforderlich.

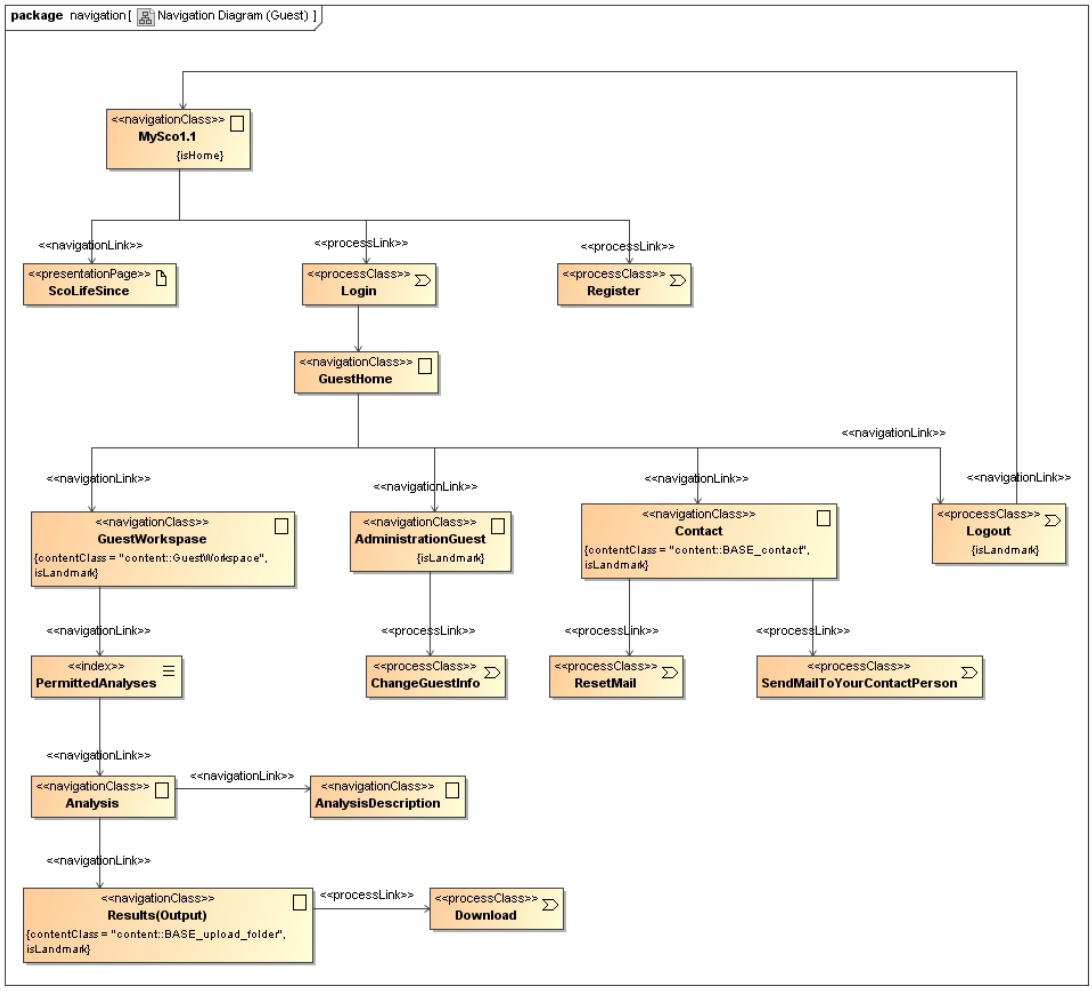


Abbildung 17. Navigation Guest

Presentation Model:

Das Präsentationsmodell vom Gast bildet wiederum eine Teilmenge des Präsentationsmodells von seinem Einlader (Owner) (wie z.B. für das Präsentationsmodell vom Administration Bereich, siehe Abbildung 18 (für Gast) im Vergleich mit der Abbildung 15 (für Owner)).

Abbildung 18. Administration für Gastbenutzerkonto

3.3.4 Implementierung

Die beschriebene Erweiterung wurde eben dem gesamten System in PHP implementiert. Für die Dynamisierung der Benutzerschnittstelle wurde JavaScript inklusive AJAX-Technologie verwendet.

Bzgl. Owner:

Das Administration Bereich wurde wie geplant um die nötigen Funktionen für Gästeverwaltung erweitert und dem entsprechend wurden alle Ebenen (wie Content, Navigation und Presentation) im Quellkode angepasst. (Siehe Anhang auf CD-ROM)

Bzgl. Gast:

S.CORE ist eine über die Jahre ausgereifte Webapplikation, aus diesem Grund stellt sich jede Erweiterung als äußerst aufwendig heraus. Abgesehen davon, wie es schon in der Anforderungsanalyse festgestellt wurde, sind alle Funktionalitäten, die dem Gast zur Verfügung gestellt sein werden, gleich den schon existierenden Funktionalitäten für den Benutzer, somit können die schon existierende Ansätze wiederverwendet werden.

Aus oben genannten Gründen wurde die Erweiterung von S.CORE auf andere Weise als geplant implementiert. Und zwar eine neue Klasse – „Gast“ erweitert nicht wie geplant die abstrakte Klasse – „*BASE_actor“*, sondern direkt Klasse – „BASE_user“ (siehe Abbildung 16). Somit hat der Gast alle Funktionalitäten und sogar den Workspace vom Owner geerbt. Die Einschränkung bei dem Zugriff auf bestimmte Services von S.CORE passiert auf der Präsentation Ebene, wo beim Erstellen von dynamischen HTML-Seiten die Überflüssigen Informationen für den Gast unsichtbar gemacht wurden.

Der gesamte Quellkode zu dem erweiterten S.CORE siehe im Anhang auf CD-ROM.

Kapitel 4 Erweiterung von UWE zur Modellierung von RIAs

4.1 RIA-Eigenschaften

Damit die oben angegebene Definition für RIA noch konkreter hervortritt, wird in diesem Abschnitt eine Liste von Eigenschaften vorgestellt, die eine moderne RIA charakterisieren:

- Kürzere Reaktionszeiten auf Benutzeraktionen, in dem ein Teil von der Anwendungslogik ins Client überlagert wird, können einige Funktionen direkt im Client aufgerufen werden, ohne Anfragen an den Server zu schicken. Dies verkürzt die Reaktionszeiten des Systems auf die Benutzeraktionen.

Systemanforderungen	Herkömmliche Webanwendung	RIA
„Muss- Felder“ im Formular dürfen nicht leer sein.	Formular wird ausgefüllt -> Submit-Button wird gedrückt -> Warten auf Server-Response -> nötige Korrekturen an dem Formular durchführen -> Submit-Button drücken -> Server-Response abwarten, usw.	Anhand JavaScript wird der Submit-Button inaktiv gemacht. Er wird erst dann aktiviert, wenn alle „Muss-Felder“ ausgefüllt sind.
Einige Felder im Formular sollen korrekte Informationen enthalten (z.B. E-Mail-Adresse soll ein „@“-Zeichen besitzen).	Formular wird ausgefüllt -> Submit-Button wird gedrückt -> Warten auf Server-Response -> nötige Korrekturen an dem Formular durchführen -> Submit-Button drücken -> Server-Response abwarten, usw.	Anhand JavaScript wird der Submit-Button inaktiv gemacht und wird erst dann aktiviert, wenn alle Felder korrekt ausgefüllt sind.
Einige Felder, wie z.B. Loginname müssen in der Datenbank eindeutig vorliegen, somit muss jeder neue Benutzername mit den schon existierenden Namen verglichen werden.	Formular wird ausgefüllt -> Submit-Button wird gedrückt -> Warten auf Server-Response -> nötige Korrekturen an dem Formular durchführen -> Submit-Button drücken -> Server-Response abwarten, usw.	Anhand AJAX-Technik, wird eine asynchrone Anfrage an den Server geschickt und schon während der Benutzer weitere Felder ausfüllt wird die Aufforderung angezeigt den Namen zu ändern.

Tabelle 8. Interaktionsszenarien zur schnellen Systemreaktion

- Kein „weißer Bildschirm“ während die Seite neu geladen wird, d.h. es gibt keine Unterbrechung von User- Aktivitäten bei Serveranfragen.

Systemfunktion	Herkömmliche Webanwendung	RIA
Seite muss neue geladen werden.	Der Server schickt die Seite mit dem neuen Inhalt. -> Der gesamte Inhalt wird gelöscht -> die Seite wird mit dem erhaltenen Inhalt vom Server komplett neue geladen. (Der Benutzer wartet und sieht in diesem Moment nur der „weiße Bildschirm“.)	Der Server schickt die Antwort auf die Anfragen vom JavaScript (oder anderen Scriptsprachen), JavaScript-Funktionen stellen fest, wie die Benutzeroberfläche sich ändern soll, nur die Differenzen zum aktuellen Zustand werden geladen, während dessen kann der Benutzer mit anderen Objekten auf der Website arbeiten.

Tabelle 9. Systemverhalten während des Ladens einer Seite

- Moderne Bedienelemente, wie zum Beispiel Baumstrukturen oder Tabs.

- Typische „Fat Client“-Operationen, wie Drag & Drop, umfassende Tastaturbedienung, Warnnachrichten beim Ausführen best. Aktivitäten:

Systemfunktionalität	Herkömmliche Webanwendung	RIA
Aktivitätsausführung, kann unerwünschte Folgen haben.	Aktivität anfordern, Server-Response abwarten, Warnungsnachricht bestätigen, Server-Request starten.	Anhand JavaScript-Funktionen wird Warnnachricht direkt im Client generiert, Aktivitätsausführung erfolgt nur nach der Bestätigung von Warnnachricht durch Benutzer.

Tabelle 10. Szenario zur Generierung von Warnungsnachrichten

4.2 Interaktive Elemente von Web 2.0 Webanwendungen im Überblick

In folgender Tabelle sind einige innovative Elemente aufgelistet, die eine moderne Webanwendung charakterisieren. Das Ziel von diesem Abschnitt ist aber nicht eine vollständige Liste von allen Elementen in modernen Webapplikationen mit RIA-Eigenschaften vorzustellen, sondern ein Überblick über die Innovationen auf diesem Gebiet zu verschaffen. Dennoch es wurde versucht, ein Akzent auf den wichtigsten Elementen von modernen Webanwendungen zu stellen. Denn somit wird die Flexibilität der im Anschluss vorgestellten Erweiterung von UWE verdeutlicht.

Interaktives Element	**Beschreibung**
Dynamisches Formular (**liveValidation**) (siehe Abschnitt 4.3.2, Punkt 1)	Bestimmte Felder in einem Formular müssen unterschiedlicher Überprüfung unterzogen werden. Zum Beispiel manche Felder dürfen nicht leer sein, manche Felder wie E-Mail-Adresse muss ein „@"-Zeichen beinhalten und von einem gültigen Mail-Server angeboten werden. Diese Überprüfung findet teilweise direkt im Client statt oder im Hintergrund wird eine Anfrage an den Server gestartet.
Dynamische Warnungen (**liveWarnings**) (siehe Abschnitt 4.3.2, Punkt 0	Beim Ausführen bestimmten Operationen wird auf dem Client (im Browser) eine Warnungsnachricht generiert, die für weitere Prozessausführung bestätigt werden soll.
Animation des Prozessfortschritts (**progressIndicator**) (siehe Abschnitt 4.3.2, Punkt 3)	Eine Animation wird auf der Clientseite durchgeführt, um dem Benutzer ein Feedback über den Prozessfortschritt zu geben.
Aktualisierung in Echtzeit (**periodicRefresh**) (siehe Abschnitt 4.3.2, Punkt 4)	Es werden bestimmte Feldern auf der Applikationsoberfläche verwendet, die eine Aktualisierung in Echtzeit anfordern (z.B. die Informationen über den aktuellen Zustand eines Objekts, das gerade auf der Serverseite bearbeitet wird, eines Files, eines Bildes, oder eines Formulars).
Automatische Ergänzung (**autoCompletion**) (siehe Abschnitt 4.3.2, Punkt 5)	Manche Input-Felder können automatisch ergänzt werden. Zum Beispiel bei der Eingabe von Postleitzahl kann die Stadt automatisch ausgegeben werden, oder bei Eingabe eines Wortes können die Varianten zur Vervollständigung angeboten werden.
Animation von der mehrfachen Auswahl **(multiSelection)** (siehe Abschnitt 4.3.2, Punkt 6)	Auswählen von mehreren Alternativen aus der Liste mit sofortiger clientseitiger Anzeige. Es gibt eine Möglichkeit die gesamte Menge komplett zu markieren. Dabei werden die Markierungen grafisch dargestellt und angezeigt.
Dynamische Informationen (**liveReport**) (siehe Abschnitt 4.3.2, Punkt 7)	Informationen über einen Objekt werden beim Überfliegen des Objekts ausgegeben. Wie z. B. die Informationen zu den Buttons (die Information kann z.B. die Folgen der Betätigung dieses Buttons beinhalten) oder Referenzen, sowie Beschreibungen zu den bestimmten Fachwörtern usw.
Bildergalerie **(gallery)** (siehe Abschnitt 4.3.2, Punkt 8)	Es wird eine Vergrößerung des Bildes dargeboten. Parallel werden auch andere Bilder auf den Computer vom Benutzer hochgeladen, dadurch entsteht die Möglichkeit für den Benutzer andere Bilder schneller (ohne weiteren Wartezeiten auf Grund des Bilddownloads) anschauen zu können.
Verfeinerte Suche (**liveSearch**) (siehe Abschnitt 4.3.2, Punkt 9)	Suche wird innerhalb der Webapplikation an einen bestimmten Parameter angepasst, beispielsweise an ein Datum.

Informative Führung (**guidedTour**) (siehe Abschnitt 4.3.2, Punkt 10)	Falls ein Benutzer bestimmte Informationen über ein bestimmtes Objekt (Produkt) innerhalb der Webapplikation braucht, dann wird ihm die Möglichkeit geboten diese Informationen anhand einer Führung anzufordern, oder man bietet die Informationen über gesamte Webapplikation anhand einer Guided Tour.
Drag&Drop **(drag&drop)** (siehe Abschnitt 4.3.2, Punkt 11)	Benutzer kann das Layout (sprich Position bestimmten Grafischen Objekten) auf der Seite ändern anhand von „Drag&Drop"-Funktionsweise
Karussell **(carrousel)** (siehe Abschnitt 4.3.2, Punkt 12)	Objekte die durch Bilder repräsentiert werden, können als Karussell auf der Präsentationsebene organisiert werden.
Kollaps **(collapse)** (siehe Abschnitt 4.3.2, Punkt 13)	Graphisches Kollabieren der Anwendung beim Ausblenden eines oder mehrerer Elemente

Tabelle 11. Interaktive Elemente

Unter anderen können auch folgende Elemente wie das Inputelement – „Slider" oder direkte „Online-Annotation" erwähnt werden. Beim „Slider" geht es um ein Präsentationselement, das den Überblick über die Elemente auf der Seite erleichtert. Anhand von „Slider" kann ein „Filter" auf die angezeigten Elemente eingesetzt werden. Der Filter kann sich auf ein bestimmtes Datum beziehen oder einfach auf die Menge von Elementen. Das Präsentationselement direkte „Online-Annotation" stellt für den Benutzer eine Möglichkeit dar, einige Elemente auf der Oberfläche zu kommentieren bzw. zu verändern. Dabei werden alle Änderungen gespeichert und auf die Serverseite übertragen. In dieser Arbeit wird auf „Slider" und direkte „Online-Annotation" nicht weiter eingegangen.

4.3 RIA-fähiges UWE

Die Interaktion mit dem Benutzer steht für eine Web 2.0 Webanwendung im Vordergrund. Genau an diesem Punkt soll UWE erweitert werden, damit dieses Konzept bei der Modellierung und Implementierung erfolgreich eingesetzt werden kann. Bis her wurde schon viel Arbeit in diese Richtung investiert (siehe [24], [26]). Somit ist, als erster Ansatz dafür, <<*User Action*>> im UWE Metamodell entstanden (siehe Abbildung 3 im Abschnitt 2.6.1). Dennoch reicht dieser Ansatz, wie es die Praxis gezeigt hat, für die Modellierung einer Rich Internet Application nicht aus. Der Mangel entsteht in der Hinsicht auf die durchaus komplexe Vorgänge, die während der Interaktion mit dem Benutzer stattfinden können und darüber hinaus komplexe Systemaktionen, die als Reaktionen auf die Handlungen vom Benutzer entstehen können. Außerdem sind ebenso aus den praktischen Erfahrungen folgende Überlegungen entstanden:

- jegliche Interaktion, die zwischen dem System und dem Benutzer stattfindet, wird durch bestimmte Ereignisse ausgelöst. Aus diesem Grund soll eine Art von Ereignissprache (Näheres dazu siehe im Abschnitt 4.3.1) definiert werden, die diese Ereignisse eindeutig beschreibt und dem Entwickler während der Modellierungsphase Konstrukte für die Beschreibung von Ereignisauslöser bereitstellt;

- das Verhalten eines interaktiven Elements kann sehr gut durch ein Zustandsdiagramm visualisiert werden, wo man sowohl die Auslöser (Events) zur Verfügung hat, als auch spätere Systemreaktion anhand von Zustandsaktivitäten gut darstellen kann.

Als Nächstes wurde das UWE Metamodell um einige Elemente erweitert. Im Rahmen dessen wurden zu den jeweiligen Stereotypen einige Metaattribute, als dynamische Erweiterungen von diesen Stereotypen, hinzugefügt. Somit wurde ermöglicht, das Element im Modell als interaktiv zu kennzeichnen. Dafür werden nach dem Anwenden von solchen Stereotypen diese Metaattribute anhand von entsprechenden Werten belegt. In der Regel wird zu diesem interaktiven Element auch ein Zustandsdiagramm hinzugefügt, um das Verhalten der Interaktion zu schildern. Die Erweiterung des UWE Metamodells wird im Abschnitt 4.3.2 detailliert und anhand von konkreten Beispielen beschrieben.

