

Kobold

Product Line Management System

Version 1.0



Abschlussbericht

Werkbold

Versionsgeschichte

- Version 1.0

Inhaltsverzeichnis

1. Einleitung	5
1.1. Zweck des Dokuments	5
1.2. Hintergrund des Projekts	5
1.3. Anforderungen	6
2. Projektergebnisse	9
2.1. Kobold	9
2.2. Dokumente	10
2.2.1. Angebot	10
2.2.2. Spezifikation	10
2.2.3. Entwurf	10
2.2.4. Handbuch	11
2.2.5. Interne Dokumente	11
3. Projektorganisation	13
3.1. Projektrahmen	13
3.2. Das Team	13
3.2.1. Rollen	13
3.3. Schulungen	15
4. Projektablauf	17
4.1. Prozessmodell	17
4.2. Projektplan	18
4.3. Analyse und Angebot	18
4.4. Spezifikation und Entwurf	18
4.5. Handbuch	19
4.6. Auslieferung	19
4.7. Nachbesserung	19
4.8. Abschlussbericht und Präsentation	19
4.9. Arbeitszeitverteilung	20
4.10. Werkzeuge	20
5. Statistische Analyse	23

5.1. Quellcodeumfang	23
5.2. Umfang der Module	24
5.3. Aufwand für die einzelnen Phasen	25
5.4. Entwicklerkooperation	25
5.5. Arbeitzeit	26
6. Persönliche Berichte	29
6.1. Necati Aydin	30
6.1.1. Tätigkeiten	30
6.1.2. Beurteilung	31
6.2. Armin Cont	32
6.2.1. Tätigkeiten	32
6.2.2. Beurteilung	34
6.3. Bettina Druckenmüller	35
6.3.1. Tätigkeiten	35
6.3.2. Beurteilung	37
6.4. Anselm R. Garbe	38
6.4.1. Tätigkeiten	38
6.4.2. Fazit	41
6.5. Michael Grosse	43
6.5.1. Tätigkeiten	43
6.5.2. Beurteilung	44
6.6. Tammo van Lessen	46
6.6.1. Tätigkeiten	46
6.6.2. Beurteilung	48
6.7. Martin Plies	49
6.7.1. Tätigkeiten	49
6.7.2. Beurteilung	51
6.8. Oliver Rendgen	52
6.8.1. Tätigkeiten	52
6.8.2. Beurteilung	54
6.9. Patrick Schneider	55
6.9.1. Tätigkeiten	55
6.9.2. Fazit	56
A. Erklärung gemäß Prüfungsordnung	59

1. Einleitung

1.1. Zweck des Dokuments

Dieses Dokument beschreibt das Studienprojekt "Kobold", das von Anfang Oktober 2003 bis Anfang November 2004 an der Universität Stuttgart am Institut für Softwaretechnologie (ISTE) durchgeführt wurde. Das Projekt wurde von neun Softwaretechnik Studenten - Necati Aydin, Anselm Garbe, Tammo van Lessen, Michael Grosse, Armin Cont, Patrick Schneider, Oliver Rendgen, Martin Plies und Bettina Druckenmüller - im Rahmen eines Studienprojekts A durchgeführt. Im Folgenden wird ein Überblick über die erzielten Ergebnisse, den Projektverlauf und die persönlichen Eindrücke der Teilnehmer gegeben.

1.2. Hintergrund des Projekts

Schlüsselerfolgsfaktoren in der Softwaretechnik sind kurze Time-to-market Zyklen, hohe Produktqualität und niedrige Kosten. Das Erzielen dieser scheinbar unvereinbaren Ziele wird durch systematische Wiederverwendung während der Entwicklung von Software möglich.

Kernpunkt des Produktlinienansatzes ist die systematische Erfassung von Unterschieden und Gemeinsamkeiten der verschiedenen Produkte und einer expliziten Abbildung dieser Aspekte in die Softwarearchitektur. Diese Grundidee hat Auswirkungen für alle Schritte im Produktentwicklungsprozess und - idealerweise - auch darüber hinaus in der Produktplanung und -strategie.

Um eine effiziente Produktlinienentwicklung in der Softwareentwicklung zu etablieren, ist es wichtig, dass diese Produktstrategie konsequent umgesetzt wird.

Hauptmotivation für die Durchführung einer Produktlinienentwicklung ist, später einzelne Produkte durch einfache Kombination von angepassten Core-Assets erstellen zu können. Eine Produktlinie ist also ideal konfiguriert, wenn die Produktlinien-Architektur für alle möglichen Produkte hinreichend spezifiziert ist. Der Produktlinien-Ingenieur hat genau diese Aufgabe. Er bestimmt die Architektur in dem er Core-Assets definiert und deren Grundbeziehungen zueinander spezifiziert (Scoping und Domain-Engineering). Soll nun ein neues Produkt in die Produktlinie aufgenommen werden (Application En-

gineering), so muss es dieser Architektur folgen. Es werden dazu bereits existierende Varianten der Core-Assets mit neuen Varianten kombiniert und dem Produkt-Ingenieur übergeben, der die Verantwortung für das Produkt übernimmt. Dieser hat dafür Sorge zu tragen, dass die Entwicklung an dem Produkt nicht die Architektur der Produktlinie verletzt.

In dieser Hierarchie ist es wichtig, dass bestimmte Vorgänge nach festen Regeln kommuniziert werden. Wenn z.B. in einem Produkt eine bereits existierende Variante eines Core-Assets verwendet wird, darf diese nicht von dem Produkt-Entwickler verändert werden. Veränderungen sind hier nur dem zuständigen Core-Asset-Entwickler gestattet. Dieser muss abwägen, ob die Veränderung sinnvoll ist, da er die Verantwortung für sein Modul, welches evtl. auch noch von anderen Produkten verwendet wird, hat. Lehnt er dies ab, so muss der Produkt-Ingenieur eine produktspezifische Variante dieses Core-Assets entwickeln lassen. Dieser Vorgang ist sehr komplex, da mehrere Personen mit ihren Entscheidungen daran beteiligt sind. Um eine Produktlinie konsequent durchsetzen zu können, müssen diese Arbeitsabläufe spezifiziert sein.

Es bietet sich nun an, die Entwicklung von Produktlinien mit Werkzeugen zu unterstützen. Damit kann man sowohl das Configuration Management vereinfachen, sowie die oben erwähnten Arbeitsabläufe modellieren und damit die Kommunikation und die Entwicklung erleichtern.

1.3. Anforderungen

Aus dem im vorherigen Abschnitt skizzierten Projekthintergrund ergaben sich folgende Anforderungen im Detail, deren Umsetzung in einem Angebotsdokument verbindlich mit dem Auftraggeber vereinbart wurden:

- Produktlinie erstellen und bearbeiten vom Produktlinien-Ingenieur
- Produkt erstellen, bearbeiten und löschen
- graphische Darstellung von Produkt- und Produktlinienarchitekturen
- Vergleich der Produkt- und Produktlinienarchitekturen in Form eines Schnittgrafen
- Erfassung von Metadaten für jede Komponente und Beziehung
- Konsistenzprüfungen durchführen
- Erzeugung eines Überblick-Dokuments mit Metadaten und Abhängigkeiten
- Produkt-Ingenieure anlegen
- Verwaltung von Core Assets
- Produkt-Entwickler verwalten

- Produktlinien und Produkte nach Metadaten durchsuchen
- Auslösen von Workflows
- Inhalt von Repositories anzeigen
- Arbeitsverzeichnis auschecken
- Arbeitsverzeichnis einchecken

2. Projektergebnisse

2.1. Kobold

Nach Abschluß der Entwicklungsarbeiten ist ein System implementiert, das, wie ursprünglich geplant, drei ausführbare Anwendungen umfasst:

Der Kobold-Server ermöglicht die zentrale Verwaltung und Persistierung von Produktlinien- und Benutzerdaten. Die im Server realisierten Sicherheitsmechanismen erzwingen dabei die Durchsetzung der festgelegten Berechtigungen und stellen die Konsistenz des Produktlinien- und Benutzerdatenmodells sicher, selbst wenn mit Hilfe der Clients der aktive Versuch der Sabotage unternommen wird.

Die Kommunikation zwischen Clients und Server erfolgt über eine verschlüsselte SSL-Verbindung. Darüberhinaus können die Administrationsfunktionen des Servers durch ein zusätzliches Administrationspasswort geschützt werden.

Die zweite ausführbare Anwendung des Kobold-Systems ist der User-Client. Über ihn können sämtliche Aufgaben des Kobold-Systems mit Ausnahme von Administrationsmaßnahmen durchgeführt werden. Der User-Client ist vollständig in die Eclipse-Plattform integriert und technisch von dieser abhängig. Pro ausgeführter Instanz kann jeweils ein Benutzer am System arbeiten, der sich mittels Benutzernamen und Kennwort am Server anmelden muss.

Die Benutzerschnittstelle ist dabei durchgehend defensiv ausgelegt, d.h. Funktionen des Clients sind nur dann anwählbar, wenn der jeweilige Benutzer dazu auch berechtigt ist und die jeweiligen Voraussetzungen dazu erfüllt sind (so kann beispielsweise ein Projekt erst angelegt werden, nachdem die Verbindung zum Server erfolgreich aufgebaut und die Berechtigung durch das System bestätigt wurde). Zusätzlich zu den serverseitigen Sicherheitsmechanismen verhindert also bereits die clientseitige Benutzerschnittstelle die beabsichtigte oder unbeabsichtigte Sabotage der Systemintegrität. Ferner wird dem Benutzer vom System selbst durch die Integration des Handbuchs im Client zusätzliche Hilfestellung bei den alltäglichen Aufgaben bei der Arbeit mit Kobold geboten (beachte dazu auch das Kapitel zum Handbuch).

Die dritte ausführbare Anwendung ist ein schlanker, text-basierter Client, der ausschließlich der Administration des Servers dient. Die Benutzer- und Administrations-Schnittstellen des Servers sind vollständig entkoppelt und die beiden Client-Applikationen voneinander unabhängig. Diese Aufteilung dient der Sicherheit der Systemintegrität. Trotz der

extensiven Sicherheitsmechanismen stellen die Administrationsfunktionen des Kobold-Systems insofern ein Sicherheitsrisiko dar, als sie bei bewusster oder fahrlässiger Verwendung die Arbeit der Benutzer potentiell erschweren können.

Zu den Administrationsaufgaben gehören das Anlegen und Entfernen von Produktlinien, deren Zuordnung zu physischen VCM-Repositories, die Registrierung und datenkonsistenzwährende Entfernung von Benutzern, sowie deren Zuordnung zu Produktlinien. Des weiteren können zahlreiche Informationen zu den auf dem jeweils administrierten Server befindlichen Daten abgefragt werden.

2.2. Dokumente

Neben den zum System gehörenden Code-Dokumenten sind im Laufe des Projekts folgende Dokumente entstanden:

2.2.1. Angebot

In einem knapp 50-seitigen Angebot wurden die Anforderungen und ein Projektlaufplan mit dem Auftraggeber vereinbart. Das Angebot beinhaltet des Weiteren eine Risikoanalyse und eine Beschreibung des Prozessmodells und der geplanten Projektorganisation. In einem angehängten Begriffslexikon werden Begriffe der Problemdomäne definiert, die zur Kommunikation mit dem Auftraggeber und innerhalb des Teams verwendet wurden.

2.2.2. Spezifikation

Da Kobold mit einem iterativen Modell entwickelt wurde, wuchs auch die Spezifikation von Iteration zu Iteration an. Die Spezifikation I legte in ihren 35 Seiten zunächst einmal das Rahmengerüst von Kobold fest. In den darauffolgenden Iterationen wurden die fehlenden Use Cases spezifiziert und umgesetzt. Zum Schluss umfasste die Spezifikation von Kobold knapp 55 Seiten.

