FAKULTÄT FÜR !NFORMATIK

Prototyp eines Web-Modelleditors auf Basis der Eclipse Technologien GMF und RAP: Konzept und Implementierung

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Magister der Sozial- und Wirtschaftswissenschaften

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Sebastian Bolnberger
Matrikelnummer 9925761

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Univ. Prof. Dr. Dimitris Karagiannis
Mitwirkung: Dr. Harald Kühn

Wien, 12.10.2008

(Unterschrift Verfasser) (Unterschrift Betreuer)

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Sebastian Bolnberger

Danksagung

Hiermit möchte ich mich herzlichst bei meinen Eltern bedanken, die mir finanziell während der Studienzeit immer zur Seite gestanden sind und mich jederzeit mit ihrer Fürsorge und ihrem Verständnis unterstützt haben. Vielen Dank auch an meine Freundin Karoline, die mich tagtäglich aufs Neue motiviert hat mein Ziel nicht aus den Augen zu verlieren.

Großer Dank gebührt auch meinem Diplomarbeitsbetreuer Dr. Harald Kühn, welcher mir während meiner Diplomarbeit immer zur Seite gestanden ist und für mich immer ein offenes Ohr hatte.

Zu guter Letzt möchte ich mich noch bei all jenen bedanken, die mir tagtäglich Rückhalt geboten und auch in komplizierten Zeiten immer an mich geglaubt haben.

Abstract

Die Idee vom automatisch generiertem *Source Code* (SC) in der Informatik ist wahrscheinlich genauso alt wie die Informatik selbst. Mit Hilfe des *Model Driven Software Development* (MDSD) Ansatzes ist es heute möglich durch das Erstellen von Modellen automatisch produzierten Programmcode zu gewinnen. Unter gewissen Vorraussetzungen kann mit dem MDSD eine höhere Entwicklungsgeschwindigkeit sowie eine bessere Handhabbarkeit der Software erzielt werden und aufgrund einer höheren Abstraktion, manifestiert durch die Modelle, kann die Softwarequalität im Allgemeinen gesteigert werden.

Das Eclipse *Graphical Modeling Framework* (GMF) nimmt sich diesem Ansatz an, indem zunächst Modelle erarbeitet werden, welche die Aufgabenstellung eindeutig beschreiben und als Input für die SC Generierung dienen. Als Output liefert das GMF automatisch realisierte Modellierungseditoren, mit Hilfe deren die gewünschte Domäne abgebildet und modelliert werden kann. Die Eclipse *Rich Ajax Platform* (RAP) ist ebenfalls ein generativer Ansatz, jedoch für die Entwicklung von Rich Internet Applications. Hierbei beschränkt sich die Entwicklung auf die Programmiersprache Java, von der ausgehend die für das Internet notwendigen Komponenten – HTML, JavaScript, *Asynchronous JavaScript and XML* (Ajax), CSS usw. – von dem RAP erstellt werden.

Die Diplomarbeit soll das GMF, das RAP und deren Voraussetzungen in einem theoretischen Teil vorstellen sowie analysieren. Ausgehend von den beiden Technologien drängt sich die Frage regelrecht auf, ob eine Kombination dieser Konzepte einen sowohl effizienteren als auch effektiveren Softwareentwicklungsprozess im Bereich von Web-Anwendungen ermöglichen kann. Der praktische Teil wird versuchen auf diese Frage eine Antwort zu finden, indem mit Hilfe dieser Ansätze ein webbasierter Modelleditor implementiert wird. In diesem Zusammenhang soll auch geklärt werden, welche gemeinsamen Schnittstellen das GMF / RAP aufweisen, um mit Minimalem manuellem Eingriff effizient arbeiten zu können. Nachdem es sich um junge Disziplinen handelt, besteht die Möglichkeit, dass solche Verknüpfungspunkte noch gar nicht konzipiert worden sind. An dieser Stelle sollen adäquate Lösungsvorschläge präsentiert werden, um zukünftig eine homoge Zusammenarbeit zu ermöglichen.

Inhaltsverzeichnis

1 Einleitung ... 1
1.1 Motivation ... 1
1.2 Aufgabenstellung ... 2
1.3 Begriffsdefinition: Web-Modelleditor ... 3
1.4 Vorgehensweise ... 5

2 Model Driven Software Development ... 8
2.1 Motivationen für den Einsatz von MDSD ... 9
2.2 Konzepte ... 12
2.2.1 Architektur von MDSD ... 12
2.2.2 Technologien zur Automatisierung ... 14
2.2.3 Qualitätsanforderungen an Modelle ... 15
2.3 Nutzen und Gewinn von MDSD ... 17

3 Überblick über die eingesetzten Technologien ... 20
3.1 Open Source – Chance oder Hindernis ... 20
3.1.1 Definitionen ... 21
3.1.2 Gründe für die Entwicklung von OS Code ... 22
3.1.3 Chance oder Hindernis ... 23
3.2 Die Eclipse Plattform ... 24
3.2.1 Anforderungen an eine moderne IDE ... 24
3.2.2 Geschichte und Ziele von Eclipse ... 25
3.2.3 Architektur ... 27
3.2.3.1 Aufbau des Plugin ... 28
3.2.3.2 Java Development Tools ... 29
3.2.3.3 Plugin Development Environment ... 29
3.3 Graphical Modeling Framework ... 30
3.3.1 Eclipse Modeling Framework ... 30
3.3.2 Graphical Editing Framework ... 31
3.3.3 Graphical Modeling Framework ... 32
3.4 Rich Ajax Platform ... 35
3.4.1 Ajax ... 36
3.4.1.1 XMLHttpRequest ... 36

3.4.1.2 Ajax vs. Klassische Webanwendungen 38
3.4.1.3 Vorteile und Nachteile 39
3.4.2 Rich Ajax Platform Architektur 42
3.4.2.1 Workbench 43
3.4.2.2 JFace 44
3.4.2.3 RAP Widget Toolkit 44
3.4.2.4 Qooxdoo 46
3.4.2.5 Equinox 46

4 Technologische Bewertung von GMF und RAP 48

4.1 Verbreitung 49
4.2 Lehre an Universitäten 50
4.3 Produktivität 51
4.4 Community 52
4.5 Performance 52
4.6 Skalierbarkeit 53
4.7 Betriebssystemunabhängigkeit vom Client 54
4.8 Betriebssystemunabhängigkeit vom Server 54
4.9 Standards 55
4.10 Lebenszyklus 55
4.11 Innovation 56
4.12 Kosten 56
4.13 Lernkurve 57
4.14 Multi-Threading 57
4.15 Offline-Fähigkeit 58
4.16 Abhängigkeit von Drittkomponenten 58
4.17 Integrierbarkeit 59
4.18 Usability 60
4.19 Entwicklungsunterstützung 60
4.20 Homogenität 61

5 Implementierung eines Web-Modelleditor Prototyps 62

5.1 Das Mindmap Modell 62
5.2 Zielsetzung 63
5.3 Konzeption und Implementierung mit Hilfe des GMF 65

5.3.1 Konzeption 65
5.3.2 Implementierung 67
5.3.2.1 Domain Model Definition 67
5.3.2.2 Graphical Definition 68
5.3.2.3 Tooling Definition 68
5.3.2.4 Mapping Definition 69
5.3.2.5 Source Code Generierung 69
5.3.2.6 Bewertung der Ergebnisse 70
5.4 Konzeption und Implementierung in der Rich Ajax Platform 70
5.4.1 Konzeption 71
5.4.1.1 Zielsetzung 71
5.4.1.2 UML Ansicht der Webapplikation 72
5.4.2 Implementierung 72
5.4.2.1 Grundlagen einer RAP Applikation 73
5.4.2.2 Implementierung des IEntryPoint Interface 75
5.4.2.3 Erreichen einer angemessenen Interaktivität 76
5.4.2.4 Bewertung der Ergebnisse 78
5.5 Analyse der Interoperabilität zwischen GMF und RAP 79
6 Zusammenfassung und Ausblick 82
6.1 Zusammenfassung 82
6.2 Ausblick 85
7 Literaturverzeichnis 87
8 Abbildungsverzeichnis 95
9 Tabellenverzeichnis 96
10 Abkürzungsverzeichnis 97
11 Fußnotenverzeichnis 98

1 Einleitung

Das erste Kapitel dieser Diplomarbeit hat sich folgender Aufgabe verschrieben: Zunächst soll die Motivation begründet werden, warum es sinnvoll ist sich mit den Themen dieser Arbeit auseinanderzusetzen. Anschließend soll die Aufgabenstellung konkretisiert werden, gefolgt von einer Abgrenzung der Begriffe Modell, Modelleditor und Web-Modelleditor im Kontext dieser Diplomarbeit. Abgeschlossen wird das erste Kapitel mit einer Übersicht, wie diese Diplomarbeit aufgebaut ist und mit welchen Maßnahmen das Ziel, den Prototyp eines Web-Modelleditors zu implementieren erreicht werden soll.

1.1 Motivation

Schon seit den Anfängen der Informatik, gibt es Ideen und Ansätze, die sich mit der automatischen Generierung von Programmcode beschäftigen. Man stützte sich dabei auf Modelle und implementierte Generatoren, die aus gegebenem Input einen entsprechenden Quellcode erzeugten. Die Vorteile von *Model Driven Software Development* (MDSD) können sich unter gewissen Vorraussetzungen in einer höheren Entwicklungsgeschwindigkeit und einer besseren Handhabbarkeit erklären lassen. Aufgrund einer höheren Abstraktion kann auch die Softwarequalität gesteigert werden.

Das Eclipse Projekt, welches sich von den Ursprüngen als erweiterbare *Open Source* (OS) Entwicklungsplattform präsentierte, startete mit dem Eclipse Modeling Project ebenfalls einen Ansatz für modellgetriebene Architekturen. Der Fokus liegt bei der Entwicklung und Förderung von modellbasierenden Technologien innerhalb der Eclipse Community. Unter diesem Dachprojekt werden verwandte Themen behandelt, beispielsweise das *Eclipse Modeling Framework* (EMF) und das *Graphical Modeling Framework* (GMF).

Mit Hilfe des GMF können grafische Editoren auf Basis von *Unified Modeling Language* (UML) Modellen oder *Extensible Markup Language* (XML) Spezifikationen erzeugt werden. Dieser Ansatz verbindet das EMF und das *Graphical Editing Framework* (GEF) und müssen nicht wie bisher händisch integriert werden. Das GMF strebt eine hohe Skalierbarkeit an und fördert den Einsatz von Domain Specific Languages.

In letzter Zeit sind auch interaktive dynamische Webinhalte immer populärer geworden, eine dabei oft eingesetzte Technologie ist *Asynchronous JavaScript and XML* (Ajax) die es ermöglicht, dass dynamisch einzelne Seiteninhalte in den Webbrowser des Client geladen werden können ohne die gesamte Seite neu vom Server anzufordern. Unter dem Projektnamen *Rich Ajax Platform* (RAP) hat sich Eclipse ein weiteres Ziel gesetzt, nämlich die Entwicklung von *Rich Internet Applications* (RIA), die Ajax User Interfaces implementieren, welche von dem Java *Standard Widget Toolkit* (SWT) abgeleitet sind und dem Gedanken des Web 2.0 entsprechen.

Das Web 2.0 ist mehr ein Schlagwort und bezeichnet keine spezielle Technik. Verschiedenste Techniken spielen hier zusammen und gestalten ein interaktives, kollaboratives sowie soziales Internet. Durch personalisierte dynamische Inhalte wandelt sich das Erscheinungsbild von Webseiten. Wegen der steigenden Interaktion der einzelnen Computer ist das Internet eine Schnittstelle zwischen den Servern, den Browsern und der Desktop-Anwendungen geworden. Die Verschmelzung der einzelnen Komponenten und die dadurch realisierte Transparenz vor dem Benutzer ist eines der Ziele vom Web 2.0.

Ausgehend von den beiden individuellen Techniken GMF und RAP drängt sich die Frage regelrecht auf, ob eine Kombination der beiden Konzepte einen sowohl effizienteren als auch effektiveren Softwareentwicklungsprozess im Bereich von Web-Anwendungen ermöglicht. Durch die Integration der beiden Konzepte in eine Applikation soll auch die Frage beantwortet werden, wie GMF und RAP kommunizieren, wo die Schnittstellen sind und wie sie im weitesten Sinne zusammenarbeiten können.

1.2 Aufgabenstellung

Im Rahmen der Diplomarbeit sollen grafische, interaktive Modellierungskonzepte und ein daraus abgeleiteter Prototyp eines Web-Modelleditors erarbeitet werden. Bei den Konzepten handelt es sich um Projekte aus der Eclipse Community, dem GMF und dem RAP. Der Prototyp als Gegenstück zu den Konzepten der Theorie soll einerseits einen Bezug zur Praxis herstellen sowie die Alltagstauglichkeit demonstrieren, andererseits einen Einblick in die Möglichkeiten der vorgestellten Konzepte bieten.

Das besondere Augenmerk soll in diesem Fall auf einer stufenweisen Erarbeitung der Konzepte gerichtet sein. Dies ist speziell notwendig, weil es sich bei diesen Technologien um relativ junge Projekte handelt, die nur durch Kenntnisse der zugrunde liegenden Ansätze effizient und im kompletten Umfang genutzt werden können.

Detaillierter betrachtet kann die Aufgabenstellung in zwei gleichwertige Teile zerlegt werden. Der erste und konzeptionelle Teil, der sich mit den grundlegenden Konzepten der Forschung, der Wissenschaft und der Informationstechnologie beschäftigt, soll nicht nur allein dem Verständnis der Thematik gewidmet sein. Zusätzlich soll er auch dazu genutzt werden, die neuen Technologien einem Qualitätstest im Sinne der Technologiereife zu unterziehen. Bei diesem Test sollen anhand von zielgerichteten Fragen die Lebensdauer, die Verbreitung, der Grad der Innovation und die Kosten beleuchtet werden.

Der mehr praktisch orientierte zweite Teil der Arbeit setzt sich mit den Konzepten selbst auseinander und führt anhand der Implementierung eines Prototyps vor, wozu die Technologien GMF und RAP fähig sind und wie man sie am effektivsten einsetzen kann. Die Architekturskizze des Prototyps ist in Abb. 1.1 aufgezeigt. Hier sieht man den Modelleditor, repräsentiert durch die Technologien HTML, dynamisches HTML, JavaScript und CSS in einen Browser eingebettet. Die Kommunikation mit dem Server erfolgt dabei über eine Ajax Engine. Um die geforderten Aufgaben erfüllen zu können muss der Server die Techniken GMF und RAP unterstützen und den generierten Code an den Browser zurücksenden.

Neben der Vorstellung und der Bewertung der beiden Technologiekonzepte und der praktischen Implementierung ist auch ein Ausblick in die Zukunft der Web-Modelleditoren Teil der Aufgabe.

1.3 Begriffsdefinition: Web-Modelleditor

Dieser Abschnitt soll dazu dienen einen Überblick über die Begriffe Modell, Modelleditor und Web-Modelleditor zu geben. Im Detail bedeutet dies eine Differenzierung dieser allgemeinen Begriffe um zu klären wie diese der Diplomarbeit zugeordnet werden können. Als erstes soll das Modell erläutert werden. Modelle werden in vielen unterschiedlichen Disziplinen, in einem zumeist unterschiedlichen Kontext, verwendet [Wiki o 2008]:

- Wissenschaftstheorie (Modelle zur umfassenderen Erkenntnis der Wirklichkeit)
- Sozialwissenschaften (Didaktische Modelle)
- Psychologie (Modelle des Menschen)
- Mathematik und Logik (Modelltheorie)
- Informatik und Wirtschaftsinformatik (Domänen-, Daten-, Referenz-, Entity Relationship Modelle, Aktivitätsdiagramme)
- Wirtschaftswissenschaft (Ökonomische Modelle)

Die Modelle, die im Rahmen dieser Diplomarbeit angesprochen werden sollen, sind konzeptionelle Modelle aus den Bereichen Informatik und Wirtschaftsinformatik. Im Speziellen zählen hierzu alle domänenspezifische-, Entity Relationship Modelle, Aktivitäts-, Klassen-, Use Case-, Mindmap Diagramme, ereignisgesteuerte Prozessketten, Zustandsautomaten usw. [Wiki o 2008], [Wiki p 2008]. Wird in weiterer Folge von Modellen gesprochen, soll auf die zuvor erstellte Definition verwiesen werden.

Nachdem eine Abgrenzung der Modelldefinition der unterschiedlichsten Disziplinen erfolgt ist, können die Charakteristika eines dazu geeigneten grafischen Editors beschrieben werden. Mit Hilfe dieses Editors können die angesprochenen Modelle auf einfachem Wege grafisch modelliert werden. Dabei können die zur Verfügung stehenden Objekte aus einer Tool Palette in den Editor gezogen werden, nach den individuellen Bedürfnissen angepasst und mit anderen Elementen verknüpft werden. Auf diese Art und Weise erhält man eine konkrete Ansicht des Modells in einem grafischen Umfeld, das in geeigneter Weise von den Editoren auch in eine maschinenlesbare Form gebracht werden kann. Modelleditoren im Rahmen der Diplomarbeit sind im Regelfall auf einem Client installiert und werden nur von diesem verwendet.

Ein Web-Modelleditor ist eine Erweiterung des zuvor beschriebenen Modelleditors, nicht in seiner Funktionalität sondern in seinem Verhalten. Ein Web-Modelleditor unterscheidet sich zunächst einmal von einem Modelleditor dadurch, dass dieser nicht auf einem Client installiert sein muss, sondern auf einem Server läuft und in einem Webbrowser den Benutzern zur Verfügung gestellt wird. Ein weiterer Unterschied betrifft die Anzahl der User, die gleichzeitig diese Software nutzen. Ist es bei einem Modelleditor lediglich der Client, auf dem er installiert ist, gibt es bei

einem Web-Modelleditor eine größere Anzahl an Clients, die zeitgleich bedient werden müssen. An dieser Stelle wäre es sinnvoll effiziente Parallelisierungsmechanismen zu implementieren. Ein weiterer Umstand, auf den geachtet werden sollte, ist der entstehende Datentransfer und die damit einhergehende Synchronisation der Clients mit dem Server. Vergleicht man diese Erkenntnis mit klassischen Web-Anwendungen, müsste nach jeder Modifikation im Web-Modelleditor die Webseite mit den entsprechenden Veränderungen neu geladen werden. Dabei entsteht ein unnötiger Datenoverhead, da auch Teile des Modells übertragen werden müssten, die überhaupt nicht verändert worden sind. An diesem Punkt ist es notwendig genaue Synchronisierungspunkte zu definieren, um die Netzwerklast zu minimieren und die Usability des Editors zu erhöhen. Eine weitere Schwierigkeit, mit der man konfrontiert ist, ergibt sich aus dem Umstand, dass die Verwendung von Grafiken und Bildern im Web eine andere ist als bei Desktop-Anwendungen. Mit anderen Worten besteht die Gefahr, dass durch das statische Verhalten von HTML eine improvisierte grafische Umsetzbarkeit des Web-Modelleditors zu bewerkstelligen ist. Das bedeutet, dass man sich mit JavaScript und dynamischen HTML weiterhelfen muss wenn man die Grenzen des reinen HTML erreicht hat.

1.4 Vorgehensweise

Der letzte Abschnitt dieses einleitenden Kapitels soll einen Überblick über den weiteren Verlauf der Diplomarbeit bieten, vor allem wie diese strukturiert ist, welche Inhalte in den unterschiedlichen Kapiteln vermittelt werden sollen und in welcher Abhängigkeit sie zur Aufgabenstellung stehen.

Das zweite Kapitel *„Model Driven Software Development“* beschäftigt sich in erster Instanz mit der Motivation, warum MDSD in der Softwareentwicklung hilfreich sein kann. Nach einem Überblick über die grundlegenden Konzepte dieser Herangehensweise wird der sich dadurch möglich gemachte Nutzen analysiert.

Das dritte Kapitel *„Überblick über die eingesetzten Technologien“* soll einen anschaulichen Einstieg in die für die Implementierung notwendigen Techniken bieten. Weil alle eingesetzten Technologien aus dem OS Umfeld stammen, wird zunächst auf diese Thematik eingegangen. Anschließend wird die Entwicklungsplattform Eclipse, auf dem die primären Technologien GMF und RAP

aufbauen, beschrieben. Darauf folgend soll das GMF näher beleuchtet werden, hierbei wird vor allem der Aufbau dieser Technologie und die Abhängigkeit zu den Ansätzen EMF und GEF illustriert. Im Abschnitt über das RAP wird auf den grundlegenden Kommunikationsmechanismus Ajax im Detail eingegangen. Abgeschlossen wird dieses Kapitel mit einer Beschreibung der einzelnen Komponenten, auf denen das RAP aufbaut.

Das vierte Kapitel *„Technologische Bewertung von GMF und RAP“* versucht die beiden Technologien anhand von unterschiedlichen Kriterien zu untersuchen. Diese Kriterien reichen von den entstehenden Kosten, über die Abhängigkeit von Drittkomponenten bis hin zur Betriebssystemunabhängigkeit von Client und Server. Mit Hilfe dieser Untersuchung kann eine Aussage über die aktuelle technologische Reife der einzelnen Technologie getroffen werden. Mit Unterstützung der gewonnenen Antworten wird versucht eine vage Abschätzung der zukünftigen Entwicklung von GMF und RAP zu liefern.

Das fünfte Kapitel *„Implementierung eines Web-Modelleditor Prototyps“* soll den implementierten Prototypen im Detail beschreiben. Ausgehend von der Wahl eines geeigneten Beispiels wird die Zielsetzung definiert und mit der Konzeption des GMF und des RAP begonnen. Als erster Schritt wird dabei das Beispielmodell in ein für das GMF lesbares Format konvertiert und auf Seite des RAP die Web-Anwendung designed.

Der nächste Schritt ist die Implementierung eines GMF Modelleditors, hierbei liegt der Fokus bei den dafür notwendigen Modellen und bei der Generierung des *Source Code* (SC). In der Implementierungsphase des RAP werden die wichtigsten Schritte erläutert, die eine RAP Anwendung von einer klassischen Web-Anwendung unterscheidet. Abgeschlossen wird dieses Kapitel mit einer Analyse der Interoperabilität zwischen dem GMF und dem RAP.

Das sechste Kapitel *„Zusammenfassung“* präsentiert die Erkenntnisse der Diplomarbeit in einer kompakten Form und untersucht, ob die zu Beginn definierte Aufgabenstellung erfüllt werden konnte. Abgeschlossen wird die Diplomarbeit mit einem gewagten Ausblick in die Zukunft von GMF und RAP, vor allem in welchem Maße sich diese Technologien weiterentwickeln müssen, um auch noch in einigen Jahren von Entwicklern genutzt zu werden.

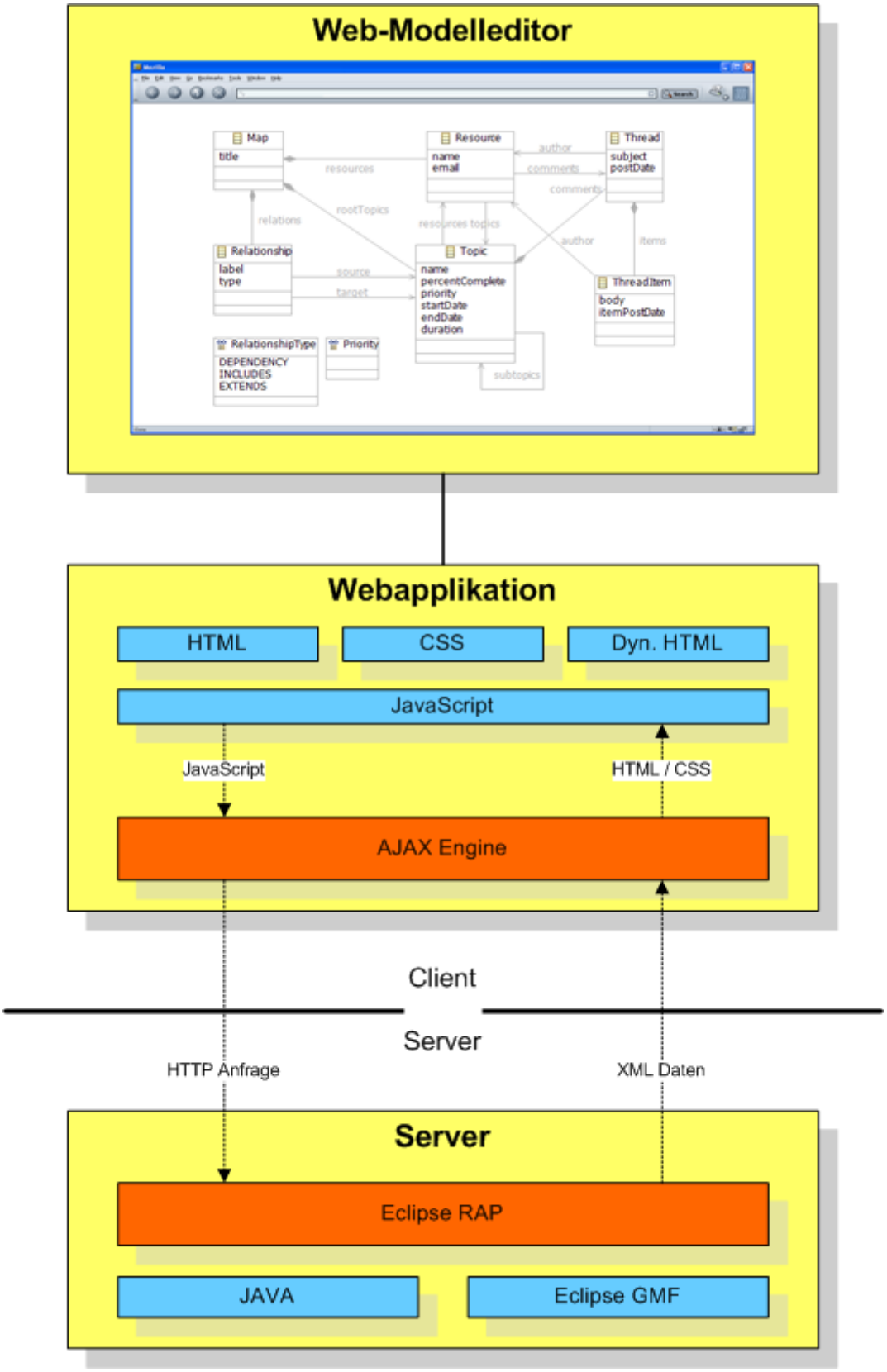

Abb. 1.1 – Architektur des Web-Modelleditor Prototyps

2 Model Driven Software Development

Dieses Kapitel über MDSD soll einen informativen Einstieg in ein spezielles Modellierungskonzept bieten, welches unter geeigneten Voraussetzungen auch zur SC Generation aus Modellen heran gezogen werden kann. Im Hinblick auf den Schwerpunkt der Arbeit, einen Modelleditor gestützt durch die Eclipse Technologien GMF und RAP in einem Browser laufen zu lassen, ist es notwendig die Grundlagen und die Visionen von MDSD zu illustrieren.

Um etwaigen Verwechslungen vorzubeugen soll an dieser Stelle der Unterschied zwischen MDSD und *Model Driven Architecture* (MDA) geklärt werden. Ein Synonym von MDSD ist auch Model Driven Development, wobei komplexe sowie umfangreiche Modelle zur Unterstützung für die Softwareentwicklung erstellt werden. Ausgehend von diesen Modellen kann der SC automatisch von Generatoren oder von Programmierern händisch erstellt werden [LeWr 2005].

Im Gegensatz dazu ist die MDA eine praktische Umsetzung der MDSD Konzepte durch eine Menge an Entwicklungstools, entwickelt von der *Object Management Group*[1] (OMG) – ein Konsortium aus Industrie und Forschung. Aufgebaut sind die Tools und die Konzepte durch existierende OMG Standards, so ist beispielsweise die Modellierungssprache UML[2] [LeWr 2005]. Mit anderen Worten beschreibt die MDSD eine Herangehensweise bei der Entwicklung von Softwareprojekten, wobei die MDA diese Konzepte praktisch umsetzt und somit in der Entwicklungspraxis einsetzbar macht.

Nach der notwendigen Abgrenzung der Begriffe kann mit der Erklärung der MDSD Konzepte begonnen werden. Dieses Kapitel untergliedert sich in den Bereich, der sich mit der Motivation für den Einsatz von MDSD Technologien auseinandersetzt, was in weiterer Folge auch auf für den Einsatz eines Web-Modelleditors ausschlaggebend sein kann. Anschließend werden die Konzepte näher betrachtet, wobei das Hauptaugenmerk bei technologischen Grundsätzen und qualitativen Anforderungen liegt. Abgeschlossen wird dieser Abschnitt mit einer Analyse des Nutzens, der durch MDSD erzielt werden kann.

---

[1] http://www.omg.org

[2] http://www.uml.org

2.1 Motivationen für den Einsatz von MDSD

Dieser Abschnitt dient vor allem dazu, herauszustreichen warum es von großer Bedeutung ist Modellierungskonzepte in die Softwareentwicklung zu integrieren. Dazu soll als erster Schritt erklärt werden, warum die Modellierung sinnvoll ist, und anschließend wird anhand von Beispielen aus dem Umfeld der Softwareentwicklung gezeigt, dass der Einsatz von Modellierungskonzepten hilfreich ist.

Um komplexe Systeme im Detail zu beschreiben werden in vielen traditionellen Entwicklungsbereichen der verschiedensten Branchen Modelle zur Unterstützung herangezogen. Beispielsweise wären Konstrukteure von Hochhäusern und Brücken ohne die Nutzung von Modellen großen Risiken ausgesetzt [Se 2003]. Sie setzen Modelle ein, um die Dimension, die Statik und die Umsetzbarkeit abschätzen zu können. Manchmal sind die Modelle auch weniger greifbar, wie Finanzmodelle oder elektrische Schaltpläne. Jedoch haben alle Modelle ein gemeinsames Ziel, sie dienen zur Abstraktion der Realität [CeNa 2004]. Modelle im Allgemeinen helfen uns bei dem Verständnis von komplexen Zusammenhängen und bieten qualitative Lösungsansätze durch ihre Fähigkeit zur Abstraktion der Aufgabenstellung auf seine Wurzeln.