Anlehnend an [5] kann man ein interaktives Element durch folgende Komponenten vollständig beschrieben:

- **Interaktion**
- **Aktion**
- **Darstellung**

Aufgrund dessen ist folgender Ansatz für die Modellierung von solchen Elementen entstanden:

- **Zustandsdiagramm** (modelliert die gesamte Benutzerinteraktion)
- **Zustandsaktivität** (repräsentiert die Systemreaktion im Modell)
- **Stereotyp\Metaattribut** (sorgen für die Darstellung der Entsprechenden Elementen im Modell)

Das Ziel von diesem Kapitel ist, konkrete Vorschläge bezüglich der Modellierung von innovativen Elementen mit RIA-Eigenschaften anzubieten. Dabei werden alle Elemente in der Liste (siehe Abschnitt 4.2) in Betracht gezogen. Einige Vorschläge werden dabei sehr konkret mit den Auszügen aus den Quellkodetexten beschrieben, andere werden nur skizzenweise geschildert.

4.3.1 Event Language

Anlehnend an JavaScript (meist verwendete Programmiersprache für die Implementierung von interaktiven Elementen) wurde folgende Liste von Ereignissen zusammengestellt, die als Auslöser für eine Interaktion dienen und in Zustandsdiagrammen als Zustandsübergangsereignisse (Events) verwendet werden:

- ***onblur(Element_Name)*** beim Verlassen des Elements.

  Dieser Auslöser kann für die Beschreibung vom Ereignis in Zustandsdiagrammen verwendet werden, wo ein Element mit dem Namen „Element_Name“ zuvor aktiviert war und der Benutzer es jetzt verlässt.

- ***onchange(Element_Name)*** bei erfolgter Änderung.

  Das ist ein Auslöser für den Fall, dass ein Element (mit dem Element_Name) einen geänderten Wert erhalten hat.

- ***onclick(Element_Name)*** beim Anklicken.

Dieser Auslöser beschreibt den Fall, wo das Element (mit dem Element_Name als Name) von dem Benutzer angeklickt wurde.

- ***ondblclick(Element_Name)*** bei doppeltem Anklicken.

  Dies ist ein Auslöser, der ähnlich dem Vorherigen ist, nur in diesem Fall erfolgt anstatt des einzelnen Klicks ein doppelter Klick an dem Element.

- ***onerror(Error_ID)*** im Fehlerfall.

  Dieser Auslöser beschreibt das Event, wo ein Prozess mit Fehlern endet. Error_ID repräsentiert dabei die Fehlermeldung, die von dem Prozess generiert wurde.

- ***onfocus(Element_Name)*** beim Aktivieren.

  Dieses Schlüsselwort beschreibt das Ereignis, der genau dann eintritt, wenn der Benutzer das Element mit dem Namen „Element_Name“ aktiviert.

- ***onload(Element_Name)*** beim Laden eines Elements.

  Beschreibt ein Ereignis, das beim Laden des Elements mit dem Namen „Element_Name“ eintritt.

- ***onmousedown(Element_Name)*** bei gedrückter Maustaste.

  Dieses Ereignis tritt ein, wenn der Anwender die Maustaste über dem Element (mit dem Namen „Element_Name“) gedrückt hält.

- ***onmousemove([Element_Name])*** bei weiterbewegender Maus. Element_Name in Klammern ist optional.

  Das Schlüsselwort „onmousemove“ wird für die Beschreibung des Events verwendet, der genau dann eintritt, wenn der Benutzer die Maus bewegt, unabhängig davon, ob die Maustaste gedrückt ist oder nicht. Dabei wird unterschieden, ob der Element_Name in Klammern vorhanden ist oder nicht. Falls das Element vorhanden ist, heißt es, dass das Element vorher angeklickt wurde und bei der Mausbewegung mitgezogen wird. Sonst ist das Event als reine Mausbewegung zu betrachten.

- ***onmouseout(Element_Name)*** beim Verlassen des Elements mit der Maus.

  Dieses Ereignis tritt ein, wenn der Anwender mit der Maus über ein Element fährt und dieses dabei verlässt.

- ***onmouseover(Element_Name)*** Das Event tritt beim Überfahren des Elements mit der Maus ein.

- ***onmouseup(Element_Name)*** beim Loslassen der Maustaste.

  Das Schlüsselwort beschreibt ein Ereignis, das genau dann eintritt, wenn der Benutzer die Maustaste über dem Element gedrückt gehalten hat und sie nun wieder loslässt.

- ***oncompletion(Process)*** Dieses Event tritt bei erfolgreicher Vollendung des Prozesses ein.

- Zeitevent - ***after(TimeInterval)*** das Ereignis tritt nach einem Zeitintervall ein.

- Zeitevent - ***on(Time)*** das Ereignis tritt zu einem bestimmten Zeitpunkt ein.

4.3.2 Modellierungsvorschläge

Im Folgenden wird näher auf die Modellierung von den interaktiven Elementen eingegangen. Die nötigen Erweiterungen im UWE- Metamodell werden im Laufe des folgenden Kapitels sowohl direkt bei der Beschreibung jedes konkreten Beispieles vorgestellt als auch auf dem abschließenden Diagramm im Überblick (siehe Abbildung 71) präsentiert. Aus den didaktischen Gründen wird das erste Beispiel „Dynamisches Formular“ sehr ausführlich beschrieben und mit den Ausschnitten aus dem Quellkode vorgestellt. Damit soll es verständlich sein, wie diese Modellierungsvorschläge entstanden sind und in wie fern sie in weiteren Arbeiten zur automatischen Generierung von Webanwendungen verwendet werden können.

1) Dynamisches Formular (liveValidation)

Ziel: Während der Formularausfüllung sollen manche Felder im Hintergrund auf Korrektheit überprüft werden. Dabei wird im Fehlerfall dem Benutzer sofort nach der Überprüfung ein Feedback in Form einer Fehlermeldung gegeben. Das dynamische Formular kann nur dann an den Server verschickt werden, wenn alle von dem Entwickler bestimmten Feldern im Formular akzeptable Werte erhalten haben.

Erweiterung des UWE Profils: Der Stereotyp „*inputElement*“ erhält ein neues Attribut „liveValidation“ vom Typ „Boolean“ mit dem Defaultwert „false“ (siehe Abbildung 19). Also, falls der Wert von dem Metaattribut liveValidation eines Elements im Formular (z.B. von einem textInput Feld) auf true gesetzt wird, dann soll dieses Feld einer Überprüfung unterzogen werden. Wie die Überprüfung genau statt findet, wird durch ein Zustandsdiagramm im Präsentationsmodell beschrieben. Dabei folgend dem Vererbungsmechanismus erhalten alle Unterklassen (wie textInput, selection, imageMap und customComponent, siehe Abbildung 71) vom *inputElement* dieses Attribut automatisch.

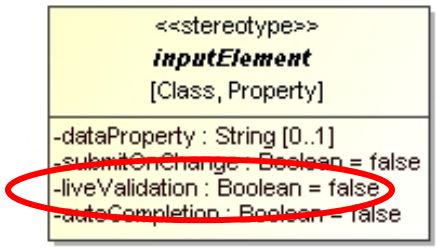


Abbildung 19. Erweiterung vom Stereotyp <<InputElement>> in UWE

Beispiel: Erstellen von einem Gastkonto (siehe Abschnitt 3.3). Falls JavaScript im Browser aktiviert ist, wird es noch während der Formularausfüllung überprüft, ob der Gastnamefeld eine gültige E-Mail-Adresse enthält, und ob dem zukünftigen Gast mindestens ein Analysemodul zugeteilt wurde, d.h. ob die entsprechende Checkbox den Status „checked“ erhalten hat. Im Fehlerfall soll die entsprechende Fehlermeldung generiert und angezeigt werden. Einige von den Fehlermeldungen sind auf der Abbildung 20 zu sehen. Wobei die rotfarbige Meldungen auf den jeweiligen Screenshots Fehlermeldungen in bestimmten Fehlerfällen repräsentieren. Auf der Abbildung 20 sind

es jeweils die Fehlermeldung zu der nicht korrekten E-Mail-Adresse, die Fehlermeldung, dass der angegebene Benutzername schon existiert und die Fehlermeldung, dass es erforderlich ist mindestens eine Analyse dem Gast zur Verfügung zu stellen, damit ein Gästekonto erstellt werden kann.

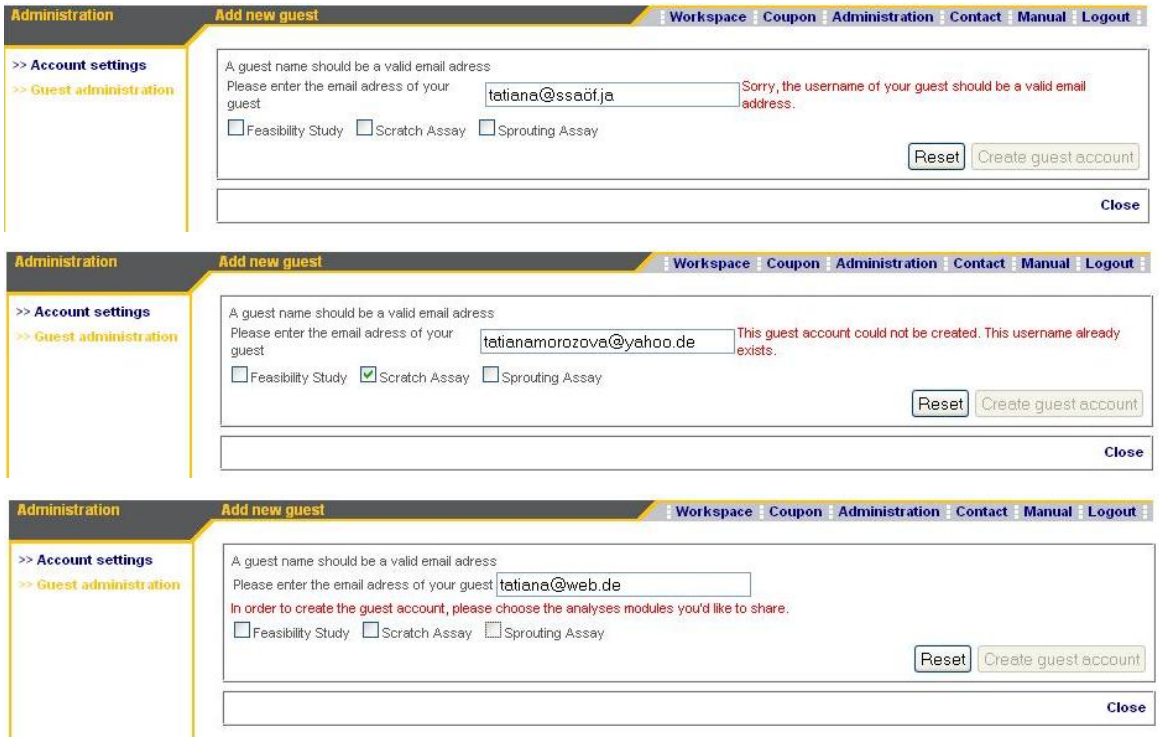


Abbildung 20. Mögliche Fehlermeldungen beim Erstellen von einem Gästekonto.

Modellierungsvorschlag für das Beispiel - „Erstellen vom einem Gastkonto“:

- ***Das Präsentationselement.*** Entsprechend der Erweiterung vom UWE Profil wird bei den Elementen „Guest Name“ und „My Analyses“ im Formular der Wert von dem Metaattribut liveValidation auf true gesetzt (siehe Abbildung 21 unten). D.h., dass diese zwei Textfelder der dynamischen Überprüfung unterzogen werden. Im Fehlerfall werden Fehlermeldungen in den entsprechenden Fehlermeldungsfeldern angezeigt. Auf der Abbildung 21 sind das jeweils „Error Message Name“ und „Error Message Analysis“ für die entsprechende Inputfelder „Guest Name“ und My Analyses“. Metaattribut „visibilityCondition“ bei den Error Message Textfeldern tragen den Wert „hidden“ (siehe Abbildung 21), was bedeutet, dass diese Felder auf der Oberfläche im unsichtbaren Zustand sind und werden nur im Fehlerfall visualisiert. (An dieser Stelle soll noch der Unterschied zwischen den Schlüsselwörtern „disabled“ und „hidden“ betont werden. Das Wort „hidden“ deutet darauf hin, dass das Element visuell versteckt vom Benutzer ist, wobei das Wort „disabled“ beschreibt den unaktivierten Zustand des Elementes auf der Benutzeroberfläche. Dabei wird das inaktive Element auf der Oberfläche sichtbar angezeigt, dennoch kann es nicht betätigt werden. Meist werden Buttons und Links auf der Benutzeroberfläche inaktiv gemacht, in diesem fall verwendet man das Metaattribut „enablingCondition“ im Modell, um den Aktivierungszustand von dem entsprechenden Element zu verdeutlichen.) Das Verhalten von dem gesamten Formular wird durch ein Zustandsdiagramm im Präsentationsmodell beschrieben.

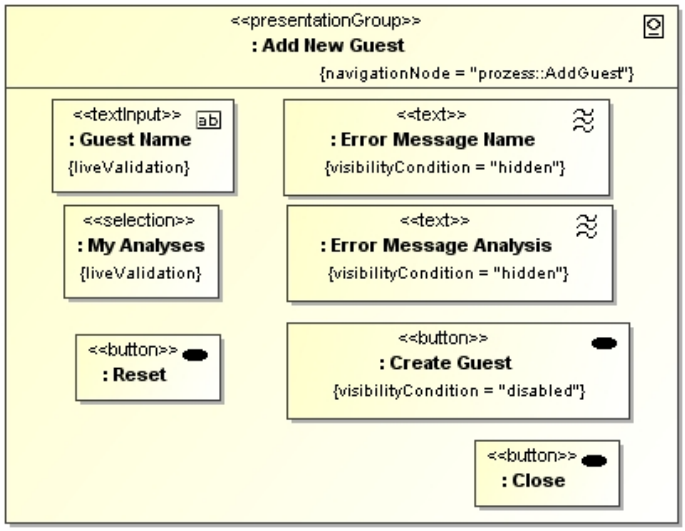


Abbildung 21. Dynamisches Formular für „AddGuest“- Aktivität

Auf der Abbildung 22 sieht man die Implementierung, die dem Layout entspricht, das auf der Abbildung 21 vorgestellten wurde.

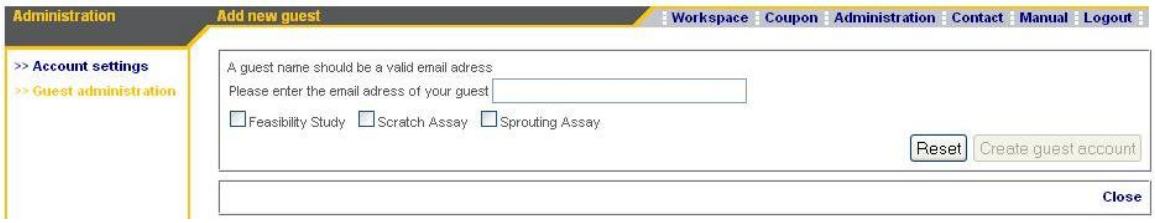


Abbildung 22. Seiten Layout für „Add New Guest“ Präsentationsgruppe.

- ***Präsentationsmodell*** wird um ein Zustandsdiagramm erweitert. Das Diagramm (siehe Abbildung 23) liefert eine übersichtliche Beschreibung von den Benutzerinteraktionen mit dem System. Auf dem Diagramm (siehe Abbildung 23) wird das Verhalten den vom „Add New Guest“ Präsentationselement (siehe Abbildung 21) dargestellt. Dabei ist der Hauptzustand „editForm“ in zwei parallele Unterzustande aufgeteilt, wodurch dieser Zustand nur in zwei Fällen verlassen werden kann. Im ersten Fall werden alle Endknoten in beiden parallelen Bereichen erfolgreich erreicht. Im zweiten Fall wird Cancel-Button betätigt (im Diagramm wird dies durch den Austrittspunkt repräsentiert). Dabei wird nicht unterschieden in welchem von beiden parallelen Bereichen oder in welchem Zustand sich das Objekt befindet, wenn der Cancel-Button gedrückt wird, wird der Zustand „editForm“ sofort verlassen. Die Parallelität von beiden Bereichen im Zustandsdiagramm verkörpert die Idee, dass Funktionen (z.B. JavaScript-Funktionen) im Hintergrund ablaufen können. Dadurch befindet sich das System, aus der Sicht des Benutzers, in beiden Unterbereichen parallel.

  Um die Trennung zwischen den Funktionen, die direkt im Browser ausgeführt werden, und den Funktionen, die im Server ausgeführt werden und anhand AJAX-Technologie im Hintergrund beim Server abgefragt werden, zu verdeutlichen, wird folgende Lösung vorgeschlagen:

- reine serverseitige Funktion (wie z.B. reine JavaScript-Funktion) wird nur im Präsentationsmodell definiert und modelliert;
- eine Funktion, die durch AJAX-Technologie aufgerufen wird, wird im Prozessmodell definiert und im Präsentationsmodell mit der Angabe des untergeordneten Paketes, wo diese Funktion definiert wurde, aufgerufen.

Ein Beispiel für diese Vorgehensweise zeigt das Aktivitätsdiagramm der Funktion „guestnameCheck“ (siehe Abbildung 25) durch die Aktivität „check_if_guest_name_is_available()“, die auf der Serverseite während des Ablaufs ausgeführt wird.