2.2.3. Entwurf

Nach Fertigstellung der ersten Spezifikation wurde ein umfassender Entwurf unter Zuhilfenahme der grafischen Entwurfssprache UML erstellt. Dabei wurde - wenn möglich - auf bekannte und bewährte Entwurfsmuster zurückgegriffen, unter anderem auf das Model-View-Controller- und Singleton-Pattern.

Die Unterstützung der später verwendeten Programmiersprache Java für abstrakte Klassenschnittstellen (Interfaces) wurde extensiv dazu verwendet, bei kritischen, vor allem aber bei den Kommunikationsmodulen zwischen Server und Clients, eine günstige Ausgangssituation für spätere Iterationen und Wartungsarbeiten zu schaffen. Dazu wurden diese Schnittstellen bereits im ersten Entwurf sehr detailliert festgesetzt. Dieses Vorgehen hat sich sehr gut bewährt.

Die Aufgaben für die Client- und Serveranwendungen wurden ebenfalls bereits im ersten Entwurf klar aufgeteilt. Neben einer Serveranwendung wurden zwei Clientanwendungen entworfen; eine zur Benutzung des Systems und eine davon unabhängige zur Administration des Servers. Die dabei festgesetzte, strenge Aufgabenteilung wurde in den späteren Iterationen mit lediglich marginalen Änderungen, wie beschrieben, umgesetzt.

2.2.4. Handbuch

Dem Benutzer steht ein rund 120-seitiges, mit Screenshots illustriertes, Handbuch zur Verfügung. Das Handbuch enthält einen Referenzteil, der die Programmdialoge beschreibt und ein Tutorial, das dem unerfahrenen Benutzer anhand eines Beispiels einen leichten Einstieg in die Bedienung von Kobold ermöglicht.

Darüber hinaus wurde der Tutorial-Teil des Handbuchs in Form von Eclipse-“Cheat-Sheets” direkt in den User-Client des Systems integriert, so dass dessen Benutzer die darin enthaltenen Informationen direkt aus dem System heraus nutzen können ohne ihre Arbeitsumgebung dazu verlassen zu müssen.

2.2.5. Interne Dokumente

Einige Dokumente wurden nur innerhalb des Teams verwendet und nicht an den Auftraggeber ausgeliefert. Zu nennen sind hier ein Anforderungsdokument, Richtlinien zu Organisation und Qualitätssicherung sowie Testpläne und -berichte.

Dazu gehörten ferner die im Rahmen der Qualitätssicherungsmaßnahmen erstellten Dokumente, konkret schriftliche Inspektionsfeedbacks, Einträge im von der Berrios-Plattform bereitgestellten Bugtrackingsystem, sowie sämtliche Walkthrough-, Review- und Sitzungsprotokolle.

3. Projektorganisation

3.1. Projektrahmen

Das Projekt wurde im Rahmen eines Studienprojekts A an der Universität Stuttgart durchgeführt. Projekteigentümer war die Abteilung Programmiersprachen und Übersetzerbau des Instituts für Softwaretechnologie.

Als Hilfestellung für die Projektmannschaft wurden begleitend zum Projekt selbst ein vom Projekteigentümer durchgeführtes Seminar zur Produktlinienentwicklung sowie eine komplette Vorlesung zur Entwicklungstechnik des Reengineering durchgeführt, an der alle Mitglieder der Projektmannschaft teilnahmen.

Aufgrund der für ein Studienprojekt untypisch hohen Verfügbarkeit an studentischen Arbeitskräften wurde die zu entwickelnde Software parallel von zwei neunköpfigen Entwicklerteams umgesetzt, die - soweit dies vom Projekteigentümer gestattet wurde - trotz der Konkurrenzsituation kooperieren und sich durch stetigen Erfahrungsaustausch gegenseitig motivieren konnten.

3.2. Das Team

Der zu diesem Bericht gehörige Auftragnehmer bestand aus einem der beiden Entwicklerteams, nämlich demjenigen, welches unter dem Namen Werkbold firmierte. Die zum Team gehörenden Studenten befanden sich zu Beginn des Projekts im fünften bzw. siebten Studiensemester. Das Bild 3.1 zeigt von links nach rechts beginnend in der hinteren Reihe: Armin Cont, Bettina Druckenmüller, Necati Aydin, Tammo van Lessen, Patrick Schneider, Oliver Rendgen, Michael Grosse, Anselm Garbe und Michael Plies.

3.2.1. Rollen

Auf die Vergabe von formalen Rollen wurden weitgehend verzichtet. Die Zuteilung der Aufgaben erfolgte gemeinschaftlich, soweit möglich auf Basis freiwilliger Übernahme, in jedem Fall aber unter Beachtung der persönlichen Fähigkeiten, während gemeinsamer Team-Treffen. Wurde eine Aufgabe von keinem der Entwickler übernommen, so wurde



Abbildung 3.1.: Das Werkbold Team

diese vom Projektleiter zugeteilt.

Projektleiter

Der Projektleiter vertrat die Interessen des Teams nach außen gegenüber dem Auftraggeber, koordinierte die Projektplanung, überwachte den Entwicklungsfortschritt und forderte den Informationsaustausch im Team und zum Auftraggeber. Die Rolle des Projektleiters wurde von Anselm Garbe übernommen.

Konfigurationsmanager

Der Konfigurationsmanager setzte die Entwicklungsumgebung auf und stellte die Verwaltung und Versionierung von Dokumenten und des Source-Codes sicher. Die Rolle

des Konfigurationsmanagers wurde von Oliver Rendgen übernommen.

Dokumentationsmanager

Der Dokumentationsmanager war der technischer Berater für alle Dokumente, die in der Entwicklung entstanden. Er war für die Erstellung des Handbuchs verantwortlich. Die Rolle des Dokumentationsmanagers wurde von Bettina Druckenmüller übernommen.

QS-Manager

Der QS-Manager war verantwortlich für die generelle Organisation und Überwachung der Qualitätssicherungsarbeiten (vgl. dynamische Rolle "Qualitätssicherer"). Unter sein Aufgabenbereich fiel insbesondere die Vermeidung von Widersprüchen bei den zu erstellenden QS-Dokumenten und die Sicherstellung deren Einhaltung durch das Team. Die Rolle des QS-Manager wurde von Armin Cont übernommen.

Entwickler

Die Aufgabe der Entwickler beinhaltete die Erstellung des Quellcodes und das Verfassen der Dokumente. Die Rolle der Entwickler wurde von allen Teilnehmern belegt.

3.3. Schulungen

Aufgrund der unterschiedlichen Vorkenntnisse der einzelnen Teammitglieder wurde der naheliegende Versuch unternommen, zumindest die gröbsten Unterschiede durch Schulungen auszugleichen, um ein gemeinsames Knowhow-Level für die weitere Zusammenarbeit im Team zu erreichen.

Die Schulungen wurden dabei von einem oder zwei Teammitgliedern durchgeführt, die in dem jeweiligen Themenbereichen besonders qualifiziert bzw. erfahren waren. Die Durchführung der Schulungen erfolgte standardisiert unter Zuhilfenahme von durch die Referenten vorbereiteten Schulungsmaterialien (Folien, Fallbeispiele, Übungen etc.) mit abschließender schriftlicher Ausarbeitung, wenn dies von den Geschulten bzw. den Referenten im jeweiligen Fall für notwendig erachtet wurde.

Nach den Schulungen waren die Referenten die ersten Ansprechpartner für Probleme, die den behandelten Themenbereichen zugeordnet werden konnten.

In den durchgeführten Schulungen wurde ein breites Spektrum unterschiedlicher Themenbereiche abgedeckt, vom Umgang mit einzelnen Werkzeugen, über Entwurfstechniken

niken, bis hin zu den eingesetzten Entwicklungs- und Dokumentationssprachen:

- 02.02.04 Eclipse SDK (Patrick Schneider)
- 02.02.04 Basics: ssh, cvs (mit Eclipse), Vorstellung der Berlios.de-Services (cvs, ftp, xdocs) und Duckforge-Time-Management (Oliver Rendgen)
- 04.02.04 Tracing (Open-Source) Projects with Maven (Tammo van Lessen)
- 23.02.04 Design Patterns (Armin Cont, Michael Grosse)
- 23.02.04 Eclipse SWT/GEF (Tammo van Lessen)
- 25.03.04 XML-RPC Kommunikation über HTTPS (Anselm Garbe)
- 25.03.04 Drools Regelengine (Bettina Druckenmüller) und GXL (Martin Plies)
- 25.03.04 iText (Necati Aydin)

4. Projektablauf

4.1. Prozessmodell

Es wurde ein evolutionäres Vorgehensmodell¹ angewendet, das sich in mehrere Iterationen unterteilte. Jede Iteration entsprach dem Wasserfall-Vorgehensmodell. Das Vorgehensmodell ähnelt dem Spiralmodell nach B.W. Boehm. Durch die Anwendung dieses Vorgehensmodells ergaben sich folgende Vorteile:

- Der Umfang der iterativen Spezifikationen und Entwürfe war wesentlich kleiner als bei einer Wasserfall-Gesamtentwicklung. Der inkrementelle Umfang resultierte in einer geringeren Fehleranfälligkeit und garantierte somit einen höheren Qualitätsstandard.
- Der Entwicklungsfortschritt konnte somit vom Auftraggeber und vom Entwicklungsteam realistischer eingeschätzt werden, da regelmäßig eine neue Auslieferung durchgeführt wurde, die vom Auftraggeber abgenommen werden musste.
- Durch die Wahl der J2SE Technologie fand die Entwicklung nach modernen *Design Patterns*² und objektorientierten Paradigmen statt. Im Gegensatz zu imperativen Technologien wurde dadurch eine geringere Kopplung der Software-Komponenten und somit eine bessere Zerlegung des Systems erreicht. Eine derart leicht durchführbare Zerlegung des Systems nach funktionalen Bestandteilen ermöglichte besonders das evolutionäre Vorgehen.
- Es konnten neue oder geänderte Anforderungen in einer späteren Iteration wesentlich flexibler im Entwicklungsprozess berücksichtigt und realisiert werden.

Ein entscheidener Unterschied zu konventionellen Vorgehensmodellen war der höhere Kommunikationsaufwand zwischen dem Auftraggeber und dem Entwicklungsteam. Dieser höhere Kommunikationsaufwand zeichnete sich in jedem Fall positiv für den Auftraggeber aus, da Anforderungen schon aufgrund der Quantität der Kommunikation besser einfließen konnten. Außerdem hatte der Auftraggeber die Möglichkeit, nach jeder abgeschlossenen Iteration, auch größere Änderungswünsche einzubringen.

¹Vgl. Literatur, u.a. J. Ludewig, Sammlung von Kapiteln zum Software Engineering

²Vgl. Literatur, Design Patterns, Erich Gamma

4.2. Projektplan

Bereits während des Vorprojekts wurde ein grober Plan für das Gesamtprojekt erstellt, der Bestandteil des Angebots wurde. Dieser grobe Projektplan wurde zu Beginn der ersten Iteration gründlich überarbeitet und sehr detailliert festgelegt.

Da dem Kunden nach Abschluß jeder Iteration die Möglichkeit gegeben wurde, aktiv auf die Entwicklung Einfluß zu nehmen und dieser auch oft davon Gebrauch machte, wurde der ursprüngliche Projektplan zu Beginn jeder neuen Iteration an die vom Kunden neu geschaffenen Realitäten angepasst.

4.3. Analyse und Angebot

Während der Anforderungsanalyse wurde in mehreren Treffen die Anforderungen des Kunden ermittelt. Die Ergebnisse wurden in einem Angebot dokumentiert, das dem Kunden zur Abnahme vorgelegt wurde. Das Angebot beinhaltete eine Beschreibung des geplanten Projekts.