In einem Artikel von IBM [CeNa 2004] ist eine Checkliste zu finden um herauszufinden, ob es bei dem gegenwärtigem Problem sinnvoll ist ein Modell dafür zu konstruieren. Die Fragestellungen sind:

- Ist der Problembereich gut bekannt?
- Wie einfach ist es die Lösung zu erreichen?
- Wie viele Personen müssen bei der Lösungserstellung zusammenarbeiten?
- Wie viel Wartungsaufwand wird für die Phase nach dem Abschluss prognostiziert?
- Welchen zeitlichen Gültigkeitsbereich wird das Endprodukt haben?

Werden diese Fragen projektbezogen ehrlich beantwortet wird deutlich, dass das fehlende Einbringen von Modellen in der Konzeptionisierungsphase weder technologisch schlau noch wirtschaftlich praktikabel ist. Man bringt sich somit um die Möglichkeit ein abstraktes Abbild der Realität zu schaffen, mit dem man Risiken frühzeitig erkennen und rechtzeitig minimieren bzw. sogar vermeiden kann [CeNa

2004]. In der Softwareentwicklung ist das Schaffen einer realen Abbildung nicht immer leicht, da diese womöglich noch gar nicht existiert, dennoch dienen Modelle dazu eine gewünschte Abbildung zu modellieren.

Es scheint, dass solche Ansätze auch für die Softwareentwicklung hilfreich sind, da es sich bei dieser Disziplin oft um höchst komplexe Abläufe handelt. Durch die Evolution bei der Softwareentwicklung ist jedoch zu erkennen, dass Modelle zumeist nur eine untergeordnete Rolle gespielt haben. Die Softwareentwicklung ist in einer offensichtlichen bedauernswerten Position. Offensichtlich bedauernswert deswegen, weil die Erwartungen an diese junge, unreife Disziplin extrem hoch sind. Die Anwendungen sind einer hohen Komplexität unterworfen, sollen reibungslos funktionieren und einen hohen Grad an Transparenz gegenüber dem Benutzer erzielen [Se 2003]. Um diese Faktoren gemeinsam Umzusetzen werden oft scheinbar unwesentliche Dinge, wie die Modellierung, vernachlässigt.

Generell hat sich aber seit der Einführung der dritten Generation von Programmiersprachen[3] Ende der 1950er Jahre kaum etwas in der Softwareentwicklung geändert. Selbstverständlich sind neue Paradigmen vorgestellt worden, wie die strukturierte und objektorientierte Programmierung, dennoch besitzt ein `if` Statement in Java dieselbe Funktionalität wie damals in Fortran. Kritisiert wird in dem Artikel von Bran Selic [Se 2003] auch der Umstand, dass Unternehmen teilweise den Fortschritt der Programmierparadigmen nicht mit Nachdruck vorantreiben. Dies wird durch tausende Zeilen händisch geschriebenen Code ausgedrückt, der zumeist noch zusätzlich manuell gewartet werden muss. Dieser klassische Weg der Softwareentwicklung kann nur durch geeignete Maßnahmen in der Ausbildung aufgebrochen und damit in eine neue Richtung geändert werden.

[3] Für den interessierten Leser können die Generationen der Programmiersprachen (GL) unter den nachfolgenden Links genauer Studieren:

- 1st GL - http://en.wikipedia.org/wiki/First-generation_programming_language
- 2nd GL; http://en.wikipedia.org/wiki/Second-generation_programming_language
- 3rd GL; http://en.wikipedia.org/wiki/Third-generation_programming_language
- 4th GL; http://en.wikipedia.org/wiki/Fourth-generation_programming_language
- 5th GL; http://en.wikipedia.org/wiki/Fifth-generation_programming_language

> *„Many practitioners have given up all hope that significant progress will result from fundamental advances in programming technologies“, Bran Selic in [Se 2003].*

Die Hoffnungen werden in die Verbesserungen von Prozessen gesetzt, wie auch schon durch den Hype von dem *Extreme Programming* (XP) und des *Rational Unified Process* (RUP) ausgedrückt wird. Das XP beschreibt ebenso wie das MDSD eine Herangehensweise bei der Entwicklung von Software, geht aber davon aus, das zu Beginn des Projekts noch nicht alle Problemstellungen eindeutig spezifiziert sind, sondern sich oft durch den Entwicklungsfortschritt ergeben bzw. vom Kunden eingebracht werden [Wiki b 2008]. Der RUP hingegen ist ein objektorientierter Ansatz, wobei der Fokus auf die Bereiche der Geschäftsprozessmodellierung, der Anforderungsanalyse, des Design, der Implementierung und des Tests gerichtet ist [Wiki c 2008].

Nach allem was wir in den letzten Jahren über die Softwareentwicklung gelernt haben, beruht sie doch zumeist auf dem Ausdruck von Ideen und ist, wenn überhaupt, durch unsere mangelnde Vorstellungskraft limitiert anstelle von technischen Barrieren. Die sich daraus ergebenden Vorteile sind die grundlegenden Ideen in den Konzepten des MDSD und ein weiterer möglicher Schritt in Richtung einer wartbareren Programmierung.

Neben den Programmiersprachen haben sich auch andere Konzepte wie beispielsweise die Modellierungstechniken stetig weiter entwickelt. Mit diesen ist es uns heute möglich Modelle zu bauen, aus denen in weiteren Schritten kompilierbarer SC generiert werden kann und das auf eine plattformunabhängige Art und Weise. Durch das Einbringen von Modellen in die Softwareentwicklung erhält man eine weitaus abstraktere Sichtweise der Aufgabenstellung, dadurch wird es möglich andere Interessensgruppen in den Designprozess einzugliedern. Durch die abstrakten Modelle und deren Transformation in technologisch spezifische können semantische Diskrepanzen zwischen entwickelten Software Bausteinen vermieden werden. Ein weiterer großer Vorteil zeigt sich durch den Anstieg der Unabhängigkeit und der Resistenz gegen Änderungen im Modell. Durch die Transformatoren und Generatoren werden die Anpassungen automatisch in den SC übernommen [FeMaCa 2007].

Jedoch genau das sind die Vorraussetzungen, die Unternehmen an die Softwareentwicklung stellen. Heterogene Systeme müssen reibungslos miteinander kommunizieren können, die Anwendungen sollen sich ohne größeren Entwicklungsaufwand an die rasch wechselnden Marktbedingungen anpassen können und das, soweit es geht, das Tagesgeschäft unberührt lassen sowie die entstehenden Kosten so niedrig wie möglich halten. Software aus heutiger Sicht ist weitaus mehr, als es noch vor ein paar Jahren war, wir stellen weitaus höhere Ansprüche an die Entwicklung und deren Nutzung. In einem heterogenen Umfeld soll sie reibungslos, robust und flexibel arbeiten. Grundlegende Änderungen in Inputs, Abläufen und Outputs aufgrund von Marktbewegungen sollen schnell und ohne größere Kosten umgesetzt werden können. Die MDSD Ansätze haben das Potential um diesen notwendigen modernen Anforderungen an Software und deren Entwicklung zu genügen.

2.2 Konzepte

Als erster Schritt soll in diesem Abschnitt das grundlegende Verständnis hergestellt werden, wie MDSD funktioniert indem das Architekturkonzept erläutert wird. Anschließend werden die technologischen Grundlagen für den reibungslosen Einsatz von Modellierungstechniken und deren qualitativen Anforderungen vorgestellt.

2.2.1 Architektur von MDSD

Der Begriff der modellgetriebenen Software Entwicklung beschreibt ein Vorgehensmodell für die Erstellung von Software bei großen Projekten. Mit Hilfe der MDSD kann die Entwicklung von Software durchgängig mit Modellen beschrieben werden. Grundsätzlich bilden die erstellten Modelle eine aufeinander aufbauende Abstraktionshierarchie, wobei die höchste Abstraktionsstufe in den grundlegenden fachlichen Modellen erreicht wird. Diese Modelle beschreiben die Aufgabenstellung in einer plattformunabhängigen Sichtweise. Eine Abstraktionsstufe niedriger befindet sich das technische Modell, welches das zuvor beschriebene fachliche Modell mittels der für diese Technik spezifischen Möglichkeiten – Objekte, Klassen, Funktionen – abbildet. In der untersten Ebene befindet sich der auf Grund des technischen Modells generierte SC [Wiki a 2008], was in der Abb. 2.1 illustriert ist.

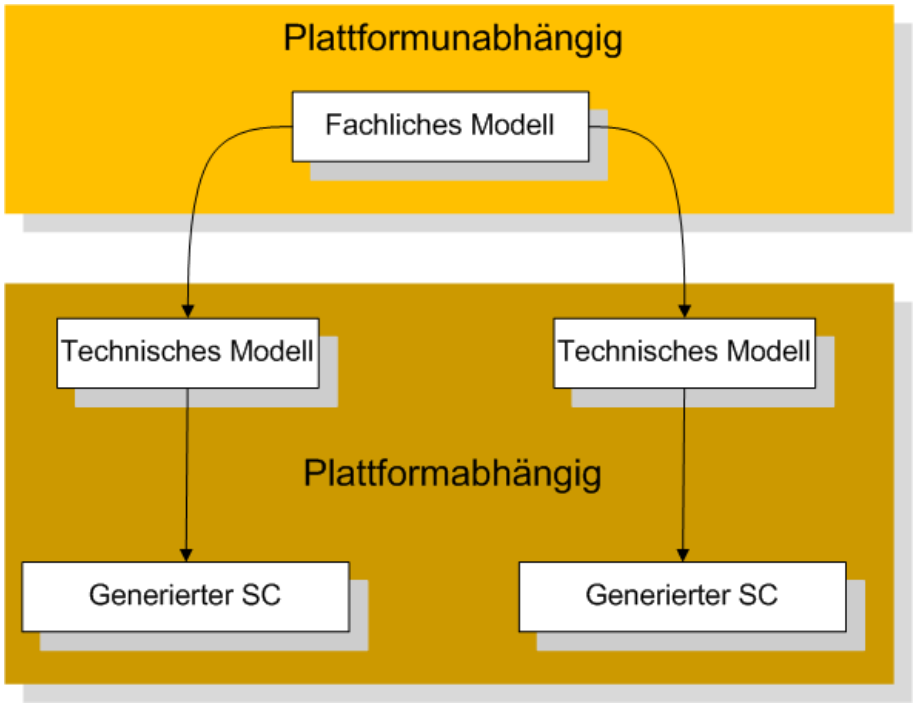

Abb. 2.1 – Abstraktionsebenen bei MDSD

Im Allgemeinen kann das fachliche Modell als Ausgangspunkt für den gesamten Softwareentwicklungsprozess gesehen werden. Ergeben sich Änderungen, werden diese nicht wie in der klassischen Softwareentwicklung in den SC eingebracht, sondern in dem Modell hinzugefügt. Durch automatische Transformationsalgorithmen werden sowohl das technische Modell als auch der SC synchron gehalten. Weil Modelle auch für nicht Entwickler verständlich sind, können alle Interessensgruppen ihre Ideen und Wünsche in den Softwareentwicklungsprozess einbringen [Wiki a 2008].

Wenn man die sich daraus ergebenden Vorteil zusammenfasst, erkennt man, dass durch die Trennung in fachliche, plattformunabhängige und technisch spezifische Modelle eine Abstraktion in Richtung der realen Welt erreicht werden kann. Das fachliche Modell kann von verschiedenen Programmiersprachen parallel umgesetzt werden, ohne grundlegend höheren Entwicklungsaufwand zu verursachen. Dadurch kann die Entwicklungsgeschwindigkeit erhöht, die manuelle Codierung durch die jeweiligen Entwickler reduziert und mitunter auch Kosten vermieden oder gesenkt werden [Wiki a 2008]. Ein weiterer Vorteil bei MDSD ist, dass der Aufwand für Änderungen und Wartung in der Software an einer zentralen Stelle, nämlich dem Modell, gemacht werden können. Dies hat zur Folge, dass die Änderungen auch in dem generierten SC wirksam werden.

2.2.2 Technologien zur Automatisierung

Sowohl die Konzepte der Modellierung als auch die Codegenerierung sind nicht neu, konnten aber in den frühen Jahren noch mit wenig erfolgreichen Meldungen Schlagzeilen machen. Der Grund dafür waren noch zuwenig ausgereifte Technologien, wie qualitative Automatisierungsmechanismen und gereifte Standards für die Modellierung und zum anderen hat sich unser Erfahrungsschatz im Bereich der Modellerstellung enorm entwickelt [Se 2003].

Die beste Möglichkeit die Produktivität und Zuverlässigkeit von Programmen nach oben zu katapultieren, ist es Automatisierungsmechanismen für die Erstellung von Programmen zu verwenden. Zu Beginn der Entwicklungen spielte die Automatisierung jedoch nur eine untergeordnete Rolle und diente vor allem unterstützend für die Erstellung der Grundstruktur von Programmen. Diese Maßnahme hatte aber noch nicht die gewünschte Auswirkung auf die Produktivität, deswegen hat sich das so genannte Round Trip Engineering[4] durchgesetzt [Se 2003]. Um aber das volle Potential von MDSD auszuschöpfen, werden an die Automatisierung zwei Vorraussetzungen gestellt, zum einen die komplette Generierung von Programmen, zum anderen die anschließende Verifizierung in einer Testumgebung. Unter der Verifizierung des generierten Programms versteht man die Analyse des Modells auf die Präsenz von erwünschten und die Absenz von unerwünschten Eigenschaften. Hierbei sind sowohl mathematische Ansätze, wie Performance Messungen denkbar, als auch die Simulation des Programms, um eine empirische Verifikation zu ermöglichen [Se 2003].

Ein weiterer Faktor der die Automatisierung unterstützt sind weltweit akzeptierte Standards. Standardisierungen liefern signifikante Impulse für Forschung und Entwicklung, weil damit wiederholte Tätigkeiten beschrieben werden können. Zusätzlich wird die Wiederverwendung durch Standards erstrebenswert und die Zusammenarbeit zwischen Tools stark vereinfacht. Für den Bereich der MDSD sind im letzten Jahrzehnt einige Standards entstanden, ein wesentlicher Bestandteil sind mit Sicherheit UML und die verschiedenen Web-Standards, wie beispielsweise XML, Simple Object Access Protocol und Ajax [Se 2003].

[4] http://de.wikipedia.org/wiki/Round_Trip_Engineering;

2.2.3 Qualitätsanforderungen an Modelle

Die Qualität von Modellen ist ein wichtiger Punkt, denn nur aufgrund von hochwertigen Modellen ist es möglich die komplexen Aufgabenstellungen und deren vielfältigen Lösungen zu veranschaulichen, ohne eine komplette Umsetzung zu implementieren. Um die Qualität von Modellen besser charakterisieren zu können, sollten Designmodelle folgenden fünf Charakteristiken genügen [Se 2003]:

- Die Abstraktion ist eine der grundlegenden Eigenschaften von Modellen, sie ist eine reduzierte Repräsentation der tatsächlichen Aufgabe. Das gute Verständnis des Problems wird durch das Verbergen von detaillierter Information, die in dieser Phase nicht nötig ist, erreicht.
- Die Lesbarkeit ist eine weitere Eigenschaft um die Verständlichkeit von Modellen zu erhöhen. Die Lesbarkeit ist direkt abhängig von der verwendeten Modellierungssprache, wobei gute Modelle Verknüpfungen verwenden um intuitive Konzepte zu veranschaulichen, damit aber die Verständlichkeit um ein hohes Maß vergrößern.
- Die Korrektheit ist dann gegeben, wenn das reale Problem mit dem gesamten Feature Katalog in dem Modell dargestellt wird.
- Die Trenderkennung in einem Modell bedeutet, dass sie vorausschauend auf Probleme hinweist und im besten Fall diese gar nicht erst auftreten lässt. Dies kann durch mathematische formale Analysen aber auch durch Simulationen erreicht werden.
- Die Wirtschaftlichkeit im Zusammenhang mit Modellen verlangt, dass die Erstellung und die Analyse der Modelle signifikant günstiger sein müssen als das tatsächlich entwickelte Produkt.

Die zuvor vorgestellten Maßnahmen wirken unterstützend beim Modellierungsprozess und helfen die Verständlichkeit von einem Modell zu erhöhen. Zusätzlich gibt es aber weitere Qualitätsanforderungen die auf eine reibungslose und übergreifende Zusammenarbeit der Modellierungskonzepte abzielen [Se 2003], dies wird in der Abb. 2.2 veranschaulicht sind.

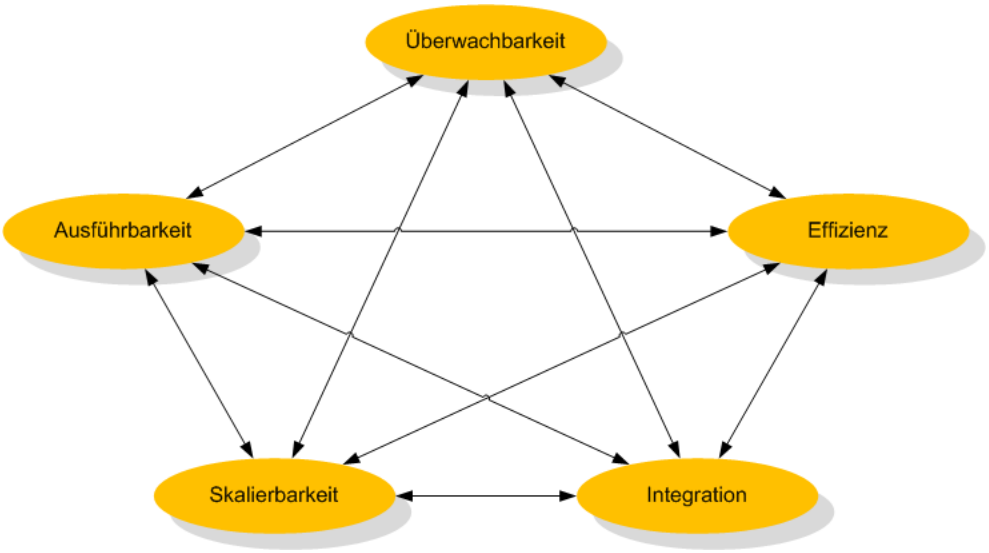

Abb. 2.2 – Pragmatiken bei MDSD

- Die Überwachbarkeit des Modells wird durch das Fehlerreporting, ein integrierter Bestandteil von Codegeneratoren, auf Modellebene erreicht. Diese Notwendigkeit ist dadurch gegeben, dass die Codegeneratoren idiosynkratisch sind, deswegen ist es oft nicht einfach in dem generierten Programm das ursprüngliche Modell zu erkennen. Die semantische Diskrepanz zwischen dem implementierten Code und den hoch abstrakten Modellierungssprachen macht es umso notwendiger dieses Hilfsmittel zur Verfügung zu stellen, um die Fehlerquellen zu lokalisieren.
- Die Ausführbarkeit des Modells soll schon in der Anfangsphase gewährleistet sein, dadurch kann schon frühzeitig eine direkte Erfahrung mit dem konstruierten System gemacht werden, was Fehler rechtzeitig aufzeigen kann.
- Die Effizienz des generierten SC ist ein wichtiges Merkmal von Programmen und kann in zwei Bereiche aufgeteilt werden, die Performance und die Speicherauslastung. Heutige Generatoren können Programme erzeugen, die in beiden Bereichen eine Effizienzsteigerung von 5% - 15%, gegenüber manuell entwickeltem Code, erreichen.
- Die Skalierbarkeit bezieht sich auf die Zeit der Generierung und Kompilierung des Modells. Mitunter ist es bei großen Modellen nicht immer sinnvoll das gesamte Projekt neu zu erstellen. Deshalb ist es für Codegeneratoren notwendig geeignete Mechanismen zu haben, die die

Auswirkungen von Veränderungen im Entwicklungsbereich analysieren und eine Codegenerierung minimieren bzw. optimieren.

- Die Integration in bestehende Entwicklungsumgebungen der eingesetzten Tools und Mechanismen von MDSD sollte mehr oder weniger nahtlos möglich sein. Mit anderen Worten sollen Vorteile von anderen Technologien ausgenutzt und integriert werden können, seien es bestehende Codebibliotheken oder andere Software Bausteine.

Der Erfolg von MDSD hängt von mehreren Faktoren ab, zum einen von der effizienten Lösung offensichtlicher technischer Belange, wie der Definition passender Modellierungssprachen und automatischen Codegeneratoren. Auf der anderen Seite dürfen die soeben geschilderten Qualitätsanforderungen nicht außer Acht gelassen werden, denn diese sind auch maßgeblich am Erfolg beteiligt.

2.3 Nutzen und Gewinn von MDSD

Dieses Unterkapitel hat die Aufgabe der Analyse des Nutzens, der durch den Einsatz von MDSD Konzepte gewonnen werden kann. In der Tab. 2.1 sind die in den kommenden Absätzen beschriebenen Nutzengewinne, die sich durch den Einsatz von MDSD ergeben können kompakt dargestellt.

Ganz allgemein gesehen ist wahrscheinlich der größte Nutzen von Modellen, dass sie einen effektiven Weg für das Verständnis einer komplexen Aufgabenstellung zur Verfügung stellen. Anders dargestellt managen Modelle die Komplexität, in dem sie die Realität der Problemstellung abstrahieren. Damit ist es Modellen möglich zur Problemstellung assoziierte Risiken aufzeigen, welche schon auf Designebene behandelt und möglicherweise eliminiert werden können [CeNa 2004].

Weil Modelle iteratives Arbeiten unterstützen, können systematische Effekte die aufgrund von Änderungen in einem Bereich auftreten auch in anderen Bereichen des Modells sichtbar werden. Normalerweise müssen sich neue Softwareprodukte in bestehende Strukturen integrieren, dafür sind geeignete Schnittstellen notwendig. Diese können einfach in die Modelle aufgenommen werden und visualisieren in einer übersichtlichen Form wie sich das neue Produkt nahtlos in eine bestehende Architektur einfügen kann. Schwachstellen im Bezug auf die

Schnittstellen können somit leicht erkannt und über Lösungen auf einer abstrakten Ebene nachgedacht werden [CeNa 2004].

Durch Modelle allgemein	· Effektiver Weg für das Verständnis komplexer Aufgabenstellungen · Machen das Design verständlich · Zeigen assoziierte Risiken auf · Kommunizieren des Designs ohne große finanzielle Mittel · Hilft der ursprünglichen Aufgabenstellung treu zu bleiben · Unterstützt iteratives Arbeiten · Visualisiert den SC und Formen der Implementierung · Systematische Effekte von Änderungen werden in den anderen Bereichen der Modelle sichtbar · Schnittstellen für die Integration in bestehende Architekturen können in die Modellierung aufgenommen werden · Schwachstellen können erkannt und über Lösungen nachgedacht werden · Durch die abstrakte, sprachunabhängige, leicht verständliche Sichtweise können Experten aus anderen Bereichen des Unternehmens in den Designprozess eingebunden werden
Gegenüber dem Auftraggeber	· Aufgabenstellung wird verstanden und kann umgesetzt werden · Auftraggeber kann in kleine Bereiche des Designs eingebunden werden
Durch Codegenerierung	· Technologisch ungebundene Sichtweise der Problemstellung · Weniger anfällig auf evolutionäre Änderungen im technologischem Umfeld · Durch Transformatoren können zeitgleich für mehrere unterschiedliche technologische Plattformen entwickelt werden

	· Minderung des Entwicklungsaufwandes · Vermeiden von semantischen Diskrepanzen von Softwareelementen · Senkung von Kosten · Wartung zentral im Modell anstatt an unterschiedlichen Stellen im Code · Änderungen im Modell werden automatisch durch die Transformatoren / Generatoren in den SC übernommen · Modelle und SC werden synchron gehalten · Effizienz des generierten Codes ist zumindest gleichwertig zu handgeschriebenen · Effizienzsteigerungen bei Performance / CPU Auslastung von 5% - 15%

Tab. 2.1 – Nutzengewinne durch MDSD Konzepte

Modelle im Sinne der MDSD sind eine technologisch ungebundene Sichtweise der Problemstellung. Das Modell und im Endeffekt das Produkt sind weniger anfällig auf evolutionäre Änderungen, die sich aus dem technologischen Umfeld ergeben. Eine Prämisse der MDSD ist es auch das Modell durch Transformatoren in plattformspezifische Sichtweisen zu überführen. Damit ist aber mit wenig mehr Entwicklungsaufwand möglich ein Produkt für mehrere Plattformen zu produzieren und semantische Diskrepanzen von Softwareelementen zu vermeiden. Dadurch kann aber auch die Wartung des Produktes an einer zentralen Stelle durchgeführt werden, nämlich in dem abstrakten plattformunabhängigen Grundmodell. Das kann nicht nur Kosten senken, sondern auch den Programmieraufwand auf den einzelnen Zielplattformen reduzieren. Letzteres ergibt sich aus dem Umstand, dass die Änderungen im Modell durch die Transformation automatisch in den SC übernommen werden [CeNa 2004].

3 Überblick über die eingesetzten Technologien

Dieses Kapitel widmet sich im Detail jenen Technologien, die bei der Erstellung eines Web-Modelleditors zwingend notwendig sind. Vorausgesetzt und nicht im Detail erläutert werden die Technologien, die zur Umsetzung von Web-Anwendungen benötigt werden, wie beispielsweise die Webkomponenten HTML, JavaScript, CSS, usw. und Java.

Als erster Schritt beschäftigt sich dieser Abschnitt mit dem Thema OS im Allgemeinen. Die notwendigen Definitionen werden skizziert und ein grober Überblick wird darüber geboten, warum viele Entwickler OS Methoden unterstützen und weiterentwickeln. OS ist zwar keine Technologie im eigentlichen Sinne, soll aber trotzdem dieses Kapitel einleiten, zum einen weil die verwendeten Technologien dieser Diplomarbeit zur Gänze auf OS Komponenten aufbauen. Zum anderen weil OS Software in den letzten Jahren eine tragende Rolle in der Softwareentwicklung spielt und es zu einem regelrechten Konkurrenzkampf zwischen freien und lizenzierten Produkte gekommen ist.

Im zweiten Schritt wird die eigentliche Entwicklungsplattform, nämlich die Eclipse Plattform, genauer vorgestellt. Speziell wird auf ihre grundlegende Architektur und das Plugin Konzept eingegangen. Im Anschluss werden die für einen Web-Modelleditor erforderlichen Eclipse Bausteine GMF und RAP erläutert. Das Kapitel zu GMF bietet einen technologischen Überblick und erklärt die Funktionsweise dieses grafischen Modellierungsansatzes. Das RAP als Komponente zur Entwicklung interaktiven Web-Anwendungen führt in die Konzepte von Ajax ein, beschreibt die grundsätzliche Architektur von RAP und erläutert die notwendigen Grundbausteine.

3.1 Open Source – Chance oder Hindernis

Die Softwareentwicklung als OS bedient sich dem Mechanismus eines verteilten und transparenten Prüfverfahrens durch alle gleichberechtigten Mitglieder der OS Community. Mit anderen Worten handelt es sich hierbei um einen Ansatz, bei dem man den produzierten SC öffentlich zugänglich macht, damit dieser verwendet und auf seine Korrektheit hin überprüft werden kann. Mithilfe dieser Methode können

sich eine bessere Qualität, höhere Zuverlässigkeit, größere Flexibilität und geringere Kosten der entwickelten Programme erzielen lassen.

Aus der OS Bewegung hat sich eine Non-Profit Organisation, die *Open Source Initiative*[5] (OSI), gebildet, die den Nutzen und die Definitionen von OS kommuniziert, sowie als Mediator zwischen den unterschiedlichen OS Communities agiert. Neben der Definition und Wahrung der Standards ist die OSI auch die zentrale Anlaufstelle für die verschiedensten OS Lizenzen. In den folgenden Unterkapiteln soll zunächst auf die Definitionen von OSI eingegangen werden. Diesem Abschnitt folgt ein Ausflug in die Psyche der Programmierer und erläutert, warum Menschen freiwillig und oft ohne Bezahlung an OS Projekten teilnehmen. In einer abschließenden Zusammenfassung werden die Vor- und Nachteile von der OS Entwicklungsmethode gegenübergestellt.

3.1.1 Definitionen

Dieses Kapitel dient zur genauen Untersuchung der OS Definitionen der OSI und soll helfen ein besseres Verständnis, betreffend der OS Thematik zu liefern. Allgemein kann gesagt werden, dass freier SC weitaus mehr bedeutet als nur den Programmcode offen zu legen. Die herausgegebenen Definitionen[6], sind im Anschluss weniger juristisch interpretiert.

Eine Lizenz von freiem SC soll keinerlei Honorar in jeglicher Form beanspruchen dürfen. Mit anderen Worten darf die Lizenz niemandem verbieten Softwareelemente, die in kommerzielle Produkte eingebaut wurden, weiterzugeben oder zu verkaufen. Zusätzlich ist sowohl der SC vom erstellten Programm als auch eine kompilierte Form zum Download zur Verfügung zu stellen. Weiters darf es keine Richtlinien geben, die eine Weiterentwicklung des SC verbieten. Eher das Gegenteil soll der Fall sein, OS Lizenzen sollen eine Modifikation fördern. Jedoch ist es zulässig zu verlangen, dass neu entstandene Programme unter einem anderen Namen oder einer höheren Versionsnummer verfügbar sein sollen. Ein wesentlicher Punkt ist es auch, dass es möglich sein soll Produkte auf den Markt

[5] http://opesource.org

[6] Der genaue Wortlaut kann unter http://www.opensource.org/docs/osd nachgelesen werden.

zu bringen, die sowohl freien als auch gesperrten SC enthalten. Dabei müssen die entstehenden Produkte nicht zwingend auch OS sein. Darüber hinaus darf die OS Komponente, die in kommerzielle Produkte eingebaut ist auch von dritten weiterhin frei verwendet werden. Präziser ausgedrückt darf eine Lizenz niemanden daran hindern die Software einzusetzen, weder für kommerzielle Zwecke noch in Bereichen der Forschung. Schlussendlich soll eine Lizenz allen Technologien zur Verfügung stehen und darf keine Aussagen über eine spezifische Umsetzung von Schnittstellen treffen [OSD 2008].