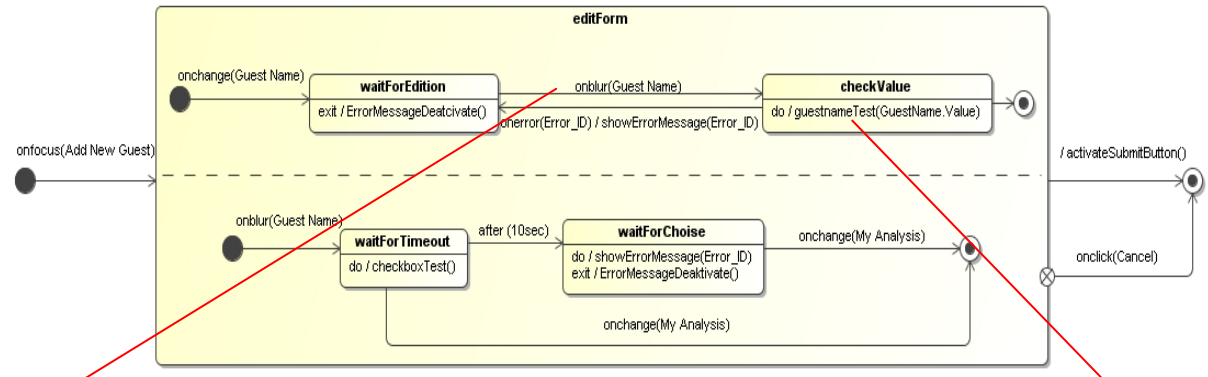


Abbildung 23. Verhalten vom dynamischen Formular

Implementierung:

Ein Auszug aus einer Datei „add_new_guest_ajax.tpl“, die das HTML-page template für „Add_new_guest“ HTML-Formular repräsentiert, zeigt die Implementierung von dem beschriebenen Formular zum Erstellen eines Gästekontos:

```
…
<td><input type="text" id="input" name="guestName" value ="{$guestname_value}" size="30"
onblur="guestnameTest(guestName.value)"></td>
…
```

Dieser Quellkode entspricht dem Element „Guest Name“ mit dem Stereotyp <<textInput>> der Präsentationsgruppe „Add New Guest“ auf der Abbildung 21. Die dynamische Validierung, als System Reaktion auf das Ereignis ***onblur(Guest Name)*** (d.h. Verlassen des Elements mit dem Namen „Guest Name“, siehe Abschnitt 4.3.1), von diesem Element soll JavaScript-Funktion „***guestnameTest(guestName.value)***“ übernehmen (Quelltext für die Implementierung von dieser Funktion siehe im Quellkodeauszug aus der Datei „checkFormFunktions.js“ unten). Im Zustandsdiagram (siehe Abbildung 23) vom Modell wird diese Funktion durch die gleichnamige do-Aktivity im Zustand „checkValue“ repräsentiert. Eine weitere Detaillierung während der Modellierungsphase können mehrere Aktivitätsdiagramme für interne do-Aktvitäten oder Übergangsaktivitäten im Diagramm darstellen, die ebenfalls im Präsentationsmodell des Projekts einzusiedeln sind. Dies verdeutlichen die Aktivitätsdiagramme für die

Funktionen guestnameTest (siehe Abbildung 24) und guestnameCheck (siehe Abbildung 25).

Ein Auszug aus der checkFormFunktions.js Quelldatei:

```
function guestnameTest(value)
{
        var test_name = guestnameCheck(value);
        var test_checkbox = checkboxCheck();
        test_name.execute();
        window.setTimeout("test_checkbox.execute()",
10000);
}

…

function guestnameCheck(value) {
        // step 1: make a new object somehow
        var that = check("guestname_error", value);
        // definition of private variables and functions
           // step 2: augmentation
        // definition of priviledged methods, which
        // have access to private variables and functions
           that.handleSuccess = function(o) {
        if (o.responseText = "already_exist")
        {

        that.setError("guest_could_not_create_loginna
me_exists");
                myFormValidation.discard("input");
        }
        else if (o.responseText = "not_e-mail")
        {// test finished and succesful

        that.setError("guestname_is_not_email");
                myFormValidation.discard("input");
        }
        else if (o.responseText = "is_ok")
        {
                that.clearError();
                myFormValidation.record("input");
        }
        else
        {
                window.alert(o.responseText);
        }
        };
           var guestname_callback = {
                failure: that.handleFailure,
                success: that.handleSuccess
        };
           that.startRequest = function() {
                var formObject =
document.getElementById('add_new_guest_form');

        YAHOO.util.Connect.setForm(formObject);
YAHOO.util.Connect.asyncRequest('POST',
'mysco.php?ajax=on&navigation=0&process=50&sub
```

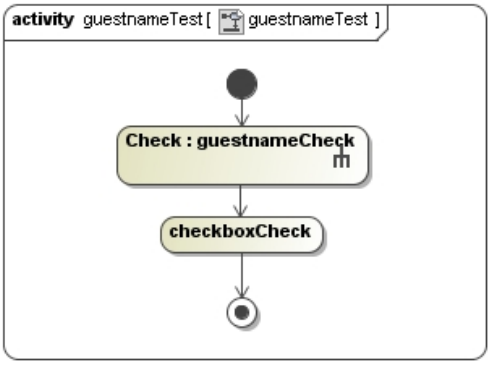


Abbildung 24. Erste Überprüfung des Gastsnamens

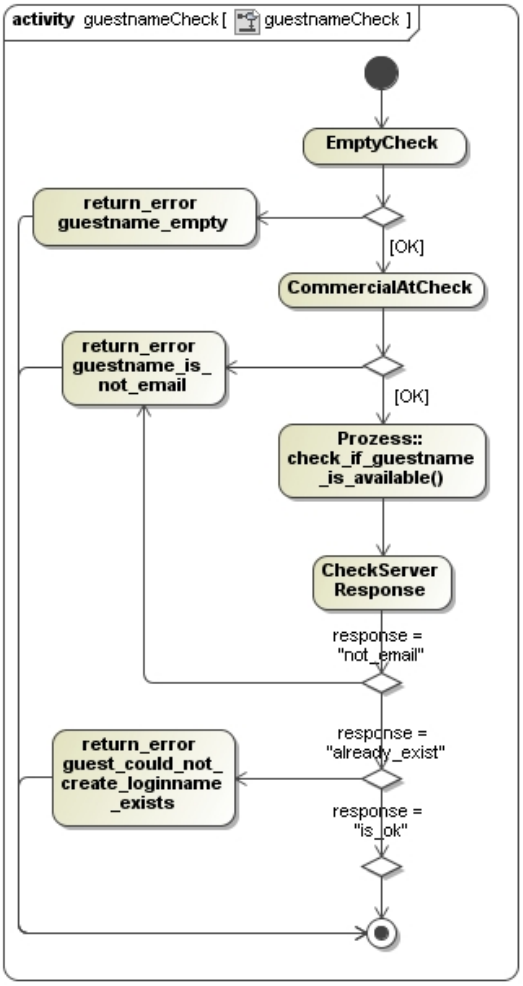


Abbildung 25. Eine detailliertere Überprüfung des Gastsnamens

```
process=53', guestname_callback, null);
        };
          that.execute = function(){
                if (that.testValue() = "")
                { //empty guest login name

        that.setError("guestname_empty");

        myFormValidation.discard("input");
                }
                else
if(!that.hasCommercialAt(that.testValue()))
                {

        that.setError("guestname_is_not_email");

        myFormValidation.discard("input");
                }
                else
                {
                        that.startRequest();
                }
                };
        // step 3: return the new object
        return that;
}
```

Wobei auf dem Diagramm die tatsächliche Ausführungsreihenfolge berücksichtigt wird, während im Quellkode (links) die Reihenfolge von Funktionsdefinitionen dargestellt wird. Dementsprechend wird „that.execute“ Funktionsobjekt als erstes aufgerufen (siehe erste drei Aktivitäten im Aktivitätsdiagramm für die Funktion „guestnameCheck“ auf der Abbildung 25) und als letztes in der „guestnameCheck“ Funktion im Quelltext definiert. Dabei wird in der Funktion guestnameTest sofort das „execute“-Teil der Funktion guestnameCheck aufgerufen. (Siehe Zeile: test_name.execute(); im Quellkode. Das Objekt test_name repräsentiert ein Objekt der Funktion guestnameCheck.)

2) *Dynamische Warnungen (liveWarnings)*

Ziel: Der Benutzer soll beim Ausführen bestimmten Operationen, auf die Folgen seiner Tätigkeit aufmerksam gemacht werden, in dem es eine Warnungsnachricht unmittelbar im Client generiert und angezeigt wird. Der gestartete Prozess wird dabei nur nach der Bestätigung vom Benutzer ausgeführt sonst wird der Prozess abgebrochen.

Erweiterung des UWE Profils: keine. In diesem Fall reicht der <<UserAction>> Stereotyp aus.

Beispiel: Löschen von einem Gastkonto (siehe Abschnitt 3.3). Falls JavaScript im Browser aktiviert ist, wird vor dem Löschen des Gasts eine Warnnachricht (siehe Abbildung 26 unten) angezeigt, die von dem Benutzer bestätigt werden soll, wenn er möchte, dass der Gast tatsächlich gelöscht werden soll.

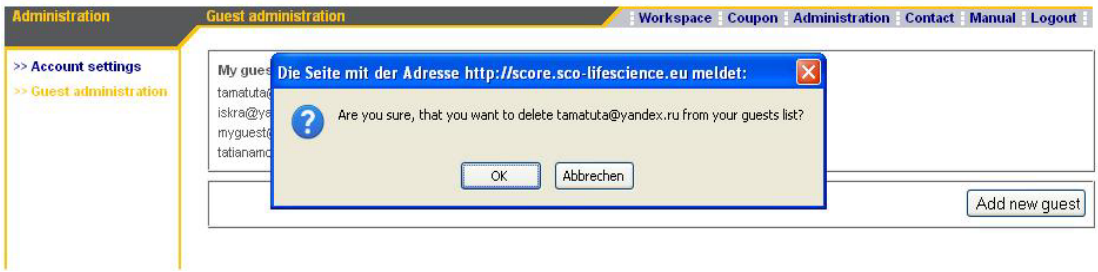


Abbildung 26. Meldung beim Versuch ein Gast aus der Gästeliste zu löschen.

Modellierungsvorschlag für das Beispiel - „Löschen vom einem Gastkonto“:

- ***Das Präsentationselement,*** das den entsprechenden Prozess starten soll, wird um eine dynamische Warnung erweitert. Die Sichtbarkeit dieser Warnung hat den Wert „hidden“, d.h., dass diese Nachricht erst verborgen ist und nur im Fall, dass der „delete“-Anchor betätigt wird, wird diese Nachricht angezeigt bzw. sichtbar gemacht. Im Beispiel wird diese Nachricht in der „GuestList“- iteratedPresentationGroup plaziert (siehe Abbildung 27).

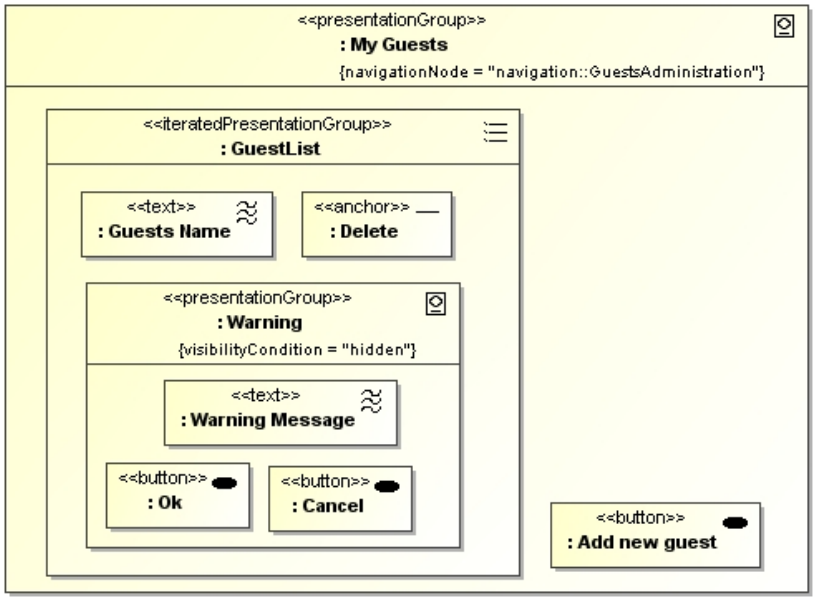


Abbildung 27. Präsentationselement „My Guests“

Die implementierte Oberfläche von dem Präsentationselement „My Guests“ sieht folgendermaßen aus (siehe Abbildung 28):

Abbildung 28. Gäste Verwaltung bei S.CORE

- ***Aktivitätsdiagramm im Prozessmodell*** für den betroffenen Prozess wird um eine Aktivität erweitert. Bei dieser Aktivität namens „WarningAck“ (Bestätigung der Warnungsnachricht durch den Benutzer) wird <<UserAction>> Stereotyp angewandt (siehe Diagramm auf der Abbildung 29 unten). Dabei wird das Verhalten von der Aktivität „WarningAck“ im Zustandsdiagramm „Warning Behavior“ (siehe Abbildung 30) beschrieben.

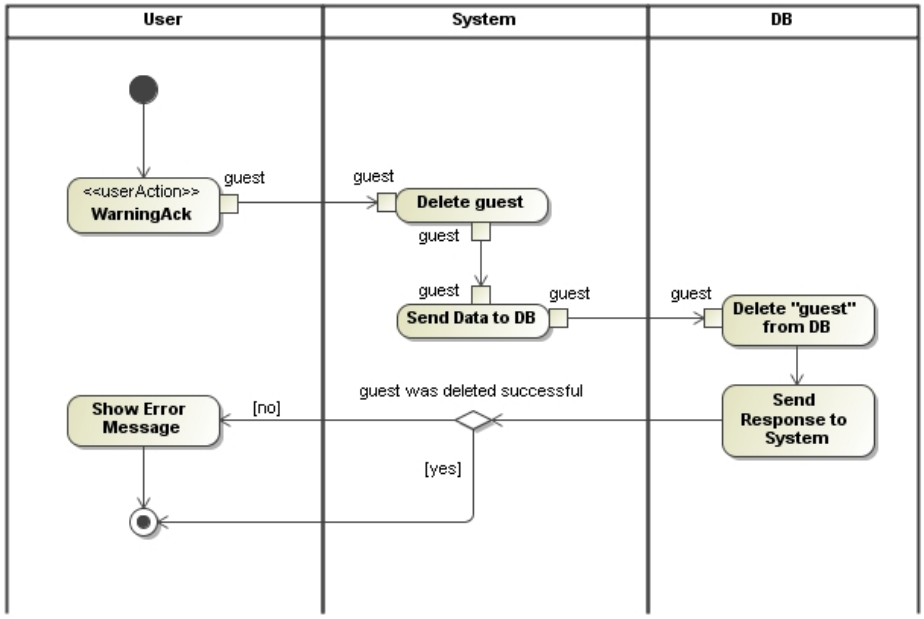


Abbildung 29. Aktivitätsdiagramm mit der dynamischen Warnung

- ***Das Präsentationsmodell*** wird um ein Zustandsdiagramm erweitert, das das Verhalten von dynamischer Warnung beschreibt. Im aufgegriffenen Beispiel bezieht sich das Zustandsdiagramm auf die Klasse „GuestList“ (siehe Abbildung 30 unten)

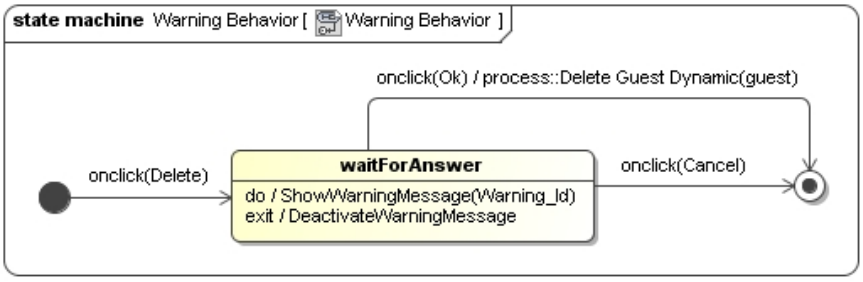


Abbildung 30. Verhalten der dynamischen Warnung

Die Übergangsaktivität „process::Delete Guest Dynamic(guest)“ deutet darauf hin, dass diese Aktivität im Prozessmodell unter dem Namen „Delete Guest Dynamic“ zu finden ist (siehe Abbildung 29) und anhand von AJAX-Technologie auf der Serverseite aufgerufen wird.

3) Animation des Prozessfortschritts (Progress Indicator)

Ziel: Der Benutzer soll während der serverseitigen Ausführung eines Prozesses ein Feedback bezüglich des Prozessfortschrittes in Form einer Animation auf der Client Seite bekommen.

Erweiterung des UWE Profils: keine. In diesem Fall ist das <<mediaObject>> Stereotyp ausreichend.

Beispiel: Up- bzw. Downloadbalken (siehe Abbildung 31) während des Up- bzw. Downloads eines Bildes in S.CORE. Während des Up- bzw. Downloads einer Datei wird ein farbiger Balken auf der Clientseite angezeigt. Anhand JavaScript wird die Simulation, von einem sich langsam mit der Farbe füllenden Balken, erstellt. Jede dritte Sekunde wird im Hintergrund beim Server nach der verbliebenen Zeit, für die Prozessausführung, nachgefragt. Aufgrund der Antwort vom Server wird die Geschwindigkeit, mit der sich der Balken mit der Farbe füllt, angepasst.