4.4. Spezifikation und Entwurf

Für die Erstellung der Spezifikation wurde zu Beginn der ersten Iteration der mit Abstand größte Aufwand getrieben. Die im Vorfeld gesammelten Anforderungen wurden dabei so umfassend wie möglich in die Spezifikation eingebunden, wobei der Kunde in Einzelfragen noch einige Male konsultiert wurde. In den nachfolgenden Iterationen wurde die Spezifikation angepasst, wenn dies notwendig wurde; die dadurch eingebrachten Änderungen waren allerdings erwartungsgemäß von sehr geringem Umfang.

Entsprechend der herausragenden Bedeutung der ersten Spezifikation, wurde diese allen zur Verfügung stehenden Qualitätssicherungsmaßnahmen unterzogen, von regelmäßigen schriftlichen Statements, über zwei formale Reviews, bis hin zu einem abschließenden protokollierten Walkthrough. Die Spezifikation wurde später die Grundlage der ersten Abschlusstests des Kobold Systems.

Für den Architekturentwurf wurde ein ähnliches Vorgehen gewählt, das heißt der Aufwand für die Erstellung des Entwurfs während der ersten Iteration war deutlich größer, als der Anpassungsaufwand in den späteren Iterationen. Als Vorbereitung wurde unmittelbar vor Beginn der Entwurfsarbeiten eine eigens dafür anberaumte Schulung zum Thema Entwurfstechniken und Design Patterns durchgeführt und bei der Erstellung des Entwurfs wurde großer Wert auf die Verwendung diverser Entwurfsmuster gelegt, um eine möglichst kopplungsfreie und wartungsfreundliche Struktur im System zu realisie-

ren.

Der Entwurf wurde in zwei großen Schritten erstellt. Ein relativ kleines Team, zu dem auch die Schulungsleiter der oben angesprochenen Schulung gehörten, erstellte zunächst einen Grobentwurf, der intensiv vom restlichen Team geprüft und nach anschließender Ausreifung durch die sehr detaillierte Ausgestaltung der wichtigsten Module und aller Schnittstellen soweit verfeinert wurde, dass er während der später folgenden Implementierungsarbeiten mit großem Gewinn eingesetzt werden konnte.

4.5. Handbuch

Das Handbuch wurde von einem einzelnen Entwickler parallel zu den Iterationen erstellt und befand sich somit immer auf dem neuesten Stand. Es enthält eine Beschreibung der graphischen Benutzeroberfläche und der Funktionalität von Kobold, sowie ein Tutorial. Das Handbuch wurde in einem Review geprüft und zusammen mit Kobold ausgeliefert.

4.6. Auslieferung

Die während der Präsentation ausgelieferten Ergebnisse sind in den beiden vorhergehenden Abschnitten beschrieben. Die Auslieferung markierte mit der Erfüllung der Projektanforderungen das vorläufige Ende des Projekts.

4.7. Nachbesserung

Im Anschluss an die Auslieferung Ende August begann eine 3-monatige Nachbesserungsphase. Es sollten lediglich Fehler behoben werden, aber keine neuen Anforderungen umgesetzt werden. Nach Behebung der meisten Fehler wird Kobold am 26.11.2004 endgültig ausgeliefert.

4.8. Abschlussbericht und Präsentation

Der Abschlussbericht ist das vorliegende Dokument. Die Abschlusspräsentation ist für den 26.11.2004 geplant. Zusammen mit einem ausgedruckten Exemplar des Abschlussberichts wird eine CD, die sämtliche Projektergebnisse beinhaltet, ausgeliefert werden.

4.9. Arbeitszeitverteilung

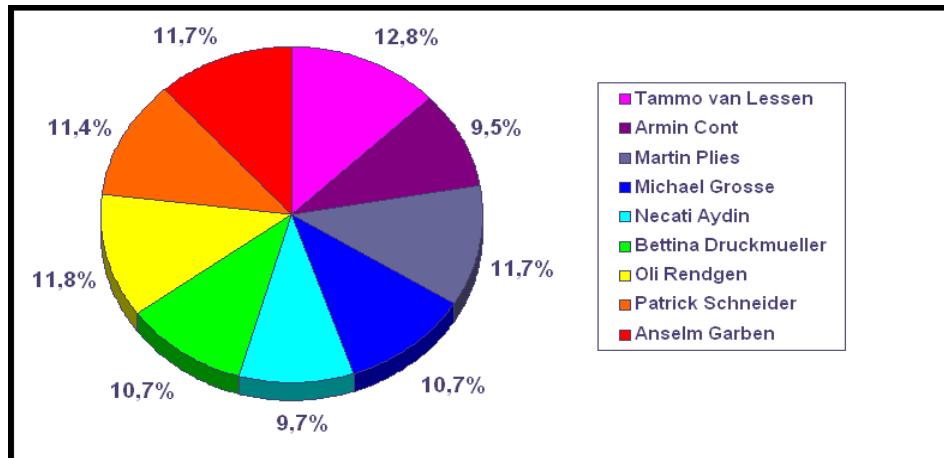


Abbildung 4.1.: Zeitliche Aufwandsverteilung auf die Entwickler

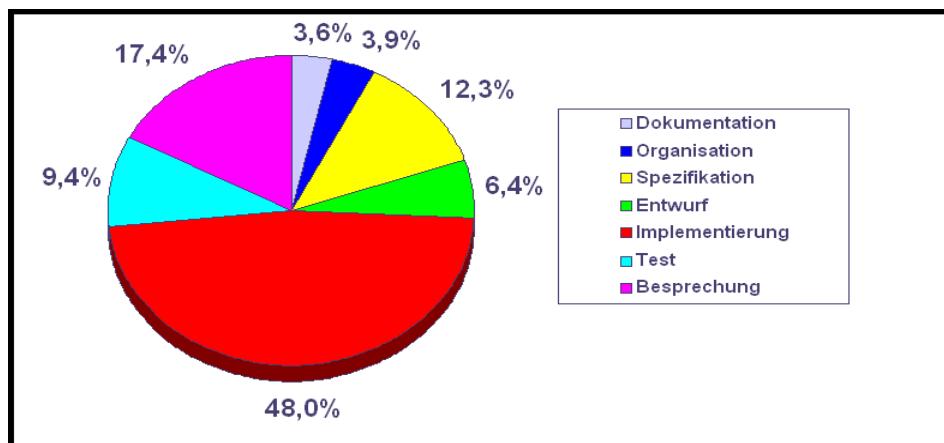


Abbildung 4.2.: Zeitliche Aufwandsverteilung auf die Aktivitäten

4.10. Werkzeuge

Das vorherrschende Kommunikationsmedium im Projekt war die Email; dabei lief das Gros der versendeten Emails über zwei eigens für dieses Projekt eingerichtete Verteilerlisten, eine Team-interne und eine das Team und den Projekteigentümer umfassende.

Zur Erstellung von Nicht-Code-Dokumenten aller Art wurde die Dokumentenerzeugungssprache TeX eingesetzt sowie der UML-Diagrammer ArgoUML für Entwurfsdokumente.

In einigen Ausnahmefällen wurden das Präsentationswerkzeug Powerpoint (für einige Schulungen) sowie diverse einfache Texteditoren eingesetzt.

Zur Erstellung von Codedokumenten wurden die Editoren der integrierten Java Entwicklungsumgebung der Eclipse-Plattform eingesetzt. Sämtliche Dokumente wurden über ein zentrales CVS-Repository verwaltet, welches über die BerliOS-Plattform zur Verfügung gestellt wurde.

5. Statistische Analyse

5.1. Quellcodeumfang

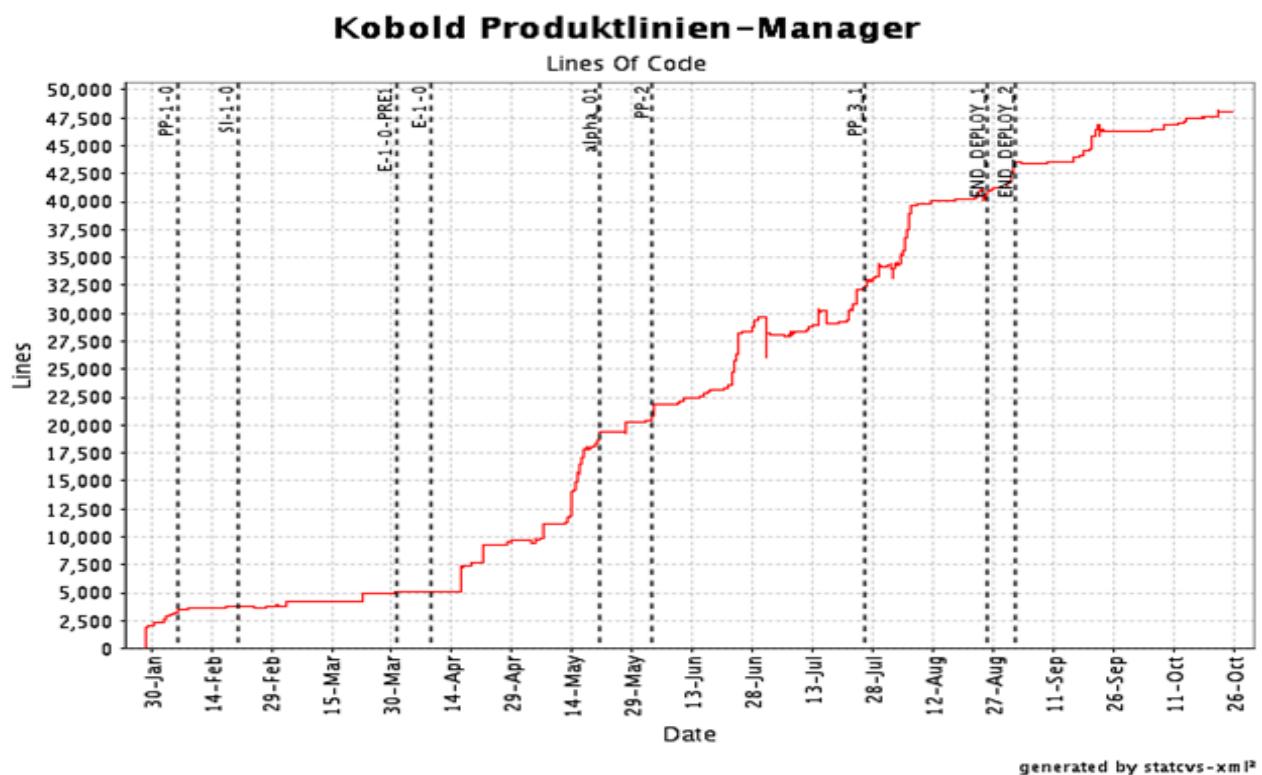


Abbildung 5.1.: Quellcode Zeilenanzahl über die Zeit

Das Diagramm zeigt die gesammelte Zeilenanzahl des Quellcodes inklusive aller Kommentar- und TeX-Zeilen über die Laufzeit des Projekts. Die vertikalen gestrichelten Linien geben die Versionsmarkierungen an.

An dem Diagramm lassen sich auch die Iterationen ablesen. Dabei stehen die Markierungen mit der Bezeichnung PP für einen jeweils neuen Projektplan und somit für eine neue Iteration. Die End-Deploy Markierungen zum Ende des Projektes stehen für einen weiteren Übergang in eine verkürzte Iteration, in der nur noch implementiert und getestet wurde.

Es ist deutlich zu sehen, wie die Anzahl der ASCII-Zeilen nach Abschluss des ersten Entwurfes (Markierung E 1-0) rapide zu steigen beginnt. Die drei flachen Sektoren im August, September und Oktober ergaben sich aufgrund des Umstands, dass die meisten Teammitglieder während der Prüfungszeit weniger für das Projekt arbeiten konnten.