3.1.2 Gründe für die Entwicklung von OS Code

Dieser Abschnitt befasst sich mit dem Phänomen, warum so viele unabhängige Programmierer immer wieder neue Entwicklungen für die OS Community liefern. Dazu soll dieses Kapitel einen kleinen Ausflug in die Psychologie machen, um besser verstehen zu können, wodurch Individuen motiviert werden freie Software zu entwickeln. A. Hars und S. Ou analysierten in ihrem Artikel [HaOu 2001] das sowohl interne als auch externe psychologische Faktoren demnach der Grund sein können bei OS Projekten teilzunehmen. Diese reichen von der intrinsischen Motivation[7], hohem Ansehen in der Gemeinde, bis hin zum persönlichen Drang und tatsächlichen monetären Gründen [HaOu 2001].

Das Konstrukt der intrinsischen Motivation kommt aus der Motivationspsychologie und besagt, das ein bestimmtes individuelles Verhalten in der Person selbst liegt. Anders erklärt ist diese Art der Motivation als eine innewohnende, weniger beeinflussbare zu charakterisieren. Dieses Verhalten wird auch ohne externe Belohnungen begonnen und aufrechterhalten. Wenn sich Belohnungen ergeben werden diese vermehrt über kognitive und affektive Prozesse erzielt. Andere interne Faktoren sind der Altruismus und ein hohes Ansehen in der Community, die eng miteinander verknüpft sind. Auf die Entwicklung von OS SC umgelegt bedeutet dies, dass es Programmierer gibt, die gute Ideen umsetzten um diese auch der Allgemeinheit zur Verfügung zu stellen und somit ganze Softwarekomponenten

[7] http://de.wikipedia.org/wiki/Intrinsische_Motivation

verbessern. Dadurch wird das eigene Ansehen in der OS Community gesteigert und die persönliche Identifikation mit der Gruppe verstärkt [HaOu 2001].

Bei den externen Faktoren handelt es sich um die Befriedigung persönlicher Bedürfnisse, mögliche zukünftige Auszeichnungen, aber auch um sich monetären Wertausgleich zu sichern. Viele OS Projekte sind ursprünglich aufgrund von persönlichen Bedürfnissen bzw. akuten Problemstellungen, initiiert wurden. Beispielsweise hat Larry Wall die Programmiersprache PERL erfunden um Webseiten generieren zu können, welche bis dato mit komplizierten C Routinen gelöst werden mussten. Die Entwicklung des Apache Web Server wurde aus ähnlichen Überlegungen gestartet. Zum anderen sehen viele Programmierer ihre Teilnahme an OS Projekten als einen Invest in die Zukunft, zur Erlangung von Einkünften unterschiedlicher Art. Zu diesen Einkünften zählen ein erhöhter Wissensstand und eine Wertsteigerung der Marke Ich, welche sich in der realen kommerziellen Welt durch bessere Jobchancen und höhere Löhne widerspiegeln kann [HaOu 2001].

3.1.3 Chance oder Hindernis

Dieser zusammenfassende Abschnitt zum Thema OS soll herausstreichen, welche Chancen diese Entwicklungsmethode gegenüber traditionellen Ansätzen hat: Gleichzeitig sollen auch die Hindernisse, welchen sich die OS Bewegung gegenüber sieht, nicht außer Acht gelassen werden.

Der am häufigsten angesprochene Punkt, nämlich dass sich OS Projekte selbst organisieren, wird von Chuck Connell [Co 2000] widerlegt. Sie werden nicht allein durch Gruppendenken oder einem anderen selbstorganisierenden Dynamismus gesteuert, vielmehr werden erfolgreiche OS Projekte von intelligenten effizienten Projektleitern – die zumeist auch die Gründer oder Ideenlieferanten sind – gelenkt. Seiner Meinung nach ist die Projektstruktur eher einer Kathedrale, als einem Basar nach geordnet – wenn man bei den Bezeichnungen von [Ra 2000] bleibt. Ein weiterer Kritikpunkt ist der Mangel an Professionalität bei kleineren Projekten, hierbei wird auf hinreichende Testphasen und Dokumentation weitestgehend verzichtet. Bei größeren Projekten werden im Bezug auf Produktivität und Codequalität klare Prozesse, eine präzise Fehlererkennung und empirische Anhaltspunkte vermisst. Durch die Offenlegung des SC mit seinen bekannten

Schwachstellen und Sicherheitslücken sehen Kritiker Tür und Tor für Hackerangriffe geöffnet [Wiki d 2008].

Die Vorteile von OS beschränken sich nicht nur auf die Entwicklung von zuverlässiger hochqualitativer Software bei einem geringen Einsatz von Kosten und Ressourcen, sondern es bietet Unternehmen neben dem Einsatz von flexiblen Technologien auch die Möglichkeit Innovationen rasch voranzutreiben. Daraus können sich Industriestandards entwickeln, die einen enormen Wettbewerbsvorteil mit sich ziehen können. Durch die Loyalität der Entwickler zu ihrem Produkt steigt die gesamte Zuverlässigkeit der entwickelten Software. Das manifestiert sich durch die Suche nach Bugs und dem Einbringen der entsprechenden Lösungen [Wiki d 2008].

3.2 Die Eclipse Plattform

Nachdem die notwendige Einführung über OS hinreichendes Wissen über dieses Thema geliefert hat, dient dieser Abschnitt dazu ein spezielles OS Projekt näher zu erläutern. Zum einen, weil es repräsentativ für die Entwicklungen im OS Bereich ist und zum anderen weil dieses Tool die Grundlage für die Erstellung des Web-Modelleditor Prototyps sein soll.

Eclipse hat in den letzten Jahren einen hohen Bekanntheitsgrad in der Entwicklerszene errungen. Aber was genau ist Eclipse eigentlich? Der erste Eindruck, den man auf www.eclipse.org erhält ist, dass Eclipse die universelle Plattform für das Erstellen von Tools ist, aber eigentlich ist Eclipse doch auch eine *Integrated Development Environment* (IDE). Die zuvor genannten Eigenschaften werden in ein und derselben Plattform vereint. Neben einer genauen Beschreibung von Eclipse wird auch auf die Anforderungen an eine moderne IDE eingegangen. Mit der Geschichte und dem architektonischen Aufbau von Eclipse wird dieses Kapitel abgerundet.

3.2.1 Anforderungen an eine moderne IDE

Die Anforderungen an eine IDE beschränken sich mitunter nicht nur auf das vorhandene Budget und die Liste an Features, die geliefert werden, sondern hängen auch oftmals von den persönlichen Vorstellungen des Entwicklers oder des Unternehmens ab [BrWi 2006].

Aus eigener Erfahrung weiß ich, dass bei Anwendungen oft unzählige Features zur Verfügung stehen, welche aber zumeist nicht in dem Ausmaß genutzt werden, wie sie eigentlich sollten. Dennoch sind sie in der Applikation enthalten und müssen als Paket gekauft werden. Sinnvoll erscheint an dieser Stelle der Ansatz, eine Basisversion auszuliefern und zusätzliche Bausteine, Komponenten oder Plugins für die individuelle Erweiterung zur Verfügung zu stellen. Diesen Ansatz der individuellen Anpassbarkeit wird in dieser Form derzeit nur von Eclipse verfolgt und soll in den nächsten Abschnitten verdeutlicht werden.

3.2.2 Geschichte und Ziele von Eclipse

Bevor auf die Architektur und technologischen Vorteile von Eclipse eingegangen wird, soll dieses Kapitel dazu genutzt werden, die Geschichte und die grundlegenden Visionen von Eclipse offen zu legen. Die grundlegende Idee hinter dem ursprünglichen Eclipse Projekt baut auf einer für alle *Betriebssysteme* (BS) offener Plattform auf, in die leicht Tools zur Erweiterung integriert werden können. Eine weitere Vision war es neben Java auch die unterschiedlichsten Programmiersprachen, wie beispielsweise C / C++, C#, COBOL, Fortran, Groovy, Nice, LISP, Jython / Python zu unterstützen [BrWi 2006], [We 2003]. Eclipse soll somit zwei Welten miteinander vereinen, zum einen handelt es sich um eine IDE, in der mit den unterschiedlichsten Sprachen programmiert werden kann, wobei Java die am besten unterstützte Sprache ist. Zum anderen ist es eine Plattform mit der Plugins zur Erweiterung entwickelt werden können.

Die Umsetzung der oben beschriebenen Ziele wurde mit der Gründung einer 100% Firmentochter von IBM, namens Object Technology International bestritten. Das dafür eingestellte Entwicklerteam hatte große Erfahrung mit dem Bau leistungsfähiger Development Tools. Die zu Beginn entwickelten Versionen sollten ursprünglich noch ihren kommerziellen Einsatz finden, ehe sich IBM mit dem Release 1.0 im Jahre 2001 entschloss, Eclipse der OS Community, unter den Lizenzbedingungen *Eclipse Public License* (EPL), zur Verfügung zu stellen [We 2003].

Die EPL ist eine benutzerfreundliche Lizenz, die es Unternehmen erlaubt, Softwarebausteine, welche mit Eclipse entwickelt worden sind in kommerzielle Produkte zu integrieren und zu vertreiben. In der Lizenz ist aber festgehalten, dass

der entwickelte SC der Eclipse Plattform zur Verfügung gestellt werden soll. Die letzten Jahre haben bewiesen, dass sich dieses System bewährt hat, denn hunderte Firmen beliefern Eclipse regelmäßig mit neuen Entwicklungen. Das war auch die Geburtsstunde der Kommunikationsplattform www.eclipse.org, wo neben den aktuellen Downloads auch Tutorials, Newsgroups und ein eigenes Wiki für die User zur Verfügung steht. Die Community erhält auch große Unterstützung von IT Solution Providern, Universitäten und engagierten Forschungsinstituten [NtE 2008].

Geleitet, gesteuert und vorangetrieben wird das gesamte Projekt von dem so genannten Eclipse Konsortium, dem namhafte Unternehmen wie IBM, Borland, Rational Software, Red Hat und SuSe[8] angehören. Um sich nur eine Größenordnung vorstellen zu können, im Jahre 2003 ist die Anzahl der Mitglieder dieses Konsortiums auf bereits 80 angestiegen. Anfang 2004 kam es zu einer wichtigen Entscheidung, die den weiteren Verlauf von Eclipse ebnete und es zu dem machte, was es heute ist, eine gemeinnützige Gesellschaft. Jegliche technologischen Entwicklungen und entstehenden SC die von der Community erstellt worden sind, sollen ohne Lizenzgebühren der Eclipse Gemeinde zur Verfügung gestellt werden. Lediglich die Einhaltung der zuvor beschriebenen EPL muss berücksichtigt bleiben [EcO 2008].

Die Funktion des Konsortiums ist wie aus dem letzten Abschnitt ersichtlich, nicht die Entwicklung von SC, sondern vielmehr die administrativen Schwierigkeiten zu meistern und die notwendige Infrastruktur, die der Plattform zugrunde liegt, zur Verfügung zu stellen. Zusätzlich schüren sie die Marketingtrommel um immer wieder neue OS Projekte zu lukrieren. Zurzeit wird an 60 verschiedenen OS Projekten geforscht, die den folgenden Aufgabenbereichen zugeordnet werden können [NtE 2008]:

- Enterprise Development
- Embedded and Device Development
- Rich Client Platform
- Rich Internet Applications

[8] Eine vollständige Liste der Mitglieder des Eclipse Konsortiums kann unter http://www.eclipse.org/membership/exploreMembership.php nachgelesen werden.

- Application Frameworks
- Application Lifecycle Management
- Service Oriented Architecture

3.2.3 Architektur

Nachdem in den vorigen Abschnitten die Visionen von Eclipse vorgestellt worden sind, konzentriert sich dieses Kapitel auf die grundlegende Architektur von der Eclipse IDE. Neben dem Aufbau durch Komponenten werden auch die standardmäßig ausgelieferten Plugins *Java Development Tools* (JDT) und *Plugin Development Environment* (PDE) erläutert.

Was die Architektur von Eclipse anbelangt, so baut diese auf einem einfachen Konzept auf. Anders als bei anderen proprietären Entwicklungsumgebungen, baut Eclipse auf einer Komponentenarchitektur auf. Eclipse ist in viele kleine Bausteine, so genannte Plugins zerlegt worden, denen Komponenten für die reibungslose Kommunikation zugrunde liegen. Miteinander können die Plugins als eine große funktionierende Entwicklungsumgebung agieren. Durch den Einsatz dieser kleinen Bausteine ist es möglich, benutzerspezifische sowie anforderungsspezifische Umgebungen aufzubauen [We 2003].

Durch diese Komponentenbauweise spaltet sich Eclipse von den handelsüblichen Ansätzen ab, die im Regelfall einer monolithischen Architektur zugrunde liegen. Will man als Entwickler Module von Drittherstellern in die Entwicklungsumgebung einbetten, ist dies relativ einfach mittels der dafür zur Verfügung gestellten APIs möglich. Will man hingegen eigene Erweiterungen integrieren, stößt man schnell an die Grenzen des Machbaren. Durch das Bausteinsystem mit den Plugins existiert praktisch fast keine Trennung von Drittherstellerkomponenten, seien es private oder industrielle. Im Allgemeinen müssen Plugins gewissen Regeln – diese werden in den kommenden Abschnitten erklärt – folgen, dann jedoch können diese reibungslos miteinander kommunizieren und sich nahtlos in die Gesamtfunktionalität einbauen lassen [We 2003].

Durch die Erläuterungen des vorigen Abschnitts wird klar ersichtlich, warum Eclipse als eine universelle Plattform angesehen werden kann. Ausgeliefert wird eine Basisumgebung, mit der einfache Anwendungen erstellt werden können. Werden die Aufgabenstellungen komplexer, können aus einem Plugin Katalog Elemente

hinzugefügt oder selbst erstellt werden, was in der Abb. 3.1 durch die Plugins „Tool“ und „My Tool“ illustriert wird. Durch diese individuellen Anpassungen handelt es sich jedoch immer noch um Eclipse als Entwicklungswerkzeug.

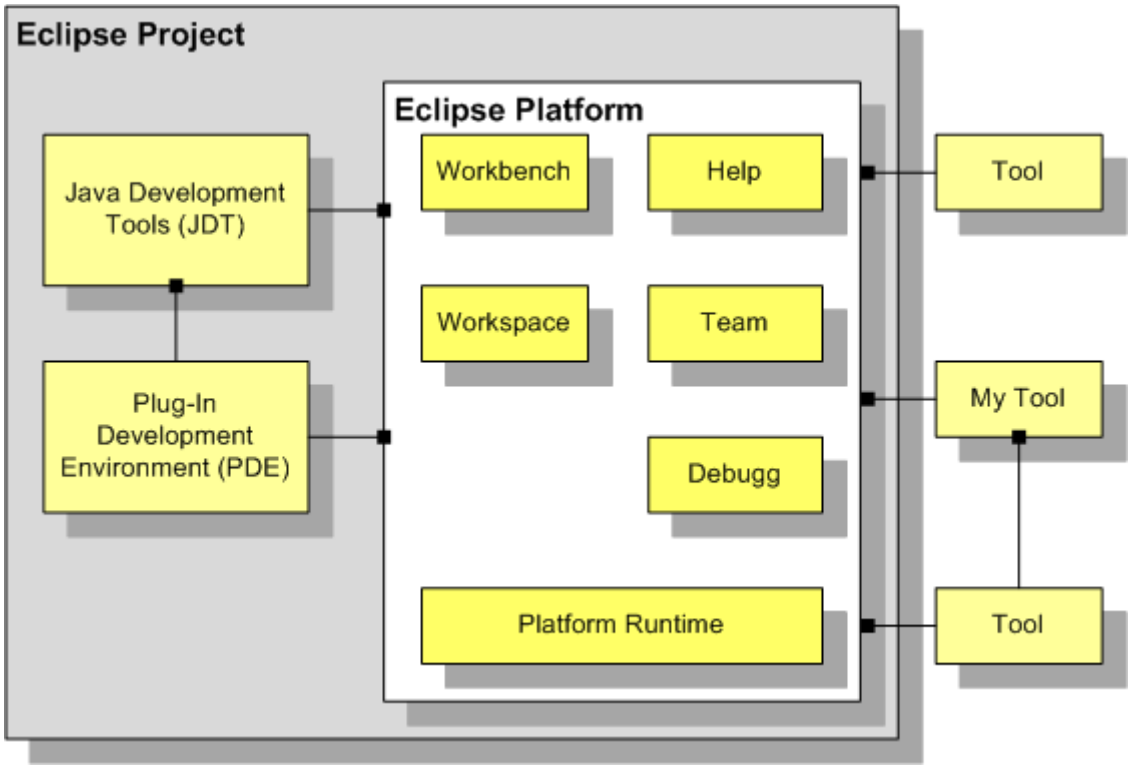

Abb. 3.1 – Basis Architektur von Eclipse [We 2003]

3.2.3.1 Aufbau des Plugin

Der kleinste Baustein in der Eclipse Welt ist das Plugin, welche eine funktionale Bandbreite von einfach überschaubaren Funktionen bis hin zu mächtigen XML Editoren haben können. Das Plugin selbst muss zusätzlich zur neuen Funktionalität auch so genannte Extension Points definieren, welche eine formale Spezifikation einer Schnittstelle beschreiben [We 2003]. Durch diesen Mechanismus ist es in Eclipse möglich, neue Plugins in die bestehende Entwicklungsumgebung zu integrieren und die Interaktivität der einzelnen Plugins zu unterstützen, siehe dazu auch Abb. 3.1.

Die Bestandteile eines Plugin lassen sich in die JAR Datei und ein dazugehöriges Plugin Manifest unterteilen. Wobei in der JAR Datei der ausführbare Code enthalten ist. Hingegen ist in dem Manifest, welches ein Plugin eindeutig identifiziert, festgehalten, welche Extension Points wie genutzt werden können und

wie der Name der JAR Datei ist. Komplexere Plugins, die aus mehreren Komponenten bestehen, werden in Eclipse als Feature bezeichnet und mit einem Feature Manifest beschrieben [We 2003].

3.2.3.2 Java Development Tools

Das JDT Projekt setzt direkt auf die Eclipse Plattform auf – siehe dazu Abb. 3.1 – liefert ein Set an Plugins und fügt die notwendigen Ressourcen hinzu um Eclipse als vollständige Java IDE verfügbar zu machen. Dadurch, dass alle Plugins mit dazugehörigen APIs ausgeliefert werden, können diese problemlos von anderen Entwicklern verwendet und erweitert werden [JDT 2008], [We 2003].

3.2.3.3 Plugin Development Environment

Wie aus der Abb. 3.1 ersichtlich ist, baut das PDE auf der Eclipse Plattform und dem JDT auf und ist für die Erstellung von Plugins und Features entwickelt worden. Das PDE liefert die notwendige Infrastruktur für die Entwicklung, das Testen und das Debuggen von Plugins und Features. Zusätzlich gibt es umfangreiche OSGi[9] Tools für die Komponentenprogrammierung, die nicht nur für die Entwicklung von Eclipse Plugins herangezogen werden können [PDE 2008], [We 2003]. Zum Thema OSGi wird noch im Abschnitt Equinox Stellung genommen.

Eclipse ist eine leistungsfähige Plattform, die sich durch ihre enorme Erweiterbarkeit durch den Aufbau von Plugins profiliert. Zusätzlich bietet sie den Entwicklern eine umfangreiche Infrastruktur für die Erstellung individueller Tools. Durch den Umstand, dass Eclipse OS ist, fließen viele der neu entwickelten Tools wieder an die stetig wachsende Gemeinde zurück, was auch wiederum die Wiederverwertbarkeit des gesamten Tools verstärkt. Auch für reine Anwendungsentwickler bietet Eclipse durch die Unterstützung von mehreren Programmiersprachen eine außerordentlich gute Basis.

[9] Detaillierte Informationen findet man unter http://www.osgi.org

3.3 Graphical Modeling Framework

Wie schon die vorherigen Kapiteln gezeigt haben, ist Eclipse eine Plattform auf der verschiedene Projekte, Plugins und Frameworks gehostet und implementiert werden können. Das GMF ist ein solcher Vertreter, stellt ein Framework innerhalb von Eclipse dar und wurde zuerst im Juni 2006 ausgeliefert. Es bietet eine generative Komponenten- und Runtime Infrastruktur zur Erstellung von grafischen Modelleditoren. Wie die Abb. 3.2 zeigt liegen die Wurzeln von GMF bei dem EMF und dem GEF [Wiki g 2008] [GMF 2008]. Bevor genauer auf das GMF eingegangen werden kann, ist es daher notwendig die grundlegendsten Eigenschaften vom EMF und GEF in den kommenden Unterkapiteln näher zu erläutern.

3.3.1 Eclipse Modeling Framework

Wie das GMF ist auch das EMF ein auf Eclipse basierendes Java / XML Framework zur Modellierung und die anschließende SC Generierung, ausgehend von einem strukturierten Modell. Anders ausgedrückt, kann das EMF für die Generierung von Tools und Applikationen aufgrund von simplen Klassenmodellen benutzt werden. Die Modelle können in annotiertem Java, in UML oder in XML Dokumenten spezifiziert und entwickelt werden. Zusätzlich können zur Modellentwicklung auch Tools, wie beispielsweise Rational Rose verwendet und nachfolgend in EMF importiert werden. Die Modelle werden von EMF rasch, effizient und korrekt in einen einfach erweiterbaren Java Code überführt, welcher einen einfachen Modellierungseditor repräsentiert [Wiki e 2008] [EMF 2008].

Für eine applikationsübergreifende Kommunikation und Zusammenarbeit ist es möglich die Java Objekte als XML Dokumente abzulegen und diese für den Austausch heranzuziehen. Wie schon erwähnt ist es möglich den generierten SC mit applikationsspezifischen Funktionen und Variablen zu modifizieren, der Code bleibt jedoch weiterhin von EMF generierbar. Genauer gesagt bleiben Veränderungen, herbeigeführt vom Benutzer, nach einer erneuten Überführung des Modells in SC erhalten. Auf der Gegenseite können Veränderungen im erzeugten Code zum Aktualisieren des ursprünglichen Modells herangezogen werden [EMF Faq 2008].

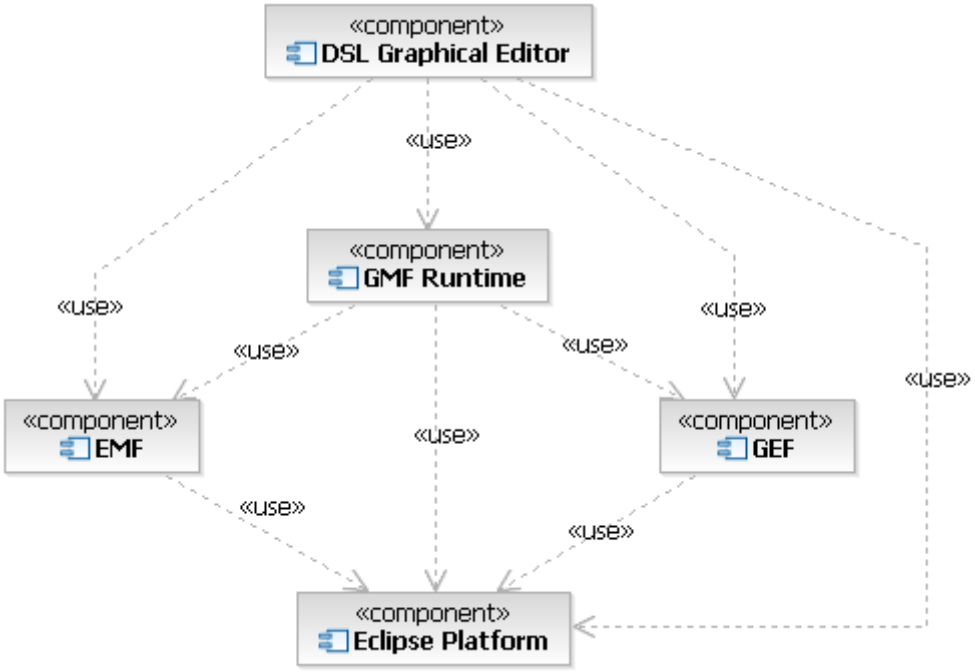

Abb. 3.2 – Komponentenstruktur von GMF [GMF Run 2008]

Wie aus der prägnanten Beschreibung von EMF hervorgeht, ist dieses ein Framework zur Generierung von Java Code aus Modellen. Das EMF selbst besitzt keine Unterstützung für grafische Elemente sondern liefert dem GMF vielmehr nur die zuvor erläuterten Funktionalitäten. Die grafische Funktionalität wird vom GEF inspiriert, welches im nächsten Kapitel besprochen wird.

3.3.2 Graphical Editing Framework

Auch das GEF ist ein auf der Eclipse Plattform basierendes Framework, anders als bei dem zuvor beschriebenen EMF bietet es jedoch die Möglichkeit umfangreiche sowie konsistente grafische Editoren zu schaffen. Das GEF erlaubt es Entwicklern aus bestehenden Modellen mächtige grafische Editoren zu kreieren [GEF 2008]. Zu diesen Modellen zählen jene die mit dem EMF entstanden sind, sie können problemlos in die GEF Umgebung integriert werden. Zusätzlich können aber auch Modelle zur Verwendung kommen die auf anderem Wege entwickelt worden sind, ähnlich dem Ansatz der auch vom EMF verfolgt wird [Wiki f 2008].

Das Framework selbst besteht aus den folgenden Komponenten, das `org.eclipse.draw2d` Plugin verfügt über Layout und Rendering Mechanismen zum Darstellen der grafischen Elemente. Wie es in Eclipse üblich ist können diese

Werkzeuge, je nach Aufgabenstellung und Applikation, frei nach den Wünschen der Entwickler erweitert werden [GEF 2008]. Die zweite Komponente wird durch den integrierten Model View Controller Mechanismus – siehe Abb. 3.3 – realisiert. Der Controller, welcher in GEF EditPart genannt wird, hat die Aufgabe sowohl das Modell als auch den dargestellten View zu beobachten. Wird einer der beiden durch eine Interaktion des Users verändert, übernimmt der Controller die Synchronisation und die Konsistenz zwischen dem Model und dem View bleibt erhalten [IBM GEF 2008].

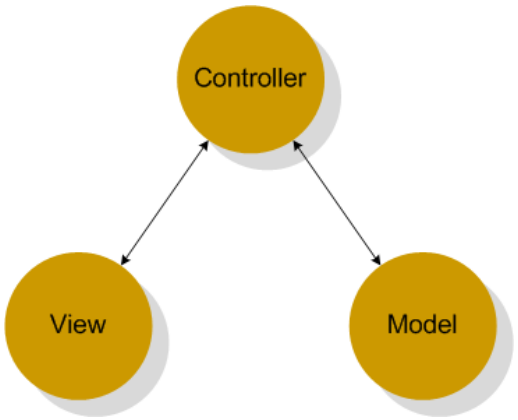

Abb. 3.3 – Das Konzept des Model View Controller.

Das GEF ist gegenüber Programmen neutral, dass ist verstärkt erkennbar durch die Möglichkeit, dass fast jede Applikation entwickelt werden kann. Einige Beispiele sind Aktivitätsdiagramme, Editoren für Klassendiagramme, State Machines oder WYSIWYG[10] Editoren [GEF 2008].

3.3.3 Graphical Modeling Framework

Das GMF ist ein neues Eclipse Projekt mit großem Potential, ein essentielles Framework für die rasante Entwicklung von standardisierten grafischen Modellierungseditoren zu werden. Unabhängig von der technischen Umsetzung bietet das GMF dem Entwickler die Möglichkeit grafische Editoren zu erstellen, beispielsweise können UML Modellierungstools, Flussdiagramme, Workflow, Business Process oder XSD Editoren generiert werden. Allgemeiner gesprochen

[10] Informationen zu **W**hat **y**ou **s**ee **i**s **w**hat **y**ou **g**et Editoren findet man bei Wikipedia unter: http://de.wikipedia.org/wiki/WYSIWYG

können grafische Editoren für die unterschiedlichsten Domain Modelle produziert werden [GMF Faq 2008]. Das GMF hat auch die Absicht eine generative Brücke zwischen dem EMF und GEF herzustellen. Die gemeinsame Verwendung von EMF und GEF ist in der Eclipse Gemeinde nichts ungewöhnliches, wie auch aus den Ausführungen der vorangegangenen Kapiteln hervor geht. Viele dieser dadurch entstandenen Plugins haben die Entwicklung vom GMF inspiriert. Das Framework selbst kann in zwei große Bereiche unterteilt werden, zum einen in die Runtime, zum anderen das Tooling. Letzteres hat die Aufgabe die notwendigen Modelle (Domain-, Notation, Grafisches Modell), die in den nächsten Schritten präsentiert werden, zu definieren. Das ist Notwendig damit die Runtime fehlerfrei und effektiv zum Generieren von Eclipse basierenden grafischen Editoren heran gezogen werden kann [GMF Run 2008]. Weiterführend soll jetzt das Tooling und die Runtime genauer beschrieben.

Wie schon zuvor erwähnt hat das Tooling die Aufgabe die nötigen Modelle zu erstellen. Die Abb. 3.4 illustriert die Modelle und Komponenten die in einem gewöhnlichen GMF Entwicklungsprozess eine entscheidende Rolle spielen. Angefangen von dem Gedanken einen grafischen Modellierungseditor zu entwickeln sind das Domain- und das grafische Definitionsmodell zu entwerfen. Das Domainmodell spiegelt die Aufgabenstellung wieder und beschreibt wie die Elemente miteinander interagieren. Das Herzstück ist das Konzept des grafischen Definitionsmodells, welches detaillierte Informationen zu den enthaltenen grafischen Elementen enthält. Zusätzlich gibt es noch die Tooling Definitionen, mit denen Menüs und Toolbars designed werden können. Erwartungsgemäß kann das grafische Modell verschiedenen Domains bzw. Applikationen zugeordnet werden und umgekehrt. Das Mapping Model hat die Aufgabe die grafischen Komponenten mit dem Domain Modell zu verknüpfen. Daraus entsteht das Generator Modell, mit diesem werden die Implementierungsdetails für die Generierungsphase definiert und umgesetzt. Dadurch erhält man das endgültige Notationsmodell, dieses hat die Aufgabe die Persistenz zwischen sich und dem Domain Modell sicher zu stellen und sich um Synchronisationsaspekte zu kümmern die durch Änderungen in den Modellen zu Stande kommen [GMF Tut 2008], [GMF Faq 2008]. Im Kapitel 5 werden diese Modelle, deren Zusammenhänge und der Ablauf während der Modellierung Anhand des Prototyps praktisch erklärt.