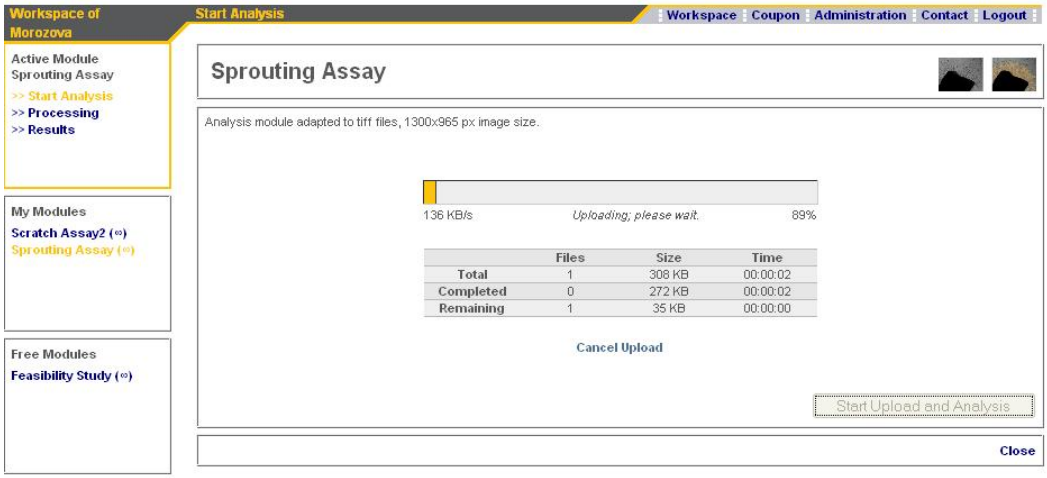


Abbildung 31. Animation vom Downloadbalken

Modellierungsvorschlag für das Beispiel - „Up- Downloadbalken“:

- ***Das Präsentationselement,*** das den entsprechenden Prozess starten soll, wird in einem Element mit dem Stereotyp <<presentationAlternatives>> mit dem neuen Element „Progress Indicator“ zusammen vereint. Auf die Klasse „Progress Indicator“ wird UWE Stereotyp <<mediaObject>> angewandt (siehe Abbildung 32).

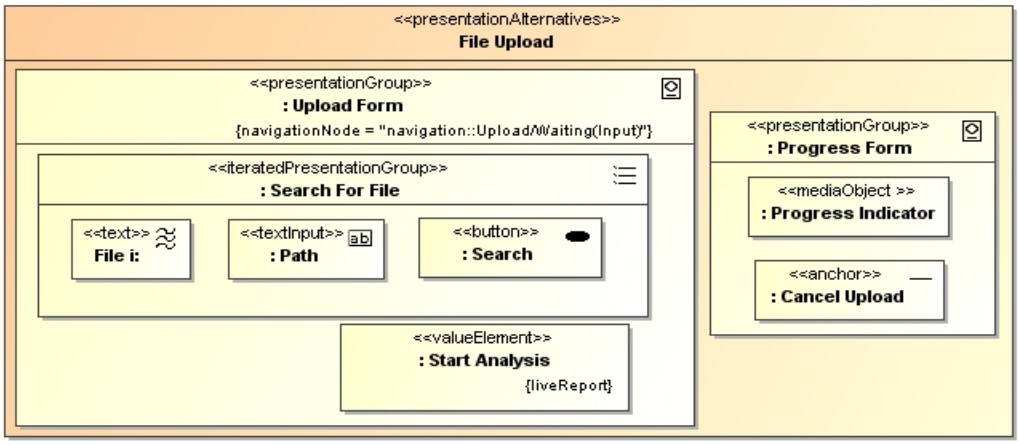


Abbildung 32. „File Upload“ Präsentationsalternative

- ***Das Präsentationsmodell*** wird um ein Zustandsdiagramm erweitert, der das Verhalten eines Objektes von der Klasse „Progress Indikator“ beschreibt (siehe Abbildung 33).

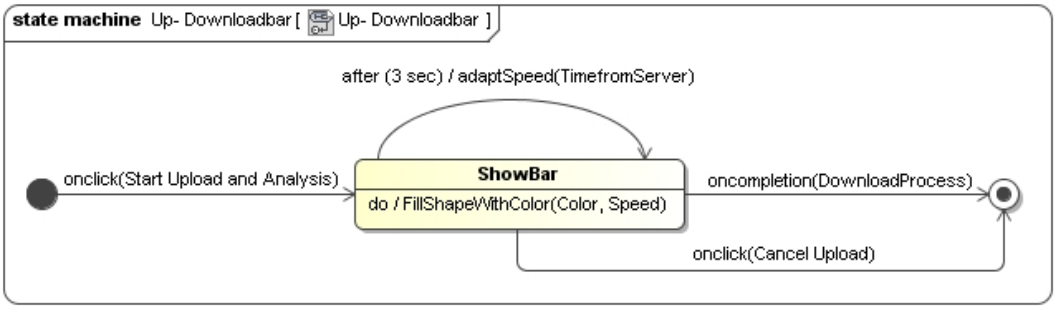


Abbildung 33. Das Verhalten von der Uploadanimation

4) Aktualisierung in Echtzeit (periodicRefresh)

Ziel: Dem Benutzer wird Information über den aktuellen Zustand von bestimmten Objekten angeboten. Diese Informationen werden periodisch beim Server angefordert.

Erweiterung des UWE Profils: Im Profil wurde der Stereotyp <<outputElement>> um ein Attribut „periodicRefresh“ vom Typ „Boolean“ mit dem Defaultwert „false“, erweitert (siehe Abbildung 34). Dementsprechend erhalten alle Klassen, die von der Klasse OutputElement erben, dieses Attribut automatisch. Falls während der Modellierungsphase eins von folgenden Stereotypen image, mediaObject oder text verwendet wird, kann der Entwickler entscheiden, ob dieses Element die interaktive Eigenschaft periodicRefresh erhalten soll oder nicht. In dem Fall, dass der Entwickler sich für ein interaktives Element entscheidet, wird der Wert von dem Metaattribut „periodicRefresh“ auf „true“ gesetzt.

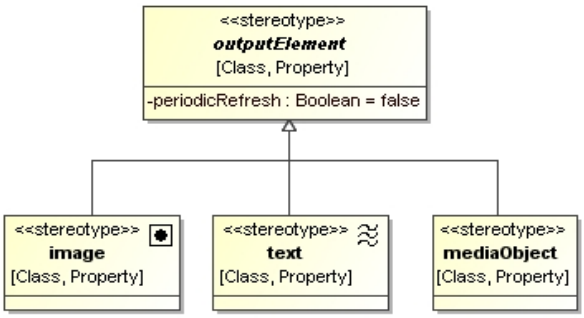


Abbildung 34. Erweiterung vom Stereotyp <<OutputElement>> in UWE

Beispiel: Zustand von einer Bildanalyse bei S.CORE (siehe Abbildung 35). Nachdem ein Benutzer ein Bild hochgeladen hat, wird automatisch dessen Analyse gestartet. Dabei erscheint neben dem Bildnamen ein Textfeld, wo der Zustand der Analyse auf dem Server angezeigt wird. Auf der Abbildung 35 sind drei Zustände vom Analyseprozess zu sehen (zur Anschaulichkeit ist das Textfeld, in dem die Zustände angezeigt werden, eingekreist). Wie auf dem Bild angezeigt wird, sind das jeweils die Zustände: Angefordert (requested), in der Bearbeitung mit der Angabe von verbliebener Zeit (processing Estimated time: < 1 min) und Erledigt (done).

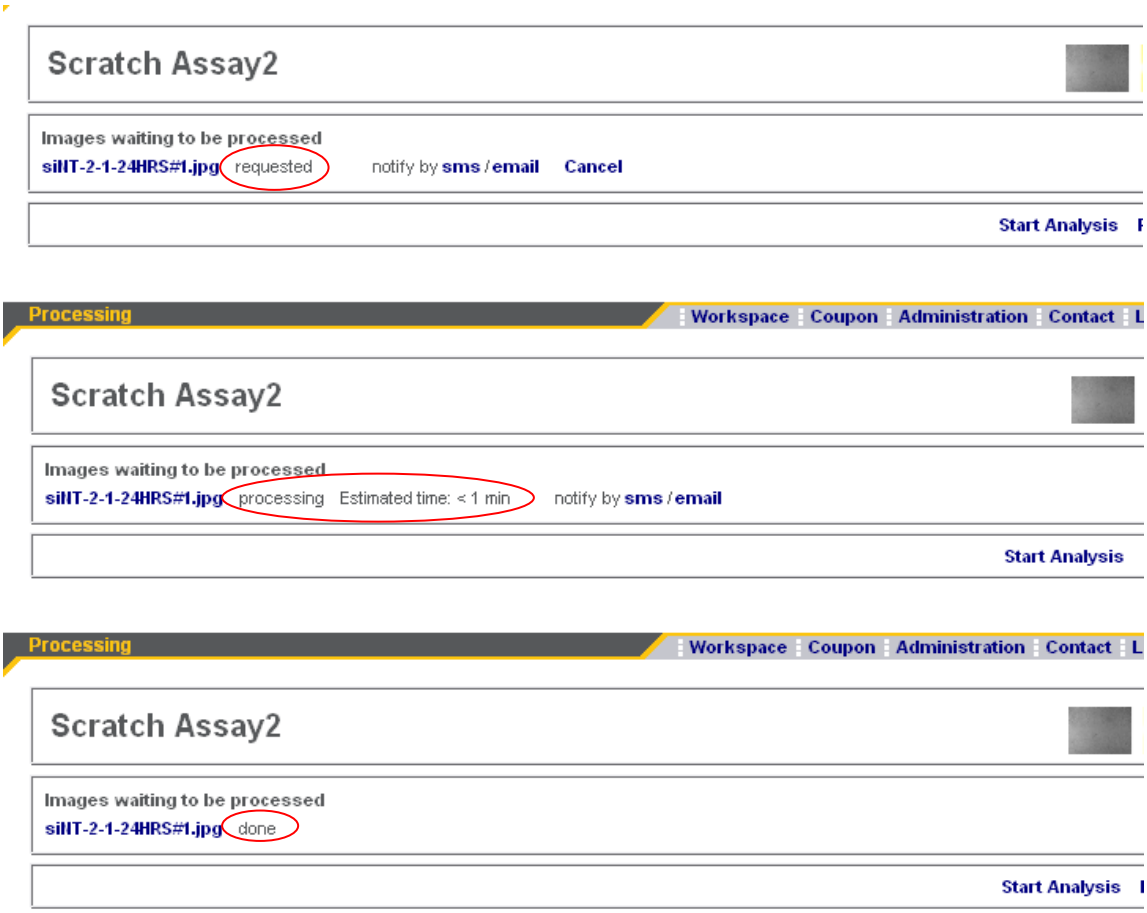


Abbildung 35. Zustand vom Analyseprozess des Bildes (siNT-2-1-24HRS#1.jpg)

Modellierungsvorschlag für das Beispiel - „Zustand vom aktuellen Analyseprozess“:

- ***Das Präsentationselement,*** das eine Liste von den Bildern darstellt, die zurzeit analysiert werden (ProcessingQueue auf der Abbildung 36), wird um einige Textfelder erweitert. Für jedes Bild wird jeweils ein Textfeld mit dem Stereotyp <<*text*>> hinzugefügt und das Metaattribut „periodicRefresh“ dieses Elements erhält den Wert „true“. Auf der Abbildung 36 wird das beschriebene Element durch Textfeld „AnalysisState“ im Modell dargestellt.

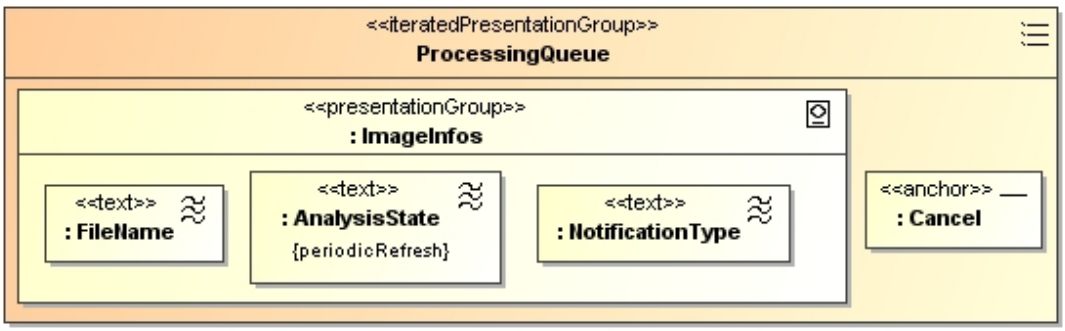


Abbildung 36. Präsentationselement „ProcessingQueue“

- ***Das Präsentationsmodell*** wird um ein Zustandsdiagramm erweitert, das das Verhalten des Textfeldes mit periodischem Aktualisieren beschreibt. Im vorgestellten Beispiel stellt das Zustandsdiagramm das Verhalten vom Präsentationselement „AnalysisState“ dar. Dabei wird dieses Diagramm zu dem Präsentationselement „ImageInfos“ hinzugefügt. Diese Platzierung des Zustandsdiagramms ist durch die Komplexität des Interaktionsprozesses zu erklären. Es werden Informationen über die Bilder gebraucht, die auf den Server zur Analyse hochgeladen wurden. Um diese Informationen beim Server anzufordern, wird der Name des Files benötigt, d.h. im Zustandsdiagramm wird ebenso ein Zugriff auf den Filenamen gefordert. Dies wird durch Gruppierung von den benötigten Elementen ermöglicht (auf der Abbildung 36 wird durch Präsentationsgruppe „ImageInfos“ dargestellt).

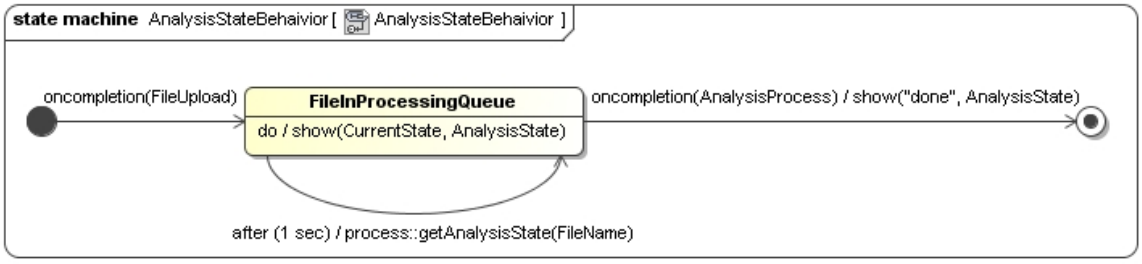


Abbildung 37. Das Verhalten von der Aktualisierung in Echtzeit

5) ***Automatische Ergängzung (autoCompletion)***

Ziel: Dem Benutzer sollte die Eingabe vom Text innerhalb der Webapplikation erleichtert werden, in dem bestimmte Textvorlagen automatisch während der Eingabe vorgeschlagen werden. Der Benutzer kann dabei entscheiden, ob er den vorgeschlagenen Text korrigiert oder nicht.

Erweiterung des UWE Profils: Die Erweiterung betrifft alle Eingabeelemente, d.h. dem Stereotyp <<inputElement>> wird ein weiteres Attribut hinzugefügt, und zwar „autoCompletion“ vom Typ „Boolean“ und dem Defaultwert „false“ (siehe Abbildung 38)

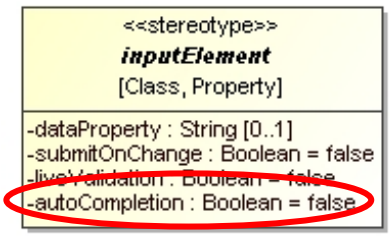


Abbildung 38. Erweiterung vom Stereotyp *inputElement* in UWE

Beispiel: Als Beispiel für diese Art der interaktiven Kommunikation mit dem Benutzer können mehrere Situationen dienen. Es könnte eine klassische Wortergänzung (word completion, siehe Abbildung 39) während der Texteingabe sein (als Abhilfe in den Fachausdrücken oder als Texteingabeerleichterung). Oder beim Ausfüllen des Formulars könnte nach der Eingabe der Postleitzahl automatisch der Stadtname eingefügt werden

(field completion). Oder bei der Eingabe, dass man ein Mitglied werden will, wird automatisch die Checkbox angeklickt, dass man alle Bedingungen des Betreibers akzeptiert (checkbox completion).

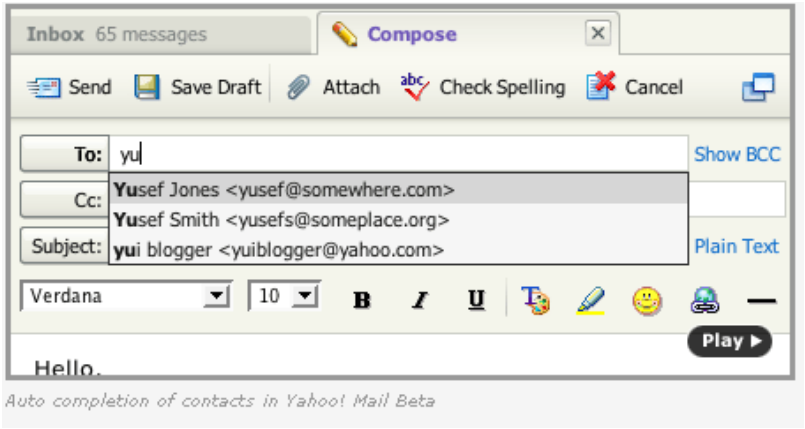


Abbildung 39. „Word completion“ bei Eingabe der gewünschten E-Mail-Adresse

Modellierungsvorschlag für das Beispiel - „field completion“:

- ***Das Präsentationselement*** „Address“ (siehe Abbildung 40) erhält den Stereotyp <<inputElement>>. Der Wert vom Metaattribut „autoCompletion“ wird auf true gesetzt. Das Verhalten von diesem Element wird durch ein Zustandsdiagramm beschrieben.