5.2. Umfang der Module

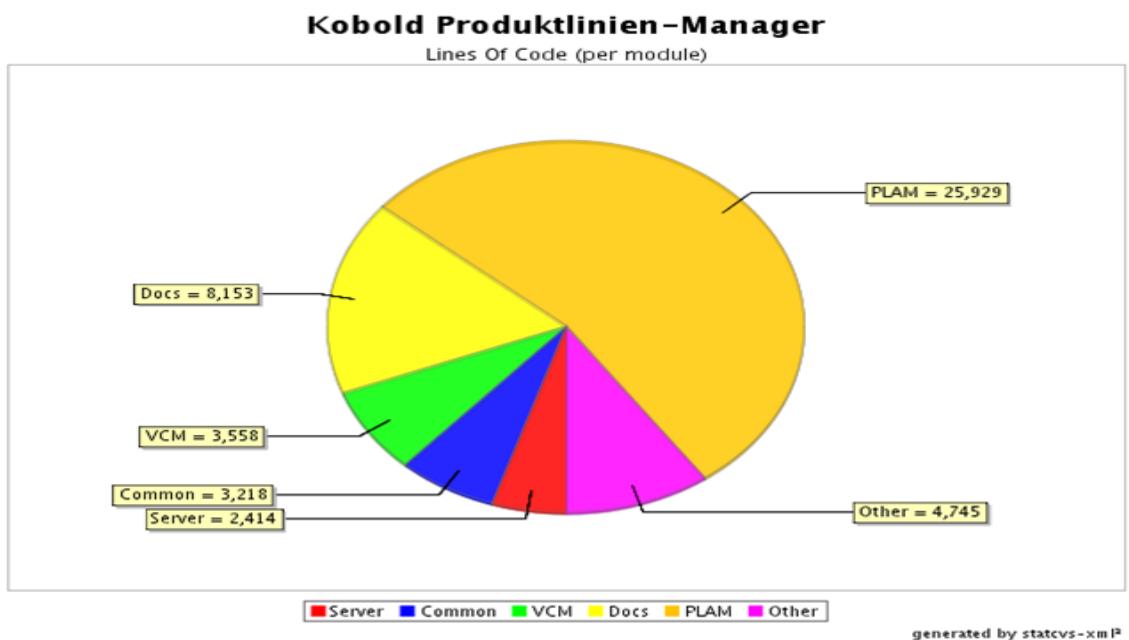


Abbildung 5.2.: Umfang der Module in Codezeilen

Dieses Diagramm zeigt Größenverhältnisse der Module, gemessen in Quellcodezeilen. Dabei fällt das PLAM-Modul besonders groß aus, da sich in diesem Modul die gesamte GUI-Implementierung befindet. Die Größen der VCM, Common und Server-Module spiegeln nicht den Aufwand wieder.

Die Daten des Tests befinden sich in dem Modul "other". Allerdings ist zu berücksichtigen, dass die meisten Testdaten dort nicht inbegriffen sind und der automatisierte Test in den Modulen des jeweils zu testenden Teiles enthalten sind.

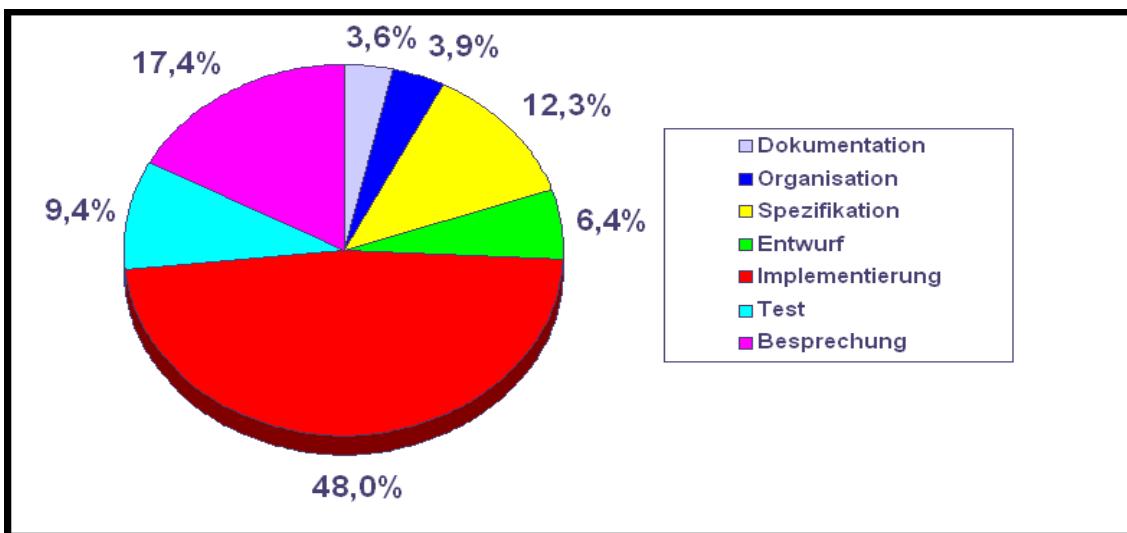


Abbildung 5.3.: Aufwand pro Phase gemessen in Arbeitszeit

5.3. Aufwand für die einzelnen Phasen

Das Diagramm zeigt den prozentuellen Anteil der Arbeitszeit, der für die einzelnen Phasen aufgebracht wurde. Auffallend ist der große Anteil der Implementierung mit 48 Prozent. Die Phase des Entwurfs ist im Vergleich zur Spezifikation etwas kurz ausgefallen. Möglicherweise war deshalb der Aufwand für die Implementierung so groß.

5.4. Entwicklerkooperation

Dateiname	Entwickleranzahl
kobold/server/SecureKoboldWebServer.java	7
kobold/common/data/Product.java	7
kobold/client/plam/editor/dialog/AssetConfigurationDialog.java	7
kobold.client.plam/plugin.xml	7
kobold/common/data/WorkflowMessage.java	6
kobold/common/data/Productline.java	6
kobold/client/vcm/preferences/VCMPreferencePage.java	6
kobold/client/vcm/communication/ScriptServerConnection.java	6
kobold/client/plam/model/productline/Variant.java	6
kobold/client/plam/model/productline/Productline.java	6

Abbildung 5.4.: Anzahl Entwickler pro Datei (Top 10)

Diese Tabelle zeigt zu einer Liste von Quellcode-Dateien die Anzahl von Entwicklern,

die jeweils gemeinsam an einer Datei gearbeitet haben. Es sind nur die zehn Dateien mit den meisten Bearbeitern angegeben, aber auch in einer Liste aller Dateien würden sich nie weniger als zwei Bearbeiter pro Datei finden lassen.

Es gab innerhalb des Teams für die Implementierung eines Moduls jeweils einen bekannten zuständigen Entwickler, der im besonderen mit dem Quellcode vertraut war. Mit diesem wurde vor grösseren Änderungen innerhalb seines Zuständigkeitsbereichs durch andere Entwickler Rücksprache gehalten, so daß Konflikte im voraus vermieden werden konnten. Kleinere Änderungen oder Fehlerkorrekturen konnten im Zweifelsfall auch nachträglich oder über das Quellcode Repository kommuniziert werden.

Diese Form des “Collective Code Ownership” ist sehr positiv zu bewerten. Es hat Reibungsverluste bei Änderungen und die Entstehung von Quellcode, der nur von einem einzelnen Entwickler verstanden und gewartet werden kann, vermieden. Alleine durch die Kenntniss, daß Quellcode von anderen gelesen wird, konnte die Lesbarkeit und Qualität erhöht werden.

5.5. Arbeitzeit

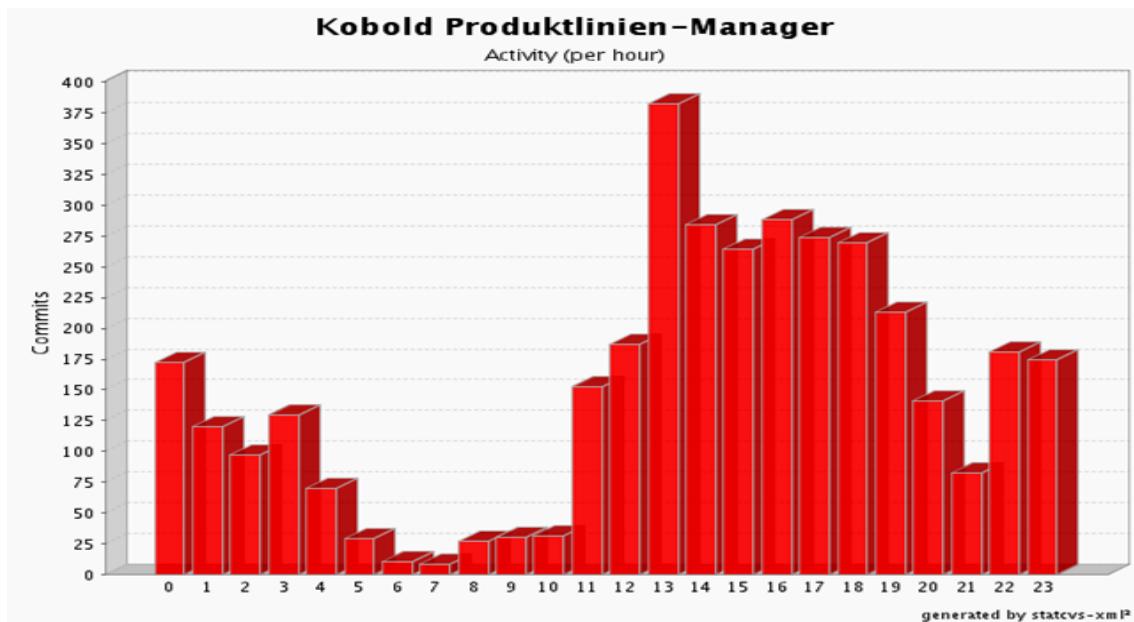


Abbildung 5.5.: Uhrzeiten der CVS Repository Aktivität

Das Diagramm zeigt die Uhrzeiten der Aktivitäten im Quellcode Repository. Eine höhere Anzahl Commits bedeutet eine höhere Aktivität. Typisch ist die erhöhte Aktivität zwischen 13 und 19 Uhr, die auch bei anderen Studienprojekten zu beobachten ist.

Offensichtlich ist das eine übliche Zeit das Arbeitsergebnis z.B. zum Ende der Kernarbeitszeit in das Repository zu comitten. Bemerkenswert sind die Aktivitäten nach 22 Uhr, die bis in die Morgenstunden reichen. Es gibt eine ausgeprägte Bandbreite von bevorzugten Arbeitszeiten wobei die Arbeiter am Nachmittag eindeutig die Mehrheit im Team stellen.

6. Persönliche Berichte

Nachfolgend sind die persönlichen Berichte der einzelnen Mitarbeiter aufgeführt. Sie rekapitulieren den Verlauf des Projekts jeweils aus dem Blickwinkel der einzelnen Entwickler und geben einen Einblick über die ganz persönlichen Erfahrungen, die sie aus diesem Projekt in ihre weitere Zukunft mitnehmen werden.

6.1. Necati Aydin



Abbildung 6.1.: Necati Aydin

Da ich, wie auch einige andere aus unserem Team, das StuProB dem StuProA vorziehen konnten, hatte ich schon einige Erfahrung aus dem StuProB gesammelt. Aber verglichen mit meinem ersten Studienprojekt war Kobold um einiges komplexer, was sich in laufe des Projekts herausstellte.