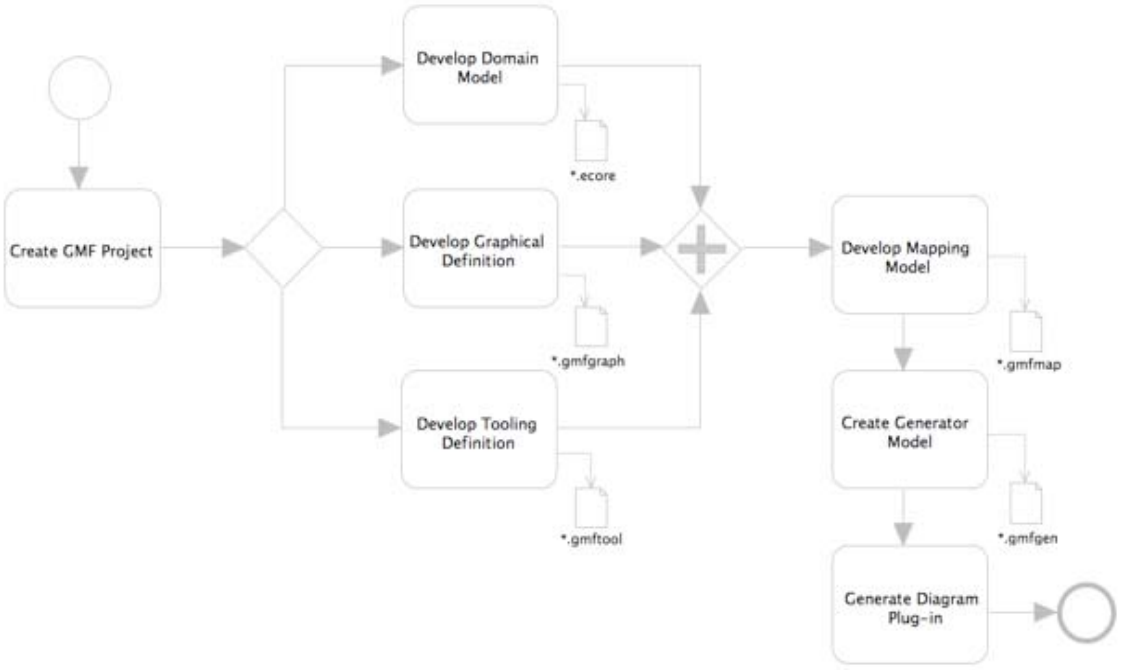

Abb. 3.4 – Arbeitsfluss zur Erstellung eines grafischen Editors mit GMF [GMF Tut 2008]

Zuletzt sind die Arbeitsweise und die Zusammenhänge der einzelnen Komponenten kennen gelernt worden, abschließend soll genauer auf den eigentlichen Mehrwert, der sich durch den Einsatz dieses Konzeptes manifestiert, eingegangen werden. Auf die Frage welche Vorteile man mit GMF gegenüber reinem EMF und GEF erzielen kann, gibt es keine Eindeutige Antwort. Es hängt vielmehr von der Aufgabenstellung ab die gelöst werden soll. GMF ist eine gute Option wenn man einen umfangreichen grafischen Editor benötigt, der viele GEF Ressourcen benutzt und mit einem EMF Domain Model arbeitet. Generell kann man sagen, dass GMF versucht eine generative Brücke zwischen den Projekten EMF und GEF zu schaffen und das Beste aus diesen beiden Welten zu vereinen [GMF Faq 2008]. Zusammengefasst können noch zusätzliche Vorteile genannt werden [GMF Run 2008]:

- Konsistentes Look and Feel auch bei anderen GMF basierten Editoren
- Die erstellten Editoren können von Drittanbietern erweitert werden, weil ein offener Zugang zum Code gewährleistet ist.
- Verbesserungen in der Runtime fließen automatisch in schon erstellte Editoren.
- Eine stabile Entwicklungsumgebung.

- Es existiert ein erweiterbares Notation Meta Model um zukünftig die Notation von der Semantik zu trennen.

3.4 Rich Ajax Platform

Ebenso wie das zuvor kennen gelernte GMF ist auch das RAP ein Plugin für das Eclipse Framework. Das RAP bietet eine Entwicklungsumgebung für die Umsetzung von Web 2.0 Anwendungen auf Basis von Java. Des Weiteren werden vom Entwickler keinerlei Kenntnisse von HTML, JavaScript oder XML vorausgesetzt, da diese Elemente aus der Java RAP Engine heraus generiert werden können [Wiki h 2008]. Generell ist ein gutes Verständnis von aktuellen Web-Technologien ein großer Vorteil, wenn Anwendungen für das Internet entwickelt werden.

Die RAP Plattform lässt sich von der *Rich Client Platform* (RCP) ableiten und erweitert diese um die Möglichkeit die erstellten Anwendungen ins Web zu bringen [StHe 2008]. Es ermöglicht eine komponentenorientierte Entwicklung und deren Zusammenbau zu mächtigen Online Anwendungen, durch ein Mashup von unterschiedlichsten, für RAP angepassten Technologien wie JFace, Workbench und SWT [ApKr 2008]. Unterstützt wird der Entwickler ebenso, weil auf das gesamte Eclipse Entwicklungsmodell und dessen Plugins zurückgegriffen werden kann. Für die clientseitige Präsentation der Objekte, wird das SWT durch das so genannte *RAP Widget Toolkit* (RWT) ersetzt, welches in Richtung Web gegenüber dem SWT optimiert wurde [RAP 2008]. Als ein zusätzliches Feature wird in der Literatur erwähnt, das mit einem geringen Adaptionsaufwand des SC, die Anwendung dahingehend angepasst werden kann, damit sie auch als reine Desktopanwendung kompilierbar und ausführbar wird [Wiki h 2008].

Dieses Kapitel bietet einen informativen Einstieg in die Welt von RAP, das beinhaltet einen Überblick über die Architektur von RAP und die damit verbundenen notwendigen technologischen Voraussetzungen. Dazu gehören der Ajax Mechanismus, Workbench, JFace, sowie das RWT, welche in den Folgenden Unterkapiteln beschrieben werden.

3.4.1 Ajax

Ajax ist keine Programmiersprache sondern ein Ansatz der es ermöglicht eine http Anfrage innerhalb einer Webseite durchzuführen, ohne dass die gesamte Seite neu geladen werden muss. Dieser asynchrone Mechanismus zur Datenübertragung zwischen dem Browser und dem Server beruht auf dem `XMLHttpRequest` Objekt, welches schon im Internet Explorer 5.0 integriert war. Erst in den letzten Jahren erlebte diese Technik einen regelrechten Boom und wurde zu der Schlüsseltechnik für Web 2.0 Anwendungen. Im eigentlichen Sinn ist Ajax ein Mashup von unterschiedlichen etablierten Web-Technologien um interaktive desktopähnliche Web-Anwendungen zu realisieren. Die in Ajax eingesetzten Technologien sind [Wiki i 2008]:

- HTML oder XHTML
- Das *Document Object Model* (DOM) hilft bei der Verwaltung von Daten und Inhalte.
- JavaScript wird zur Manipulation des DOM heran gezogen und dient für die teilnehmenden Komponenten als Schnittstelle zwischen Client und Server.
- Der `XMLHttpRequest` ist das eigentliche Kernstück in Ajax und ermöglicht eine asynchrone Kommunikation.
- XML bzw. jedes andere textbasierte Format kann für die zu transferierenden Daten herangezogen.
- CSS wird für das Layouting auf der Webseite verwendet.
- XSLT kann für die Datentransformation benutzt werden.

3.4.1.1 XMLHttpRequest

Der Großteil der eingesetzten Technologien sollte wohl vertraut sein, der `XMLHttpRequest` hingegen ist die Schlüsselstelle des Ajax Mechanismus und zählt wahrscheinlich nicht zur bekanntesten Komponente. Deswegen soll dieser Baustein exakter beleuchtet werden. Das `XMLHttpRequest` Objekt implementiert ein Interface[11] – einen Auszug ist in Abb. 3.5 dargestellt – das von einer Scripting

[11] Eine ausführliche Beschreibung aller Methoden, Attribute, Eventhandler und Fehlermeldungen findet man auf der Seite des W3C: http://www.w3.org/TR/XMLHttpRequest/

Engine, mit dem Ziel http Funktionen am Client durchzuführen, beansprucht wird. Als Beispiele können an dieser Stelle das Absenden eines Formulars oder das Laden von bestimmten Daten vom Server genannt werden. Nachdem der Name des Objektes eingeschränkte Funktionalitäten vermuten lässt, dem aber nicht so ist, sollen die folgenden Informationen dazu dienen Missverständnisse aus dem Weg zu räumen. Zunächst unterstützt das `XMLHttpRequest` Objekt jedes textbasierte Format für den Transport zwischen Client und Server, nicht nur XML. Zusätzlich können Anfragen auch über https und nicht nur über http gestellt werden. Abschließend sollen, ausgehend vom W3C [W3C XHR 2008] Standard zumindest die http Methoden DELETE, GET, HEAD, POST, PUT, OPTIONS unterstützt werden.

Für die Entwickler sind die in Abb. 3.5 dargestellten Informationen über Events und Attribute von Bedeutung. Der Eventhandler `onreadystatechange` wird immer dann ausgeführt wenn ein Event am `XMLHttpRequest` Objekt festzustellen ist. Dieser führt zu einer Statusänderung des Objektes und wird im nur lesbaren Attribut `readyState` abgelegt. Ein `XMLHttpRequest` Objekt kann folgende Statusphasen einnehmen [W3C XHR 2008]:

- UNSENT: Dieser Status wird erreicht nachdem ein `XMLHttpRequest` Objekt erzeugt wurde.
- OPENED: Dieser Status wird durch das erfolgreiche ausführen der `open()` Methode eingeleitet. Es können die Request Header gesetzt und die Anfrage mit der `send()` Methode gesendet werden.
- HEADERS_RECEIVED: Dieser Status wird erreicht wenn alle Response Header empfangen wurden.
- LOADING: Während des Empfangs der Antwort vom Server ist das Objekt in diesem Status.
- DONE: Repräsentiert den Status wenn der Transfer entweder komplett erfolgt ist oder Fehler während der Übertragung aufgetreten sind.

Als nächster Schritt wird der Ablauf innerhalb einer Ajax Anwendung und die Kommunikationsaspekte zwischen Client und Server beschrieben. Zur deutlicheren Abgrenzung werden die Ergebnisse mit jenen einer klassischen Web-Anwendung

verglichen, um herauszustreichen wie das Ajax Konzept die Webseitenprogrammierung effizient unterstützen kann.

```
interface XMLHttpRequest {

  attribute EventListener onreadystatechange;

  const unsigned short UNSENT = 0;
  const unsigned short OPENED = 1;
  const unsigned short HEADERS_RECEIVED = 2;
  const unsigned short LOADING = 3;
  const unsigned short DONE = 4;
  readonly attribute unsigned short readyState;

  .
  .
  .

};
```

Abb. 3.5 – Eventhandler und Status im `XMLHttpRequest` Interface [W3C XHR 2008]

3.4.1.2 Ajax vs. Klassische Web-Anwendungen

Die Funktionsweise von klassischen Client Server Architekturen dürfte bekannt sein, dennoch ist es interessant den Unterschied zu Ajax Anwendungen hervorzuheben. Unterstützt werden diese Erklärungen durch die Abb. 3.6 sowie 3.7 a und b. Betrachtet man zunächst die Abb. 3.6, die die auszutauschenden Daten und die Kommunikation zwischen dem Browser und dem Server darstellt, ist schon clientseitig eine Differenz zu bemerken. Bei der klassischen Anwendung stellt der Browser direkt eine http Anfrage an den Webserver, bei der Ajax Anwendung hingegen wird eine JavaScript Funktion aufgerufen die in der Ajax Ebene verarbeitet und dann der dementsprechende Request an den Server gestellt wird. Serverseitig gibt es kaum Unterschiede, außer das nicht zwingend HTML / CSS, sondern auch XML oder JavaScript, an den Client zurück gesendet werden kann. Abgefangen werden die Daten wieder von der Ajax Schicht, die im Anschluss daraus das eigentliche HTML und die nötigen CSS Dateien generiert welche der Browser zur Darstellung verwendet. Wie gut ersichtlich ist das große Novum die vermittelnde Ajax Schicht, die diese asynchrone Kommunikation erst ermöglicht.

Ein interessanter Unterschied ist auch zu beobachten, wenn man die http Anfragen über die Zeit hinweg bei klassischen und Ajax Anwendungen vergleicht, was in Abb. 3.7 a und b illustriert ist. Die Arbeitsweise der synchronen Übertragung ist leicht erklärt, auf eine Benutzeraktivität folgt die Datenübertragung, die serverseitige Verarbeitung und die neuerliche Übertragung der geforderten Daten.

In diesem Modell gibt es keine Möglichkeit, dass der Benutzer während er auf Serverantworten wartet, eine zusätzliche Aktivität setzt. Anders bei dem Ajax Modell, der Benutzer kann jederzeit Aktionen im Browser setzten, diese werden von der Ajax Engine übernommen und an den Server zur Bearbeitung geschickt. Sobald die Ajax Schicht die Ergebnisse erhalten hat werden sie im Browser dargestellt. Dem Benutzer wird somit das Gefühl vermittelt sich in einer desktopähnlichen Umgebung zu befinden, wo die Anfragen kontinuierlich gestellt um danach asynchron und sequentiell abgearbeitet zu werden.

3.4.1.3 Vorteile und Nachteile

Aus jeder Technologie ergeben sich Vor- und Nachteile die es zu beachten gibt. Dieser Abschnitt widmet sich der Thematik, wie man die Vorteile effizient ausnützen kann, beleuchtet aber auch die Kritikpunkte denen sich ein Webentwickler gegenüber sieht. Die, wenn beachtet und durchdacht gelöst werden, zu keinerlei großen Nachteilen von Ajax Seiten gegenüber klassischen Web-Anwendung führen werden.

Der wohl bedeutendste Vorteil ist das nur bestimmte, vom Entwickler speziell ausgewählte Bereiche einer Webseite, nach einer Benutzerinteraktion neu geladen werden müssen. Dadurch besteht durch den Ajax Mechanismus die Möglichkeit dem Server bestimmte Rechenschritte, wie beispielsweise das generieren des HTML Codes, abzunehmen. Daraus resultiert eine Entlastung des Servers und des Weiteren wird weniger Bandbreite verbraucht, da nur JavaScript Code und Daten übertragen werden. Die statischen Elemente einer Webseite müssen in diesem Fall nicht durch jede Benutzereingabe neu vom Server berechnet und gesendet werden [Wiki i 2008] [Wiki j 2008].

Zusätzlich ist Ajax eine frei zugängliche Technologie die von Webbrowsern getragen werden, die eine Unterstützung von JavaScript integriert haben und können somit unabhängig vom BS arbeiten. Zur reibungslosen Ausführung einer Ajax Webseite ist keine zusätzliche Installation eines Plugins mehr notwendig, was einen transparenteren Zugang für den Benutzer schafft [Wiki i 2008].

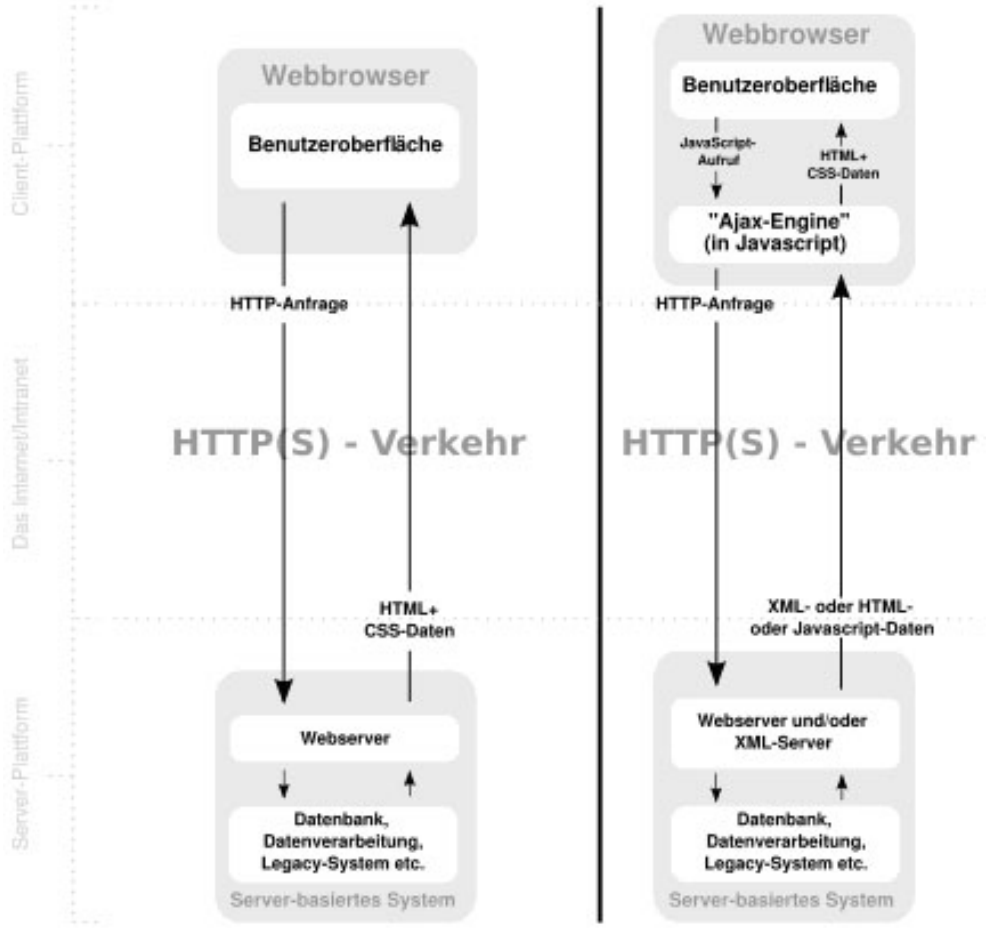

Abb. 3.6 – Transferierte Daten bei Ajax vs. Klassicher Web-Anwendung [Wiki i 2008]

Einer der am häufigsten beklagten Kritikpunkte ist die Funktionalität der „Zurück“ Schaltfläche im Browser. Die Ajax Technologie kann, in dem jetzigen Stand der Implementierung, nicht garantieren, dass durch einen Klick auf dieses Browserelement der davor liegende Zustand wieder hergestellt werden kann. Das liegt daran, dass normalerweise nur statische Seiten im Verlauf des Browsers gespeichert und dynamische Änderungen ignoriert werden. Dies ist für die Entwickler einer Ajax basierenden Web-Anwendung eine Herausforderung, vielfach wird auf unsichtbare IFrames, in denen der Verlaufzustand abgelegt wird, gesetzt. Eine weitere Möglichkeit wäre es, die Zustände am Server abzulegen und die „Zurück“ Knopf Funktion über JavaScript an eine serverseitige Evaluierung des letzten Status umzuleiten. Eine weitere Herausforderung, die in einem ähnlichen Zusammenhang steht ist es, dem Benutzer die Möglichkeit zu geben Lesezeichen auf gewisse Zustände einer Ajax Applikation zu geben. Dazu werden zumeist URL

Fragment Identifier (der Teil einer URL der nach einer # steht) herangezogen, welche dieses Problem beheben [Wiki i 2008] [Wiki j 2008].

Ein anderes Problem das bei der Entwicklung von Ajax Applikation bedacht werden muss ist die Netzwerklatenz. Kann diese nicht als vernachlässigbar angesehen werden, sollten Ladevorgänge dem Benutzer visuell bestätigt werden. Es könnte ansonsten das Gefühl aufkommen, dass die Anwendung sehr träge bis gar nicht mehr reagiert [Wiki i 2008], [Wiki j 2008].

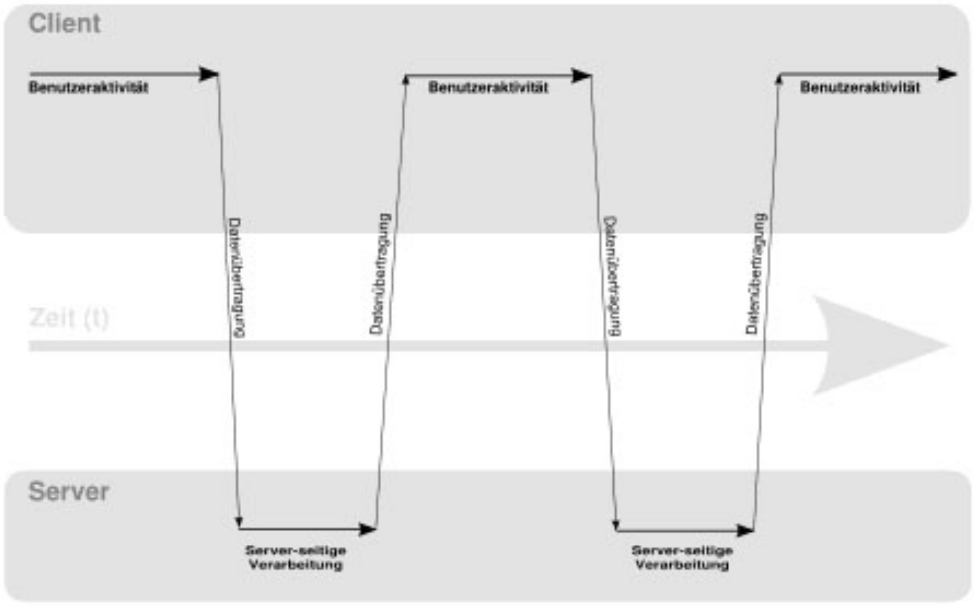

Abb. 3.7 a – Ablauf bei der Client Server Kommunikation in klassischen Web-Anwendungen [Wiki i 2008]

Der Webserver sieht sich einem ganz anderen Problem gegenüber, er soll auf Benutzer Events reagieren. Generell können dies Server aber nur wenn ein eindeutiger Befehl vom Client dafür gesendet wird. Nachdem der Browser aber nicht abschätzen kann wann so ein Event eintritt, müsste er kontinuierlich seinen Status an den Server übermitteln. Da dies das Netzwerk zu sehr belasten würde ist das nicht praktikabel. Im Allgemeinen wird dieses Polling an bestimmten Stellen durch den Entwickler im SC erzwungen. Dazu ist es aber notwendig das der erstellte Thread nach Beendigung der Anfrage für kommende bestehen bleibt, und nicht wie üblich nach einer Request Response Session geschlossen wird [Wiki i 2008].

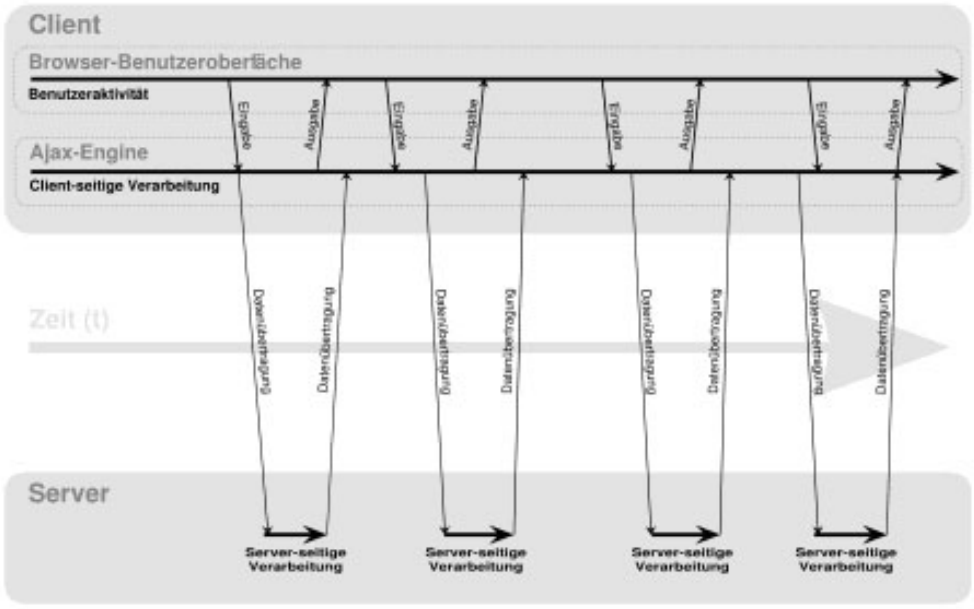

Abb. 3.7 b – Ablauf bei der Client Server Kommunikation bei Ajax Anwendungen [Wiki i 2008]

3.4.2 Rich Ajax Platform Architektur

Zum besseren Verständnis vom RAP ist es wichtig zu Wissen, dass Eclipse auf dem Prinzip des Rich Client aufbaut. Der Rich Client besteht abstrakt betrachtet aus einer Schicht zur Datenverarbeitung, einer darüber liegenden Schicht mit einer Benutzeroberfläche und einer Schicht die es möglich macht zusätzliche Plugins anzudocken.

Auf diesen Ideen baut auch das RAP auf, welche eine Sammlung von logischen und visuellen Elementen für die Entwicklung von Desktopapplikationen mit der Programmiersprache Java zur Verfügung stellt [Wiki h 2008]. Das RAP ist ebenfalls von diesen Ideen inspiriert und von der RCP in Richtung Webentwicklung adaptiert worden. Zusätzlich wurde der schon beschriebene Ajax Mechanismus integriert um eine asynchrone Serverkommunikation zu ermöglichen. Die schematische Architektur der beiden Ansätze ist in Abb. 3.8 gegenübergestellt. Im Folgenden sollen die einzelnen Komponenten des RAP genauer vorgestellt und der Unterschied zum RCP deutlich gemacht werden

Die Abb. 3.8 zeigt, dass der grundlegende Aufbau von RAP und RCP gleich ist. Sowohl RAP als auch RCP wird von den Workbench und JFace Komponenten gestützt. Nachdem der große Unterschied zwischen RAP und RCP darin liegt, das letzteres reine Desktopanwendungen erzeugt, ist auch klar ersichtlich, dass das

SWT zur Darstellung der Elemente im Browser nicht herangezogen werden kann. Bei RAP wurde das SWT dahingehend angepasst, dass Elemente in den Browser gerendert werden können. Dazu gibt es das RWT, was auf Equinox aufbaut und mit Qooxdoo eine JavaScript Komponente besitzt.

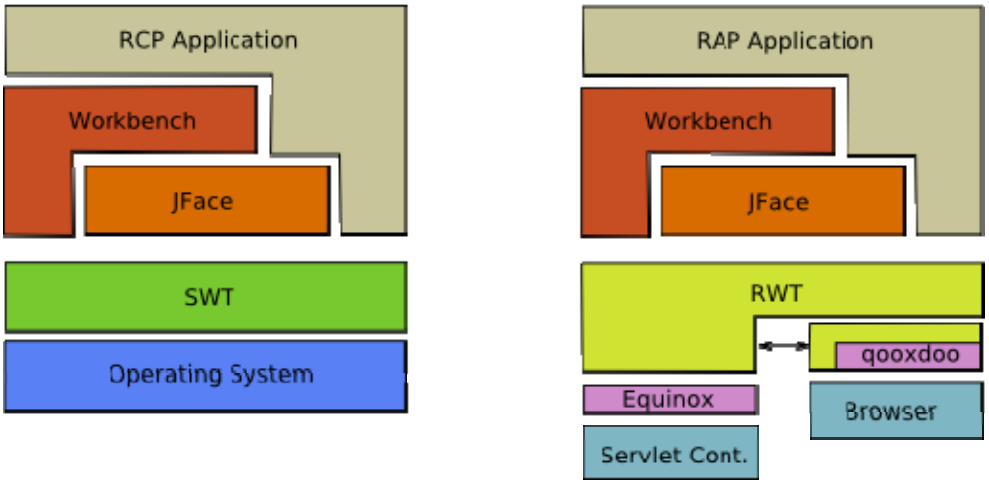

Abb. 3.8 – Architektur von der RAP [RAP Abt 2008]

Ein Punkt der nicht außer Acht gelassen werden darf ist, dass eine RAP Applikation mehrere User bedient, im Gegensatz zu Desktopanwendungen die mit RCP erstellt wurden. Die mit RAP entwickelten Applikationen müssen somit für mehrere gleichzeitige Benutzer optimiert werden [ApKr 2008]. Auf diese Problematik und deren Lösungsansätze soll später noch genauer eingegangen werden. Die folgenden Abschnitte bieten einen grundlegenden Einblick in die aufgezählten Komponenten.

3.4.2.1 Workbench

Der Eclipse Workbench definiert die Struktur des *User Interface* (UI), mit dem Ziel der nahtlosen Integration der unterschiedlichen Tools. Diese können mit so genannten Extension Points, welche durch den Workbench beschrieben werden an die Arbeitsumgebung gekoppelt werden. Zusätzlich zählen auch die Präsentation und die Koordination der Benutzeroberfläche zu den Aufgabenbereichen des Workbench [Wrkb 2008]. Um die Arbeitsumgebung für die Entwicklung von Web-Anwendungen ähnlich strukturiert zu gestalten, wurden Adaptionen an den Workbench Fenstern, den Perspektiven und dem Selektionsservice vorgenommen. Jedoch wurde versucht die grundlegende Struktur des Workbench Konzepts nicht zu stark zu verändern, es mussten jedoch grundlegende Adaptionen bei der

Benutzerverwaltung und den Speicherkonzepten vorgenommen werden [RWT 2008]. Zu den geänderten Speicherkonzepten wird im kommenden Abschnitt zum Thema Equinox noch Stellung genommen.

3.4.2.2 JFace

JFace ist ein spezielles Toolkit, das bei der Umsetzung der UI Programmierung hilft. Grundsätzlich ist JFace für die Zusammenarbeit mit SWT gedacht und beinhaltet Komponenten wie Bilder, Schriften, Dialoge, Fortschrittsanzeigen sowie verschiedene Wizards. Zusätzlich gibt es zwei nennenswerte Features, der Action Mechanismus bietet dem Entwickler die Option objektbezogene Aktionen zu definieren, unabhängig von der Darstellung im UI. Der View Mechanismus ist ein auf einem Modell basierender Adapter für spezielle SWT Widgets zur einfachen Repräsentation von strukturierten Applikationsdaten wie beispielsweise Listen, Tabellen oder Bäume [JFc 2008]. Für die Umsetzung von RIA wurde auf das vorhandene Toolkit aufgesetzt, dadurch entstanden für die Webentwicklung entsprechende Sichten, Dialoge, Aktionen und eine Bilder Registrierung [RWT 2008].