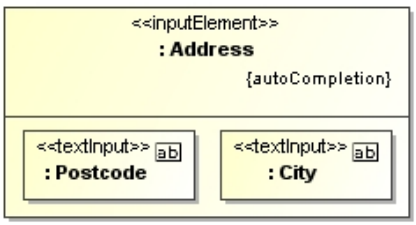


Abbildung 40. Präsentationselement Address im Formular „Account Settings“

- ***Das Präsentationmodell*** wird um ein Zustandsdiagramm erweitert, das das Verhalten des interaktiven Elementes mit automatischer Ergänzung darstellt (für beschriebenes Beispiel siehe Abbildung 41). Eine Besonderheit in diesem Beispiel spielt die Tatsache, dass das Inputelement „Address“ zwei Unterelementen „Postcode“ und „City“ besitzt. Sie wurden aus folgendem Grund mit einander vereinigt, dass ein Zustandsdiagramm immer das Verhalten von einem Objekt einer Klasse beschreibt. In diesem Fall wurde jedoch Zugriff sowohl auf „Postcode“ als auch auf „City“ benötigt. Das Diagramm (siehe Abbildung 41) ist dann folgender Maßen zu lesen: beim Verlassen des Textfeldes „Postcode“ (onblur(Postcode)) wird dessen Wert zurückgegeben (getValue(Postcode)) und im Zustand „CheckValue“ wird eine mögliche Variante für den Stadtnamen aus der Liste von Städten (CitySet) ausgewählt (serchForCompletion(Postcode.Value, CitySet)). Nach der erfolgreicher Suche wird ein passender Vorschlag zurückgegeben (getResultValue(Completion)). Danach wird

im Zustand „WaitForCorrect“ der gefundene Wert (Completion.Value) im Feld „City“ angezeigt. Dabei wird der Zustand „WaitForCorrect“ durch Benutzerauswahl vom Eingabetextfeld „City“ (onfocus(City)) verlassen. Danach kann das Feld korrigiert werden oder der Vorschlag wird angenommen.

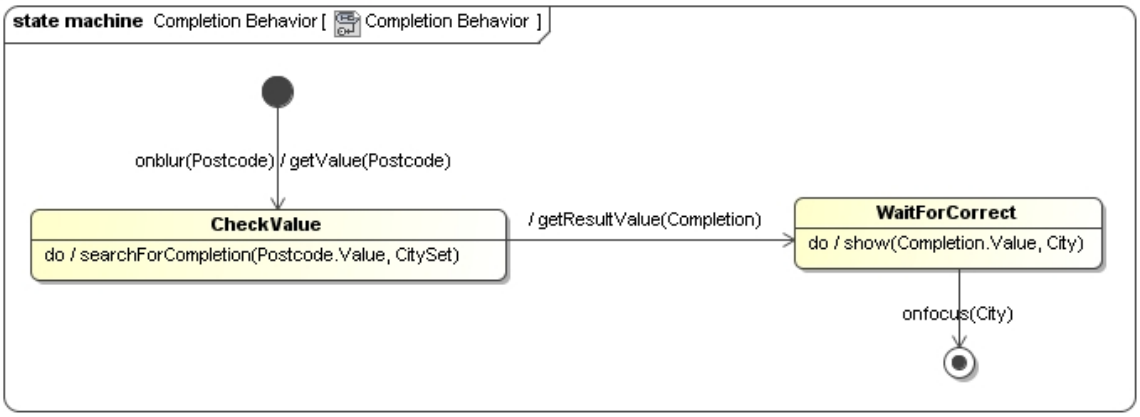


Abbildung 41. Das Verhalten von automatischer Ergänzung

6) Animation von mehrfacher Auswahl (MultiselectionIndicator)

Ziel: In vielen herkömmlichen Webanwendungen wird schon die Möglichkeit angeboten, beim Selektieren von einigen Elementen aus einer angebotenen Liste, eine mehrfache Auswahl zu treffen. Dies wird mittels einer externen Option, wie z.B. eines externen Anklickfeldes in der Checkbox, realisiert. Wenn dieses externe Feld angeklickt wird, heißt es automatisch, dass z.B. alle Felder in der Liste ausgewählt sind. Dennoch bleiben die tatsächlich ausgewählten Felder auf der Benutzeroberfläche leer. Die Animation von einer mehrfachen Auswahl dient der besseren Anschaulichkeit der Multiselektion für den Benutzer. Die Animation gibt dem Benutzer in irgendeiner Form (wie z.B. ein Button oder ein grafisches Objekt) die Möglichkeit alle ausgewählte Felder tatsächlich (d.h. in direkter grafischer Ausführung) anzuklicken.

Erweiterung des UWE Profils: keine. In diesem Fall ist das Metaattribut „multiple“ vom Stereotyp <<selection>> ausreichend (siehe Abbildung 42).

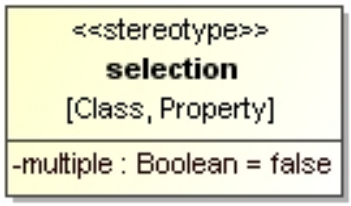


Abbildung 42. Stereotyp „selection“ in UWE

Erweiterung des UWE Metamodells: Die Metaklasse „Selection“ wird um eine „Teil des Ganzen“-Beziehung (in UML wird diese Beziehung durch die Komposition dargestellt) zu

sich selbst erweitert. Durch diese Erweiterung wurde berücksichtigt, dass ein multiselektives Objekt eine Menge von Objekten mit Einzelselektion als Teile haben darf. Durch diese Menge wird angegeben auf welche Objekte die Animation von Multiselektion sich erweitert. (Siehe Abbildung 43)

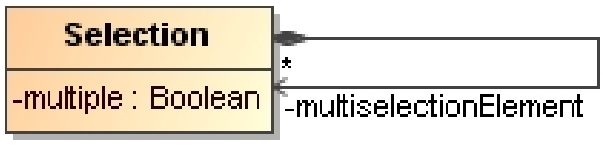


Abbildung 43. Erweiterung vom UWE Metamodell

Beispiel: Mehrfache Auswahl von Resultaten nach einer erfolgreich ausgeführten Bildanalyse bei S.CORE (siehe Kapitel 3). Dabei wird anhand von kleinen Pfeilen jeweils in jeder Zeile und jeder Spalte die Möglichkeit der mehrfachen Auswahl dargestellt. Auf der Abbildung 44 (unten rechts) sieht man den doppelten Pfeil, der die Möglichkeit bietet, alle Optionen auf einmal auszuwählen.

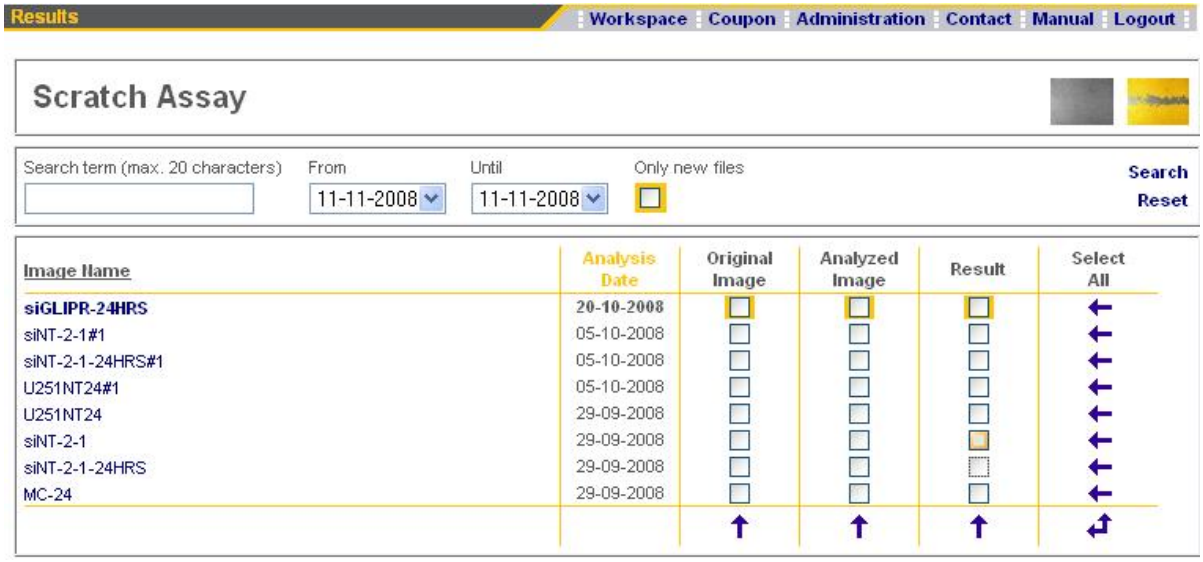


Abbildung 44. Resultatsliste von analysierten Bildern bei S.CORE

Dabei nach dem Anklicken von den Pfeilen werden jeweilige Auswahlfelder sofort in grafischer Ausführung auf der Clientseite anhand von Hacken animiert. Das Bild vom entsprechenden Pfeil wird umgedreht, was die mehrfache Auswahl anschaulich darstellt. (Siehe Abbildung 45 und Abbildung 46 jeweils für die mehrfache Auswahl in einer Spalte und für die mehrfache Auswahl aller möglichen Felder gleichzeitig). Im Fall der weiteren Betätigung des Pfeilbildes werden alle betroffenen Felder entsprechen deselektiert oder wieder selektiert.

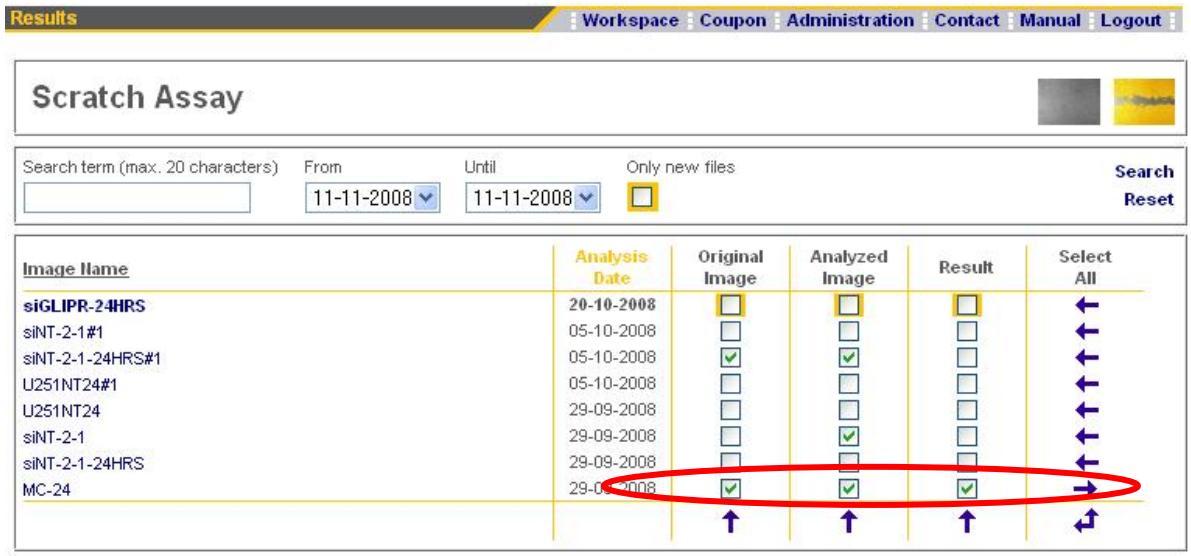


Abbildung 45. Gleichzeitige Auswahl von allen Optionen in der letzten Zeile

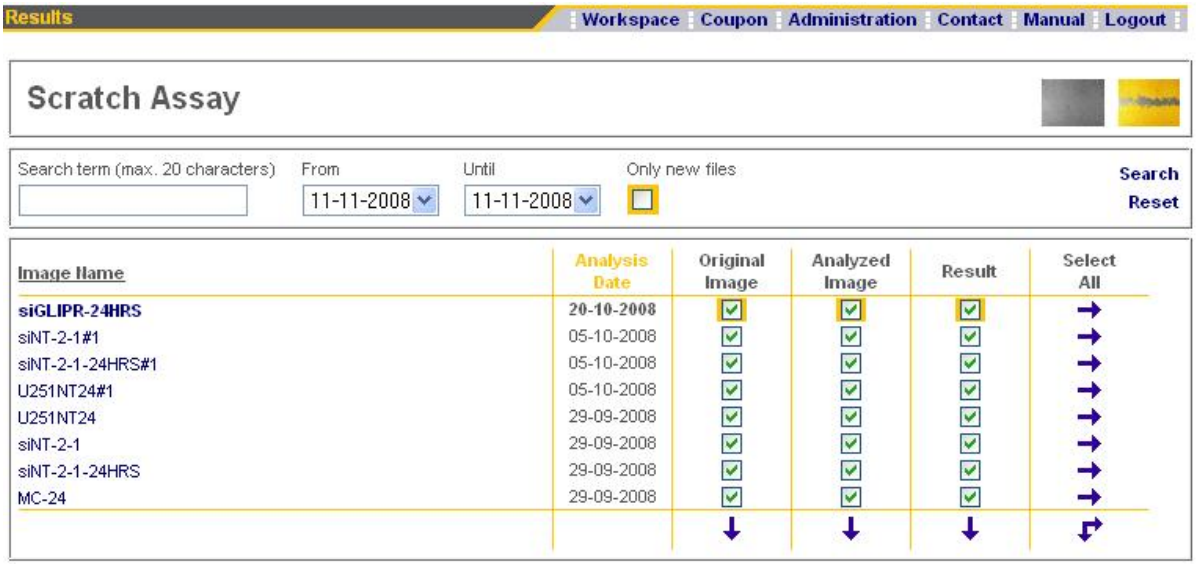


Abbildung 46. Gleichzeitige Auswahl von allen möglichen Optionen

Modellierungsvorschlag für das Beispiel - „Multiselektion von Resultaten“:

- Es werden ***Präsentationselemente***, die die Multiselektion repräsentieren zu der Auswahlliste hinzugefügt. Dabei erhalten solche Elemente den Stereotyp <<selection>> und das Metaattribut „multiple“ wird auf „true“ gesetzt. Auf der Abbildung 47 werden solche Präsentationselemente durch „RowMultiselection“, „SelectAll“, „OriginalImageMultselection“, „AnalysedImageMultiselection“ und „ResultMultiselection“ Elemente dargestellt.

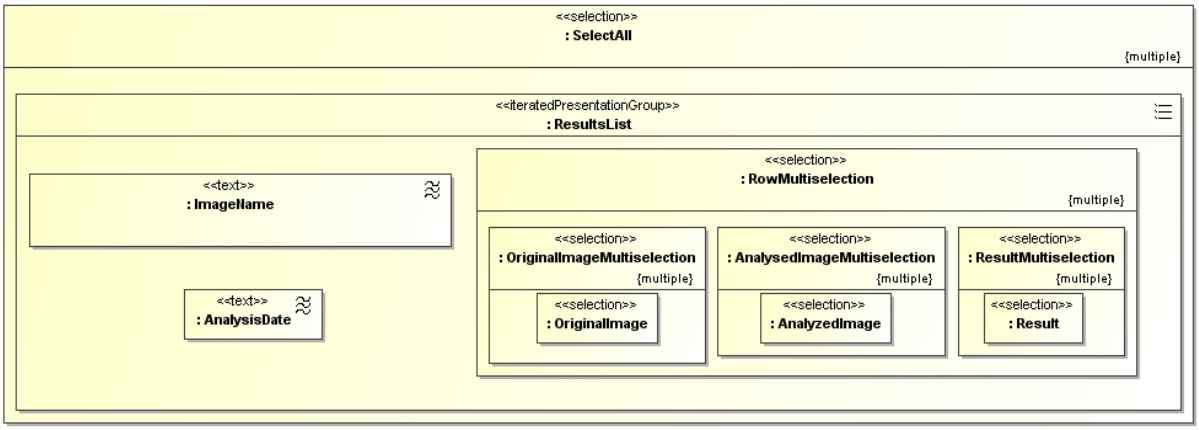


Abbildung 47. Präsentationselement „SelectAll“

- ***Das Präsentationsmodell*** wird um ein oder mehrere Zustandsdiagramme erweitert, die das Verhalten des jeweiligen interaktiven Elementes mit mehrfacher Auswahl darstellen. Das Zustandsdiagramm für „RowMultisection“ kann beispielsweise wie auf der Abbildung 48 dargestellt werden (weitere Zustandsdiagramme sind analog). Das Aufrufen von der Funktion „turnArrowPicture()“ ist für das Umdrehen vom Bild zuständig. Dieses Umdrehen von jeweiligen Pfeilbildern repräsentiert bei S.CORE die Multiselektion (siehe Abbildung 45).

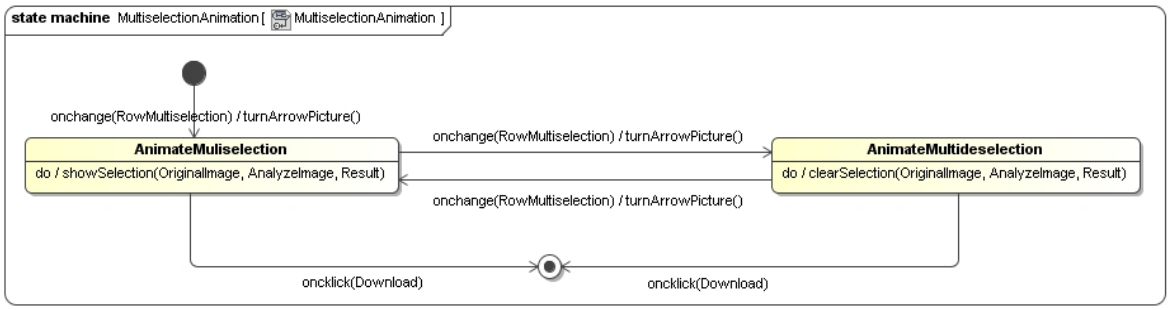


Abbildung 48. Das Verhalten von „RowMultiselection“ Animation

7) Dynamische Informationen (liveReport)

Ziel: Der Benutzer kann einige zusätzliche Informationen über ein Objekt auf der Benutzeroberfläche erhalten. Dabei werden diese Informationen anhand von kleinen Kommentaren beim Überfliegen des entsprechenden Objektes mit der Mouse angeboten.