6.1.1. Tätigkeiten

Im Vorprojekt bestand unser Team aus Patrick, Anselm, Oli, Tammo und mir. In dieser Phase des Projekts waren wir im wesentlichen alle an der Analyse der Anforderungen und der Ausarbeitung eines Angebots beteiligt.

Nachdem wir den Zuschlag für das Projekt bekamen und die restlichen Teammitglieder hinzukamen, versuchten wir die Ergebnisse der Analysen vom Vorprojekt beider Teams zusammenzuführen. Dabei konnte man beobachten, wie unterschiedlich die Anforderungen aufgefasst wurden waren. Nachdem wir dann, so gut es ging, auf einen gemeinsamen Nenner kamen, konnte es dann auch mit der Spezifikation beginnen, an der wir auch noch alle beteiligt waren.

In den frühen Phasen des Hauptprojekts nahm ich an den Reviews für den Projektplan und der Spezifikation teil. In der ersten Iteration war meine Aufgabe den “Role-tree”(Navigationsbaum) zu implementieren. Dies erwies sich schwieriger, als ich es im Vorfeld angenommen hatte. Zum einen war ich mit Eclipse noch nicht so vertraut und zum anderen waren in dieser Phase noch Entwurfsfragen bezüglich des Datenmodells noch offen.

In der zweiten Iteration arbeitete ich am Dokument-Generator, der für ein beliebiges Asset alle Komponenten, Varianten, Benutzer, etc. ausliest und daraus ein pdf-Dokument generiert. Hierzu musste ich mich erst einmal mit “iText” beschäftigen und mich hier einzubringen. Dies war dann auch mein Schulungsthema. Gegen Ende der zweiten Iteration führten wir dann ein Code-Review durch, an dem ich als Gutachter teilnahm.

Am Anfang der dritten und letzten Iteration (geplant waren zuerst vier), war ich immer öfter mit der Erstellung und Verschönerung von Dialogen beschäftigt. Zudem wurden mir immer wieder Bugs zugewiesen, die ich dann auch bearbeiten konnte. Zu dieser Phase des Projekts gab es auch keine klaren Zuordnungen mehr. Die Tasks die keinem Projektmitglied direkt zugewiesen waren, konnten von jedem bearbeitet werden, der gerade etwas Zeit hatte.

In einer späteren Phase des Projekts führte ich Systemtests mit anderen Teammitgliedern zusammen durch. Die hier aufgefundenen Fehler wurden dann auch gleich in Bugtracker aufgeführt. Es war auch interessant, mal den Entwicklungsfortschritt und die Realisierung des Projekts des “Konkurrenz”-Teams zu sehen. Dies war mir gegönnt, da ich mit einem weiteren Teammitglied, ihr Tool (PLAM) testen konnte. Hierbei konnte ich feststellen wie unterschiedlich Kobold und PLAM sind.

6.1.2. Beurteilung

Trotz einiger Schwierigkeiten war Kobold ein Erfolg. Durch dieses Projekt konnte ich sicherlich mehr Erfahrungen sammeln, die für mein Studium und für andere Projekte sehr hilfreich sein können. Zum Erfahrungsgewinn zählt für mich das Erlernen neuer Technologien, wie Eclipse, iText, etc., und das freundschaftliche Miteinander der Teammitglieder, obwohl man sich vorher nicht kannte.

Die zwischenzeitlichen Motivationsprobleme, die durch die unterschätzte Komplexität von Eclipse hervorgerufen wurde, wurden schnell behoben, indem die Teammitglieder immer wieder bereit waren sich gegenseitig zu helfen. Diese gegenseitige Hilfsbereitschaft verstärkte den Zusammenhalt des Teams. Zudem war die Atmosphäre im Team immer sehr locker.

Vielen Dank an alle Beteiligten!

6.2. Armin Cont



Abbildung 6.2.: Armin Cont

6.2.1. Tätigkeiten

Entsprechend dem Charakter einer Vorprojektphase (Strudelmodell) hatte ich das Vergnügen, mit meinen Teammitgliedern gemeinsam die durchzuführenden Aufgaben anzugehen. Neben Analyse und Grobentwurf arbeitete ich hauptsächlich an der Planung der Organisation des Projekts, was sowohl die Aufteilung der Rollen mit einschloss als auch die Konzeption eines dazu passenden Entwicklungsmodells. Die Integration der Vorarbeiten resultierte in der Erstellung eines umfassenden Angebots für den Auftraggeber.

Nach der Angebotserteilung bestand meine erste Aufgabe darin, zusammen mit Anselm Garbe durch die Erstellung eines ersten, gemeinsamen Projektplans entsprechend dem Angebot der Werkbold-Gruppe die konzeptionellen Grundlagen für die Entstehung eines schlagkräftigen Entwicklerteams für das weitere Projekt zu legen.

Neben der danach anstehenden Arbeiten an der Spezifikation unseres künftigen Systems bereitete ich zusammen mit meinem Kollegen Michael Grosse die Schulung des Teams zur Verwendung von Entwurfsmustern im Software-Architekturentwurf vor, die unmittelbar vor Beginn der Entwurfsarbeiten von uns gehalten wurde.

Naheliegend war dann meine extensive Teilnahme an den Entwurfsarbeiten selbst, während derer wir das (auch) durch die Schulung im Team etablierte Design-Know-How zur Erstellung eines zunächst groben und später weiter verfeinerten Entwurfs zum Einsatz bringen konnten.

Unglücklicherweise war es mir aufgrund einer länger andauernden, schweren Erkrankung nicht vergönnt, aktiv an den im Anschluß daran beginnenden Implementierungsarbeiten der ersten Iteration teilzunehmen, was mich jedoch bei meiner Rückkehr zum Ende der besagten ersten Iteration in die außerordentliche Lage versetzte, das bis zu dem Zeitpunkt entstandene System aus einem im höchsten Maße unvoreingenommenen Blickwinkel zu bewerten.

Tatsächlich war es für mich eine sehr mühsame Aufgabe, das bestehende System anhand des umgesetzten Codes in seiner Gesamtheit zu verstehen, obwohl der ursprüngliche Entwurf noch weitestgehend erkennbar war. Wie sich herausstellen sollte, war diese Aufgabe tatsächlich nur mit der tatkräftigen Unterstützung einzelner Kollegen möglich, die mir das System persönlich erklären mussten, um mich de facto in die Lage zu versetzen, wieder produktiv für das Projekt tätig zu werden.

Während der nachfolgenden Monate konzentrierten sich meine Aufgaben im Rahmen der Implementierung hauptsächlich auf die Umsetzung des Administrationsclients und die Erweiterung bzw. Anpassung des Kobold-Servers. Nach Fertigstellung der rudimentären Administrationsfunktionen, zeigte sich die Notwendigkeit der Abhärtung des Servers zur Sicherstellung der Datenmodellintegrität, was aufgrund meines inzwischen sehr guten Verständnisses des Servers von mir durchgeführt wurde.

Durch die mir zugetragenen Qualitätssicherungsaufgaben, konnte ich durch meine Kritik und meine katastrophalen Erfahrungen beim Versuch des Verständnisses des Gesamtsystems nach meinem Wiedereintritt positiv auf die bis dato im Projekt vorherrschende Dokumentations- und Kommentierungskultur einwirken. Nicht zuletzt der hohen Disziplin und Sachlichkeit der Teammitglieder beim Umgang mit persönlicher Kritik ist es zu verdanken, dass der von mir in diesem Zusammenhang erbrachte Aufwand trotz der anarchischen Natur des Studienprojekts nicht gänzlich vergebens war und zu nachvollziehbaren Ergebnissen geführt hat.

Naheliegend war meine Tätigkeit an den vor allem gegen Ende der letzten Phasen fortcierten Testarbeiten, für die ich neben meiner Teilnahme bei deren Durchführung die vorbereitenden Testdokumente zunächst von der Spezifikation und später vom Handbuch ableiten durfte. Meine letzte große Aufgabe im Projekt war die gemeinsam mit meiner Kollegin Bettina Druckemüller durchgeführte Erstellung des gemeinsamen Abschlussberichts.

6.2.2. Beurteilung

Müsste ich eine Zeitmaschine besteigen, um mir selbst vor Beginn dieses Projektes Rat zu geben, ob ich an diesem Projekt teilnehmen sollte oder nicht, so würde ich mich energisch dafür aussprechen, und zwar aus den folgenden Gründen: Neben dem unschätzbarer Erfahrungsgewinn im Umgang mit den von uns eingesetzten und von mir im Vorfeld sträflich unterschätzten Technologien der Eclipse-basierten Javaentwicklung, hatte ich das große Glück eine bis dato von mir nicht bekannte Kultur der Softwareentwicklung kennenzulernen. So wäre es im Rahmen der Softwareprojekte, an denen ich zuvor teilgenommen hatte, völlig unvorstellbar gewesen, die Konfigurationsverwaltung des Codes automatisch von einer Software durchführen zu lassen; einem Sakrileg wäre es gleichgekommen, jedem Entwickler das Recht einzuräumen, jedes beliebige Code-dokument ohne Rücksprache zu ändern und sich dabei auf die Eigenverantwortung und Disziplin des entsprechenden Entwicklers zu verlassen.

Die Erfahrung, dass dies in einem qualifizierten und angemessen disziplinierten Umfeld tatsächlich funktionieren kann, hat in diesem Sinne meinen Horizont erweitert, auch wenn meine Aversion gegenüber einem solchen Vorgehen aufgrund der mit unter eingetretenen Probleme eher bestätigt wurde. Den von kompromissloser Sachlichkeit und Kritikfähigkeit geprägten, offenen Umgang der Teammitglieder untereinander empfand ich, gerade im Hinblick auf meine teilweise sehr negativen Erfahrungen außerhalb der Universität, als geradezu paradiesisch. Diese von mir bis dahin nie erlebte Atmosphäre wird mir künftig als - erreichbares - Ideal auch bei Softwareprojekten in einem weniger akademisch-aufgeklärten Umfeld dienen.

6.3. Bettina Druckenmüller



Abbildung 6.3.: Bettina Druckenmüller

6.3.1. Tätigkeiten

Das Studienprojekt A Kobold war für mich nach dem Softwarepraktikum im vierten Semester das zweite Softwareprojekt überhaupt, denn leider konnte ich außerhalb der Universität noch keinerlei Erfahrung sammeln oder gar an einem solchen Projekt teilnehmen.

Da die Hälfte des Teams schon in höheren Semestern war und es für diese teilweise schon das zweite Studienprojekt war, war auch ihr Grundwissen größer als meines. Als ich in das Team hinein kam, hatte ich zunächst schwer damit zu kämpfen, mir die Ideen und Pläne aus dem Angebot des Teams vorzustellen und zu begreifen. Positiv überrascht war ich von der Bereitschaft meiner Kollegen, mir geduldig alle Fragen zu beantworten und mich in die Bereiche einzuarbeiten, mit denen ich noch keine Erfahrung gemacht hatte. Auch die Schulungen innerhalb des Teams waren eine gute Möglichkeit zum Einstieg.

Während des Vorprojektes hatte ich die Rolle des Ansprechpartners/Projektleiters inne. Die Anforderungsanalyse, die Organisation des Teams, die Erstellung des Grobentwurfes und des Angebotes fand in gemeinschaftlicher Arbeit statt. Zusätzlich sorgte ich

dafür, dass regelmäßig Protokolle geschrieben wurden und alle Teammitglieder an ihre Zeiterfassung und Termine dachten.