3.4.2.3 RAP Widget Toolkit

Die Notwendigkeit eines Toolkits wie SWT ist der Eclipse Community sehr wichtig, da jedoch die Konzepte nicht zu 100% auf Web-Anwendungen zugeschnitten sind, wurde das RWT entwickelt. Es kombiniert eine Widget Infrastruktur mit einem clientseitigen JavaScript Framework für das Rendering der UI Komponenten. Zum Zeitpunkt der Diplomarbeit sind noch nicht alle Konzepte von RWT umgesetzt, befinden sich aber in ständiger Weiterentwicklung [RWT 2008].

Betrachtet man das RWT genauer, – vergleiche Abb. 3.8 – so baut es auf serverseitigen OSGi und Runtime Komponenten auf, dem so genannten Equinox Framework. Ähnlich zu SWT gibt es eine umfangreiche Widget Bibliothek, welches mit dem Qooxdoo Framework gerendert und im Webbrowser des Client dargestellt wird. Die Konzepte rund um Equinox und Qooxdoo werden in den kommenden Abschnitten präsentiert.

Abgeleitet vom SWT besitzt das RWT nichtsdestotrotz unvermeidliche Unterschiede. Zum einen sind diese aus der verteilten Natur von Web-

Anwendungen hervorgegangen, zum anderen sind sie durch die starre HTML Dokumentstruktur entstanden. An erster Stelle ist das Event Handling zu nennen, denn zum Unterschied von SWT muss der Event zunächst an den Server geleitet werden, bevor dieser bearbeitet werden kann. Eine weitere Problematik ist das synchron halten von Client und Server. Die im UI präsenten Widgets werden serverseitig als Objekte betrachtet, deswegen muss bevor Events am Server ausgeführt werden können eine Synchronisierung zwischen dem Client State und dem Server State durchgeführt werden [RWT 2008].

Um diese gewünschte Funktionalität zu erreichen wird beim RWT der http Request in vier unterschiedliche Phasen unterteilt und sequentiell abgearbeitet. Jede einzelne Phase hat ihre bestimmte Aufgabe und ermittelt die notwendigen Ressourcen für die nächste. Zusammengefasst wird das im RWT Lebenszyklus, der im Hintergrund arbeitet und für den Entwickler zumeist nicht sichtbar ist. Dennoch tragen diese Informationen dazu bei, dass das Event- und Synchronisierungskonzept besser verstanden werden kann. Der Lebenszyklus besteht aus folgenden Phasen [RWT 2008]:

- Prepare UI Root: Ist dafür Verantwortlich, dass Einstiegspunkte aufgerufen werden. Ist mit der Main Funktion aus SWT Applikationen vergleichbar.
- Read Data: Diese Phase holt sich Parameter mit Statusinformationen die zu bestimmten Widgets gehören.
- Process Action: Das ist die jene Phase wo die geforderte Interaktion des Users als Aktion am Server ausgeführt wird. Dazu ist es dem Programmierer möglich verschiedene Event Listener für die Widgets zu implementieren.
- Render: In dieser Phase wird der neue JavaScript Code erzeugt und an den Browser gesendet.

Die konkreten Widgets einer RAP Anwendung müssen so gebaut sein, dass sie einer einheitlichen API folgen und zusätzlich transparent mit dem Lebenszyklus kommunizieren können. Um dem Entwickler die Umsetzung zu ersparen werden für jedes Widget eigene Adapter kreiert die diese Aufgaben erfüllen. In der Instanz eines Widgets werden jene Informationen gespeichert die durch den Entwickler mittels des API gesetzt worden sind. Als nächsten Schritt erhält das Widget einen Adapter, der zusätzliche Informationen für den Server enthält. Beispielsweise kann

dieser Adapter Information beinhalten zu welchem Zeitpunkt ein Wert an den Client gesendet werden soll. Schlussendlich erhält jedes Widget eine gemeinsame Instanz eines Lebenszyklus Adapters. Dieser hat die Verantwortung das Widget auf dem Server mit Änderungen auf der Seite des Clients synchron zu halten. Der Adapter dient auch dazu, dass man dem Client in Abhängigkeit der Browserfähigkeiten, Seiten entweder in Ajax oder im klassischen HTML / CSS retour sendet [RWT 2008].

3.4.2.4 Qooxdoo

Das Qooxdoo Framework liefert Entwicklern einen innovativen Ansatz um Ajax Applikationen für unterschiedliche Browser, mit objektorientiertem JavaScript, zu erstellen. Dieses OS Projekt, welches 2005 gestartet wurde, vereint gleichermaßen ein vielfältiges UI Toolkit als auch eine optimierte Client Server Kommunikation. Innerhalb von RAP wird Qooxdoo für das Rendering der RWT Komponenten im Webbrowser herangezogen, wobei die verwendeten Widgets über benutzerdefinierte Events gesteuert werden können. Zurzeit wird dieses clientseitige JavaScript Framework von Firefox, Internet Explorer, Oper und Safari unterstützt [Qxd 2008], [Wiki k 2008].

Um einer RIA das Look & Feel einer Desktopanwendung zu geben, bietet Qooxdoo die Möglichkeit unterschiedliche Themes zu erstellen, nicht nur um der Anwendung ein spezielles Design zu verpassen, sondern vor allem um den Benutzer eine gewohnte bedienerfreundliche Umgebung zu liefern [Wiki k 2008]. Zusammengefasst ermöglicht Qooxdoo die Entwicklung von Web-Anwendungen mit der Unterstützung von plattformunabhängigen Werkzeugen auf Basis von JavaScript. Die Ajax Komponente liefert dazu noch eine performante Client Server Kommunikation zum optimierten asynchronen Laden von Informationen [Ecke 2008].

3.4.2.5 Equinox

Equinox ist ein auf Java basierendes OSGi Framework und integraler Bestandteil des Eclipse Projektes. Betrachtet man das Framework von der SC Seite, so ist es eine genaue Umsetzung der OSGi Kernspezifikation. Damit ist es möglich Services zu entwickeln die auf OSGi aufbauen [Eqx 2008], [Wiki l 2008]. In diesem Zusammenhang soll prägnant die Idee hinter OSGi erklärt werden.

Die OSGi Technologie ist eine dynamisches System von Modulen für die Programmiersprache Java. Mit anderen Worten bietet diese Technologie standardisierte Ansätze damit Applikationen aufgrund von kleinen wieder verwertbaren kollaborativen Komponenten aufgebaut werden können. Mittels einer *Service Oriented Architecture* (SOA) besteht die Möglichkeit, dass sich die einzelnen Module zu Arbeitsgemeinschaften zusammenschließen können. Auf die bekannten Vorteile von modularen Strukturen und SOA soll in diesem Kontext nicht weiter eingegangen werden [OSGi 2008], [Wiki m 2008].

Allgemeiner betrachtet ist das Ziel des Equinox Projektes einerseits, eine erstklassige OSGi Community zu bieten, aber auch die Vision von Eclipse als eine Plugin Landschaft voranzutreiben. Um dem gerecht zu werden, ist Equinox für die komplette Entwicklung und die Auslieferung des OSGi Frameworks verantwortlich. Abseits von Eclipse arbeitet dieses Projekt an folgenden Themen [Eqx 2008]:

- Implementierung aller Aspekte der OSGi Spezifikation
- Forschung und Entwicklung an zukünftigen OSGi Versionen
- Entwicklung von Infrastrukturkomponenten die für die Verwaltung und die Stabilität von OSGi basierenden Systemen notwendig sind

Im Zusammenhang mit dem RAP steuert Equinox das serverseitige OSGi und die Laufzeitpakete bei. Für jede Web-Anwendung gibt es aus Speichergründen nur genau eine OSGi Instanz, die sich die Benutzer untereinander teilen müssen. Das ist auch der im Kapitel Workbench angesprochene Unterschied zu SWT, wo jedem User eine Instanz zugeteilt wird. In der verteilten Struktur des Internet, mit einer unberechenbaren Anzahl von Usern, würde dies zu massiven Speicherproblemen führen [RWT 2008].

4 Technologische Bewertung von GMF und RAP

Dieses Kapitel soll den theoretischen Teil dieser Diplomarbeit abschließen, indem es eine Bewertung der beiden vorgestellten Eclipse Technologien GMF und RAP vornehmen soll. Bei dieser Technologiebewertung handelt es sich um vorgegebene Fragestellungen[12], die jeder Technologie separat gestellt werden. Mit anderen Worten werden sowohl das GMF als auch das RAP isoliert begutachtet, eine Analyse der Interoperabilität findet man im Kapitel 5.5. Durch diese Herangehensweise lässt sich ein Überblick bieten, der Aufschluss darüber geben kann, in welche Richtung sich die jeweilige Technologie entwickeln könnte. Im Rahmen der Diplomarbeit ist dieses Kapitel deswegen wichtig, weil Technologien nicht nur vorgestellt und eingesetzt werden, sondern deren Wesen kritisch hinterfragt wird, was im Großen und Ganzen ein abgerundetes Bild von GMF und RAP liefern soll.

Als ersten Schritt visualisiert die Tab. 4.1 die gewonnenen Ergebnisse, die dazugehörenden Begründungen werden darauffolgend geliefert. Dabei werden zunächst die Fragestellungen vorgestellt und danach jeweils für das GMF und das RAP einzeln beantwortet. Es wurde versucht jeder Antwort eine Referenz aus der Literatur beizufügen, bei einigen Punkten würde die Analyse jedoch den Rahmen der Diplomarbeit überschreiten. Beispielsweise würde eine Recherche, an welchen zentraleuropäischen[13] Universitäten diese Technologien gelehrt werden, alleine schon Monate dauern. Deswegen wurden diese Fragen auch Experten aus Industrie und Forschung, die mit diesen Technologien vertraut sind, vorgelegt.

Fragestellung	GMF	RAP
Verbreitung	mäßige Verbreitung	geringe Verbreitung
Lehre an Universitäten	wird teilweise gelehrt	wird nicht gelehrt
Produktivität	produktiv	sehr produktiv
Community	große Communities	große Communities
Performance	gerade ausreichend	gerade ausreichend

[12] Diese Fragen wurden von Dr. Harald Kühn, Mitglied der Geschäftsführung von BOC Information Systems GmbH, zur Verfügung gestellt.

[13] Interessanter wäre sicherlich eine weltweite Analyse, das allein könnte aber schon eine eigene Arbeit füllen.

Skalierbarkeit	etwas skalierbar	gut skalierbar
OS Unabhängigkeit vom Client	Ja	Ja
OS Unabhängigkeit vom Server	Ja	Ja
Standards	Ja	Ja
Lebenszyklus	Die Technologie wird die nächsten 10 Jahre maßgebend sein.	Die Technologie wird die nächsten 10 Jahre maßgebend sein.
Innovationen	Eventuell geringere Änderungen in der Zukunft, die aber positiver Natur sein werden.	Technologie ist noch sehr jung und wird wahrscheinlich noch vielen Veränderungen unterworfen sein.
Kosten	keine	keine
Lernkurve	sehr hoher Aufwand	geringer Aufwand
Multi-Threading	Ja	Ja
Offline-Fähigkeit	Ja	Nein
Abhängigkeit von Drittkomponenten	Ja	Ja
Integrierbarkeit	Mit Aufwand integrierbar	Mit Aufwand integrierbar
Usability	Viele Usability Features	Keine Usability Features
Entwicklungsunterstützung	IDE mit grafischem Editor vorhanden	IDE mit grafischem Editor vorhanden
Homogenität	Erträgliche Homogenität	Erträgliche Homogenität

Tab. 4.1 – Ergebnisse der Technologiebewertung

4.1 Verbreitung

Die Frage zielt darauf ab, inwieweit diese Technologien bereits verbreitet sind. Bei Technologien mit geringer Verbreitung besteht die Gefahr, dass diese nicht weiterentwickelt und somit eingestellt werden. Die für diese Technologien getätigten Investitionen würden somit verloren gehen.

GMF

Das gezeigte Ergebnis einer mäßigen Verbreitung wird durch folgende Aussagen begründet. Das GMF erfährt große Unterstützung von IBM und Borland, die auch die meisten und größten Komponenten, seit der ersten Version im Juni 2006 beigesteuert haben [GMF Dash 2008]. Die beiden Produkte Acceleo[14] und Obeo[15] sind kommerziell genutzte Applikationen, die auf das GMF aufbauen, zusätzlich gibt es mehrere kleinere Projekte, wie eine kleine Internetstudie ergab [Mosk 2008], [HeLLC 2008]. Einen interessanten Aspekt lieferte die Auswertung der Herkunft der beigesteuerten GMF Projekte und Erweiterungen, nämlich dass es eine starke Konzentration von GMF Benutzern in Europa gibt [Ohloh 2008].

RAP

Versucht man sich mit einer webgestützten Suche nach RAP Projekten, stößt man in den meisten Fällen nur auf kleine Demos und Prototypen [RAP Dash 2008]. Begründet kann dieser Umstand damit werden, dass die Version 1.0 vor nicht einmal einem Jahr, nämlich im Oktober 2007 erschienen ist [RAP Abt 2008]. Das Projekt erfährt zwar eine große Unterstützung durch die Unternehmen Innopract[16], 1&1[17] und CAS[18], jedoch befindet es sich immer noch in der Aufbauphase mit vielen fehlenden Features. Diese Umstände könnten viele interessierte Entwickler noch von dem Einsatz des RAP abhalten und führen zu dem Ergebnis einer noch geringen Verbreitung.

4.2 Lehre an Universitäten

Der Hintergedanke dieser Fragestellung ist es, herauszufinden, wie einfach es ist kompetente Entwickler mit den entsprechenden Kenntnissen zu gewinnen. Werden die Technologien an einer großen Anzahl an Universitäten gelehrt, kann das ein guter Indikator dafür sein.

[14] http://www.acceleo.org

[15] http://www.obeo.fr

[16] http://www.innoopract.com

[17] http://www.oneandone.com

[18] http://www.cas.de

GMF

Einige Institute darunter das BIG auf der TU Wien, bieten Lehrveranstaltungen an, in denen auf MDSD Aspekte eingegangen werden. Mitunter wird das GMF als beispielhafte Anwendung erwähnt. Es ist aber zu beobachten, dass sich einige Diplomarbeiten mit dieser Technologie auseinandersetzten [BIG 2008]. Das liefert die Bewertung, dass das GMF teilweise an Universitäten gelehrt wird.

RAP

Kennt man Universitäten und Lehrpläne im Allgemeinen, kann man davon ableiten, dass diese durch einen langwierigen Prozess gewonnen werden. Dadurch wird klar, dass junge Trends wie das RAP es nicht in die Lehre schaffen. Da das RAP aber aus unterschiedlichen Konzepten aufgebaut ist und diese vereint, ist davon auszugehen, dass diese bestimmt an den Universitäten gelehrt werden. Dennoch beziehen sich diese nicht auf das RAP im Speziellen und somit kann man zu dem Urteil kommen, dass diese Technologie nicht gelehrt wird.

4.3 Produktivität

Die Frage nach der Produktivität kann auch mit anderen Worten beschrieben werden, nämlich, ob man mit diesen Technologien schnell Ergebnisse erzielen kann. Bei Programmiersprachen gibt es bereits diesbezügliche Studien um die unterschiedliche Produktivität herauszufinden.

GMF

Die Produktivität des GMF kann durch folgende Aussagen geklärt werden: Prinzipiell erhält man rasch einen lauffähigen Editor, wenn man ein in XML vorliegendes Grundmodell zur Verfügung hat. Möchte man diesen Editor erweitern bzw. verändern, bedeutet dies, dass man in die Generierung der XML Modelle eingreifen muss. Wie produktiv dies vonstatten geht, ist abhängig vom individuellen Wissenslevel. Um die erstellten Änderungen in den Editor zu übernehmen, genügt ein Klick und der Editor wird neu generiert. Allgemein kann man von einer produktiven Technologie sprechen.

RAP

Das Ergebnis welches zeigt, dass diese Technologie sehr produktiv ist, wird durch folgende Erkenntnisse gestärkt. Das Framework schottet den Entwickler von den

eigentlichen Webkonzepten HTML, CSS, JavaScript und Ajax ab und übernimmt die Generierung derselben. Nachdem diese Technologien ihre Wurzeln in der RCP Entwicklung haben, kann das damit erlangte Wissen zumeist zu 100% wiederverwendet werden, da sich ein Eventhandler prinzipiell nicht unterscheidet, egal ob er einer RCP Anwendung oder einer RAP Anwendung zugeordnet ist. Das RAP selbst trifft für den Entwickler diese Entscheidung und produziert den dafür erforderlichen SC der Webseite.

4.4 Community

Welchen Rückhalt haben diese Technologien durch gemeinsame Entwicklergemeinschaften? Gibt es beispielsweise Internet Communities, welche die Weiterentwicklung und das Bug Fixing der Technologien vorantreiben? Dadurch können schneller Lösungen bei auftretenden Fragestellungen gefunden werden.

GMF / RAP

In diesem Fall kann die Bewertung für beide Technologien gemeinsam getätigt werden, da es sich bei dem GMF und dem RAP um Technologien handelt, die den gleichen Hintergrund haben, nämlich das Eclipse Projekt. Eclipse ist aber gerade als Paradebeispiel einer bestens funktionierenden, starken und durchsetzungsfähigen Community zu nennen. Mit einem starken Rückhalt, wie der Eclipse Gemeinschaft haben sich die beiden Technologien ein gutes Fundament für die Zukunft gebaut.

4.5 Performance

Die Frage nach der Performance ist vor allem für den kommerziellen und industriellen Einsatz von großer Bedeutung und zielt darauf ab ob die Technologien hinreichend performant für die Entwicklung und den Einsatz großer, industrieller Projekte sind.

GMF

Prinzipiell kann ausgesagt werden, dass Eclipse in einem kommerziellen Umfeld zum Einsatz kommt und Themen wie die Performance eine große Rolle spielen. Im Detail werden die erstellten Modelle des GMF reibungslos und in einer akzeptablen Zeit in SC umgesetzt, Probleme treten im Editor erst dann auf, wenn

Modellierungen mit ihren Elementen sehr groß werden. Dabei ist dann zu beobachten, dass es lange Ladezeiten beim Starten des GMF Editors gibt und das Modifizieren der Objekte im Editor nicht einwandfrei vonstatten geht [ECL Forum a 2008].

RAP

Beobachtet man die RAP News [RAP Note 2008] und die betreffenden Foren, so ist erkennbar, dass die Performance in der Task Liste nicht an oberster Stelle steht und mitunter bis dato vernachlässigt wurde. Eine so junge Technologie wie das RAP steht unter gewaltigem Druck, zunächst einmal alle gewünschten Features abzudecken, und wird sich aller Voraussicht erst später mit diesem Thema befassen. Zusätzlich hängt die Performance bei einer Web-Anwendung nicht allein von der am Server ausgeführten Komponente ab, sondern auch von der Latenz, dem verwendeten Browser und dessen möglicherweise vorhandenen Speicherlöcher.

4.6 Skalierbarkeit

Die Frage nach der Skalierbarkeit steht in engem Zusammenhang mit der zuvor getätigten Untersuchung nach der Performance. Hierbei ist es wichtig herauszufinden wie diese Technologien skaliert werden können um eventuelle Anforderungen an die Performance zu erfüllen.

GMF

Über die Skalierbarkeit des GMF kann ausgesagt werden, dass eine prinzipielle Skalierbarkeit vorhanden ist, das aber mit größer werdenden Modellen die Speicherauslastung zu einem Problem werden kann. Illustriert kann dieses Phänomen durch folgendes Beispiel werden, indem man davon ausgeht, dass ein Modell mit 40 Bücherregalen mit jeweils 100 Büchern existiert. Dabei wird jedes Element als Java `EObject` im Heap Speicher abgelegt. Das bedeutet aber, dass 4000 Objekte plus deren Verbindungen einzeln im Speicher abgelegt werden. Das jedes Objekt separat abgelegt wird, sollte bei der Entwicklung mit dem GMF nicht außer Acht gelassen werden [ECL Con 2008], [ECL Forum b 2008].

RAP

Nachdem das RAP Web-Anwendungen (mit vielen gleichzeitigen Benutzern) konzipiert, deren Interaktivität am Server gesteuert wird, war die Skalierbarkeit von Beginn an ein großes Thema [RAP Rel 2008]. Die gute Skalierbarkeit soll ebenfalls anhand eines Beispiels demonstriert werden. Bei 500 parallelen Usern konnte bei einer Demoanwendung eine stabile Speicherauslastung von ca. 100 MB am Server festgestellt werden [RAP Scale 2008]. Dadurch kann diese Technologie als gut skalierbar angesehen werden.

4.7 Betriebssystemunabhängigkeit vom Client

Hierbei handelt es sich schlichtweg darum, ob diese Technologien für mehrere Client BS konzipiert sind oder ob sie zielgerichtet für eine Plattform entwickelt wurden.

GMF / RAP

Bei beiden Technologien handelt es sich um Eclipse Projekte, die zur Gänze auf Java basieren und somit diese Frage schon beantworten. Die Plattformunabhängigkeit der Programmiersprache Java ermöglicht es auch den beiden Technologien für unterschiedliche BS zur Verfügung zu stehen [ECL Dwnld 2008]. Beim RAP im Speziellen handelt es sich um eine Web-Technologie, die nur HTML, CSS und Ajax produziert, was ein weiteres Kriterium der Plattformunabhängigkeit ist.

4.8 Betriebssystemunabhängigkeit vom Server

Die Frage nach der Unabhängigkeit von Server BS ist ähnlich der zuvor untersuchten Umstände nach Client BS. Sind diese Technologien für unterschiedliche Server BS konzipiert, oder sind sie zielgerichtet für eine Plattform entwickelt worden.

GMF / RAP

Ähnlich bei der Untersuchung der Clientunabhängigkeit kommt man auch für die Server zum selben Schluss. Auf den Servern wird reiner Java Code ausgeführt somit gibt es keine Probleme bei der Portierung auf ein anderes Server BS.

4.9 Standards

Bezüglich der Standards sucht man nach der Antwort ob es welche gibt, oder ob man mit diesen Technologien von anderen Herstellern abhängig ist. Die umgesetzten Standards sollen an dieser Stelle nur genannt, nicht aber deren Funktionsweise erklärt werden.

GMF

Nachdem es sich bei dieser Technologie um ein Eclipse Projekt handelt erfolgt die primäre Entwicklung in Java. Zusätzlich werden noch UML 2 und XML als Standardtechnologien eingesetzt.

RAP

Diese Technologie verhält sich ähnlich wie die zuvor beschriebene, die grundlegende Entwicklung erfolgt in Java. Des Weiteren werden die Standard Web-Technologien – HTML, CSS, Javascript und Ajax – sowie das serverseitige OSGi für die Erstellung von Web-Anwendungen verwendet.

4.10 Lebenszyklus

Welche Erwartungen an die Lebensdauer liegen hinter den Technologien? Anders ausgedrückt, sind diese Technologien auch noch in 10 Jahren in der Entwicklung präsent, oder wurden sie schon von anderen verdrängt?

GMF

Man kann davon ausgehen, dass diese Technologie über die nächsten Jahre hinweg bestehen kann. Begründet wird diese Aussage mit dem schon bis Ende 2009 vorliegenden detaillierten Entwicklungsplan für den nächsten Release [GMF Plan 2008]. Zusätzlich kann man einen weiteren Trend erkennen, nämlich einen Anstieg von Editoren für unterschiedliche Domain Specific Language, welche die Nutzung dieser Technologie fördern kann [ECL Mod 2008]. Auch ein Interview mit dem Projektleiter des GMF, Richard Gronback, ist ebenfalls ein Indikator, dass diesem Framework eine lange Lebensdauer bevorsteht. Das nächste große Ziel ist es eine Webunterstützung für die Editoren zu erzielen [Gronb 2008].

RAP

Auch dieser Technologie kann man eine lange Lebensdauer zugestehen, zum Einen, weil der Begriff des Web 2.0, welcher unter anderem interaktive Webseiten beschreibt, noch sehr jung ist und sich in diesem Umfeld bestimmt interessante Projekte finden lassen. Zum Anderen sind auf Java basierende Ajax Toolkits, wie es das RAP ist, eine Erleichterung für die Entwickler, weil nicht alle einzelnen Webkomponenten manuell programmiert werden müssen. Auch der langfristige Projektplan dieser Technologie zeigt, dass diese noch im Aufbau ist [RAP Plan 2008].

4.11 Innovation

Die Innovation von Technologien kann auch anders umschrieben werden, handelt es sich schon um gefestigte und ausgereifte Technologien, oder sind noch viele dynamische Weiterentwicklungen zu erwarten?

GMF

Das GMF baut mittlerweile auf ein gefestigtes Fundament auf und sieht in naher Zukunft weitere Optimierungen des Editors und die Erweiterung des Frameworks in Richtung dem Web vor [GMF Plan 2008]. Letzteres besitzt bestimmt einen hohen innovativen Charakter und wird das GMF an sich selbst wettbewerbsfähiger werden lassen.

RAP

Wie schon bei anderen Punkten angesprochen, handelt es sich bei dem RAP um eine noch sehr junge Disziplin, die sich noch in stetiger Weiterentwicklung befindet. Vor allem weil immer noch nicht alle gewünschten Features in der derzeitigen Version vorhanden sind, wie eine komplette SWT Umsetzung auf Ajax oder ein ausgereiftes Eventhandling für die Benutzerinteraktion [RAP Plan 2008].

4.12 Kosten

Welche Kosten, vor allem in welcher Höhe, sind zu erwarten wenn man diese Technologien in einem kommerziellen Umfeld einsetzen möchte? Die Kosten sind bezogen auf eventuelle Lizenzen oder Royalties und nicht auf das einzusetzende Kapital, beispielsweise für die Bezahlung von Mitarbeitern.

GMF / RAP

Beide Technologien haben ihre Wurzeln in dem OS Umfeld, für die Nutzung entstehen somit für die Entwickler keinerlei Kosten. Die Verwendung ist aber wie schon erwähnt an die Eclipse GPL gebunden, die auch eingehalten werden muss.

4.13 Lernkurve

Die Fragestellung nach der Lernkurve hinterfragt, mit welchem individuellen Einsatz diese Technologien zu erlernen sind?

GMF

Wie schon bekannt ist, baut das GMF auf dem EMF und GEF auf, die einem komplexen Ansatz folgen. Zusätzlich werden XML und UML Kenntnisse benötigt, die für die Erstellung des Inputmodells erforderlich sind. Es kann mit einer langen Einarbeitungszeit gerechnet werden, wenn keinerlei Wissen über die dem GMF zugrunde liegenden Frameworks vorhanden ist, da diese Technologie eigenen Mechanismen und Konzepten zur Generierung eines Editors folgt.

RAP

Diese Technologie kann mit einem geringen Aufwand erlernt werden. Sie beinhaltet zwar unterschiedliche Web-Technologien, diese müssen aber zur Verwendung vom RAP nicht erlernt werden. Die Umsetzung erfolgt mit Java und dem vom SWT abgeleiteten RWT, wobei hier noch nicht viele Objekte für die Verwendung im Browser umgesetzt sind.

4.14 Multi-Threading

Inwieweit unterstützen diese Technologien Applikationen, die auf einer Multi-Threading Architektur aufgebaut sind? Dieses Kriterium ist eng verknüpft mit den Fragestellungen zur Performance und der Skalierbarkeit. Mit Hilfe einer ausgeprägten Multi-Threading Fähigkeit können nicht nur extrem performante, sondern auch außerordentlich interaktive Anwendungen realisiert werden.

GMF / RAP

Dadurch, dass beide Technologien den Ursprung in derselben Programmiersprache haben, kann diese Fragestellung wieder sowohl für das GMF als auch für das RAP gleichzeitig beantwortet werden. Die Programmiersprache

Java wurde von Grund auf zur Multi-Threading Fähigkeit hin konzipiert und stellt dafür optimierte Konzepte und Klassen zur Verfügung [DPunkt 2008]. Speziell für das RAP ist diese Eigenschaft notwendig, da eine Web-Anwendung in der Regel viele Benutzer gleichzeitig bedienen muss, die Parallelisierung kann dadurch zum Vorteil ausgenutzt werden.

4.15 Offline-Fähigkeit

Können diese Technologien sowohl in einem Online als auch in einem Offline Umfeld genutzt werden? Dieser Aspekt erlangt vermehrt an Bedeutung, da der Einsatz von autonomen mobilen Geräten immer mehr ansteigt.

GMF

In der aktuellen Version dieser Technologie, kann die SC Generierung ein Eclipse Plugin oder eine RCP Applikation erzeugen. Letztere muss sich in von Zeit zu Zeit mit dem Server synchronisieren um arbeiten zu können. An dieser Stelle müsste genauer untersucht werden, ob in die Generierung in einem solchen Maße eingegriffen werden kann, dass ein praktikables Offline arbeiten möglich ist [GMF RCP 2008]. Wird jedoch ein Plugin für die Eclipse IDE erzeugt, so ist dieses nur Offline nutzbar, man kann somit von einer Offline Fähigkeit dieser Technologie sprechen.

RAP

Mit Hilfe des RAP können komplexe interaktive Web-Anwendungen geschaffen werden. Dieser Umstand alleine bestätigt schon, dass eine Offline Fähigkeit, in einem solchen vernetzten Umfeld, in dem diese Technologie agiert, keinen Vorteil bringen kann und somit auch nicht zur Verfügung steht.

4.16 Abhängigkeit von Drittkomponenten

Welche Voraussetzungen müssen diese Technologien erfüllen, damit man überhaupt mit ihnen arbeiten kann? Genauer ausgedrückt, welche Komponenten von Drittherstellern müssen installiert sein, damit die Technologien eingesetzt werden können (beispielsweise ein bestimmter Browser, Datenbanken, usw.)?

GMF

Das GMF benötigt als einzige Grundlage die Eclipse Plattform, in die diese Technologie integriert werden kann. Solange ein Modell Editor als Plugin läuft, werden keine weiteren Komponenten benötigt, für die Realisierung als eine RCP wird eine entsprechende Server Software notwendig – beispielsweise ein Apache Tomcat.