Erweiterung des UWE Profils: Da diese Funktionalität für praktisch jedes Element der Benutzeroberfläche relevant sein kann, wird der Stereotyp „valueElement“ um ein Metaattribut „liveReport“, mit dem Typ „Boolean“ und dem Defaultwert „false“, erweitert (siehe Abbildung 49).

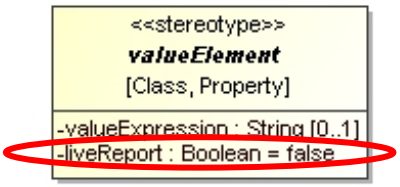


Abbildung 49. Erweiterung vom Stereotyp „valueElement“ in UWE

Beispiel: Erklärung zu den Buttons und Anchors. Beim Überfliegen von allen Buttons und Anchors im S.CORE werden die Informationen angezeigt, die den Benutzer daraufhin aufmerksam machen, was passieren würde, wenn er dieses Element anklicken würde. Auf der Abbildung 50 sieht man die Erklärung zu dem „Start Analysis“- Anchor (unten rechts), der daraufhin deutet, dass das Anklicken von diesem Element den Upload-Prozess und die Bildanalyse startet.

Abbildung 50. Dynamische Information zu dem „Start Analysis“- Anchor bei S.CORE

Modellierungsvorschlag für das Beispiel - „Erklärung zu Buttons und Anchors“:

- Analog zu den vorherigen Beispielen wird beim Anwenden des jeweiligen Stereotypes auf ***das entsprechende Präsentationselement*** das Metaattribut „liveReport“ auf „true“ gesetzt. Auf der Abbildung 51 sind das „Start Analysis“ und „Results“ Elemente, die die interaktive Elementerweiterung - „liveReport“ besitzen.

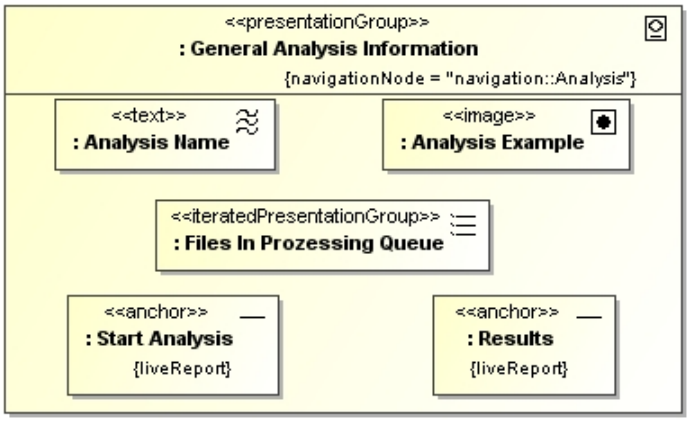


Abbildung 51. „General Analysis Information“- Präsentationselement

- ***Das Präsentationsmodell*** wird um ein Zustandsdiagramm erweitert, das das gewünschte Verhalten der dynamischen Information beschreibt. Ein mögliches Verhalten von der dynamischen Information für das Präsentationselement „Start Analysis“ ist auf der Abbildung 52 vorgestellt.

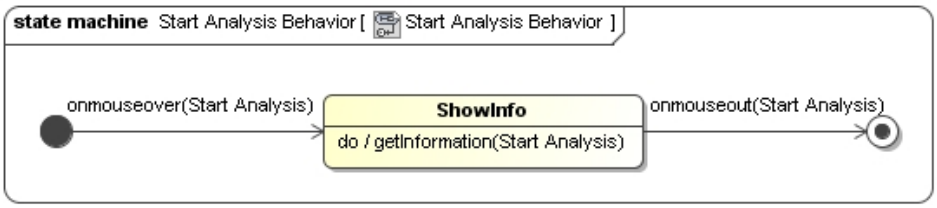


Abbildung 52. Das Verhalten von der dynamischen Information bezüglich des „Start Analysis“- Präsentationselementes

8) Bildergalerie (gallery)

Ziel: Der Benutzer soll eine Möglichkeit bekommen, eine Menge von Bilder relativ schnell und bequem in der vergrößerter Ausführung betrachten zu können. Dabei beim Anklicken eines von den Bildern, das einer Galerie gehört, wird es in der vergrößerter Ausführung angezeigt. Während der Zeit, die der Benutzer für detaillierte Betrachtung des ausgewählten Bildes braucht, werden im Hintergrund andere Bilder aus der Galerie vom Server in den Client geladen. Dadurch können weitere Bilder sehr schnell und bequem in der angebotenen Galerie angeschaut werden.

Erweiterung des UWE Profils: Der Stereotyp „outputElement“ wird um ein Metaattribut „gallery“ mit dem Type „String“ erweitert (siehe Abbildung 53). Dabei, falls ein interaktives Element eine Bildergalerie repräsentieren soll wird dem Metaattribut „gallery“ der Name von der entsprechenden Galerie als dessen Wert angegeben. Diese Erweiterung verbreitert sich gemäß dem Vererbungsprinzip auf alle Unterklassen vom Stereotyp „OutputElement“, d.h. auf die Stereotypen „image“, „text“ und „mediaObject“.

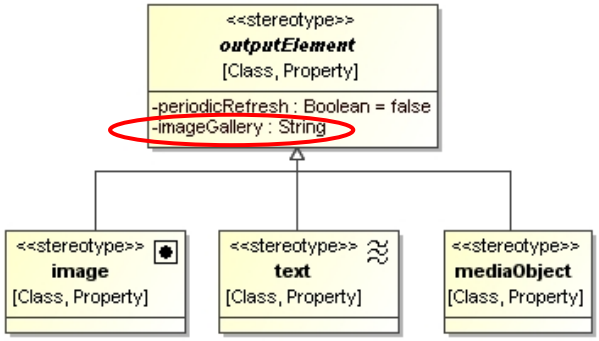


Abbildung 53. Erweiterung vom Stereotyp „outputElement“ in UWE

Beispiel: In S.CORE werden Bildergalerien für resultierende Bilder nach einer Bildanalyse angeboten. Dabei das Anklicken vom Namen eines zu der Analyse geschickten Bildes startet die dazugehörige Bildergalerie. Die Abbildung 54 zeigt eine

Galerie, die alle Bilder, die zu der Bildanalyse vom Bild, mit dem Namen - „siGLIPR-24HRS“, gehören.

Abbildung 54. Bildergalerie zu dem analysierten „siGLIPR-24HRS“- Bild

Modellierungsvorschlag für das Beispiel - „Bildergalerien zu den Bildanalysen“:

- Das Metaattribut „gallery“ vom ***Präsentationselement*** - „ImageName“ im Präsentationsmodel erhält den Wert – „concat(ImageName.Value, „_Gallery“)“. Die Funktion- concat(String1, String2) konkateniert bzw. fügt den tatsächlichen Namen von dem jeweiligen Bild und den String „_Gallery“ zusammen (siehe Abbildung 55). Im oben aufgegriffenen Beispiel würde der resultierende String dem tatsächlichen Wert - „siGLIPR-24HRS_Gallery“ gleich bedeuten.

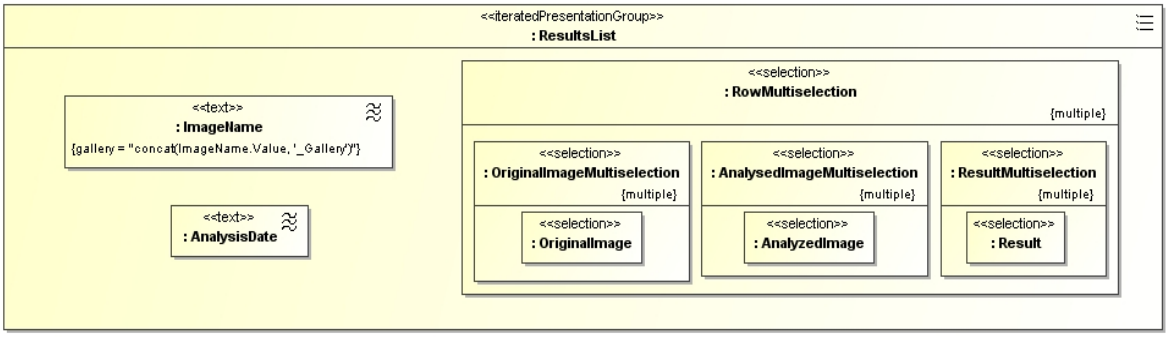


Abbildung 55. Präsentationselement „ResultsList“

Nach dem Anklicken von einem der Bildnamen aus der Liste (repräsentiert durch die Klasse „ResultsList“) erscheint die jeweilige Galerie zu diesem Bild. In dieser Bildergalerie können der Reihe nach der OriginalImage, AnalyzedImage oder Result angezeigt werden. Das Layout von der Galerie wird auf der Abbildung 56 durch das Präsentationselement „Gallery“ dargestellt.

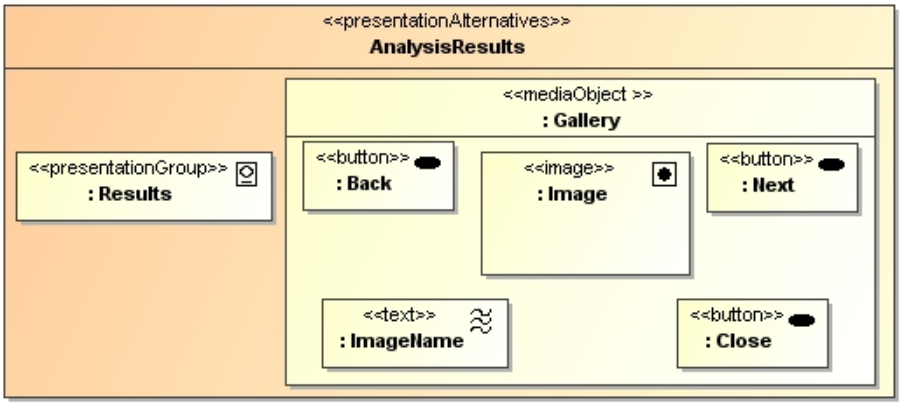


Abbildung 56. Präsentationselement „AnalysisResults“

- ***Das Präsentationsmodell*** wird um ein Zustandsdiagramm erweitert, das das Verhalten von der Bildergalerie beschreibt. (Siehe Abbildung 57)

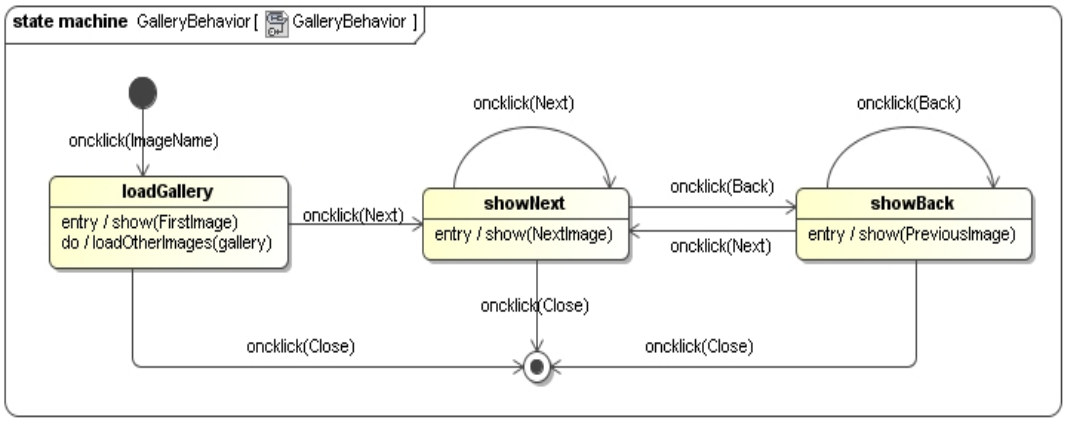


Abbildung 57. Das Verhalten von der Bildergalerie in der „ResulsList“

9) Verfeinerte Suche (liveSearch)

Ziel: Der Benutzer bekommt eine Möglichkeit, die Webanwendung zu durchsuchen, dabei wird jeweilige Suche an ein oder mehrere Parameter angepasst. Zum Beispiel kann die Suche angepasst an ein bestimmtes Datum durchgeführt werden, oder angepasst an ein textuelles Muster, oder beides zusammen. Die Parametrisierung der Suche erfolgt durch eine Eingabemaske auf der Benutzeroberfläche.

Erweiterung des UWE Profils: Der Stereotyp „uiElement“ wird um ein Attribut „liveSearchCondition“ vom Typ „String“ erweitert (siehe Abbildung 58). Alle Unterklassen vom Stereotyp „uiElement“ (d.h. alle Präsentationsstereotypen) erhalten dieses Attribut automatisch (siehe Abbildung 71).

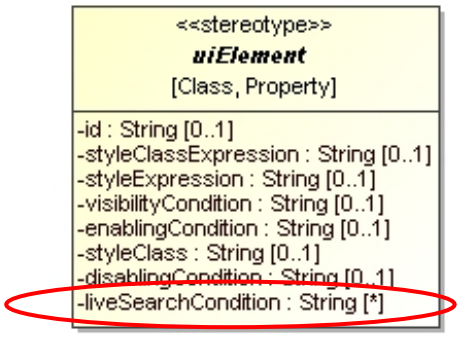


Abbildung 58. Erweiterung vom Stereotyp „uiElement“ in UWE

Beispiel: Durchsuchen von Resultaten der Bildanalysen bei S.CORE. Die analysierten Bilder werden in S.CORE im Benuzer-Workspace gesammelt. Dabei können Resultate anlehnend an ein vom Benutzer angegebenes Datum durchsucht werden. Der Benutzer kann das Datum aus einer Liste auswählen, danach spielt dieses Datum beim Durchsuchen die Rolle des Durchsuchungsparameters. (Siehe Abbildung 59)

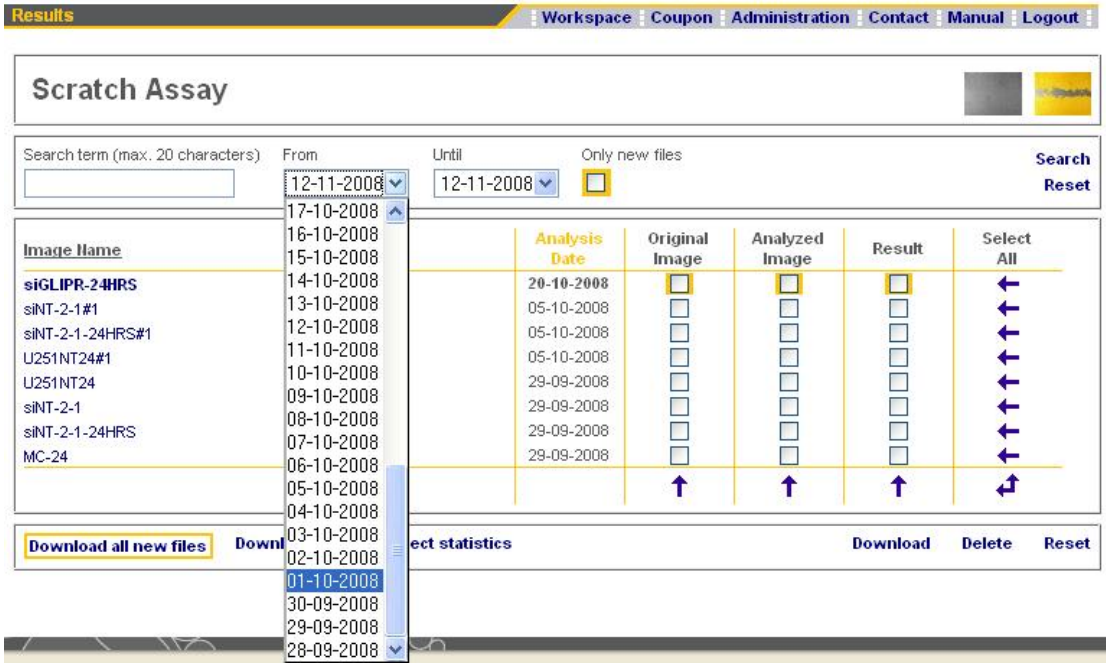


Abbildung 59. Parametrisierung von verfeinerter Suche durch Datumsauswahl

Modellierungsvorschlag für das Beispiel - „Durchsuchen von Resultaten“:

- Dem Präsentationsmodel wird ***ein Präsentationselement*** „Searchbox“ hinzugefügt, das die Durchsuchungsparameter auf der Benutzeroberfläche verwirklicht (auf der Abbildung 60 sind das jeweils ImageNameExample, FromDate, UntilDate, OnlyNewFiles Präsentationselemente, die entsprechende Parameter für Suche nach bestimmten Namen, nach bestimmten Datum bzw. im bestimmten Zeitintervall oder nach nur neue Files repräsentieren). Außerdem wird dem Modell ein Präsentationselement „Search“ hinzugefügt. Dieses Element vereint alle Elemente, die

für die Suche und für die spätere Darstellung von Resultaten relevant sind. Darauf wird der Stereotyp <<presentationGroup>> angewandt und dessen Metaattribut „liveSearchCondition“ wird mit einem String belegt, der die Beziehung zwischen den Durchsuchungsparametern und den Elementen, die durchsucht werden sollen, beschreibt. Im beschriebenen Beispiel sind folgende Beziehungen nachvollziehbar, ImageName = ImageNameExample.Value (somit soll falls angegeben den Wert von dem Präsentationselement ImageNameExample gleich dem Bildnamen sein), FromDate.Value < AnalysisDate < UntilDate.Value (d.h., dass das Analysedatum kleiner als der Wert des FromDate und größer als der Wert des UntilDate Elementen sein soll). (Siehe Abbildung 60)