Zu Beginn des Hauptprojektes erstellte ich zusammen mit Patrick Schneider die Spezifikation der ersten Iteration. Dies bereitete mir einige Probleme, da ich bisher nur Spezifikationen gekannt hatte, die zum Großteil aus UseCases bestanden. Hier aber sollte das Rahmengerüst spezifiziert werden, das ich mir selbst nur schlecht vorstellen konnte. Da ich außerdem bis zu diesem Zeitpunkt keinerlei Erfahrung mit TeX hatte, musste ich zuerst ein paar Stunden in die Einarbeitung investieren. Als Dokumentationsmanagerin war dies aber sowieso unumgänglich, weil alle Dokumente in TeX geschrieben wurden, und so waren diese Stunden gut investiert.

Am Entwurf und in der Anfangsphase der Implementierung konnte ich aufgrund einer Krankheit nur mit Unterstützung teilnehmen. Da ich mich für die Spezifikation schon intensiv mit der Regelengine Drools beschäftigt hatte, übernahm ich für Entwurf und Implementierung das Modul der Workflows. Ich half meinem Kollegen Michael Grosse beim Entwurf der Workflow-Klassen, schrieb die eigentliche Regelengine und gab meinen Kollegen Michael Grosse und Martin Plies einen Crashkurs im Regelschreiben, damit sie mir bei dieser Aufgabe helfen konnten.

Gegen Ende der ersten Iteration begann ich, meine Rolle als Dokumentationsmanagerin voll auszuleben: Ein Handbuch musste erstellt werden. Diese Aufgabe behielt ich bis zuletzt bei und führte sie parallel zu meinen anderen Aufgaben aus. Ich gab acht, dass das Handbuch immer das aktuelle System widerspiegelte, was mir sehr erleichterte, den Überblick über das System nicht zu verlieren. Im Laufe des Projektes habe ich das Handbuch sicherlich ein paar Dutzend Mal vollständig erneuert, da sich sowohl die Bedienungsoberfläche als auch die Funktionalität ständig änderten.

Ab der zweiten Iteration war ich in der Lage, die Workflows alleine zu bearbeiten. Leider hatte sich im Entwurf mittlerweile so viel verändert, dass die komplette Regelmenge neu geschrieben werden musste. Es mussten außerdem neue Regeln hinzugefügt werden, die dem Benutzer als Beispiel dienen sollten.

Meine Arbeiten in der Implementierung stellten nach einer intensiven Einarbeitung in Eclipse und Drools keine Probleme dar. Da ich mit Java schon einige Erfahrung gemacht hatte, konnte ich leicht damit arbeiten. Für die Spezifikationen der weiteren Iterationen erstellte ich die Workflow-relevanten UseCases und setzte sie natürlich auch in die Tat um.

Im Laufe des Projektes nahm ich außerdem an allen qualitätssichernden Maßnahmen wie Reviews und Tests teil. Zusätzlich hielt ich eine Schulung zu Drools, damit der Rest des Teams einen Einblick in diese Technologie bekam.

Im Zuge der Auslieferung erstellte ich für den Client ein sogenanntes CheatSheet, das in Kobold erscheint und dem unerfahrenen Benutzer eine kleine Einleitung bietet. Außerdem war es meine Aufgabe, die Spezifikation zu überarbeiten und auf den aktuellen Stand zu bringen.

Zum Abschluss des Projektes arbeitete ich zusammen mit meinem Kollegen Armin Cont an der Erstellung dieses Abschlussberichtes.

6.3.2. Beurteilung

Für mich persönlich war das Studienprojekt ein voller Erfolg, da ich sehr viel Erfahrung sammeln konnte.

Durch das Projekt kam ich mit den unterschiedlichsten Dingen in Kontakt, mit denen ich davor noch nicht gearbeitet hatte. Darunter fallen TeX, CVS, Duckforge, BerliOS, Client-Serveranwendungen, Version Control, Eclipse, Perl, Regelengines, etc.. Auch implementierungstechnisch konnte ich mir viel Wissen aneignen.

Zusätzlich zum software-relevanten Lerneffekt lässt sich aus Kobold auch viel projekttechnische Erfahrung mitnehmen. So habe ich zum Beispiel gelernt, wie wichtig eine gute Spezifikation und ein guter Entwurf für eine reibungslose Implementierung sind. Und auch auf qualitätssichernde Maßnahmen werde ich in zukünftigen Projekten besonders Wert legen. Diese sorgen nicht nur dafür, dass Fehler frühzeitig erkannt werden, sondern auch dass bei einem großen Team jeder den Überblick über das gesamte System behält und im Notfall an jeder Stelle helfen kann.

6.4. Anselm R. Garbe



Abbildung 6.4.: Anselm R. Garbe

Das Studienprojekt A Kobold war mein zweites Studienprojekt, da ich neben anderen Team-Mitgliedern bereits mein Studienprojekt B im Bereich Automatisierung vorgezogen hatte.

6.4.1. Tätigkeiten

Ich gehörte von Anfang an zum Team Werkbold und nahm im gesamten Projektverlauf die Rolle des Projektleiters ein.

Vorprojekt

Im Vorprojekt war ich zusammen mit meinen Team-Mitgliedern Tammo, Oli, Patrick und Necati mit der Angebotserstellung befasst. Ich strebte anfänglich eine Realisierung mit Ada und GTK2 an, wurde allerdings von Tammo und Patrick überzeugt, ein Angebot mit der J2SE Technologie zu wagen, da auch unser Auftraggeber sich dieser Idee offen gegenüber äußerte. Ich konzentrierte mich gemäß meiner Aufgabe als Projektleiter im Angebot um die projekt- und prozesstechnischen Vorgaben. Unser Angebot beruhte

maßgeblich auf Erfahrungen aus dem vorgezogenen Studienprojekt B sowie auf verschiedenen OpenSource-Projekten, die einzelnen Team-Mitglieder hobbymäßig betrieben. Ich verfolgte ein iteratives Prozessmodell, in dem jede Iteration dem klassischen Wasserfall-Phasenmodell entsprach. Ursprünglich planten wir 4-5 Iterationen (was letztendlich auch eintrat), zwischenzeitlich korrigierten wir allerdings die Iterationszahl auf 3 nach unten.

Darüber hinaus entwarf ich mit den anderen Werkbolden, vor allem unter Anleitung von Tammo, eine Grundarchitektur des Kobold Systems, um dem Auftraggeber im Angebot eine in Worte gefasste Vision nahezulegen, aus welchen Komponenten das endgültige System bestehen wird und wie diese zusammenpassen.

Nach dem Zuschlag für unser Angebot kümmerte ich mich um die Integration des Cures-Teams und um die Durchsetzung und Vergegenwärtigung unseres Angebotes beim Cures-Team. Immerhin stellte sich die Situation für das Cures-Team nicht so leicht, da wir als einziges Team der vier Vorprojekte eine J2SE-basierte Realisierung angeboten hatten und das Cures-Team nun in den sauren Apfel beißen mußte, ein ganz anders geartetes Produkt zu realisieren.

Hauptprojekt

Um das Kennenlernen zu vereinfachen, veranstalteten wir mit unseren Konkurrenten des AssetDevel Teams und unseren Auftraggebern zunächst eine Kennenlernen-Party im Fakultätsratssaal.

Bereits im Werkbold-Team des Vorprojektes herrschten unterschiedliche Kenntnisse der Eclipse- und J2SE Technologie vor, was sich auch durch die Integration des Cures-Teams nicht verbesserte. Aus diesen Gründen, die wir im Vorfeld bereits abgeschätzt hatten, führten wir am Anfang des Hauptprojektes verschiedene Schulungsmaßnahmen zu den Technologien durch, die in Kobold Verwendung fanden. Ich hielt hierzu einen Vortrag zum Thema XML-RPC - ein RPC Mechanismus, dessen Kommunikation in Form eines bestimmten XML-Formates realisiert wird.

Die erste größere Aufgabe im Hauptprojekt war neben der Integration des Cures-Teams und der Vergegenwärtigung der nun gemeinsamen Produktvision, die Erstellung eines angepassten Projektplanes. Hierzu gingen wir wie angeboten vor, in dem der Projektplan iterationsabhängig zu Beginn jeder Iteration für diese verfeinert wurde und somit alle aufgebrauchten Aufwände transparent für den Kunden sichtbar gemacht werden konnten.

Ich strebte eine Projektleitung an, die möglichst wenig Autorität an den Tag legte, um so ein angenehmes Team- und Projektklima zu schaffen, was leider nur teilweise gelang, so dass ich hin und wieder Aufgaben für die sich kein Freiwilliger fand zuteilen mußte. Neben der Verantwortlichkeit zur Aufgabenverteilung organisierte ich auch regelmäßige Termine zur Team-Synchronisation und ggf. zur Rücksprache oder kurzen

Präsentationen mit dem Auftraggeber.

Zur Erfassung der Projektaufwände hielt ich bei Bedarf die Team-Mitglieder an, ihre Aufwände zu erfassen, was erfahrungsgemäß bei Studienprojekten sehr widerwillig regelmäßig durchgeführt wird. Da jede Iteration in bestimmtem Maße dem Phasenmodell entsprach, wurden in jeder Iteration die Spezifikation verfeinert, der Entwurf konkretisiert und die Implementierung mehr und mehr vervollständigt. Aufgrund häufiger Technologie-Überraschungen mußte die Spezifikation und Implementierung in den Folge-Iterationen hin und wieder verfeinert oder überarbeitet werden.

Als Projektleiter fühlte ich mich für eine fristgerechte Fertigstellung und Auslieferung der einzelnen Meilensteine verantwortlich. Trotz der Teamgröße von 9 Entwicklern war ich an der Spezifikations- und Entwurfserstellung sowie Implementierung maßgeblich beteiligt, da ich mich persönlich eher als Entwickler sehe, als ein Manager.

Im Laufe der ersten Iteration mußten wir den Projektplan deutlich ändern und verschmolzen die ersten beiden Iterationen miteinander, da sich ein deutlich höherer Aufwand für den verfolgten Meilenstein der Implementierung zeigte. Die zweite Iteration verlief ziemlich fristgerecht, die dritte Iteration und zum Projektende hin verwässerte sich allerdings die Definition eines definierten Auslieferungszustandes, wobei wir am Projektende rückblickend tatsächlich 4 Iterationen durchgeführt haben, wenn man die dritte Iteration als definiertes Ende des offiziellen Hauptprojektes betrachtet und die 4. Iteration als Nachbesserungsphase.

Es bleibt anzumerken, dass aufgrund der Unvollständigkeit des Systems in den ersten Iterationen mit der Qualitätssicherung erst relativ spät begonnen wurde, was wiederum zu einer spät einsetzenden Stabilisierung des Systems führte.

In der Spezifikation beschrieb ich vor allem die Grob-Architektur und die Client-Server Kommunikation. Aufgrund des architektonischen Charakters meiner Beiträge zur Spezifikation war ich gleichbedeutend am Entwurf beteiligt. In der Implementierung kümmerte ich mich zu Beginn um die XML-RPC basierte Kommunikation zwischen Client und Server, sowie um die Persistierung der kommunizierten Datenobjekte auf dem Server und die SSL-Anbindung der XML-RPC Kommunikation. Die Client-Server Kommunikation galt als Kernkomponente und wurde somit zum Anfang implementiert. In der Mitte der Implementierung (etwa Sommer 2004) kümmerte ich mich vor allem um einzelne Dialoge und verschiedener Zuarbeiten in der Client-Server Kommunikation. Zum Ende der Implementierung (in der Nachbesserungsphase) war ich mit der tiefgreifenden Fehlerbereinigung der VCM-Anbindung beschäftigt.

Da ich bis auf die Details der Grapheneditor-Visualisierung in allen Bereichen des Kobold Systems recht guten Einblick durch meine Implementierungstätigkeit finden konnte, war es mir leicht möglich Implementierungsarbeitspakete den einzelnen Team-Mitgliedern zuzuteilen.

Zum Projektende kümmerte ich mich um die Aufgabenverteilung bzgl. der Abschlusspräsentationsplanung und der Erstellung des Abschlussberichtes, sowie fester Termin-

vorgaben.