RAP

Auch das RAP benötigt als Basis die Eclipse Plattform und eine entsprechende Server Software um die Verarbeitung der Benutzerinteraktion mit der Applikation durchführen zu können. Zusätzlich wird ein Ajax kompatibler Browser benötigt. Laut RAP Homepage [RAP Plan 2008] zählen zurzeit folgende dazu:

- Internet Explorer 7.0 oder höher
- Gecko Engine (Firefox, Netscape)
- Webkit Engine (Safari, Google Chrome)

4.17 Integrierbarkeit

Können die gewählten Technologien in andere integriert oder mit anderen erweitert werden? Wenn ja, welche Mechanismen bieten diese Technologien dafür an?

GMF

Das GMF an sich integriert schon bestehende Technologien wie das EMF und das GEF, darüber hinaus gibt es die Möglichkeit kommerzielle Produkte, wie die schon erwähnten Acceleo und Obeo, zu implementieren, welche GMF Editoren integrieren. Eine hoher Wissenstand über die SC Generierung und die Details des GMF wird aber in jedem Fall vorausgesetzt.

RAP

Auch diese Technologie hat eine beachtliche Anzahl unterschiedlicher Technologien integriert. Sie vereint Java mit den Web-Technologien. Des Weiteren gibt es die Möglichkeit externe JavaScript Frameworks für erweiterte Darstellungsoptionen einzubetten. Interne Java Schnittstellen, wie beispielsweise zu dem SWT sind auch vorhanden, obwohl hierbei noch nicht alle Objekte adressierbar sind.

4.18 Usability

Welche Möglichkeiten bieten diese Technologien an, die Benutzerfreundlichkeit der entwickelten Anwendung zu unterstützen oder sogar maßgebend zu erhöhen?

GMF

Bei der Usability vom GMF müssen zwei Seiten unterschieden werden. Zum Einen den theoretischen Aufbau der Modelle, die als Grundlage für den Editor dienen und zum Anderen der Editor selbst. Erstere beschäftigt sich mit den in XML erstellten Modellen, die aber durch einen grafischen Baum editierbar sind. Man muss also nicht manuell in den XML Code eingreifen. Mit dem Editor selbst wird nur mehr über eine grafische Umgebung gearbeitet, ein Eingriff in den SC ist an dieser Stelle nicht mehr nötig.

RAP

Die junge RAP Technologie beschränkt sich zurzeit noch auf simple Java Editoren, in denen der Java SC manuell programmiert werden muss. Momentan gibt es auch nicht die Möglichkeit die grundlegenden Elemente einer Web-Anwendung (Buttons, Textfelder, Bilder, …) über einen Designmechanismus aufzubauen. Der dafür notwendige Java Code muss vom Entwickler eigenständig erstellt werden.

4.19 Entwicklungsunterstützung

Welche Unterstützung wird dem Entwickler geboten, wenn er mit diesen Technologien arbeitet? Anders formuliert, stehen spezielle Entwicklungsumgebungen und Tools zur Verfügung, welche die Entwicklungsproduktivität positiv beeinflussen?

GMF / RAP

Die Eclipse IDE wird für beide Technologien als Entwicklungstool eingesetzt. Zum Einen wird dadurch eine nahtlose Integration der Technologien in die IDE garantiert. Zum Anderen ergeben sich dadurch Vorteile, die man durch den Einsatz von mächtigen Entwicklungsumgebungen gewohnt ist (Refactoring, Versionskontrolle, Automatisierungsmechanismen, usw.).

4.20 Homogenität

Welchen Homogenitätsgrad erreichen die Technologien? Mit anderen Worten, sind sie ein Bündel aus verschiedenen Ansätzen, oder handelt es sich um einheitliche Technologien?

GMF

Die Homogenität von dieser Technologie ist von dem Standpunkt abhängig, von dem man sie betrachtet. Um die Arbeit mit dem GMF überhaupt zu ermöglichen sind die schon zuvor erwähnten Basistechnologien notwendig. Diese bilden aber auch die Standards in dem Umfeld der MDSD. Die Bewertung siedelt sich somit in der Mitte an und vergibt dem GMF eine erträgliche Homogenität.

RAP

Ähnlich wie bei dem GMF ist auch bei dem RAP der Betrachtungswinkel wichtig. Diese Technologie vereint, völlig transparent vor dem Benutzer, viele Web-Standards um mächtige interaktive Web-Anwendungen zu schaffen. Ohne dieses Mash Up könnte das RAP aber seine Aufgaben nicht erfüllen, was aber wiederum auf Kosten der Homogenität geht.

5 Implementierung eines Web-Modelleditor Prototyps

Mit diesem Kapitel beginnt die Diplomarbeit die praktische Umsetzung des zuvor erarbeiteten theoretischen Fundaments. In Folge wird schrittweise der Prototyp eines Web-Modelleditors entwickelt, wie es die praktische Aufgabenstellung der Diplomarbeit vorsieht. Zu Beginn soll ein simples Beispiel gefunden werden, dass das zugrunde liegende Modell beschreibt, gefolgt von einer konkreten Zielsetzung, mit der das abschließende Ergebnis verglichen werden kann. Im Anschluss werden die Details in der Konzeption und der Implementierung sowohl von dem GMF und dem RAP veranschaulicht. Abgeschlossen wird dieses Kapitel mit einer eingehenden Analyse über die Interoperabilität der beiden Technologien.

5.1 Das Mindmap Modell

Die Implementierung des Prototyps umfasst die Realisierung eines einfachen Mindmap Editors. Eine Mindmap ist eine Gedächtnis- oder Gedankenkarte, die helfen soll gesammeltes Wissen kompakt und in einer strukturierten Form darzustellen. Im Gegensatz dazu steht das Brainstorming Modell, wobei hier das Wissen in einem zunächst unübersichtlichen Chaos in Form von einzelnen Elementen vorliegt [Wiki n 2008]. Um ein wenig mehr ins Detail zu gehen, ist es nützlich, die Abb. 5.1 zur Veranschaulichung heranzuziehen. Klar zu erkennen ist das zentrale Element, welches das eigentliche Hauptthema darstellt. Ausgehend von diesem spannt sich ein Netz von Unterthemen mit konkreten Spezifikationen auf. Anders ausgedrückt besitzt das Hauptthema Assoziationen zu Unterthemen, welche wiederum eigene detailreichere Unterthemen assoziieren können. Durch dieses Konzept lassen sich weitschichtige Karten aufbauen.

Abb. 5.1 zeigt eine funktionelle und übersichtliche Mindmap, die dabei helfen kann Ideen zu sammeln, sowie Protokolle zu erstellen. Weiters können sie bei der Strukturierung von wissenschaftlichen Arbeiten und Vorträgen von großem Wert sein. Die in Abb. 5.1 dargestellte Struktur dieser Diplomarbeit war eine hervorragende Grundlage und verhinderte, dass jenes zu Beginn gesteckte Ziel über den Verlauf der Arbeit nicht aus den Augen verloren wurde.

Das Mindmap Modell eignet sich aber auch aus weiteren Gründen für die Entwicklung eines webgestützten Modelleditors. Die simple zugrunde liegende

Logik des Modells, bestehend aus Thema, Subthema und Verbindung zwischen diesen beiden, hilft die Konzentration auf die komplexen Technologien GMF und RAP zu lenken. Das bedeutet, dass man nicht schon beim Verständnis des Modells an geistige Grenzen stößt. Des Weiteren hat es sich angeboten die Gedächtniskarte als Beispiel heranzuziehen, da es innerhalb der Eclipse Community eine Vielzahl an Tutorials, Beispielen und Hilfethemen zu diesem Modell gibt, die in kniffligen Situationen eine erlösende Hilfe waren.

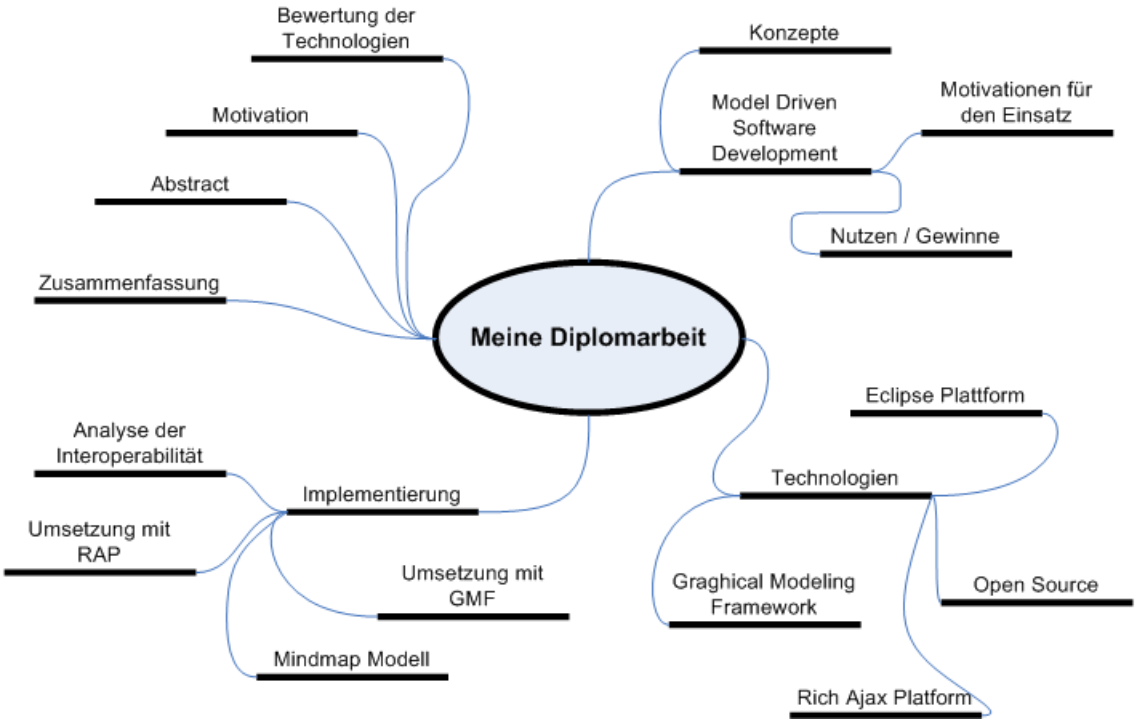

Abb. 5.1 – Mindmap zu dieser Diplomarbeit (MS Visio)

5.2 Zielsetzung

Dieser Abschnitt widmet sich eingehend der genauen Beschreibung des Prototyps, der im Zuge dieser Diplomarbeit entwickelt werden soll. Eingangs wurde gewünscht, dass ein Modell als Editor in einen Webbrowser integriert werden soll. Um dieses Ziel zu erreichen, ist es zunächst notwendig, das zuvor beschriebene Mindmap Modell mittels dem GMF als eine Eclipse Anwendung zu konzipieren. Dadurch erhält man die Kontrolle über die zugrunde liegenden Modelle, die für die Generierung des XML Codes der grafisch modellierten Mindmap notwendig ist. Mit diesem Ergebnis ausgestattet soll mit dem RAP ein interaktives Web-Frontend entwickelt werden, welches die Modellierung einer Mindmap im Browser zulässt. Dazu werden die Daten zum Server transferiert, diese in das Modell einspeist und

das generierte XML im Browser darstellt. Diesen Ansatz spiegelt die Darstellung in Abb. 5.2 b wieder. Im Gegensatz dazu illustriert die Abb. 5.2 a einen GMF Ansatz, der neben der SC Generierung als Plugin und RCP auch eine lauffähige RAP Version erzeugen kann. Auf beide Ansätze wird im Abschnitt 5.5 noch genauer eingegangen.

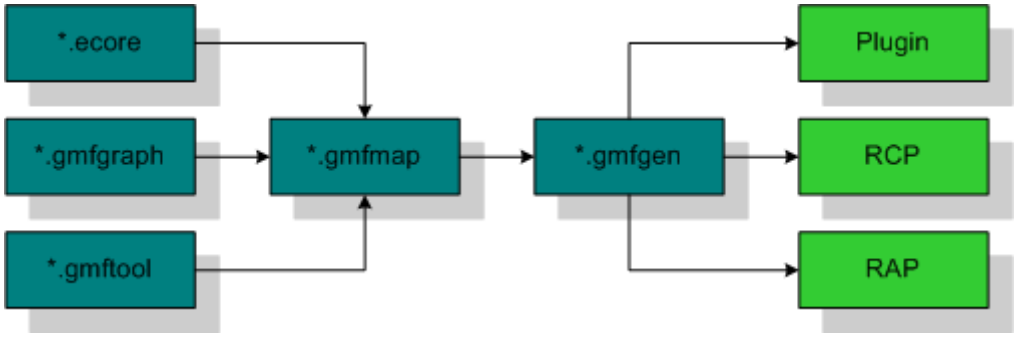

Abb. 5.2 a – Idealer Ansatz bei der Entwicklung eines Web-Modelleditors

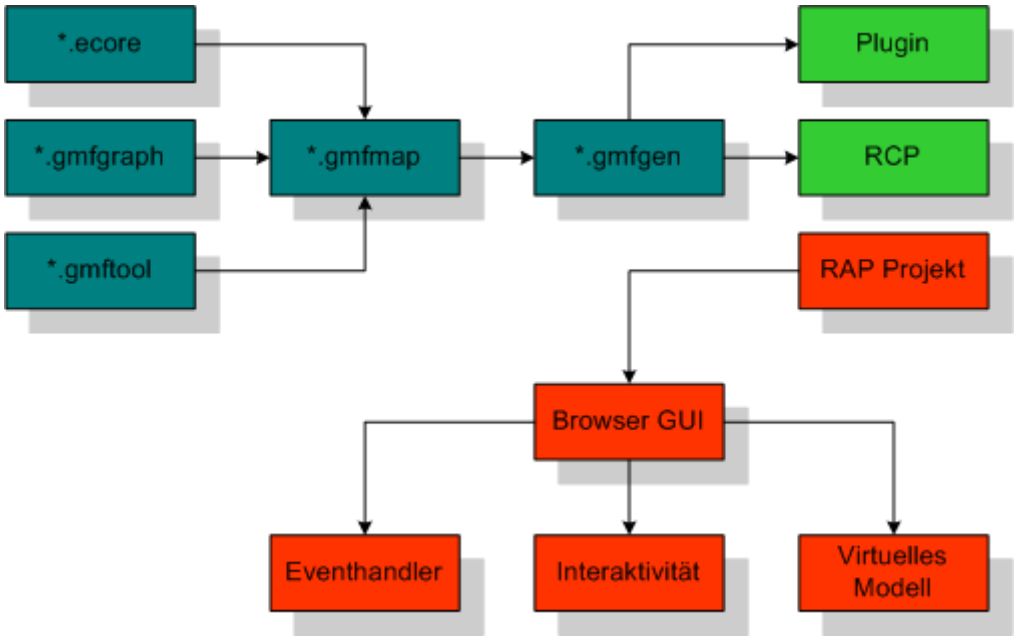

Abb. 5.2 b – Lösungsansatz der praktischen Aufgabe dieser Diplomarbeit

Auf der beiliegenden CD findet der interessierte Leser den zugrunde liegenden SC, die als Eclipse Projekte importiert werden können. Durch eine geeignete Konfiguration der Entwicklungsumgebung und der anschließenden Runtime Einstellungen für das GMF [GMF Tool 2008] und das RAP [RAP Launch 2008] können diese individuell begutachtet und getestet werden. Zu allen Projekten, die im Folgenden aufgelistet werden, ist ein JavaDoc verfügbar:

- org.eclipse.gmf.examples.mindmap – Modell Definitionen

- org.eclipse.gmf.examples.mindmap.diagram – Vom GMF generierter SC
- org.eclipse.gmf.examples.mindmap.edit – Vom GMF generierter SC
- org.eclipse.rap.examples.mindmap – RAP Web-Frontend

5.3 Konzeption und Implementierung mit Hilfe des GMF

Dieser Abschnitt befasst sich eingehend mit der Konzeptions- und Implementierungsphase des Mindmap Modells mit Hilfe des GMF. In der ersten Phase wird das Ziel definiert, das mit dem GMF erreicht werden soll. Dieser Schritt beginnt damit, dass das Mindmap Modell entwickelt und in ein für das GMF geeignetes Inputformat transformiert wird. Der nächste Abschnitt beschäftigt sich im Detail mit den einzelnen Modellen, die für das GMF und die anschließende SC Generierung notwendig sind.

5.3.1 Konzeption

Um das definierte Ziel eines Web-Modelleditors zu erreichen gilt es als ersten Schritt die Stärken des GMF auszunutzen und eine grafische Modellierungsumgebung als ein Eclipse Plugin zu schaffen. Darauf folgend soll das grafische Modellierungstool im Hintergrund mit einer geeigneten Modellrepräsentation zusammenarbeiten und die modellierten Elemente in maschinenlesbaren SC umwandeln.

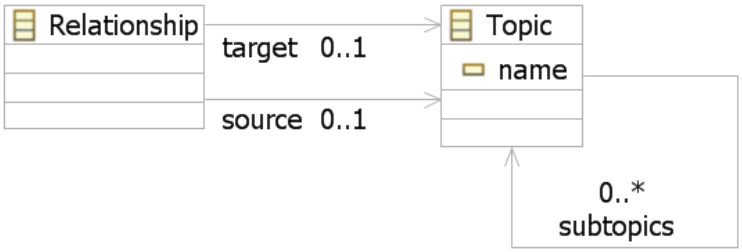

Abb. 5.3 – Modell des Mindmap Beispiels

Wie auch schon zuvor in dem Kapitel über das GMF beschrieben, ist es notwendig diesem Framework ein geeignetes Input Modell zur Verfügung zu stellen. Solche Modelle können in annotiertem Java oder in XML entwickelt werden oder durch externe Tools zur Modellentwicklung erstellt und anschließend in das GMF importiert werden [Wiki e 2008]. Für das Mindmap Beispiel wurde zunächst mit Hilfe von UML eine anschauliche Repräsentation des Modells erstellt, siehe Abb. 5.3. Ausgehend von einem Hauptthema, kann es mehrere Subthemen geben, die

mit dem Hauptthema durch eine Relation verbunden sind. Man könnte das Modell auch dahingehend erweitern, dass mehrere Hauptthemen definiert werden können und dass man Themen Notizen hinzufügen kann. Dies würde jedoch über das eigentliche Ziel hinauswachsen und soll deshalb im Rahmen dieser Diplomarbeit nicht behandelt werden.

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage
   xmi:version="2.0"
   xmlns:xmi=http://www.omg.org/XMI
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="mindmap"
   nsURI="http://www.example.org/mindmap" nsPrefix="mindmap">

 <eClassifiers xsi:type="ecore:EClass" name="Relationship">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="label"
   unique="false" eType="ecore:EDataType
   http://www.eclipse.org/emf/2003/XMLType#//String">
  ...
  </eStructuralFeatures>
  <eStructuralFeatures xsi:type="ecore:EReference" name="source"
   eType="#//Topic">
  ...
  </eStructuralFeatures>
  <eStructuralFeatures xsi:type="ecore:EReference" name="target"
   eType="#//Topic">
  ...
  </eStructuralFeatures>
 </eClassifiers>