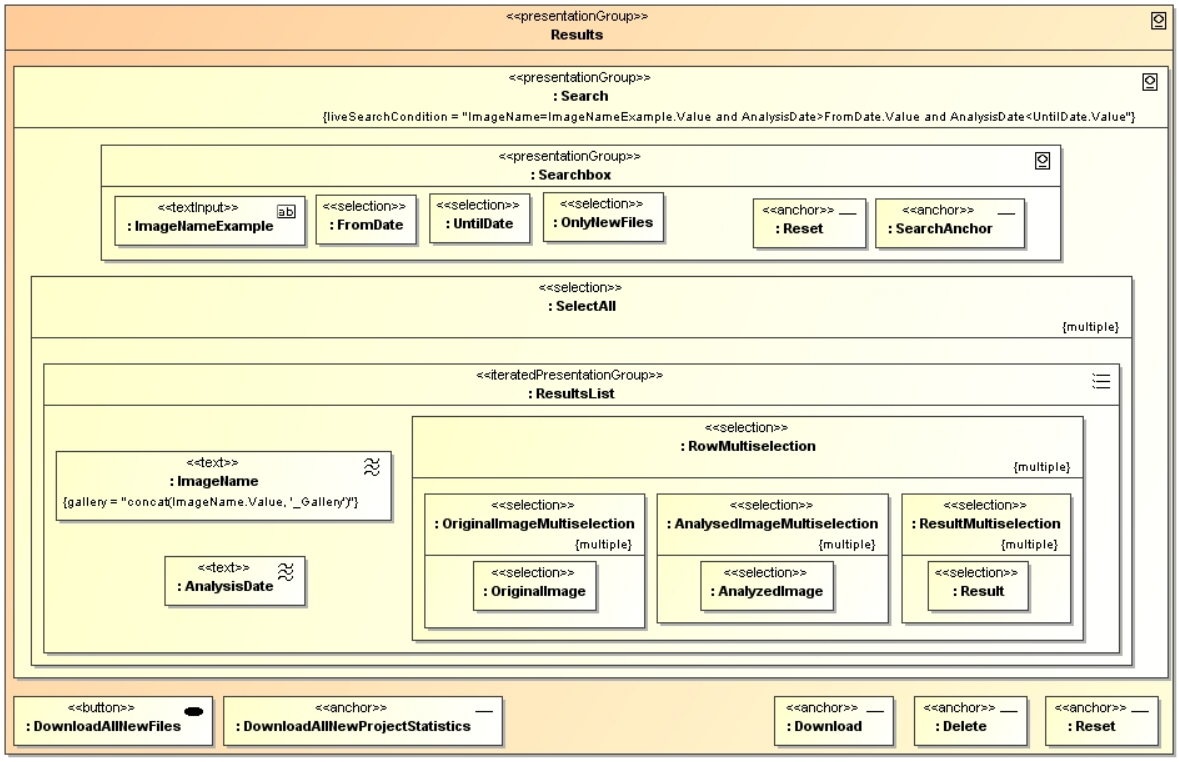


Abbildung 60. Präsentationselement Results

- ***Das Präsentationsmodell*** kann um ein Zustandsdiagramm erweitert werden, das das Verhalten des interaktiven Elementes - „Search“ von der Abbildung 60 darstellt und alle Zusammenhänge zwischen den in der Searchbox gesetzten Parameter und den tatsächlichen Objekten, die durchsucht werden sollen, beschreibt.

10) Informative Führung (GuidedTour)

Ziel: Dem Benutzer werden interaktive meist multimediale Informationen über bestimmte Objekte geboten. Meist stellen diese Informationen eine Art von Auskunft dar und dienen der Beschreibung entweder eines Produktes oder der Webapplikation selbst.

Erweiterung des UWE Profils: keine. Je nach der Darstellungsart kann der Stereotyp <<mediaObject>> oder ein Stereotyp mit dem Wert „true“ des Metaattributs „liveReport“ verwendet werden.

Beispiel: Als Beispiel kann man der GuidedTour bei Nokia, für die neueste Geräte auf dem Markt (siehe Abbildung 61), nennen. (Dabei wäre die Modellierung von solch einer Führung anhand von interaktiven Elementen „dynamische Infos“ (siehe Punkt 7) möglich.) Wie genau „GuidedTour“ funktionieren soll, kann mit einem Zustandsdiagramm im Präsentationsmodel beschrieben werden.

Abbildung 61. GuidedTour bei Nokia für das Gerät 7650

Ein weiteres Beispiel für GuidedTour ist die multimediale Hilfe bei „Yahoo! Tech“. Dabei wird die Einführung anhand eines kleinen Filmes durchgeführt. In diesem Fall kann das Präsentationselement im Präsentationsmodel den Stereotyp <<mediaObject>> tragen. Die Benutzerinteraktion mit dem System kann mit einem Zustandsdiagramm im Präsentationsmodell beschrieben werden.

Abbildung 62. Multimediale Hilfe bei Yahoo! Tech

11) Drag&Drop

Ziel: Dem Benutzer wird die Möglichkeit geboten, die Lage von manchen Elementen auf der Benutzeroberfläche zu verändern. Dabei anhand Drag&Drop-Technik (ziehe und lass fallen), die sich bei Desktopanwendungen etabliert hat, werden einige Elemente verschoben. Dadurch ändert sich das Layout von der gesamten Applikationsoberfläche.

Erweiterung des UWE Profils: Stereotyp <<presentationGroup>> wird um ein Metaattribut „drag&drop“ mit dem Typ - „Boolean“ und dem Defaultwert „false“ erweitert. (Siehe Abbildung 63) Im Fall, dass bei einem Präsentationselement mit dem Stereotyp <<presentationGroup>> der Wert von dem Metaattribut „drag&drop“ auf „true“ gesetzt wird, sind die einzelne Elemente innerhalb dieser Gruppe verschiebbar.

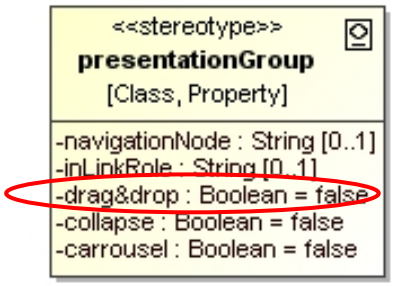


Abbildung 63. Erweiterung vom Stereotyp <<presentationGroup>> in UWE

Beispiel: Als Beispiel kann freie Gestaltung von Startseite bei Google (siehe [w8]) im Rahmen des Projektes – „iGoogle“ dienen. Dabei wird das Layout unter dem Benutzernamen vom jeweiligen Benutzer gespeichert. Alle Arbeitfelder (Kalender, Suchfenster, Wetterauskunft u.a.) können beliebig auf der Seite verschoben werden. (Siehe Abbildung 64)

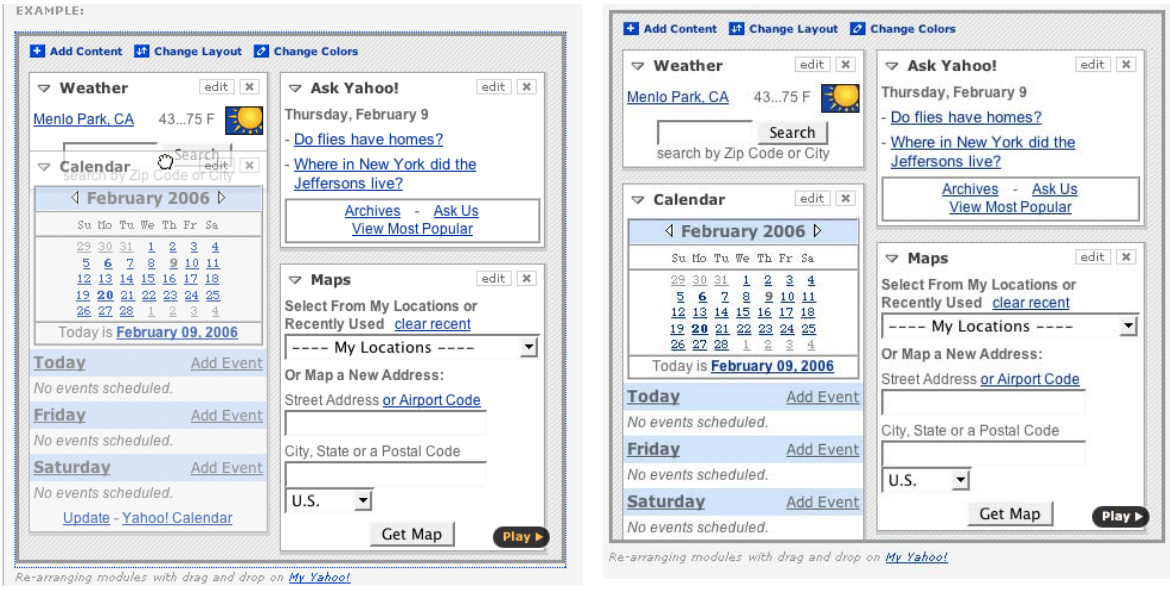


Abbildung 64. Freie Gestaltung von iGoogle Seite anhand Drag&Drop-Technik

12) Karussell (carrousel)

Ziel: Eine platzsparende und pfiffige Idee zu der Organisation einer Liste von Objekten. Sie gibt dem Entwickler eine Möglichkeit Bilder oder andere grafische Objekte in der Form von so genannten Karussell zu organisieren. Dabei wird ein vergleichbar kleines Fenster angeboten. In dieses Fenster passen maximal ein Paar Bilder. In diesem Fenster oder in der Nähe wird ein Bedienungselement angeboten, wobei beim Betätigen dieses Elementes ändern sich die Bilder im Präsentationsfenster. Der Benutzer hat dabei den Eindruck, dass die Bilder sich drehen, wie auf einem Karussell. Genauer gesagt wird beim Benutzer ein Gefühl erweckt, dass man ein Karussell dreht, auf dem alle Bilder platziert sind. Dies macht das System unterhaltsam und interessant.

Erweiterung des UWE Profils: Stereotyp <<presentationGroup>> wird um ein Metaattribut „carrousel“ mit dem Typ „Boolean“ und dem Defaultwert „false“ erweitert. (Siehe Abbildung 65) Analog zu drag&drop, wird das Setzen vom Metaattribut „carrousel“ auf „true“ gleich dem interaktiven Layout von inneren Elementen einer Gruppe bedeuten.

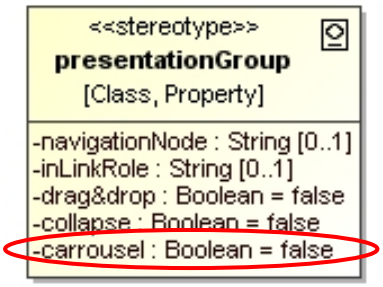


Abbildung 65. Erweiterung vom Stereotyp <<presentationGroup>> in UWE

Beispiel: Ein hervorragendes Beispiel für solche Organisation von Medialen und Graphischen Objekten stellt die Organisation von Rezepten bei „Yahoo! Food“ (siehe Abbildung 66). Dabei kann der Benutzer sehr schnell und bequem eine Menge von Rezepten durch blättern (Siehe Abbildung 66) und ein aufgefallenes Rezept anschauen. Oder die Darstellung von den Filmlisten bei „Yahoo! Movies“ (siehe [w9])

Abbildung 66. Karussell von Rezepten bei „Yahoo! Food“

13) Kollaps (collapse)

Ziel: Der Benutzer bekommt eine Möglichkeit nicht interessante oder nach Meinung des Benutzers unnötige Elemente auszuschalten. Dabei kommt den multimedialen Effekt vom Kollabieren zum Einsatz. Die grafische Oberfläche wird wörtlich zusammen geklappt. Dabei wird das verschwundene Element ansatzweise, zum Beispiel anhand eines Tags oder eines Buttons, auf der Seite dargestellt. Zum Ausklappen der Applikation soll in der Regel dieses Element betätigt werden, dann entsteht eine vollständige Version von der Benutzeroberfläche.

Erweiterung des UWE Profils: Eben dem vorherigen Beispiel wird Stereotyp <<presentationGroup>> um ein Metaattribut „collapse“ mit dem Typ „Boolean“ und dem Defaultwert „false“ erweitert. (Siehe Abbildung 67)

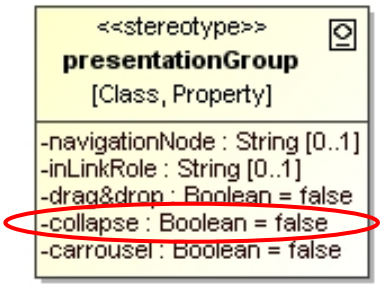
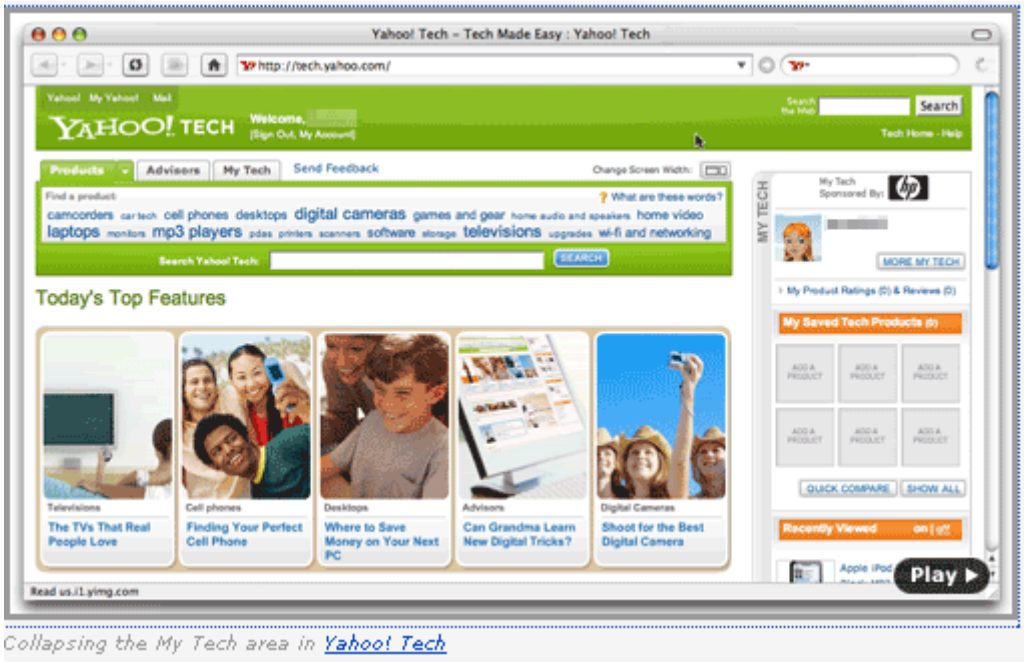


Abbildung 67. Erweiterung vom Stereotyp <<presentationGroup>> in UWE

Beispiel: Als Beispiel kann eine andere durch Yahoo entwickelte Webanwendung - „Yahoo! Tech“ dienen. Benutzerdefinierte Elemente vom Präsentationselement - „MY TECH“ können verborgen werden. Die vollständige Version vom Layout auf „Yahoo! Tech“ sieht man auf der Abbildung 68.



Collapsing the My Tech area in Yahoo! Tech

Abbildung 68. Vollständiges Layout von Yahoo! Tech

Nach dem Kollabieren sieht man eine schmalere und übersichtlichere Version vom Layout. (Siehe Abbildung 69.)

Collapsing the My Tech area in Yahoo! Tech

Abbildung 69. Layout im kollabierten Zustand

Abschließende Diagramme:

Alle beschriebenen Erweiterungen von UWE betreffen das Präsentationsmodell. Auf der Abbildung 70 wird das abschließende Diagramm für alle beschriebenen Erweiterungsvorschläge von UWE Metamodell dargestellt. Metaklasse „Selection“ hat eine „Teil des Ganzen“-Beziehung (Komposition in UML) zu sich selbst als Erweitung bekommen. Durch diese Erweiterung wurde berücksichtigt, dass ein multiselektives Objekt eine Menge von Objekten mit Einzelselektion als Teile haben muss. Durch diese Menge wird angegeben, auf welche Objekte die Animation von Multiselektion sich erweitert.