6.4.2. Fazit

Das Studienprojekt Kobold hat meinen Horizont reichhaltig erweitert. Da ich zuvor noch nie die Rolle eines Projektleiters eingenommen hatte, war es für mich eine vollkommen neue Erfahrung dieses besondere Maß an Verantwortung zu tragen.

Aus Sicht des Projektleiters lag die schwierigste Aufgabe für mich im Umgang mit den Team-Mitgliedern, da ich mich bemühte eine lockere und wenig autoritäre Atmosphäre zu schaffen. Allerdings verhält es sich mit der Teamführung vermutlich ähnlich wie mit der Erziehung, weshalb frühe Fehler oder Nachlässigkeiten permanent bestraft werden. In großen Teams, bei denen ich die einzelnen Team-Mitglieder nicht von Anfang an genau einschätzen kann, würde ich in Zukunft zu etwas mehr Autorität neigen, um somit Termine reibungsloser einzuhalten.

Gleichwohl hat meine Projektleitung dazu geführt, dass es zu keinen nennenswerten Konflikten innerhalb des Teams gekommen ist. Es hat sich gezeigt, dass die Fähigkeiten und Interessen der einzelnen Team-Mitglieder ganz unterschiedlich waren und blieben. Anfangs zog ich den Trugschluss, das man alle Team-Mitglieder auf ein gemeinsames Niveau bzw. auf einen einheitlichen Kenntnisstand bringen kann, was mir jedoch heute als unrealistisch erscheint. Dies mag für von Anfang an eingespielte Teams oder Abteilungen zutreffen, jedoch nicht für ein Team eines Studienprojektes.

Aus Sicht des Entwicklers bin ich mir nicht sicher, ob die Wahl der Eclipse-Plattform sowie der J2SE Technologie bei unserer Team-Zusammensetzung und den mitgebrachten Vorkenntnissen, die richtige war. Die Komplexität, die sich in der Eclipse-Welt verbirgt, hat meines Erachtens nach wenigstens 1/3 des gesamten Aufwandes verschlungen. Dieser Aufwand wäre vielleicht mit GTK2/Ada geringer ausgefallen. Andererseits hat es allen Team-Mitgliedern und mir viel gebracht, sich in die Eclipse-Welt einzuarbeiten, da im kommerziellen Bereich Java-Technologie heutzutage in vielen Unternehmen eingesetzt wird.

Architektonisch würde ich das System statt der Client-Server Lösung mit einer Versions-Kontrollsystem Abhängigkeit heute vollkommen unabhängig von Versionskontrollsyste- men wie CVS planen und anbieten. D.h. ich würde eine Datenbank (z.B. in Form eines Dateisystems) zur Persistierung sowie zur Speicherung der Konfiguration verwenden, statt der implementierten Client-Server Lösung. Es hat sich als sehr beschränkt erwiesen, von einem gegebenen Versionskontrollsystem abhängig zu sein.

Aus Sicht des Studenten würde ich generell die Empfehlung aussprechen, Studienprojekte zeitlich zu komprimieren, d.h. die beiden vorgesehenen Studienprojekte nacheinander vollzeittechnisch in den Studienplan zu integrieren, d.h. 4 Monate pro Studienprojekt als Vollzeitprojekte durchzuführen. Die Aufwände, die sich durch Prüfungssituatio- nen bzw. durch temporäre mentale Umorientierungen der Entwickler ergeben sind nicht

zu unterschätzen.

Abschliessend bleibt zu sagen, dass neben aller Kritik das Studienprojekt harmonisch verlief und ich die Erfahrungen, die ich sammeln durfte nicht missen möchte. Sie werden mir ganz wegweisend behilflich sein.

Bedanken möchte ich mich ganz herzlich für das Engagement unseres Teams sowie bei den Mitarbeitern der Abt. Programmiersprachen und Compilerbau. Sie haben sich sehr bemüht uns zu unterstützen. Ganz besonders beeindruckt war ich vom Engagement im Produktlinien-Seminar!

6.5. Michael Grosse

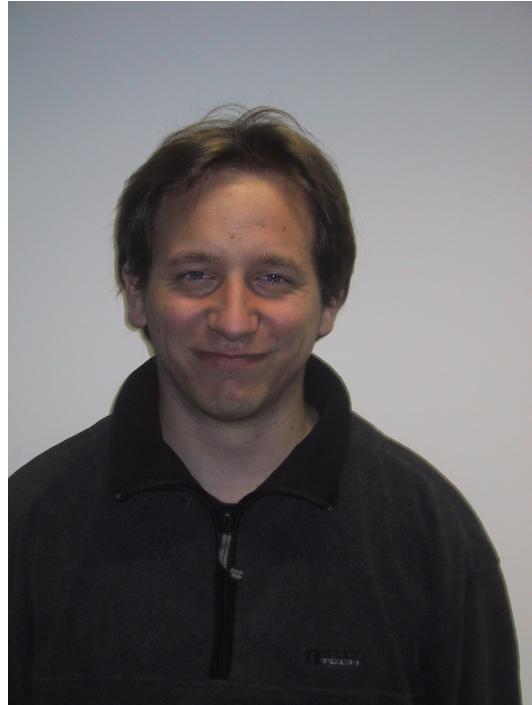


Abbildung 6.5.: Michael Grosse

6.5.1. Tätigkeiten

Das Studienprojekt A - Kobold war für mich das erste Softwareprojekt in dieser Größenordnung. Da einige in unserem Projektteam das Studienprojekt B vorgezogen hatten, hatte ich vor allem am Anfang des Projektes noch recht wenig Erfahrung in der Arbeit an so großen Projekten

Meine Tätigkeiten lassen sich am besten in die Tätigkeiten des Vorprojekts und des Hauptprojektes untergliedern. Während des Vorprojektes nahm ich, wie alle anderen, an allen Analysegeprächen mit den Kunden teil und erstellte mit meinen anderen Teammitgliedern zusammen hieraus einen groben Entwurf sowie das daraus resultierende Angebot.

Für das interne Seminar bereitete ich das Thema Design Patterns zusammen mit Armin Cont vor. Durch dieses Seminar wurden die Grundlagen für die anstehenden Entwurfsbesprechungen gelegt und so konnten sowohl Armin Cont, meine Person, sowie ein weiteres in Entwurfsdingen erfahrenes Projektmitglied, Tammo van Lessen, dieses durch die Schulung erworbene Wissen in der Erstellung des Grobentwurfs einbringen.

Nachdem die wichtigsten internen Schulungen gehalten waren, stand die Einarbeitung in Java und hier speziell in die recht komplexe Struktur von Eclipse an. Aufgrund meiner früheren Programmiererfahrung stellte Java an sich kaum ein Problem dar. Die von uns gewählte Eclipse Plattform jedoch ein umso größeres. Meine erste Aufgabe innerhalb der Programmierung war die Entwicklung einer Übersicht für unsere Workflowmeldungen.

Hierfür musste zuerst einmal herausgefunden werden, welche der vielen von Eclipse zur Verfügung gestellten Darstellungsmöglichkeiten für das Projekt am sinnvollsten verwendet werden könnte. Aufgrund der, wie sich herausstellte, großen Komplexität dieser Aufgabe, bearbeitete ich diese Aufgabe zusammen mit Martin Plies.

Nachdem diese Aufgabe erfolgreich zuende gebracht war, wurde mir aufgetragen, zusammen mit Bettina Druckenmüller die Erstellung der Workflows zu entwickeln. Bei dieser Aufgabe war ich zwar nur in den Anfangsphasen dabei, erhielt aber hierdurch einen sehr guten Eindruck des von uns eingesetzten Drools-Mechanismuses.

Meine nächste Aufgabe bestand darin, alle Benutzersteuerungsdialoge, wie Benutzer anlegen, Benutzernamen ändern und Benutzer löschen zu erstellen. Diese Aufgabe fiel mir recht leicht, da ich mich an der Struktur schon bestehender ähnlicher Dialoge recht gut orientieren konnte.

Mit zunehmendem Projektverlauf bekam die Qualitätssicherung einen größeren Stellenwert, so dass ich bei Reviews und den einsetzenden Tests zunehmend eingesetzt wurde. Außerdem betreute ich die Testdurchführung unseres Partnerteams.

Gegen Ende der Entwicklungsarbeiten wurde ich zunehmend als SSpringer für kleinere Ausbesserungsarbeiten an Code-Teilen eingesetzt. Diese Aufgaben gaben mir einen recht guten Gesamtüberblick über das Kobold-System.

Bei den Abschlusstests lag meine Aufgabe darin, diese Tests zu organisieren und durchzuführen, sowie danach die entstandenen Fehlermeldungen in unseren Bugtracker einzutragen.

Zum Abschluss des Projektes wirkte ich noch an der Erstellung der Abschlusspräsentation mit.

6.5.2. Beurteilung

Alles in allem gesehen würde ich sagen, dass Kobold ein perfektes Projekt zum Lernen war. Aufgrund der sehr komplexen eingesetzten Technik, welche ursprünglich völlig unterschätzt worden war kamen unsere Zeitpläne regelmäßig durcheinander.

Das von uns gewählte evolutionäre Entwicklungsmodell zeigte in diesem Projekt seine Risiken, was mir persönlich zeigte, wie wichtig die Qualitätssicherung innerhalb eines Projektes ist.

Für mich persönlich ergab sich ein recht tiefer Einblick in die Dynamiken einer größeren Gruppe. So fanden nur sehr wenige Besprechungen wirklich zur vorher ausgemachten Uhrzeit statt. Diese daraus resultierende Dynamik war allerdings nicht, wie leicht vermutet, hinderlich, sondern war in vielen Punkten durchaus produktiver, da sowohl private Gespräche, als auch kleine Absprachen mit wenigen Teammitgliedern so schon vor der Besprechung abgehakt werden konnten und sich diese so nicht mehr innerhalb des Gruppentreffens störend auswirkten. Auch lernte ich, wie wichtig eine frühzeitige und vor allem durchgängige Qualitätssicherung für das Wohl und Wehe eines Projektes ist.

Was mich positiv überrascht hatte war zum einen der große Zusammenhalt innerhalb der Gruppe, wie auch die gute Zusammenarbeit mit unserem "Konkurrenzteam".

Auch wenn die Arbeitsleistungen der einzelnen Mitglieder in diesem Projekt nicht immer homogen waren, so lässt sich aus meiner Sicht heraus sagen, dass alle Teammitglieder an einem Strang zogen und ein ständiger Wille, dieses Studienprojekt mit Erfolg zu beenden, ständig zu spüren war.

Der Lernerfolg dieses Projektes lässt sich in folgende Stichwörter fassen: Das evolutive Entwicklungsmodell, Eclipse als Plattform und der Stellenwert der Qualitätssicherung.

6.6. Tammo van Lessen



Abbildung 6.6.: Tammo van Lessen

Wie für einige andere Teammitglieder ist das Studienprojekt A “Kobold” bereits das zweite Studienprojekt für mich gewesen.

6.6.1. Tätigkeiten

Nach der Projektvergabe bildeten Anselm, Patrick, Oli, Necati und ich das “Team Werkbold” mit dem Ziel über das Vorprojekt den Zuschlag für das Hauptprojekt zu bekommen.