 <eClassifiers xsi:type="ecore:EClass" name="Topic">
  <eStructuralFeatures xsi:type="ecore:EReference" name="comments"
   upperBound="-1" eType="#//Thread" containment="true"
   resolveProxies="false">
  ...
  </eStructuralFeatures>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="name"
   unique="false" eType="ecore:EDataType
   http://www.eclipse.org/emf/2003/XMLType#//String">
  ...
  </eStructuralFeatures>
  <eStructuralFeatures xsi:type="ecore:EReference" name="subtopics"
   upperBound="-1" eType="#//Topic">
  ...
  </eStructuralFeatures>
 </eClassifiers>
</ecore:EPackage>
```

Abb. 5.4 – Auszug aus dem XML Inputdokument für GMF

Anhand des UML Modells kann ein GMF spezifisches XML Dokument (vergleiche Abb. 5.4, sowie Abschnitt 3.3.1) erstellt werden, das die definierten Vorgaben erfüllt und dem GMF als Input File genügt um den Anstoß für die Generierung eines Modelleditors zu starten. Als Ausgangsdaten für die Implementierung wurde über die Konzeptionisierungsphase Folgendes erreicht:

- Anreicherung von Basiswissen über Eclipse und das GMF (Version 2.01)
- Simples Basismodell gefunden durch das Mindmap Modell
- Reduzieren der Funktionalität auf ein produktives Minimum
- Abbilden des Mindmap Modells in UML
- Transformation von UML in XML

5.3.2 Implementierung

Nachdem sich das vorherige Kapitel mit der Konzeption bezüglich der Realisierung eines Modelleditors mittels dem GMF beschäftigt hat, widmet sich dieser Abschnitt den unabdingbaren Schritten, die im Rahmen der Implementierung durchzuführen sind, um einen lauffähigen Editor als ein Eclipse Plugin zu erstellen. Im Folgenden soll auf fünf der wichtigsten Schritte in der Evolution eines GMF Modelleditors genauer eingegangen werden. Die Zusatzinformationen, die von den Implementierungsdetails abweichen, sind, wenn nicht anders angegeben, aus der Eclipse Hilfe zum Thema GMF Tooling entnommen [GMF Tool 2008].

5.3.2.1 Domain Model Definition

Das GMF an sich beschreibt das Domain Modell mit einem `*.ecore` File, seinem View `*.ecore_diagram` und einem EMF nahen `*.genmodel`. Der Ausgangspunkt an dieser Stelle ist das in XML transformierte Mindmap Modell, welches während der Konzeption entstanden ist, und nun in ein `mindmap.ecore` File importiert wird. In weiterer Folge kann man aus dieser Basis einen View generieren, der auch editierbar ist. Im Anschluss daran entsteht die angesprochene Verknüpfung zum EMF, durch eine automatische Modelltransformation des `mindmap.ecore` in das `mindmap.genmodel` File. Daraus entstanden ist eine EMF spezifische Repräsentation des Mindmap Modells. Dieser Schritt ist deswegen von großer Bedeutung, weil das GMF dadurch die Fähigkeit vom EMF erlangt, den SC für die Objekte zu generieren. Zusammenfassend kann man sagen, dass man durch die Erstellung des Domain Modells die grundlegende Logik hinter dem Editor entwickelt hat, wobei die meisten Schritte aufgrund des `mindmap.ecore` File generiert werden konnten.

5.3.2.2 Graphical Definition

Als nächsten Schritt ist es notwendig die grafischen Definitionen des Mindmap Editors zu definieren. Im Speziellen das Aussehen eines Themas, eines Subthemas und einer Verknüpfung zwischen den beiden. Dies wird mit dem Erstellen eines `mindmap.gmfgraph` File erreicht, welches auf Basis des mindmap.ecore File generiert wird. Um das Verständnis zu erhöhen soll an dieser Stelle auf die Abb. 5.5 verwiesen werden, welches dieses File darstellt. Es ist zu erkennen, dass die Elemente auf einem Canvas dargestellt werden. Die ursprünglichen Mindmap Objekte Topic, Subtopic und Relation werden durch die korrespondierenden Elemente Node, Diagram Label und Connection repräsentiert. Zusätzlich ist definiert, dass ein Topic / Subtopic als Rechteck dargestellt wird und die Verbindung eine Linie in Form einer Polyline Connection ist. An dieser Stelle scheint es wichtig zu bemerken, dass das abgebildete `mindmap.gmfgraph` File die default mäßige Generierung aus dem `mindmap.ecore` File ist. An dieser Stelle besteht die Möglichkeit das Aussehen händisch zu modifizieren. Dazu stellt das GMF einige grafische Mechanismen zur Verfügung (Layouts; grafische Grundelemente wie Rechtecke, Ellipsen; Linienarten usw.).

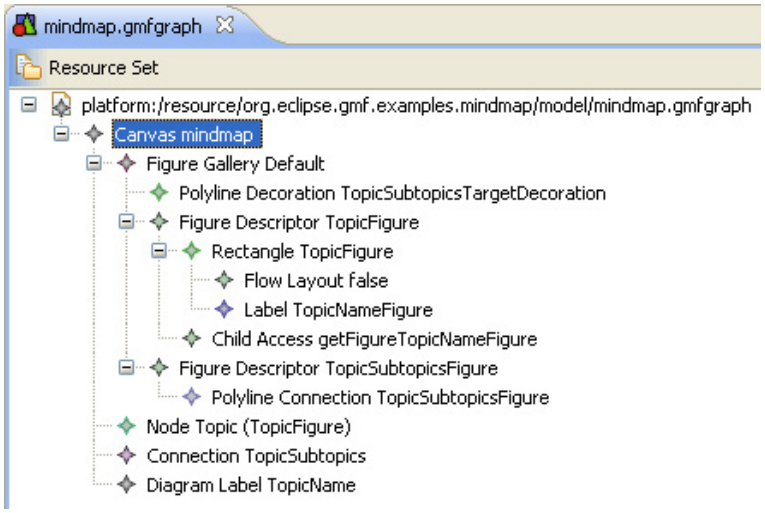

Abb. 5.5 – mindmap.gmfgraph

5.3.2.3 Tooling Definition

Im Gegensatz zur grafischen Definition beschreiben die Tooling Definitionen die Werkzeugpaletten und die sich daraus ergebenden Aktionen für die grafischen Elemente des Editors. Zusätzlich können auch andere periphere Strukturen, wie Menüs und Toolbars modifiziert werden. Das entsprechende default mäßige

`mindmap.gmftool` File kann ebenfalls aus dem `mindmap.ecore` File heraus generiert werden, siehe dazu Abb. 5.6. Aufgrund der Ausgangssituation wurde eine Mindmap Palette generiert, die zwei Tools zur Verfügung stellt. Zum einen eines zum Erzeugen eines Themas, zum anderen eines zum Erstellen eines Subthemas. Sofern man nicht weiter in zusätzliche Tooling Definitionen eingreifen möchte, kann zum nächsten Schritt übergegangen werden.

5.3.2.4 Mapping Definition

In der Entwicklung eines auf dem GMF basierenden Modelleditors ist die Mapping Definition ein Schlüsselelement, weil es in Folge dazu benutzt wird den letzten Transformationsschritt zum Code Generierungsmodell zu gewährleisten. Um dies zu erreichen muss das `mindmap.gmfmap` File erzeugt werden. Dieses wird nicht nur aus dem `mindmap.ecore`, sondern auch auf Basis von dem `mindmap.gmfgraph` und dem `mindmap.gmftool` File heraus generiert. Mit anderen Worten ist das Mapping eine Zusammenfassung der bisherigen Entwicklungen in einem einzigen File. Zusätzlich definieren die Mappings eine reibungslose Integration von dem Domain, dem grafischen und dem Tooling Modell, anhand von automatisch erstellten Regeln.

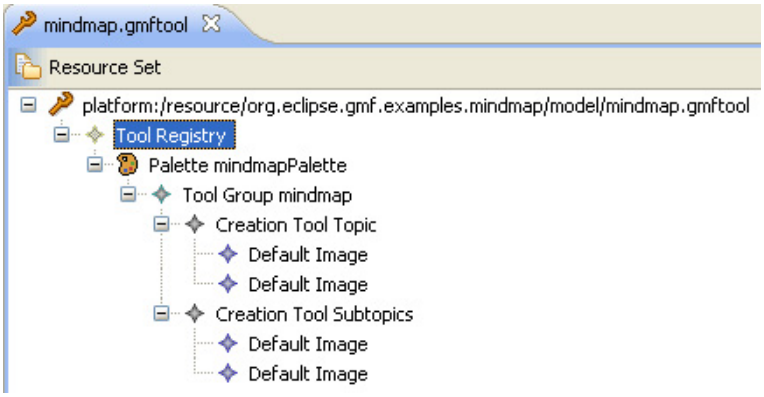

Abb. 5.6 – mindmap.gmftool

5.3.2.5 Source Code Generierung

Bevor es zur eigentlichen SC Generierung kommen kann, muss noch ein letztes Modell erstellt werden, nämlich das `mindmap.gmfgen`. Es ist vergleichbar mit dem weiter oben genannten EMF `mindmap.genmodel`, hierbei wird aber sämtlicher SC generiert, der für das Ausführen des Eclipse Plugins notwendig ist. Um dieses letzte Modell zu erhalten kann es aus der zuvor beschriebenen Mapping Definition

heraus generiert werden. Anschließend kann der SC Generator für das Plugin gestartet werden. Die Entwicklung eines auf dem GMF basierender Modelleditors ist somit an dieser Stelle beendet. Durch die vorangegangenen Schritte ist das Grundgerüst eines einfachen GMF Mindmap Modelleditors geschaffen worden, der durch Starten einer neuen Eclipse Application Runtime getestet werden kann (siehe Abb. 5.7).

5.3.2.6 Bewertung der Ergebnisse

Zum Abschluss dieses Abschnittes sollen die erzielten Ergebnisse anhand der Abb. 5.7 zusammengefasst werden und in Anbetracht der Zielsetzung kritisch hinterfragt werden. In Abb. 5.7 ist das Ergebnis der SC Generierung aufgrund der entwickelten Modelle dargestellt. Die beiden Herzstücke des Modelleditors sind das grafisch editierbare Arbeitsblatt in der Mitte und die Palette zum Erstellen von Topics, sowie Subtopics auf der rechten Seite. Die restlichen Bereiche werden von Standard Eclipse Views in Beschlag genommen.

Das Ziel der Diplomarbeit, einen webbasierten Modellierungseditor zu entwickeln, ist in diesem Stadium zwar noch nicht erreicht, dennoch ist ein erster großer Schritt getan worden. Es wurde mit dem GMF ein lauffähiger Modelleditor erstellt, der für eine Portierung in das Internet mit Hilfe des RAP herangezogen werden kann. Die darauf folgenden Kapitel beschäftigen sich mit der Konzeption und der anschließenden Implementierung des Web-Frontend für den Mindmap Editor.

5.4 Konzeption und Implementierung in der Rich Ajax Platform

Dieser Abschnitt befasst sich im Detail mit der Konzeptions- und Implementierungsphase des Mindmap Modells mit Hilfe des RAP. In der ersten Phase wird das Ziel definiert, dass mit dem RAP Ansatz erreicht werden soll. Diese beginnt damit, dass ein Klassendiagramm erstellt wird, das für die Implementierung herangezogen werden soll. Anschließend werden die Grundlagen eines neuen RAP Projektes erläutert, gefolgt von der Beschreibung eines Einstiegspunktes am Server für die Web-Anwendung. Abgeschlossen wird dieses Kapitel mit einer Klärung der interaktiven Komponenten und einer kritischen Bewertung der gewonnenen Ergebnisse.

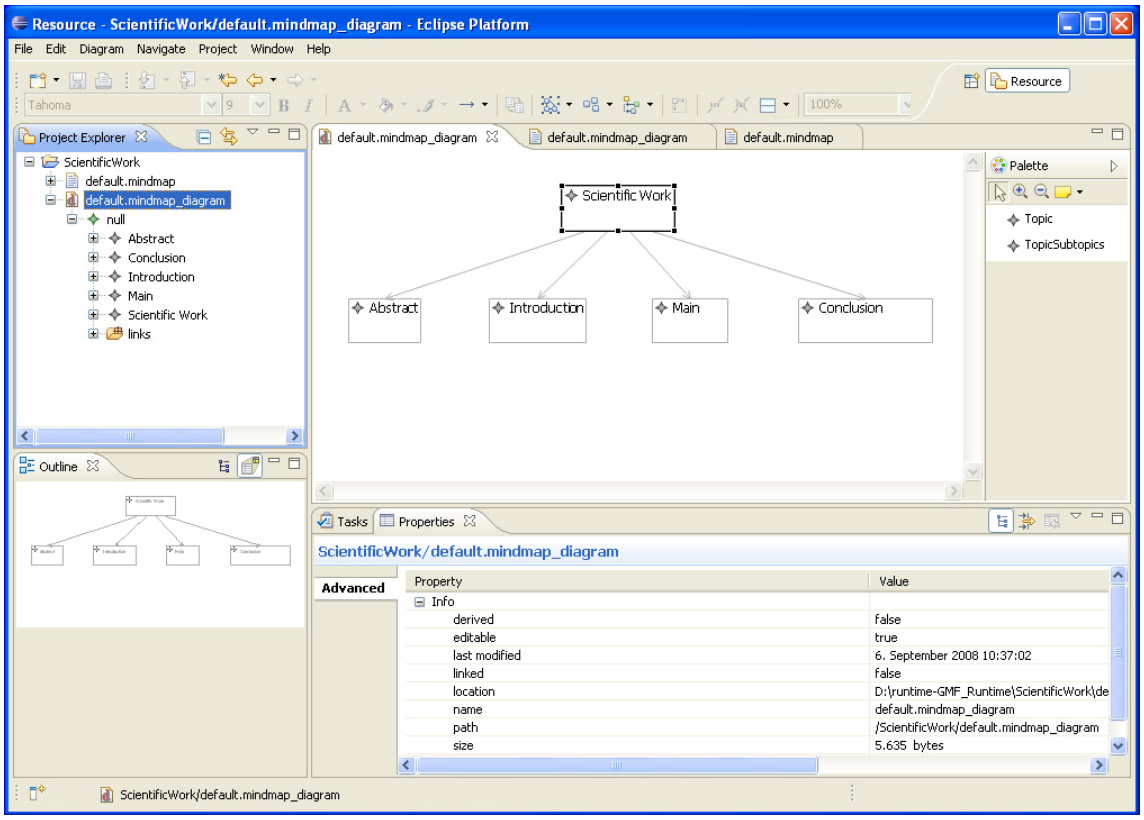

Abb. 5.7 – Einfacher Mindmap Editor als Eclipse Plugin

5.4.1 Konzeption

Die RAP Konzepte setzten sich aus Ajax, HTML, Javascript, CSS, usw. zusammen, dennoch verspricht man sich von diesem Framework, dass die Kenntnis der zuvor genannten Web-Technologien nicht zwingend zur Umsetzung einer RAP Web-Anwendung notwendig sind. Vielmehr soll es möglich sein eine reine Java basierende Web-Anwendung zu entwickeln, die sich automatisch um Aspekte der Server Client Kommunikation und um das Layout kümmert. Dies macht das RAP zu einer interessanten Alternative im Gegensatz zur klassischen interaktiven Webentwicklung. Ob diese Versprechungen der Praxis standhalten können, werden die nächsten Kapitel herausfinden. Beginnend mit der Zielsetzung des RAP Frontend wird die logische Klassenstruktur aufgebaut und anschließend schrittweise erklärt.

5.4.1.1 Zielsetzung

Der zweite Baustein am Weg zu einem Web-Modelleditor ist es eine Browserapplikation (vergleiche Abb. 1.1) zu schaffen, die es erlaubt, die Mindmap Objekte (Themen, Subthemen und Verbindungen) in einen Editor zu laden und

diese interaktiv zu verändern. Verändern bedeutet in diesem Sinne eine Namensänderung des Themas sowie ein individuelles Positionieren auf der Arbeitsfläche. Anschließend soll es möglich sein einen Event auszulösen, welcher die Elemente im Editor in das zuvor entwickelte GMF Modell am Server einspielt und den generierten XML Code an den Browser zurücksendet, um ihn dort in ansprechender Form darzustellen.

5.4.1.2 UML Ansicht der Web-Anwendung

Der nächste Schritt ist die Ausarbeitung eines UML Klassendiagramms, um die notwendigen Abhängigkeiten und die klassenspezifischen Charakteristika für eine Mindmap Web-Anwendung zu definieren. Abb. 5.8 illustriert das Java Paket `org.eclipse.rap.examples.MindmapGUI` und soll in Folge schrittweise erläutert werden. Die gesamte Verwaltung der Web-Anwendung geht von der Klasse `MindmapDemo.java` aus, die das Interface `IEntryPoint` implementiert. Diese Klasse erstellt die grundlegenden *Graphical User Interface* (GUI) Funktionalitäten und definiert die Eventhandler für die Werkzeugpalette, wie sie auch im GMF verfügbar ist. Diese Eventhandler werden durch die Klassen `MyClickListener.java` (die von der `SelectionAdapter` Klasse abgeleitet ist) und `MyMouseListener.java` (welche das `MouseListener` Interface implementiert) realisiert. Die Klasse `ImageLoader.java` kann immer dann verwendet werden, wenn GUI spezifische Grafiken in den Browser geladen werden sollen. Ein Thema selbst wird durch die Klasse `Topic.java` spezifiziert, welches ihrerseits von der Klasse `Composite` abgeleitet ist und somit als eine SWT Grafik im Editor angezeigt werden kann. Ein Thema hat eine `Relation.java` Klasse, welche die Verbindungen zwischen Topic und Subtopic implementiert. Der schon genannten Eventhandler `MyMouseListener.java` wird dazu verwendet um die Elemente im Editor individuell zu verschieben. Genauere Details zu den Interfaces und Klassen wird im anschließenden Kapitel Implementierung erläutert.

5.4.2 Implementierung

Die folgenden Kapitel, die die Entwicklung einer Web-Anwendung mit dem RAP beschreiben, bauen auf das oben genannte Klassendiagramm auf. Es sollen essentielle Punkte beschrieben werden, die bei der Erstellung einer RAP Anwendung notwendig sind. Zunächst wird auf die grundlegende Konfiguration

einer neuen RAP Applikation eingegangen, dann soll die Klasse `MindmapDemo.java` genauer beleuchtet werden, da sie der Einstiegspunkt der Web-Anwendung ist und die komplette GUI Konfiguration übernimmt. Auf die Kommunikation zwischen dem Client und dem Server wird ebenfalls eingegangen, sowie eine abschließende Bewertung der erzielten Ergebnisse geliefert.

5.4.2.1 Grundlagen einer RAP Applikation

Die hier beschriebenen Grundlagen beziehen sich weniger auf programmiertechnische Kenntnisse, sondern vielmehr auf Vorbereitungen, die getroffen werden müssen, damit ein neues RAP Projekt kompilierbar wird und die Client / Server Kommunikation übernimmt. Als ersten Schritt müssen im Plugin Manifest `manifest.mf` das notwendige User Interface Plugin, auf das die neue Web-Anwendung aufbaut, und die die für die Kommunikation notwendigen Servlet Pakete hinzugefügt werden [RAP Start 2008]:

- `org.eclipse.rap.ui`
- `javax.servlet`
- `javax.servlet.http`

Der zweite wichtige Schritt in der Vorbereitung ist es, einen geeigneten Einstiegspunkt am Server für die Web-Anwendung zu definieren. Dieser Einstiegspunkt ist vom Typ `org.eclipse.rap.ui.entrypoint`, besitzt eine eindeutige ID, einen Parameter, der in der URL angegeben wird um die Anwendung zu starten und jene Klasse, die die Anwendung initialisiert [RAP Start 2008]. Im Fall der Mindmap Web-Anwendung ist das die Klasse `MindmapDemo.java`. Damit sind die grundlegenden Voraussetzungen getroffen und die Programmierung der einzelnen Klassen zu beginnen.

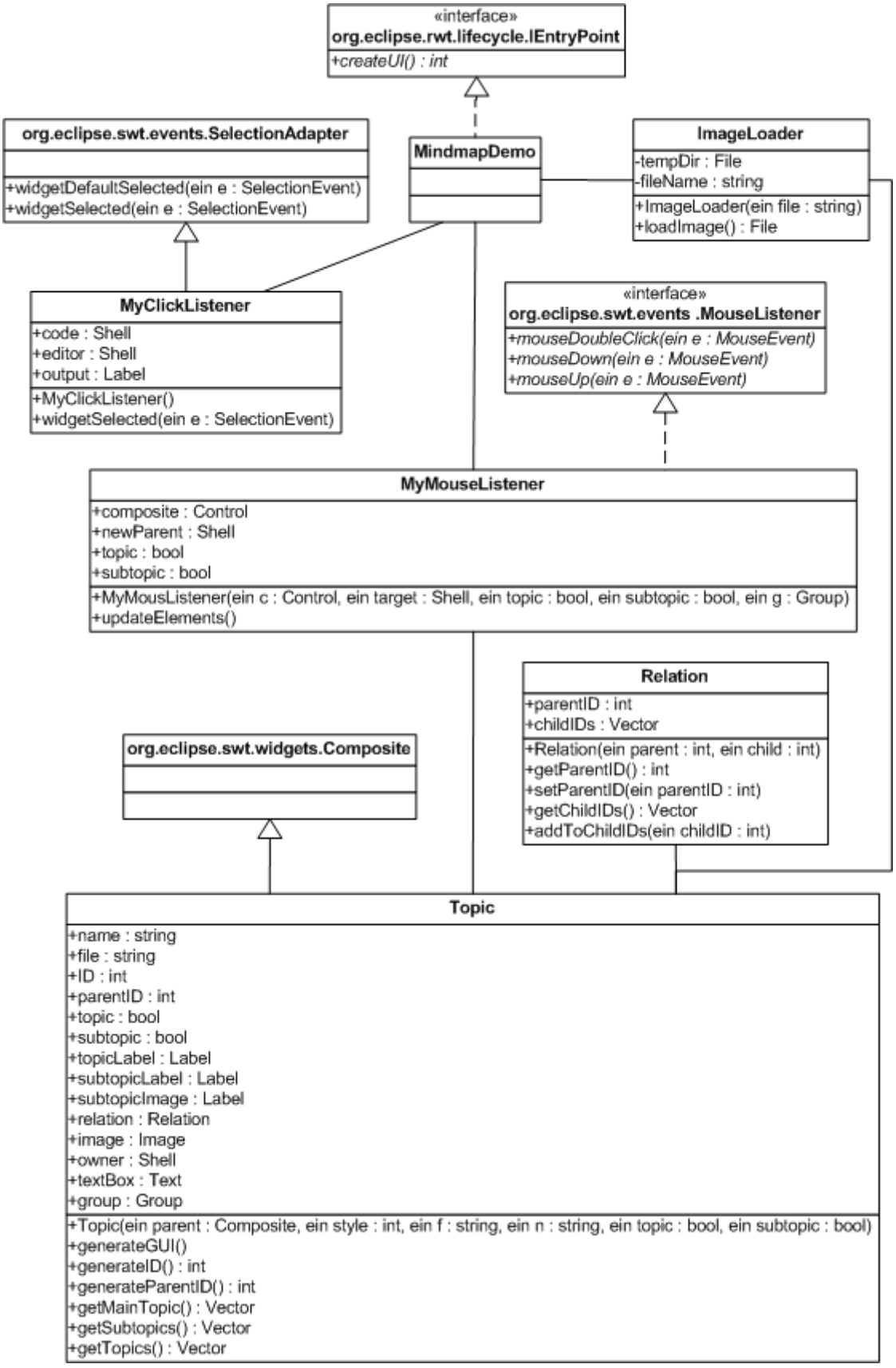

Abb. 5.8 – MindmapDemo Klassendiagramm

5.4.2.2 Implementierung des IEntryPoint Interface

Der Einstiegspunkt einer RAP Applikation ist in erster Linie dafür verantwortlich die Elemente der Benutzeroberfläche – Display, Shell, Grafiken usw. – zu definieren. Um dieser Aufgabe nachkommen zu können muss die Klasse `MindmapDemo.java` das Interface `IEntryPoint` und die damit verbundene Methode `createUI()` implementieren. In dem Umfeld der RAP Programmierung kann diese Methode mit der `main()` Methode von SWT oder Konsolen Applikationen verglichen werden [JDoc a 2008], [RAP Start 2008]. Zur Illustration wird in Abb. 5.9 ein Ausschnitt aus dieser Klasse präsentiert und im Folgenden besprochen.

```
public class MindmapDemo implements IEntryPoint {

 public int createUI() {
  Display display = new Display();

  final Shell editor = new Shell(display, SWT.SHELL_TRIM);
  editor.setBounds(150, 10, 500, 400);
  editor.setText("Editor");
  editor.open();

  .
  .
  .

  while (!elements.isDisposed() && !editor.isDisposed() &&
         !code.isDisposed() ) {
   if (!display.readAndDispatch())
    display.sleep();
   }