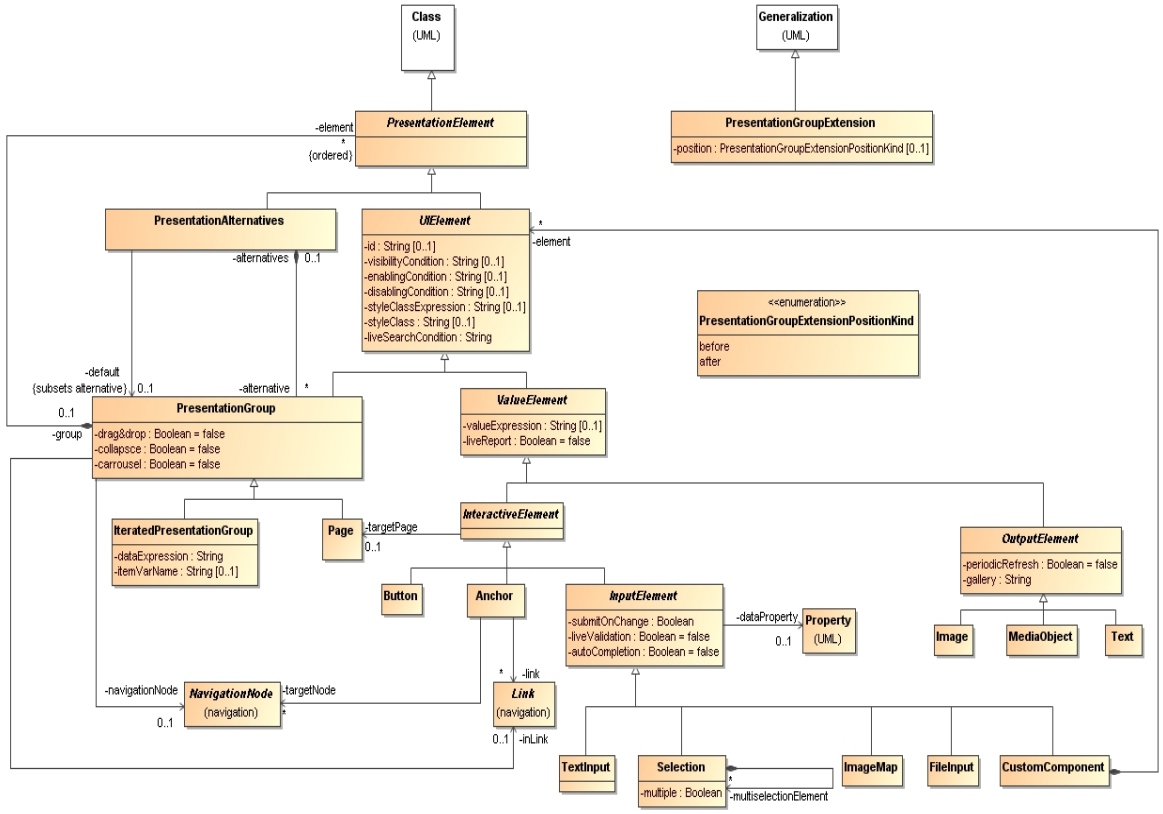


Abbildung 70. Erweitertes UWE Metamodell: Präsentationsteil

Die nachfolgende Abbildung 71 stellt die gesamte Erweiterung des UWE Profils im Präsentationsteil dar. Die vorgenommnen Erweiterungen sind auf dem Bild eingekreist. Aus der Abbildung 71 wird deutlich, dass die ausgearbeitete Erweiterung von UWE Profil sehr übersichtlich geworden ist. Dennoch wurden dadurch viele mächtige Elemente auf der Modellierungsebene als Abhilfe für den Entwickler geschaffen. Dabei sind die Uwe-Modellierungselemente sehr flexibel geworden, denn dem Entwickler ist die Freiheit gelassen, während der Modellierung zu entscheiden, ob das Element interaktive Funktionalitäten erhalten soll oder nicht. Die beschriebene Erweiterung stellt eine Erweiterung mit übergreifender Wirkung dar, denn anhand vom Vererbungsprinzip die getroffene Erweiterung von allen Unterelementen automatisch übernommen wird.

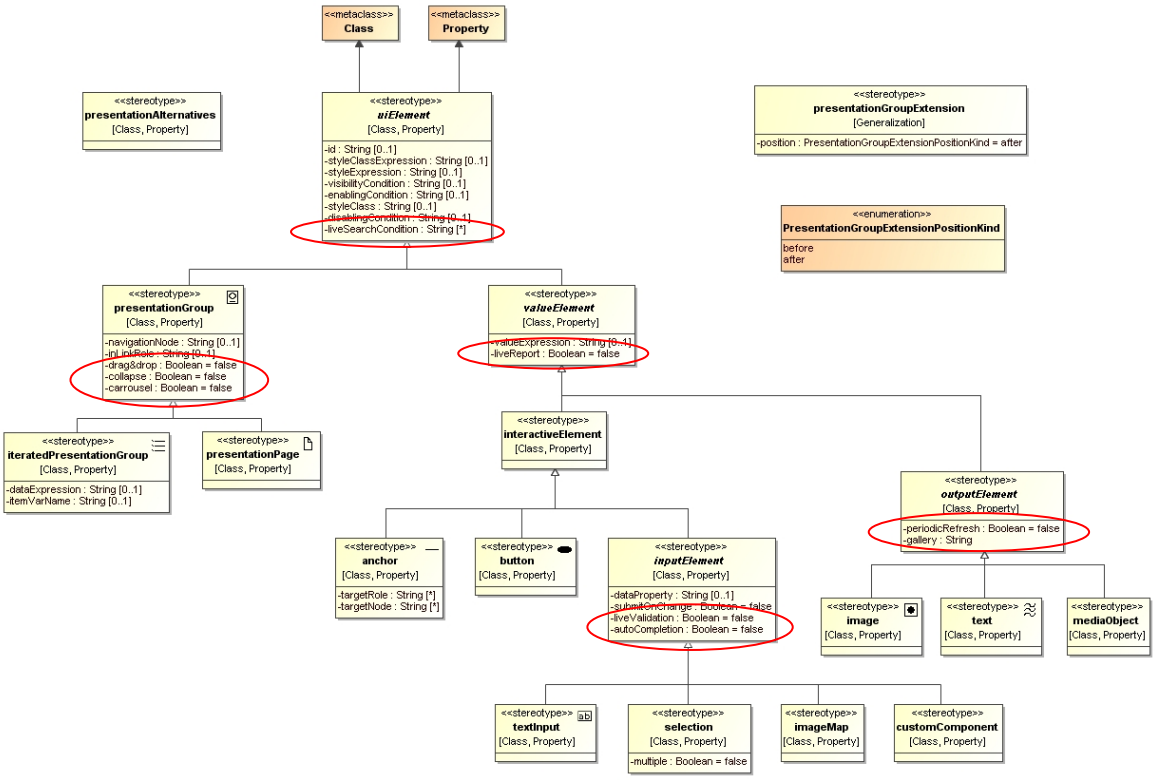


Abbildung 71. Erweitertes UWE Profil: Präsentationsstereotypen

Alle Erweiterungen zu den Präsentationsstereotypen sind noch mal in der Tabelle 12 aufgelistet. Alle Unterelemente erhalten diese Erweiterungen anhand des Vererbungsprinzips automatisch.

Stereotyp	Erweiterung
uiElement (user interface element)	Wurde um ein Metaattribut: **liveSearchCondition** erweitert.
presentationGroup	Wurde um die Metaattribute: **drag&drop, collapse und carrousel** erweitert.
valueElement	Wurde um ein Metaattribut: **liveReport** erweitert.
inputElement	Wurde um die Metaattribute: **liveValidation und autoCompletion** erweitert.
selection	Auf der Metamodellebene wurde um eine **Kompositionsbeziehung** zu sich selbst erweitert.
outputElement	Wurde um die Metaattribute: **periodicRefresh und gallery** erweitert.

Tabelle 12. Erweiterte Elemente in UWE

Kapitel 5 Resultate und Ausblick

Im Rahmen dieser Diplomarbeit wurde zum Einen die moderne Webanwendung S.CORE erweitert, die bei Firma S.CO LifeScience GmbH entwickelt wurde. Dabei wurde eine Grundlage für mehrfaches Benutzen eines Benutzerkontos geschaffen. Bei der Implementierung kamen moderne Technologien wie AJAX zum Einsatz. Parallel wurden die Modelle auf den neusten Stand im Entwicklungsprozess vom S.CORE gebracht. Dabei wurde die neueste Version vom UWE Profil und Metamodell als methodische Vorlage für die Modellierung verwendet.

Zum Anderen wurde eine ausführliche Untersuchung von heutigen Trends auf dem Gebiet von modernen Technologien für die Entwicklung und Implementierung von zeitgerechten Webapplikationen durchgeführt. Dabei wurde eine Liste von modernen interaktiven Präsentationselementen erstellt, die zurzeit der Verfassung der vorliegenden Arbeit bei der Implementierung von Web 2.0 Webanwendungen anhand von modernen Technologien zum Einsatz kamen.

Abgesehen davon wurden basierend auf der ausgearbeiteten Liste von Präsentationselementen konkrete Vorschläge zur Modellierung von solchen Elementen mittels des UWE Ansatzes gemacht. Dafür wurde UWE überarbeitet und erweitert. Innerhalb dieser Erweiterung wurde eine Ereignissprache anlehnend an JavaScript beschrieben. Viele der Vorschläge wurden im Rahmen der Erweiterung von S.CORE validiert und in die Praxis umgesetzt.

In den anschließenden Arbeiten kann automatische Generierung von Web 2.0 Webanwendungen basierend auf den Resultaten von dieser Arbeit in die Praxis umgesetzt werden. Dafür sollen entsprechende Modelltransformationen beschrieben werden. Somit wird der Quellkode generiert. Bei der Generierung kann sowohl PHP als auch Java Server Faces (JSF) als Zielprogrammiersprache gewählt werden.

Eine weitere Erweiterung stellt eine detaillierte Ausarbeitung der Vorschläge für die Modellierung von interaktiven Elementen, die im Rahmen dieser Arbeit nicht ausführlich betrachtet sind. Analog dazu besteht der Bedarf in konkreten Modellierungsvorschlägen, für die im Abschnitt 4.2 erwähnte Elemente, wie „Slider“, „Filter“ und direkte „Online-Annotation“. Dabei können einige Mängel in der Ereignissprache aufgedeckt werden. Dies würde einige Erweiterungen im „Event Language“ bewirken.

Literatur

Webanwendungen

[w1] http://de.youtube.com/
[w2] http://de.wikipedia.org/wiki/Wikipedia:Hauptseite
[w3] https://www.google.com/accounts/ServiceLogin?service=mail&passive=true&rm=false&continue=http%3A%2F%2Fmail.google.com%2Fmail%2F%3Fui%3Dhtml%26zy%3Dl&bsv=1k96igf4806cy&ltmpl=googlemail
[w4] https://login.yahoo.com/config/mail?&.src=ym&.intl=de
[w5] http://www.tafiti.com/
[w6] http://www.sco-lifescience.de/
[w7] http://www.score.sco-lifescience.eu/
[w8] http://www.google.de/ig?hl=de&source=iglk
[w9] http://food.yahoo.com/
[w10] http://movies.yahoo.com/

Referenzen

[1] RIA-Technologien im Kurzvergleich. http://www.it-republik.de/php/artikel/1870, letzter Zugriff: 18.08.08;

[2] RIA- Zauber- oder Unwort? http://entwickler.de/zonen/portale/psecom,id,99,news,44231,p,0.html, letzter Zugriff: 19.08.08;

[3] What is Web 2.0? http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, letzter Zugriff: 21.08.08;

[4] Cristian Darie, Bogdan Brinzerea, Filip Chereches-Tosa, Mihai Bucica: „Ajax und PHP. Interaktive Webanwendungen für das Web 2.0 Erstellen.“ 2007 Carl Hanser Verlag, München Wien

[5] RIA Patterns. Best Practices for Common Patterns of Rich Interaction http://billwscott.com/share/presentations/2007/e7/RIA-Best-Practice-UI11WebApps.pdf, letzter Zugriff: 07.09.08;

[6] Wikipedia http://www.wikipedia.org/, letzter Zugriff: 10.11.08

[7] Kai Jäger: “Ajax in der Praxis”. 2008 Springer- Verlag Berlin Heidelberg

[8] Nora Koch, Alexander Knapp, Gefei Zhang, Hubert Baumaeister: „UML- based Web Engineering. An Approach based on Standards“, chapter 7, 157-191 In Web Engineering: Modelling and Implementing Web Applications. Gustavo Rossi, Oscar

Pastor, Daniel Schwabe and Luis Olsina (Eds.), ©Springer, HCI, Vol 12, January 2008

[9] Object Management Group – Unified Modeling Language (UML), http://www.uml.org, letzter Zugriff 24.10.08

[10] Helmut Balzert: „Lehrbuch der Software-Technik. Bd.1. Software-Entwicklung." 1996, 1998, 2001 Spektrum Akademischer Verlag, Heidelberg

[11] What is Web 2.0? http://www.oreilly.de/artikel/web20.html, letzer Zugriff: 20.08.08

[12] Jaime Gómez, Cristina Cachero: "OOH Method: Extending UML to Model Web Interfaces." http://gplsi.dlsi.ua.es/iwad/ooh_project/papers/igp02.pdf, 2003, Idea Group, Inc.

[13] Nora Koch and Anderas Kraus (LMU München, Germany), C. Cachero and S. Meliá (Universidad de Alicante, Spain): "Modeling Web Business Processes with OO-H and UWE." http://www.pst.informatik.uni-muenchen.de/personen/kochn/iwwost03-koch-others.pdf, letzter Zugriff: 07.11.08, In Daniel Schwabe, Oscar Pastor, Gustavo Rossi, and Luis Olsina, editors, *Proc. 3rd Int. Workshop on Web-Oriented Software Technology (IWWOST2003)*, pages 27-50, 2003.

[14] Daniel Schwabe, Gustavo Rossi: „The Object-Oriented Hypermedia Design Model (OOHDM)." http://www.telemidia.puc-rio.br/oohdm/oohdm.html, letzter Zugriff 27.10.08

[15] Web Modeling Language (WebML), http://www.webml.org/, letzter Zugriff: 01.11.08

[16] Nathalie Moreno, Piero Fraternali and Aantonio Vallecillo: "WebML modelling in UML." 2007 IET Software, p. 67-80

[17] Gustavo Rossi, Matias Urbieta, Jeronimo Ginzburg, Damiano Diatante, Alejandra Garrido: "Refactoring to Rich Internet Applications. A Model-Driven Approach", 2008 IEEE

[18] Michael Mahemoff: „Ajax Design Patterns." 2006, O'Reilly Media Verlag, Sebastopol, USA

[19] Jan Zahalka, Diplomarbeit unter Betreuung von Dr. Nora Koch, Robert Stabl: „Web Engineering für asynchrone Anwendungen", 2006, Ludwig-Maximilians-Universität (LMU), Deutschland

[20] Web Engineering Group: „Model-Driven Web Engineering. UWE Approach." 2008, Ludwig-Maximilians-Universität (LMU), Germany

[21] Martin Nußbaumer, Dissertaion: „Entwicklung und Evolution dienstorientierter Anwendungen im Web Engineering." 2007, Universität Karlsruhe (TH), Fakultät für Informatik, Deutschland

[22] Frédéric Fondement , Raul Silaghi: „Defining Model Driven Engineering Processes.“ 2006, Software Engineering Laboratory, Swiss Federal Institute of Technology in Lausanne, Switzerland

[23] Ulrich Wolffgang, Diplomarbeit unter der Betreuung von Dipl.-Wirt.Inform. Christoph Lembeck in der Zusammenarbeit mit „Viadee“ IT-Unternehmensberatung: „Modellgetriebene Entwicklung von Webapplikationen.“ 2008, Westfällische Wilhelms- Universität Münster, Institut für Wirtschaftsinformatik, Deutschland

[24] Christian Kroiß, Nora Koch: „The UWE Metamodel and Profile – User Guide and Reference. Version 1.0.“ 2008, Ludwig-Maximilians-Universität (LMU), Germany

[25] Andreas Kraus, Dissertation: “Model Driven Software Engineering for Web Applications.” 2007, Ludwig-Maximilians-Universität (LMU) München, Germany

[26] Christian Kroiß, Diplomarbeit unter der Betreuung von Dr. Nora Koch, Gefei Zhang: „Modellbasierte Generierung von Web-Anwendungen mit UWE (UML-based Web Engineering).“ 2008, Ludwig-Maximilians-Universität (LMU), Deutschland

[27] Design Pattern Library. http://developer.yahoo.com/ypatterns/, letzter Zugriff: 29.08.08

[28] Jenifer Tidwell: “Designing Interfaces: Patterns for Effective Interaction Design.” 2005, O’Reilly Media, Inc

[29] Was ist Ajax? http://www.admin-wissen.de/eigene-tutorials/webentwicklung/ajax-tutorial/einfuehrung-in-ajax/, letzter Zugriff: 15.08.08

[30] HTML Dateien selbst erstellen. http://de.selfhtml.org/index.htm, letzter Zugriff: 16.10.08

[31] Douglas Crockford: „The JavaScript Programming Language“ http://video.yahoo.com/watch/111593/1710507, letzter Zugriff: 30.08.08

[32] Smarty – die kompilierende PHP Template-Engine. http://www.smarty.net/manual/de/, letzter Zugriff: 28.08.08

[33] PHP Manual. http://de.php.net/manual/de/, letzter Zugriff: 20.09.08

[34] PHP Tutorial. http://tut.php-quake.net/de/communication.html, letzter Zugriff: 29.09.08

[35] Jochen Stärk: „Objektorientierung in PHP 5.“ http://rzdocs.uni-hohenheim.de/phptutorial/objektorientierung.html, letzter Zugriff 15.09.08

[36] JSON Message, SendReseve. http://www.seemysites.net/forum/viewtopic.php?t=5, letzter Zugriff 17.09.08

[37] JavaScript. http://www.homepage-total.de/javascript/, letzter Zugriff 20.09.08

[38] Jevon Wright, Jens Dietrich: “Survey of existing Languages to Model Interactive Web Applications.” 2008, Institute of Information Sciences and Technology, Massey University, Palmerston North, New Zeland

[39] Chao Wang, Fortgeschrittenenpraktikumsarbeit unter der Betreuung von Gefei Zhang: „Comparison and Evaluation of RIA User Interface Design Approaches.“ 2008, Ludwig-Maximilians-Universität (LMU), Germany

[40] Magic Draw CASE Tool. http://www.magicdraw.com/, letzter Zugriff 02.11.08

Anhang auf CD-ROM

1. Quellkode zur Webanwendung S.CORE (S_CORE.zip)
2. Die erweiterten Modelle vom S.CORE (level1_my.mdzip)
3. Das erweiterte UWE Metamodell (UWE Metamodell_new.mdzip)
4. Das erweiterte UWE Profil (UWE Profil.mdzip)