  return 0;
 }
}
```

Abb. 5.9 – Auszug aus dem Einstiegspunkt

Die Methode `createUI()` beginnt damit, dass ein neues Display Widget erzeugt wird, auf das ein Shell Widget gelegt wird. Im Anschluss werden alle anderen Widgets und Objekte definiert, die das anfängliche GUI aufbauen. Der letzte Abschnitt sorgt dafür, dass solange bestimmte Widgets im Browser am Leben sind, überprüft wird, ob die Event Queue noch Events beinhaltet. Diese werden dann vom Display an die zuständigen Bereiche verteilt und abgearbeitet. Wenn die Queue leer werden sollte, wird das Display in einen Ruhezustand versetzt, das soll vor allem die CPU Belastung senken, bis ein neuerlicher Event in die Queue gelangt [JDoc b 2008].

5.4.2.3 Erreichen einer angemessenen Interaktivität

Die Interaktivität ist eine absolute Kernfunktionalität eines Editors und darf auf keinen Fall in einem Web-Modelleditor fehlen. Generell lässt jedoch die statische Struktur von HTML diese gewünschte Interaktion nicht zu, deswegen kommt an dieser Stelle das vom RAP erzeugte Ajax ins Spiel, welches Benutzeraktionen an die entsprechenden Eventhandler am Server zur Verarbeitung weiterleitet. Haben diese ihre Aufgabe erfüllt, retournieren sie ihr Ergebnis über die Ajax Schicht an den Browser (wie die Abb. 5.10 a, b und c zeigen).

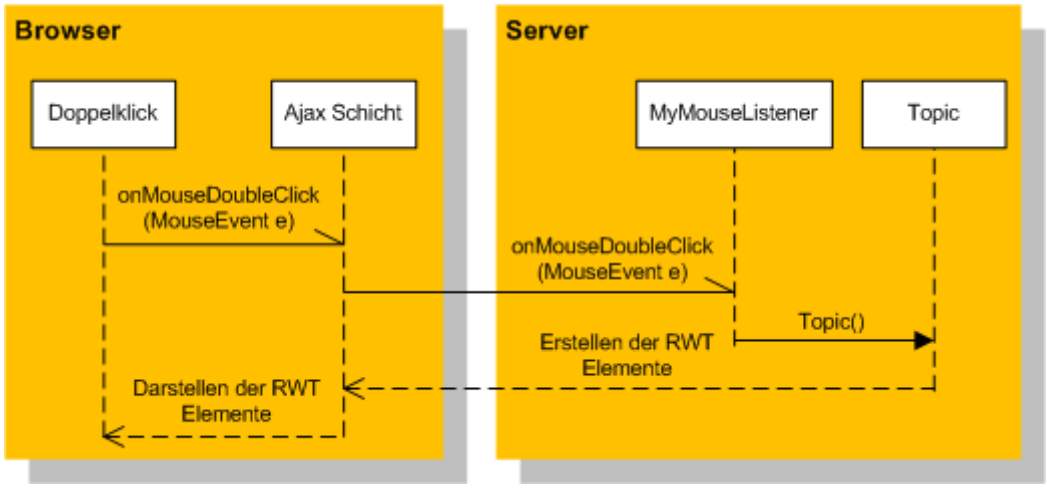

Abb. 5.10 a – Interaktion bei einem Doppelklick auf ein Topic

Um die simplen interaktiven Aufgaben der Mindmap Demo zu gewährleisten gibt es die zwei Klassen `MyMouseListener.java` und `MyClickListener.java`. Letzterer wartet darauf, dass ein Button gedrückt wird, wonach die im Editor erstellten Informationen an den Server und somit in das GMF Modell eingespielt und der daraus resultierende XML Code generiert wird. Der SC wird an den Browser zurück geschickt und in geeigneter Form präsentiert[19]. Die Klasse `MyMouseListener.java` implementiert das Interface `MouseListener` und die damit verbundenen Methoden:

- `mouseDoubleClick(MouseEvent e)`: Diese Methode wird nur von Elementen der Werkzeugpalette angesprochen und fügt dem Editor aus der Palette ein Thema oder ein Subthema hinzu (vergleiche Abb. 5.10 a).

[19] Die Klasse `MyClickListener.java` ist nur zum Teil implementiert. Gründe dafür werden im Kapitel 5.6 näher erläutert.

- `mouseDown(MouseEvent e)`: Diese Methode wird nur von Elementen des Editors angesprochen und berechnet für das ausgewählte Element die aktuelle Position (vergleiche Abb. 5.10 b).
- `mouseUp(MouseEvent e)`: Diese Methode wird nur von Elementen des Editors angesprochen und berechnet für das ausgewählte Element im Editor die aktuelle Position des Mauszeigers und verschiebt es an diese Stelle (vergleiche Abb. 5.10 c).

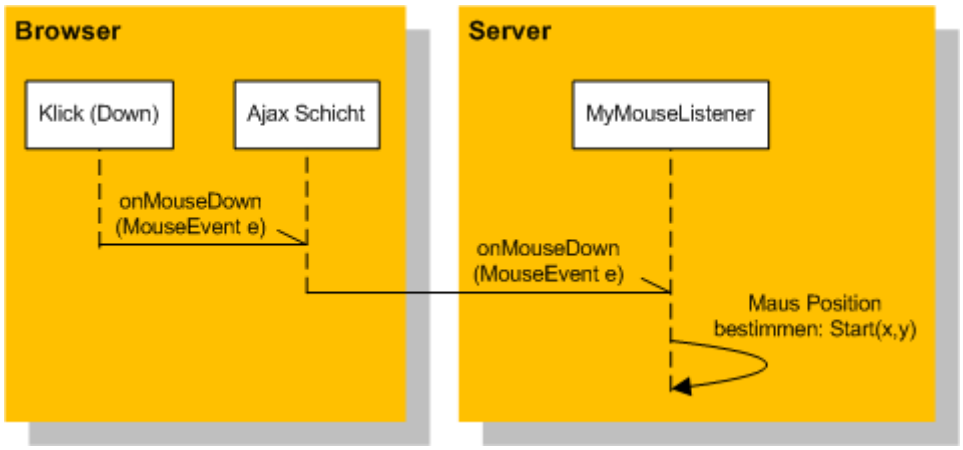

Abb. 5.10 b – Interaktion bei einem Klick (Down) auf ein Topic

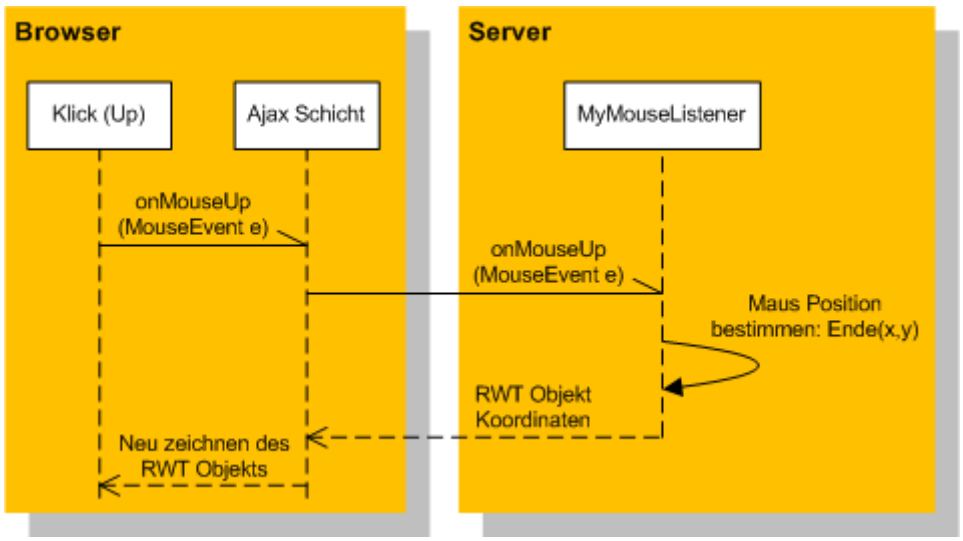

Abb. 5.10 c - Interaktion bei einem Klick (Up) auf ein Topic

Zusätzlich existiert eine Methode `updateElements()`, die aufgerufen wird, wenn Themen und deren Beziehungen neu aus der Werkzeugpalette heraus erzeugt wurden. Die Veränderungen, betreffend der Elemente des Editor, werden neu berechnet und im Browser aktualisiert. Durch diese beiden Klassen erreicht man

ein minimales Maß an Interaktivität, welche im Rahmen eines Prototyps zulässig ist, jedoch für Systeme mit kommerziellem Hintergrund absolut unzureichend ist.[20]

5.4.2.4 Bewertung der Ergebnisse

Der Abschluss dieses Kapitels soll dazu genutzt werden ein Resümee über die Entwicklung eines Web-Frontend für das Mindmap Beispiel mit dem RAP zu ziehen. Das RAP wurde eingesetzt, um einen bequemen Ansatz zu verfolgen, bei dem es prinzipiell nicht notwendig ist sich mit Details von HTML, Ajax, JavaScript und CSS auseinander zu setzen. Das erste Ergebnis das aus den zuvor präsentierten Entwicklungen zu nennen ist, dass mit dem RAP eine reine Java basierte Web-Anwendung realisiert werden kann. Die notwendigen Scripte für das Layout und für die Kommunikation zwischen dem Browser und dem Server werden vom Framework generiert.

Daraus ergibt sich aber schon der erste Kritikpunkt, nämlich jener, dass man ohne ein Eingreifen in den generierten SC bald an die Grenzen des Machbaren stößt, beispielsweise, wenn es um Themen der grafischen Präsentation und der Benutzerinteraktion geht. Im Speziellen lassen es die derzeitig verfügbaren SWT Komponenten nicht zu, dass man wie gewohnt einfache Zeichenoperationen – beispielsweise Punkte und Linien – ausführen kann [DevX 2008], [InfoQ 2008]. Diese Erkenntnis erklärt auch den Umstand, dass es nicht möglich war eine grafische Verbindung zwischen dem Hauptthema und den Subthemen herzustellen. Praktisch wurde dieses Problem umgangen, indem das Hauptthema eine Liste mit seinen Subthemen beinhaltet, die ausgelöst durch einen Event aktualisierbar sind.

Eine zweite Einschränkung ergab sich bei den Bemühungen, die Elemente im Editor individuell von einer Position zu einer anderen zu ziehen. Zurzeit gibt es keine Drag and Drop Unterstützung und keinen Mouse Listener, der auf die Zeigerbewegung reagiert [InfoQ 2008]. Seit dem letzten Release von RAP 1.1 im Juni 2008 besteht zumindest die Möglichkeit auf Mausevents wie Doppelklick, Mausklick Down / Up zu reagieren [RAP News 2008].

[20] Durch Eingreifen in den vom RAP erzeugten Ajax Code besteht prinzipiell die Möglichkeit zusätzliche und qualitativ höherwertige interaktive Komponenten zu programmieren. Dieses würde aber den Rahmen der Diplomarbeit und die des Prototyps sprengen.

Dennoch ist es gelungen unter solchen Umständen eine simple Web-Anwendung für eine Mindmap Demo zu entwickeln, die in Abb. 5.11 dargestellt ist. Die Oberfläche ist in drei Bereiche, einer Werkzeugpalette, einem Editor und einem SC Output unterteilt. Durch einen Doppelklick auf Elemente in der Palette wird das entsprechendes Element im Editor erzeugt und die Abhängigkeiten zwischen den Elementen in einer Liste bei dem Hauptthema aufgezeigt. Durch Klicken, Ziehen und Loslassen der Elemente im Editor besteht die Möglichkeit die Themen auszurichten. Durch das Klicken auf den Update Knopf werden die Informationen aus dem Editor gesammelt und an den Server übergeben.

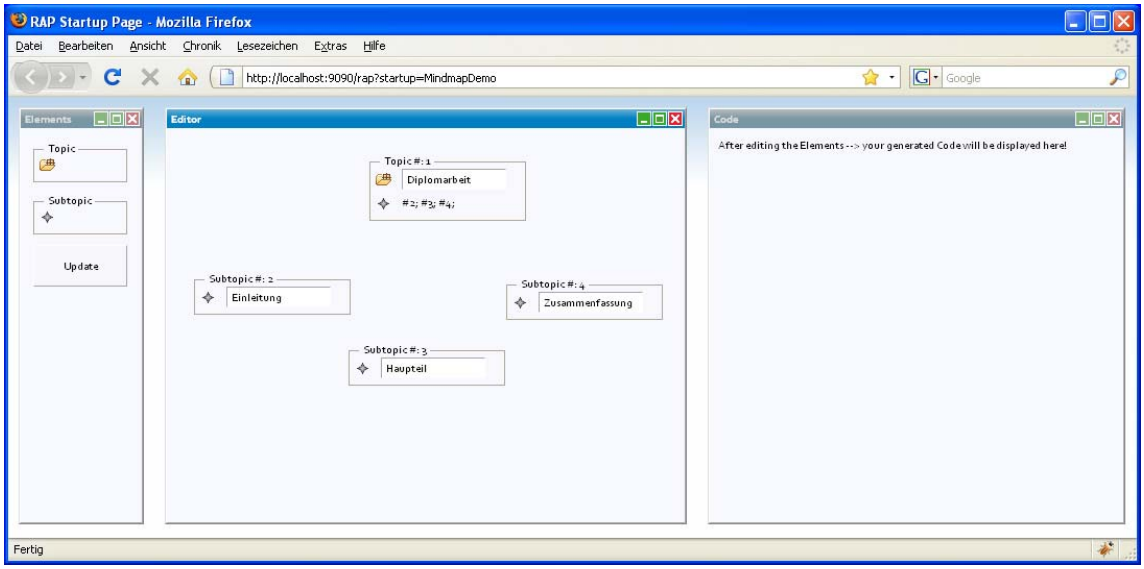

Abb. 5.11 – GUI der Mindmap Demo

5.5 Analyse der Interoperabilität zwischen GMF und RAP

Dieser abschließende Abschnitt der praktischen Aufgabe befasst sich genauer mit den Schnittstellen, die zwischen dem GMF und dem RAP vorhanden sind oder noch erstellt werden müssen, damit eine optimale und effiziente Zusammenarbeit der beiden Frameworks gewährleistet werden kann.

Bevor genauer auf die nicht vollständig implementierte Java Klasse eingegangen werden kann, soll noch einmal die ursprüngliche Idee des Web-Modelleditors anhand der Abb. 5.2 a / b in Erinnerung gerufen werden. Abb. 5.2 b zeigt ein Spiegelbild dessen, was in dem praktischen Teil der Diplomarbeit erarbeitet wurde. In Abb. 5.2 a befindet sich ein Lösungsansatz, der beide Frameworks optimal nutzt und mit einem Minimum an persönlicher Arbeit einen Modelleditor in einen Browser

einbettet. Mit anderen Worten würde zu den zwei schon bestehenden automatisch generierbaren Outputkomponenten Plugin und RCP eine dritte hinzukommen, nämlich das RAP.

Betrachtet man die Arbeitsschritte Abb. 5.2 b, so ist gut zu erkennen, dass für einen Web-Modelleditor zunächst ein Editor mittels des GMF erstellt werden muss. Mit dem RAP wird derselbe Editor auf Basis von Web-Technologien neu entwickelt, um im Browser dargestellt werden zu können. Zusätzlich erstellt die RAP Schicht ein virtuelles Modell, welches die im Editor erstellten Elemente repräsentiert. Zusammengefasst wird der Editor und das Modell doppelt implementiert, oder anders ausgedrückt, für unterschiedliche Plattformen programmiert. An dieser Stelle ist leicht einzusehen, dass dies zu einer ineffizienten Art und Weise der Softwareentwicklung führt und dass bei ansteigender Größe und Komplexität des Modells der Wartungsaufwand in unermessliche Höhen steigen kann.

Die Abb. 5.2 a schlägt einen anderen Ansatz vor, der diesen Entwicklungsaufwand enorm reduzieren könnte. Die Frage ist nur, unter welchen Umständen dieses Ziel zu erreichen ist bzw. ob es aus technologischer Sicht zum Zeitpunkt der Diplomarbeit überhaupt mit einem angemessenen Aufwand machbar ist. Wie man in der Abb. 5.2 a erkennt, spaltet sich der Arbeitsfluss nach der Erstellung des `*.gmfgen` auf, somit wäre hier der ideale Ansatzpunkt, um die RAP Konzepte einzubringen. Das bedeutet aber in weiterer Folge, dass ein geeigneter RAP Renderer entwickelt, sprich, dass in die bestehende SC Generierung des GMF manuell eingegriffen werden müsste. Im Rahmen dieser Diplomarbeit ist dieser Aufwand nicht vertretbar, wobei noch erschwerend hinzukommt, dass es über den Teilbereich, den Generierungsmechanismus, nur wenige Informationen gibt. Zusätzlich sollte man sich aber auch die Frage stellen, ob dieser Ansatz unter den derzeitigen technischen Voraussetzungen überhaupt lösbar ist. Die Antwort auf diese Frage werden die kommenden Absätze geben.

Als erster Schritt soll überprüft werden, ob das RAP überhaupt die Vorraussetzungen hat um die vom GMF bevorzugten Aufgaben zu erfüllen. Das GMF ist ein Modelleditor mit komplexen Darstellungsmöglichkeiten für dessen Objekte und besitzt die Möglichkeit Elemente zu verbinden und diese frei zu positionieren. Das RAP in seiner derzeitigen Version besitzt keine Möglichkeit primitive Zeichenoperationen im Browser durchzuführen, ein dynamisches

Verbinden von Elementen mit Linien oder Pfeilen ist also nicht möglich. Auch bei der Komplexität von grafischen Elementen stößt das RAP an die Grenze seiner Machbarkeit, dicht gefolgt von einer sehr geringen Unterstützung von Mausevents. Zurzeit werden nur ein Doppelklick, das Drücken / Loslassen der Maustaste registriert. Die Möglichkeit eines verfolgbaren Mauszeigers oder einer Drag and Drop Funktionalität ist mit einer der nächsten Versionen geplant. Mit anderen Worten kann zusammengefasst werden, dass das GMF mit der Draw2D API für die grafische Präsentation arbeitet, die vom RAP nur in geringstem Ausmaß realisiert wird [DevX 2008], [InfoQ 2008].

Das bedeutet aber für den Ansatz in Abb. 5.2 a, dass dieser noch nicht umsetzbar ist, aber die Hoffnung auf Machbarkeit mit den steigenden Versionen der beiden Frameworks zunimmt und sich die beiden einander annähern. Der Grundstein der Zusammenarbeit von GMF und RAP scheint aber schon gelegt, wie der leitende Entwickler des RAP Projektes Jochen Krause [Krause 2008] in dem Interview [InfoQ 2008] bestätigt. Hier kommuniziert er, dass der Wunsch nach umfassenden grafischen Editoren groß ist und dass man versuchen wird die Draw2D API für das RAP in den nächsten Versionen komplett zu implementieren.

6 Zusammenfassung und Ausblick

Der Abschluss der Diplomarbeit zum Thema eines Web-Modelleditor Prototyps soll an dieser Stelle noch einmal die gewonnenen Erkenntnisse präsentieren. In einer kompakten Zusammenfassung der einzelnen Kapitel wird das erzielte Wissen dargestellt und im Kontext zur Diplomarbeit bewertet. Im Anschuss daran soll ein persönlicher Ausblick in die Zukunft der beiden Eclipse Technologien GMF und RAP gewagt werden. Das Hauptaugenmerk liegt hierbei auf den notwendigen Anpassungen, welche die Technologien durchführen müssen, um weiterhin eine bestimmende Rolle in der Softwareentwicklung spielen zu können.

6.1 Zusammenfassung

Das Ziel war es einen Prototypen eines Web-Modelleditors zu entwickeln, der sich der automatisierten Eclipse Technologien GMF und RAP bedient. Dafür war es notwendig sich mit den Konzepten der MDSD auseinander zu setzen. Darauf aufbauend konnte sich den primären Entwicklungstechnologien von Eclipse angenommen werden. Mit den dadurch gesammelten Informationen war es möglich den Prototypen zu implementieren.

Im Kapitel 2 *„Model Driven Software Development“* wurde ein Vorgehensmodell für die Softwareentwicklung vorgestellt, welches praktisch vom GMF umgesetzt wird. Es besagt, dass dadurch eine Software durchgängig mit Modellen beschrieben werden kann. Grundsätzlich bilden die dabei erstellten Modelle eine aufeinander aufbauende Abstraktionshierarchie, die höchste Abstraktionsstufe wird dabei in dem fachlichen Modell erreicht. Hierbei wird Aufgabenstellung in einer plattform-unabhängigen Sichtweise wiedergegeben. Eine Abstraktionsstufe darunter befindet sich das technische Modell, welches das zuvor erwähnte fachliche Modell mittels der für diese Technik spezifischen Möglichkeiten abbildet. In der untersten Ebene findet man den auf Grund des technischen Modells generierten SC.

Durch das Einbringen von Modellen in die Softwareentwicklung können unterschiedliche Formen von Gewinnen erreicht werden. Man erhält eine abstraktere Sichtweise der Aufgabenstellung, dadurch wird es möglich andere Interessensgruppen in den Designprozess einzugliedern. Ein weiterer großer Vorteil zeigt sich durch den Anstieg der Unabhängigkeit und der Resistenz gegen

Änderungen im Modell. Durch die Transformatoren, Generatoren und deren Regeln werden die Anpassungen im Modell automatisch in den SC übernommen.

Im Kapitel 3 *„Überblick über die eingesetzten Technologien“* wurden die essentiellen Technologien des Web-Modelleditors vorgestellt. Darunter fallen das grafische Modellierungsframework GMF und das RAP als Ausgangspunkt für interaktive Web-Anwendungen. Diese Techniken, sind als OS Software verfügbar, weswegen sie die OS Definitionen von OSI umsetzten, welche in der Realität durch die EPL realisiert werden. Die durch OS Software einhergehenden Vorteile beschränken sich nicht nur auf die Entwicklung von zuverlässiger hochqualitativer Software bei einem geringen Einsatz von Kosten und Ressourcen. Zusätzlich bietet es Unternehmen die Möglichkeit zum Einsatz von flexiblen Technologien, aber auch um Innovationen rasch voranzutreiben.

Die Eclipse Plattform wurde für die Entwicklungen als IDE, aus unterschiedlichen Gründen herangezogen. Zum einen sind das GMF und das RAP für Integration in Eclipse konzipiert, zum anderen baut diese IDE auf einem anderen Konzept auf als die meisten proprietären Entwicklungsumgebungen. Eclipse ist in viele kleine Bausteine, so genannte Plugins, zerlegt worden. Miteinander können die Plugins als eine große funktionierende Entwicklungsumgebung agieren. Durch den Einsatz dieser Bausteine ist es möglich, benutzer- sowie anforderungsspezifische Umgebungen aufzubauen.

Das GMF, welches dem MDSD Ansatz folgt, vereint die schon existierenden Projekte EMF und GEF, wobei ersteres ein Framework zur Generierung von Java SC aus Modellen ist. Das EMF selbst besitzt keine Unterstützung für grafische Elemente, sondern liefert dem GMF nur die Funktionalität aus einem geeigneten Input Modell SC zu generieren. Das GEF hingegen erlaubt es, aus bestehenden Modellen, mächtige grafische Editoren zu kreieren. Dazu zählen vor allem jene, die mit dem EMF entwickelt worden sind. Für die grafische Darstellung wird die Draw2D API verwendet, welches sich um die Aspekte von Layout und Rendering kümmert. Das GMF kombiniert die beiden Frameworks und ermöglicht dadurch die Entwicklung von grafischen Editoren für die unterschiedlichsten Domain Modelle.

Im Kontrast dazu steht das RAP, mit dem komplexe interaktive Web 2.0 Anwendungen auf Basis von Java erstellt werden können. Die dazu benötigten

Webkomponenten werden von der RAP Engine generiert. Die Schlüsseltechnik ist der Ajax Mechanismus, der es ermöglicht eine http Anfrage innerhalb einer Webseite durchzuführen, ohne dass die gesamte Seite neu geladen werden muss. Zusätzlich differenziert sich die RAP Architektur von jener der RCP durch eine RWT Komponente (die SWT Objekte werden dazu für die Darstellung im Browser optimiert), Equinox (einem OSGi Framework) und einem Servlet Container.

Im Kapitel 4 *„Technologische Bewertung von GMF und RAP“* wurde durch gezielte Fragestellungen versucht spezielle Antworten von den jeweiligen Technologien zu erhalten. Für beide Technologien gelten eine starke Unabhängigkeit von Client / Server BS sowie eine große Unterstützung von Standards. Die Entwicklung erfolgt im von Eclipse zur Verfügung gestellten IDE. Durch den Aufbau als OS Software entstehen hierbei keine Kosten bei der Benutzung und eine große Community unterstützt bei vielen Problemen. Die Ergebnisse zu Fragen der Performance, der Integrität und der Homogenität machen es deutlich, dass Investitionen in diesen Bereichen notwendig sind um einer konkurrenzfähigen Zukunft entgegenblicken zu können. Weiters verfügen beide über eine Multi-Threading Fähigkeit, sind im positiven Sinne abhängig von Drittkomponenten und es kann beiden Technologien eine lange Lebensdauer vorausgesagt werden.

Speziell zum GMF ist zu bemerken, dass es sich um eine produktive Technologie handelt, die aber nur durch einen großen Lernaufwand greifbar wird. Auf Universitäten werden erst teilweise Kurse angeboten, die MDSD Inhalte vermitteln. Daraus ergibt sich eine eher mäßige Verbreitung im Entwicklerumfeld. Durch die grafische Oberfläche stehen viele Usability Features zur Verfügung, diesbezüglich kann man sich nur geringere Innovationen vorstellen.

Das RAP zeichnet sich durch einen geringen Lernaufwand aus, kann als gut skalierbar und sehr produktiv angesehen werden. Zurzeit findet man noch wenig bis keine Usability Features. Der Innovationscharakter dieser noch sehr jungen Technologie ist aber entsprechend groß, somit kann auch diesbezüglich noch mit großen Veränderungen gerechnet werden. Als entscheidenden Nachteil kann gesehen werden, dass das RAP noch nicht an den Universitäten gelehrt wird und somit eine eher geringe Verbreitung aufweist.

Das Kapitel 5 *„Implementierung eines Web-Modelleditor Prototyps“* hat sich im Wesentlichen mit den notwendigen Erläuterungen zum praktischen Teil der Diplomarbeit beschäftigt. Im Großen und Ganzen betraf dies die Beschreibung des begleitenden Beispiels, das Mindmap Modell und der Umsetzung mit den Technologien GMF und RAP. Der interessante Aspekt war jedoch die Analyse der erzielten Ergebnisse, im besondern betreffend die Interoperabilität dieser Ansätze.

Die Zusammenarbeit des GMF mit dem RAP in einer Art und Weise, die per Knopfdruck aus dem erstellten Modell eine RIA produziert, ist nicht nur für die beiden Projektleiter der Eclipse Projekte wünschenswert, sondern wäre auch eines der Ziele dieser Arbeit gewesen. Dies würde aber bedeuten, dass an geeigneter Stelle – um die Trennung von fachlichem und technischem Modell zu gewährleisten – die SC Generierung hätte umgeschrieben werden müssen. Zusätzlich wird man auf weitere technologische Barrieren stoßen, das betrifft vor allem die unterschiedliche Herangehensweise wie Grafiken dargestellt werden. Im GMF mit der Draw2D API und im RAP mit dem RWT. Diese Ansätze sind strikt inkompatibel, das impliziert, dass die Objekte der Draw2D API im RWT zur Gänze neu implementiert werden müssen, was zum Abschluss dieser Arbeit nur in einem geringen Maß geschehen ist. Eine weitere Differenz manifestiert sich in der Unterstützung von Benutzer Events, welche für die Interaktivität eine große Rolle spielen. Im GMF in einer Vielzahl vorhanden, sind sie im RAP jedoch erst minimal vorhanden. Auch in diesem Fall müssen für die Eventlistener spezielle RAP Adapter implementiert werden, die den RWT Lebenszyklus realisieren. Zusammenfassend kann gesagt werden, dass das RAP in seiner derzeitigen Version keine Möglichkeiten besitzt alle vom GMF gewünschten Features umzusetzen.

6.2 Ausblick

Nach der kompakten Zusammenfassung der Ergebnisse wird dieser Abschnitt dazu genutzt um eine persönliche Interpretation der derzeitigen Situation und der möglichen Zukunft zu liefern. Den besprochenen Problemen bei der Integration des RAP in das GMF kann ich mich nur anschließen. Ich führe diese Unausgewogenheit der implementierten Features auf das derzeitige Entwicklungsstadium von RAP Version 1.1 zurück. Bis dato war die Konzentration

auf die reibungslose Umsetzung der Kommunikationsmechanismen gerichtet, wobei die Widget Komponente ein wenig zu kurz kam. Betrachtet man aber die langfristigen Pläne des RAP, so ist eindeutig erkennbar, dass eine komplette Umsetzung der Draw2d API und der Eventlistener als RWT Adapter geplant ist. Ohne diese Weiterentwicklungen und zusätzlichen raschen Releasezyklen wird die Akzeptanz dieser Technik bei der Community mit Sicherheit nicht steigen.

Das GMF ist meiner Ansicht nach noch nicht offen genug, um auch die Flexibilität von Web-Anwendungen ausnützen zu können. Möglicherweise war dies auch nicht von Beginn an geplant, weil die Nachfrage nach webbasierten Modelleditoren in der Konzeptionsphase des GMF noch nicht gegeben war. Mittlerweile hat sich diese Ansicht gewandelt und wurde auch schon vom Projektteam des GMF aufgegriffen. Erst mit dieser Öffnung in Richtung neuer Konzepte in enger Kombination mit einer konsequenten Weiterentwicklung des RAP können mächtige interaktive Modellierungstools für das Internet konzipiert werden.

7 Literaturverzeichnis

[ApKr 2008] Frank Appel, Jochen Krause; RAP the Web; June 2008; http://www.eclipsecon.org/2008/index.php?page=sub/&id=199;

[BIG 2008] Business Informatics Group; September 2008; http://www.big.tuwien.ac.at/index.html;

[BrWi 2006] K. Brüssau, O. Widder; Eclipse, die Plattform; 2006; http://entwickler-press.de/ep/psecom,id,1,buchid,34.html;

[CeNa 2004] Gary Cernosek, Eric Naiburg; The Value of Modeling; June 2004; http://www.ibm.com/developerworks/rational/library/6007.html

[Co 2000] Chuck Connell; Open Source Projects Manage Themselves? Dream On; September 2000; http://www.chc-3.com/pub/manage_themselves.htm;

[DevX 2008] Riccardo Govoni; The Rich Get Richer: Eclipse Rich AJAX Platform Builds on RCP; September 2008; http://www.devx.com/webdev/Article/36101/1954

[DPunkt 2008] Java Multithreading ability; September 2008; http://www.dpunkt.de/java/Programmieren_mit_Java/Multithreading/1.html

[Ecke 2008] Andreas Ecker; Qooxdoo: Open Source Ajax Framework; June 2008; http://www.slideshare.net/ecker/qooxdoo-open-source-ajax-framework;

[ECL Con 2008] Anthony Hunter, Mohammed Mostafa; Extending your DSM by leveraging the GMF Runtime; September 2008; http://www.eclipsecon.org/2007/index.php?page=sub/&id=3624

[ECL Dwnld 2008] Download Eclipse for various Operating Systems; September 2008; http://www.eclipse.org/downloads/

[ECL Forum a 2008] Posting at the Eclipse Forum indicating problems with big GMF Models; September 2008; http://www.eclipse.org/newsportal/article.php?id=14835&group=eclipse.modeling.gmf#14835

[ECL Forum b 2008] Posting at the Eclipse Forum alert scalability issues at the GMF Models; September 2008; http://dev.eclipse.org/newslists/news.eclipse.modeling.gmf/msg02699.html

[ECL Mod 2008] Eclipse Modeling Proposals; September 2008; http://www.eclipse.org/proposals/modeling/

[EcO 2008] Eclipse Organisation; 2008; http://www.eclipse.org/org/;

[EMF 2008] Eclipse Modeling Framework; May 2008; http://www.eclipse.org/modeling/emf/;

[EMF Faq 2008] Eclipse Modeling Framework Faq; May 2008; http://wiki.eclipse.org/EMF-FAQ;

[Eqx 2008] Equinox; June 2008; http://www.eclipse.org/equinox/;

[FeMaCa 2007] José Eduardo Fernandes, Ricardo J. Machado, Joao Álvaro Carvalho; Model Driven Software Development for Pervasive Information Systems Implementation; 2007; http://ieeexplore.ieee.org/search/freesrchabstract.jsp?arnumber=4335251&isnumber=4335220&punumber=4335218&k2dockey=4335251@ieeecnfs&query=%28model+driven+software%29%3Cin%3Emetadata&pos=22;

[GEF 2008] Eclipse GEF Overview; May 2008; http://www.eclipse.org/gef/verview.html;

[GMF 2008] Graphical Modeling Framework; May 2008; http://www.eclipse.org/modeling/gmf/;

[GMF Dash 2008] Contributions to GMF; September 2008; http://dash.eclipse.org/dash/commits/web-app/summary.cgi?company=y&month=x&project=modeling.gmf

[GMF Faq 2008] Graphical Modeling Framework Faq; May 2008; http://wiki.eclipse.org/Graphical_Modeling_Framework_FAQ;

[GMF Plan 2008] GMF Release Plan; September 2008; http://www.eclipse.org/projects/project-plan.php?projectid=modeling.gmf

[GMF RCP 2008] GMF RCP Support, September 2008; http://wiki.eclipse.org/GMF_RCP_Support;

[GMF Run 2008] Graphical Modeling Framework Runtime; May 2008; http://www.eclipse.org/articles/Article-Introducing-GMF/article.html;

[GMF Tool 2008] GMF Tooling Tutorial; September 2008; http://help.eclipse.org/ganymede/topic/org.eclispe.gmf.doc/tutorials/tooling/step1.html;

[GMF Tut 2008] Graphical Modeling Framework Tutorial; May 2008; http://wiki.eclipse.org/index.php/GMF_Tutorial;

[Gronb 2008] Interview with GMF leading developer Richard Gronback; September 2008; http://eclipse.dzone.com/articles/the-gmf-project-an-interview-w

[HaOu 2001] A. Hars, S. Ou; Working for Free? – Motivations of Participating in Open Source Projects; 2001; http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/7255/20032/00927045.pdf?tp=&arnumber=927045&isnumber=20032;

[HeLLC 2008] HELINKS LLC Project implementing the GMF; September 2008; http://wiki.eclipse.org/Eclipse_DemoCamps_2008_-_Ganymede_Edition/Karlsruhe/Session_abstracts

[IBM GEF 2008] Overview of GEF; May 2008; http://www.ibm.com/developerworks/opensource/library/os-gef/;

[InfoQ 2008] Eclipse Ganymede: An in-depth look at RAP (Rich Ajax Platform)*;* September 2008; http://www.infoq.com/news/2008/06/eclipse-ganymede-rap

[JDoc a 2008] Object Reference to org.eclipse.rwt.lifecycle; September 2008; http://help.eclipse.org/ganymede/topic/org.eclispe.rap.help/help/html/reference/api/org/eclipse/rwt/lifecycle/package-summary.html

[JDoc b 2008] Object Reference to org.eclipse.swt.widgets.Display; September 2008; http://help.eclipse.org/ganymede/topic/org.eclispe.rap.help/help/html/reference/api/org/eclipse/swt/widgets/Display.html

[JDT 2008] Java Development Tools; 2008; http://www.eclipse.org/jdt;

[JFc 2008] JFace; June 2008; http://wiki.eclipse.org/index.php/JFace;

[Krause 2008] RAP Project Leading Developer; September 2008; http://www.eclipse.org/webtools/people/person.php?name=krause

[LeWr 2005] Grace A. Lewis, Lutz Wrage; Model Problems in Technologies for Interoperability: Model Driven Architecture; May 2005; http://www.sei.cmu.edu/publications/documents/05.reports/05tn022.html;

[Mosk 2008] Homepage of the Moskitt Application; September 2008; http://www.moskitt.org/eng/moskitt-tecnologia-empleada/

[NtE 2008] New to Eclipse; 2008; http://www.eclipse.org/home/newcomers.php;

[Ohloh 2008] Spreding of the GMF Projects; September 2008; https://www.ohloh.net/projects/5020

[OSD 2008] Open Source Definition; April 2008; http://www.opensource.org/docs/osd;

[OSGi 2008] OSGi Technology; June 2008; http://www.osgi.org/About/Technology;

[PDE 2008] Plug-In Development Environment; 2008; http://www.eclipse.org/pde;

[Qxd 2008] Qooxdoo: the new era of web development; June 2008; http://qooxdoo.org/about;

[Ra 2000] Eric Steven Raymond; The Cathedral and the Bazaar; 2000; http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/;

[RAP 2008] Rich Ajax Platform; May 2008; http://www.eclipse.org/rap/;

[RAP Abt 2008] Rich Ajax Platform; June 2008; http://www.eclipse.org/rap/about.php;

[RAP Abt 2008] About the RAP Project; September 2008; http://www.eclipse.org/projects/project_summary.php?projectid=technology.rap

[RAP Dash 2008] Contributions to RAP; September 2008; http://dash.eclipse.org/dash/commits/web-app/summary.cgi?company=y&month=x&project=technology.rap

[RAP Launch 2008] Launching a RAP Application; September 2008; http://help.eclipse.org/ganymede/topic/org.eclispe.rap.help/help/html/getting-started/launcher.html;

[RAP News 2008] RAP News Blog; September 2008; http://www.eclipse.org/rap/noteworthy/news_11.php;

[RAP Note 2008] RAP New and Noteworthy; September 2008; http://www.eclipse.org/rap/noteworthy/

[RAP Plan 2008] RAP Release Plan; September 2008; http://www.eclipse.org/projects/project-plan.php?projectid=technology.rap

[RAP Rel 2008] RAP Release Informations; September 2008; www.eclipse.org/project-slides/Eclipse_RAP_1_1_ReleaseReview.pdf

[RAP Scale 2008] RAP scalability issues; September 2008; http://www.innoopract.com/fileadmin/user_upload/Dokumente/Web-enabled_RCP_Applications_with_the_Rich_Ajax_Platform_pdf.pdf

[RAP Start 2008] RAP Hello World Demo; September 2008; http://help.eclipse.org/ganymede/topic/org.eclispe.rap.help/help/html/getting-started/hello-world.html

[RWT 2008] RAP Widget Toolkit; May 2008; http://wiki.eclipse.org/WidgetToolkit;

[Se 2003] Bran Selic; The Pragmatics of Model Driven Development; 2003; http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1231146;

[StHe 2008] Ralf Sternberg Rüdiger Hermann; Getting started with RAP development; June 2008; http://www.eclipsecon.org/2008/index.php?page=sub/&id=318;

[W3C XHR 2008] The XMLHttpRequest Object; June 2008; http://www.w3.org/TR/2008/WD-XMLHttpRequest-20080415/;

[We 2003] Markus Weyerhäuser; Die Programmierumgebung Eclipse; 2003; http://www.sigs.de/publications/js/2003/cebit/weyerhaeuser_JS_cebit_03.pdf;

[Wiki a 2008] Modellgetriebene Software Entwicklung; March 2008; http://de.wikipedia.org/wiki/Modellgetriebene_Softwareentwicklung;

[Wiki b 2008] Extreme Programming; March 2008; http://de.wikipedia.org/wiki/Extreme_programming;

[Wiki c 2008] Rational Unified Process; April 2008; http://de.wikipedia.org/wiki/Rational_Unified_Process;

[Wiki d 2008] Open Source; April 2008; http://en.wikipedia.org/wiki/Open_source;

[Wiki e 2008] Eclipse Modeling Framework; May 2008; http://en.wikipedia.org/wiki/Eclipse_Modeling_Framework;

[Wiki f 2008] Graphical Editing Framework; May 2008; http://en.wikipedia.org/wiki/Graphical_Editing_Framework;

[Wiki g 2008] Graphical Modeling Framework; May 2008; http://en.wikipedia.org/wiki/Graphical_Modeling_Framework;

[Wiki h 2008] Rich Ajax Platform; May 2008; http://de.wikipedia.org/wiki/Rich_Ajax_Platform;

[Wiki i 2008] Asynchronous JavaScript and XML; May 2008; http://de.wikipedia.org/wiki/Ajax_%28Programmierung%29;

[Wiki j 2008] Asynchronous JavaScript and XML; May 2008; http://en.wikipedia.org/wiki/Ajax;

[Wiki k 2008] Qooxdoo Framework; June 2008; http://de.wikipedia.org/wiki/Qooxdoo;

[Wiki l 2008] Equinox OSGi Framework; June 2008; http://de.wikipedia.org/wiki/Equinox_%28OSGi-Framework%29;

[Wiki m 2008] OSGi; June 2008; http://de.wikipedia.org/wiki/OSGi;

[Wiki n 2008] Mindmap Model; September 2008; http://de.wikipedia.org/wiki/Mind_Map;

[Wiki o 2008] Modell; September 2008; http://de.wikipedia.org/wiki/Modell

[Wiki p 2008] Modell in der Wirtschaftsinformatik; September 2008; http://de.wikipedia.org/wiki/Modell_(Wirtschaftsinformatik)

[Wrkb 2008] Workbench; June 2008; http://wiki.eclipse.org/Workbench;

8 Abbildungsverzeichnis

Abb. 1.1 Architektur des Web-Modelleditor Prototyps
Abb. 2.1 Abstraktionsebenen bei MDSD
Abb. 2.2 Pragmatiken bei MDSD
Abb. 3.1 Basis Architektur von Eclipse [We 2003]
Abb. 3.2 Komponentenstruktur von GMF [GMF Run 2008]
Abb. 3.3 Das Konzept des Model View Controller [IBM GEF 2008].
Abb. 3.4 Arbeitsfluss zur Erstellung eines grafischen Editors mit GMF [GMF Tut 2008]
Abb. 3.5 Eventhandler und Status im XMLHttpRequest Interface [W3C XHR 2008]
Abb. 3.6 Transferierte Daten bei Ajax vs. Klassicher Webapplikation [Wiki i 2008]
Abb. 3.7 a Ablauf bei der Client Server Kommunikation in klassischen Webanwendungen [Wiki i 2008]
Abb. 3.7 b Ablauf bei der Client Server Kommunikation bei Ajax Anwendungen [Wiki i 2008]
Abb. 3.8 Architektur von der RAP [RAP Abt 2008]
Abb. 5.1 Mindmap zu dieser Diplomarbeit (MS Visio)
Abb. 5.2 a Idealer Ansatz bei der Entwicklung eines Web Modelleditors
Abb. 5.2 b Lösungsansatz der praktischen Aufgabe dieser Diplomarbeit
Abb. 5.3 Modell des Mindmap Beispiels
Abb. 5.4 Auszug aus dem XML Inputdokument für GMF
Abb. 5.5 mindmap.gmfgraph
Abb. 5.6 mindmap.gmftool
Abb. 5.7 Einfacher Mindmap Editor als Eclipse Plugin
Abb. 5.8 MindmapDemo Klassendiagramm
Abb. 5.9 Auszug aus dem Einstiegspunkt
Abb. 5.10 a Interaktion bei einem Doppelklick auf ein Topic
Abb. 5.10 b Interaktion bei einem Klick (Down) auf ein Topic
Abb. 5.10 c Interaktion bei einem Klick (Up) auf ein Topic
Abb. 5.11 GUI der Mindmap Demo

9 Tabellenverzeichnis

Tab. 2.1 Nutzengewinne durch MDSD Konzepte

Tab. 4.1 Ergebnisse der Technologiebewertung

10 Abkürzungsverzeichnis

Ajax	Asynchronous JavaScript and XML
BS	Betriebssystem
DOM	Document Object Model
DSL	Domain Specific Language
EMF	Eclipse Modeling Framework
EPL	Eclipse Public License
GEF	Graphical Editing Framework
GL	Generation der Programmiersprachen
GMF	Graphical Modeling Framework
GUI	Graphical User Interface
IDE	Integrated Development Environment
JDT	Java Development Tools
MDA	Model Driven Architecture
MDSD	Model Driven Software Development
OMG	Object Management Group
OS	Open Source
OSI	Open Source Initiative
PDE	Plug-In Development Environment
RAP	Rich Ajax Platform
RCP	Rich Client Platform
RIA	Rich Internet Applications
RUP	Rational Unified Process
RWT	RAP Widget Toolkit
SC	Source Code
SOA	Service Oriented Architecture
SWT	Standard Widget Toolkit
UI	User Interface
UML	Unified Modeling Language
XML	Extensible Markup Language
XP	Extreme Programming

11 Fußnotenverzeichnis

1 http://www.omg.org

2 http://www.uml.org

3 Für den interessierten Leser können die Generationen der Programmiersprachen (GL) unter den nachfolgenden Links genauer Studieren:
1st GL - http://en.wikipedia.org/wiki/First-generation_programming_language
2nd GL; http://en.wikipedia.org/wiki/Second-generation_programming_language
3rd GL; http://en.wikipedia.org/wiki/Third-generation_programming_language
4th GL; http://en.wikipedia.org/wiki/Fourth-generation_programming_language
5th GL; http://en.wikipedia.org/wiki/Fifth-generation_programming_language

4 http://de.wikipedia.org/wiki/Round_Trip_Engineering;

5 http://opensource.org

6 Den genauen Wortlaut der Definitionen können unter http://www.opensource.org/docs/osd nachgelesen werden.

7 http://de.wikipedia.org/wiki/Intrinsische_Motivation

8 Eine vollständige Liste der Mitglieder des Eclipse Konsortiums kann unter http://www.eclipse.org/membership/exploreMembership.php nachgelesen werden.

9 Detaillierte Informationen findet man unter http://www.osgi.org

10 Informationen zu What you see is what you get Editoren findet man bei Wikipedia unter: http://de.wikipedia.org/wiki/WYSIWYG

11 Eine ausführliche Beschreibung aller Methoden, Attribute, Eventhandler und Fehlermeldungen findet man auf der Seite des W3C: http://www.w3.org/TR/XMLHttpRequest/

12 Diese Fragen wurden von Dr. Harald Kühn, Mitglied der Geschäftsführung von BOC Information Systems GmbH, zur Verfügung gestellt.

13 Interessanter wäre sicherlich eine weltweite Analyse, das allein könnte aber schon eine eigene Arbeit füllen.

14 http://www.acceleo.org

15 http://www.obeo.fr

16 http://www.innoopract.com

17 http://www.oneandone.com

18 http://www.cas.de

19 Die Klasse `MyClickListener.java` ist nur zum Teil implementiert. Gründe dafür werden im Kapitel 5.5 näher erläutert.

20 Durch Eingreifen in den vom RAP erzeugten Ajax Code besteht prinzipiell die Möglichkeit zusätzliche und qualitativ höherwertige interaktive Komponenten zu programmieren. Dieses würde aber den Rahmen der Diplomarbeit und die des Prototyps sprengen.