Vorprojekt

Neben der aktiven Teilnahme an diversen Meetings zur Analyse der Anforderungen interessierte mich besonders, wie man ein solches Projekt am elegantesten angehen kann und welche Frameworks uns zur Verfügung stehen. Als langsam klar wurde, dass der Kunde nicht ausschließlich auf eine Implementierung mit Ada mit GTK bestehen würde, habe ich die Eclipseplattform als Alternative vorgeschlagen. Da die Produktlinienverwaltung nicht nur für Manager sondern gerade für Programmierer entwickelt

werden sollte ist eine integrierte Entwicklungsumgebung meines Erachtens das passende Framework für unser Projekt. Mit dieser Begründung, der Tatsache, dass einige im Team mehr Erfahrung mit Java als mit Ada hatten und dass Eclipse auch softwaretechnisch sehr interessant ist (Erich Gamma hat hier sämtliche seiner GoF-Patterns umgesetzt), stimmten die anderen Teammitglieder schnell zu. Um sicher zu gehen, dass Eclipse auch tatsächlich geeignet ist, sollte ein Prototyp entwickelt werden. So war es nun meine Aufgabe anhand eines Prototypen die Machbarkeit des Grapheneditors in der Eclipseplattform zu überprüfen. Dazu wählte ich GEF (Graphical Editing Framework, <http://www.eclipse.org/gef>) als Framework. Nach einiger Einarbeitung konnte ich dem Team den Prototypen präsentieren. Da der Prototyp performant genug für die Anforderungen war, war schnell unsere Entscheidung für Eclipse gefallen. Nachdem die Anforderungen analysiert waren, formulierten wir das Angebot. Daran waren Anselm und ich maßgeblich beteiligt.

Aus vier Angeboten entschied sich der Kunde für zwei Projekte. Dazu gehörte auch unser Projekt "Kobold". Unser Team wurde nun um vier Teammitglieder erweitert.

Hauptprojekt

Die ersten Meetings nach der Zusammenlegung waren etwas schwieriger als erwartet, denn die Vorstellungen beider Team von dem Projekt waren doch etwas unterschiedlich. Nach längerem hin und her einigten wir uns und es wurde mit der Anforderungsdokumentation begonnen. Da die Eclipseplattform sehr umfangreich und komplex ist, entschlossen wir uns, uns gegenseitig zu schulen, damit wir alle einen möglichst gleichen Wissensstand zum Implementierungsbeginn haben. Ich bereitete zwei Schulungen vor: In der ersten stellte ich im Rahmen eines inf.misc-Vortrags das Build- und Projektmanagementtool "Maven" vor, mit dessen Hilfe wir Metriken, Statistiken sowie die Projekthomepage generieren. Der zweite Vortrag behandelte das Grapheneditorframework GEF, dessen Implementierungsdetails und Patterns ich intern vorstellte. Nach diesen Schulungen lag es auch nahe, dass ich während des Entwurfs und der Implementierung für den graphischen Editor zuständig war, den ich dann auch in Eigenregie entwickelte. Bevor es dazu kam, stand erst der Entwurf des Datenmodells an auf dem die anderen Komponenten dann operieren sollten. Im weiteren Verlauf stellte sich jedoch heraus, dass das Modell nicht alle Anforderungen erfüllte, so dass ich zusammen mit Martin die Architektur des Modells überarbeitete.

Parallel zu der Entwicklung des Grapheneditors kümmerte ich mich um die Integration des Models in die Eclipse-Workbench, programmierte den Navigationsbaum und die lokale Messageverwaltung.

6.6.2. Beurteilung

Zunächst begann das Studienprojekt sehr viel versprechend. Interessantes Thema, interessante Leute, interessante Frameworks. Mit dem Entwurf zeigte sich allerdings, dass das Eclipse-Framework vielleicht doch eine Nummer zu groß für ein Studienprojekt dieser Größenordnung war, da es eine intensive Auseinandersetzung mit den dort eingesetzten Patterns und Mechanismen erfordert, die in dem zeitlich begrenzten Rahmen leider nicht bei allen möglich war. Das war etwas ernüchternd. Diese Erfahrung ist aber nur eine der vielen, die ich im Verlauf des Projekts gemacht habe. Ich denke, dass gerade dies die wichtigen Dinge sind, die man bei einem Studienprojekt mitnehmen kann. So musste ich lernen, wie man mit unterschiedlichen Arbeitsweisen und Wissensständen umgehen muss, wie man Konflikte im Team vermeidet, wie wichtig Dokumentation ist, etc. Alles Dinge, die man schon vorher wusste, aber anders priorisiert hat. Ich war überrascht, wie fatal die Auswirkungen von vermeintlichen Kleinigkeiten sein können.

Für mich war das Studienprojekt ein sehr erkenntnisreiches und spannendes Studienprojekt, welches mir viel Spass aber auch einiges Kopfzerbrechen über das Miteinander im Team bereitet hat.

Trotzdem oder auch gerade deshalb sehe ich das Projekt als Erfolg an.

6.7. Martin Plies



Abbildung 6.7.: Martin Plies

Dieses Projekt war mein erstes Softwareprojekt dieser Größe. Projekterfahrungen mit mehreren Leuten hatte ich zuvor nur aus dem Sopra. Auch das Entwickeln von Plugins für Eclipse war für mich neu, da ich Eclipse zuvor nur als Editor für kleinere Java-programme verwendet hatte.

6.7.1. Tätigkeiten

Das Vorprojekt führte ich zusammen mit Armin, Bettina, und Michael durch. Wir erledigten die Aufgaben im wesentlichen zusammen..

Ich nahm an allen Analysegesprächen teil und versuchte zusammen mit den Teammitgliedern die Anforderungen möglichst genau und vollständig zu erfassen.

In der Spezifikationsphase hielt ich mich eher zurück, um mich intensiver auf meine noch ausstehende Prüfung vorzubereiten. Später erstellte ich ein Anforderungsdokument, in dem ich die Einzellanforderungen aus den Angeboten der Teams und dem von der Abteilung Programmiersprachen und Compilerbau herausgegebenen Paper zusammenfasste.

Im Zusammenhang mit den durchgeführten Schulungen schaute ich mir GXL an und stellte es in einem Vortrag vor.

In der ersten Implementierungsphase erstellte ich zusammen mit Bettina und Michael einen genaueren Entwurf der Workflows. Wir definierten einen Beispielworkflow und konkretisierten den Entwurf, um möglichst abstrakte Workflows über die Client Server-schnittstelle schicken zu können. Im folgenden kümmerte ich mich um die Anzeige der Workflows auf dem Client. Die Art der Darstellung musste erst gesucht werden. Die hohe Komplexität von Eclipse und die fehlenden Kenntnisse machten anfangs immer wieder Probleme. Auch die zur Darstellung verwendete swt-Komponenten bereiteten anfangs Probleme, da die verfügbare Dokumentation nur mäßig war.

Da am Entwurf des Datenmodells mehrfach Änderungen nötig wurden, schaltete ich mich in die Diskussion ein und überlegte hauptsächlich zusammen mit Tammo, wie das Datenmodell am besten aussehen sollte.

In der 2. Phase arbeitete ich nicht weiter an den Workflows. Ich kümmerte mich um den GXL-Export des Koboldgraphs und den GXL-Import von Bauhausgraphen. Da ich mich jetzt besser damit auskannte, ging dies deutlich zügiger.

Danach erstellte ich den Edgecontainer. Dieser benutzt zwei Adjazenzlisten, um die Kanten effizient zu verwalten und stellt die nötigen Methoden zur Verfügung, um die eingehenden oder abgehenden Kanten der benötigten Kantentypen zurückzugeben.

Da ich zu Ende der Phase noch Zeit hatte, übernahm ich die Erstellung des Productcomposer zur Produktgestaltung. Dieser ist dafür zuständig, den Anwender bei der Erstellung eines Produktes zu unterstützen und nach der Auswahl eines Assets die anderen Assets dahingehend zu markieren, ob sie auch zum Produkt gehören oder nicht.

In der Gesamtheit ist der Productcomposer ein recht komplexer Algorithmus. Ich verwendete die Tiefensuche um durch den Graphen zu traversieren.

Dadurch, dass wir Metaknoten in unserem Graph verwendeten, wurde der Productcomposer deutlich komplizierter. Durch die Und- und Oderknoten kann es komplizierte Abhängigkeiten geben, die vom Productcomposer unterstützt werden müssen.

Entwurf und Implementierung gingen aber trotzdem vergleichsweise zügig von der Hand.

Danach erweiterte ich den Productcomposer, so dass der Productcomposer nach Auswahl eines Produktes das Produktmodell erstellt.

Als nächstes erweiterte ich das Löschen von Assets. Das Löschen der Assets im Produkt fehlte noch. Wenn der Löschvorgang zurückgenommen wurde, funktionierte das Wiederherstellen der Kanten auch noch nicht.

Gegen Ende des Projekts erweiterte ich die Darstellung der Nachrichtenliste, um Filtermöglichkeiten und gab dem Benutzer die Möglichkeiten, die Nachrichten zu sortieren. Schließlich entfernte ich noch Fehler in der Dokumentation, um die Erstellung der Javadoc-Kommentare möglich zu machen. Zudem entfernte auskommentierten Code

aus dem Projekt.

6.7.2. Beurteilung

Kobold hat mir einen großen Erfahrungsgewinn gebracht.

Es war eine Erfahrung für mich in Teamarbeiten ein Softwareprojekt dieser Größenordnung durchzuführen. Hier muss ich hinzufügen, dass es in unserer Gruppe sehr harmonisch zuging.

Ich habe erlebt, wie wichtig es ist, gerade bei einem iterativen Vorgehensmodell eine ständige und genaue Fortschrittskontrolle zu machen und wie wichtig es ist, dass in den frühen Phasen Kernkomponenten in guter Qualität erstellt werden.

Eclipse hat gezeigt, wie wichtig es doch ist, die Umgebung zu kennen, in der oder mit der entwickelt wird. Der Aufwand, den uns Eclipse bereitet hat, wurde leider stark unterschätzt.

Da ich Eclipse inzwischen kenne, vermag ich es jetzt in späteren Projekten gut einzusetzen.

6.8. Oliver Rendgen



Abbildung 6.8.: Oliver Rendgen

6.8.1. Tätigkeiten

Bei dem Studienprojekt Kobold handelt es sich um mein 2. Studienprojekt. Durch die bisherigen Erfahrungen in Projekten und mein 1. Studienprojekt war mir das Vorgehen und der Ablauf innerhalb des Projektes schnell klar und so konnte ich mich auf die Inhalte konzentrieren.

Das Vorprojekt begann mit der Analyse, geprägt durch die Vorstellung des Kunden und kleinere Analysegepräche direkt mit Mitarbeitern. Nachdem die ersten Fakten gesammelt waren sortierte unser Team die Fakten in Meetings und formte daraus erste Vorstellungen eines Rahmengerüstes. Zu dieser Zeit kamen auch erste Gedanken auf, nicht wie üblich Ada, sondern Java als Implementierungssprache zu verwenden. Erfreulicherweise war der Kunde mit diesem Vorschlag einverstanden. Anhand der weiteren Gedanken wurde bald ein grober Entwurf erstellt, der dann zu dem Angebot führte. Das Angebot stellte einen guten Überblick über unsere Ideen dar, die als funktionale Anforderung einflossen. Auch, dass wir nach dem evolutionären Vorgangsmodell vorgehen wollten, war damit festgelegt.

Dadurch, dass unser Angebot angenommen wurde, vergrößerte sich unsere Gruppe um die Teilnehmer der anderen Gruppe. Um die neuen Teilnehmer in unsere bisherigen Gedanken und Designentscheidungen einzuführen, sowie alle Teilnehmer auf den gleichen Wissensstand zu bringen, wurden interne Schulungen und Seminare durchgeführt. Da einige Teilnehmer bisher nur Windows benutztten, führte ich in einem Seminar in die Grundlagen von SSH, CVS inkl. Einbindung in Eclipse und die Berrios-Dienste ein. Somit sollte jeder Teilnehmer in der Lage sein auf die unter Versionkontrolle stehenden (Quell-)Dokumente zuzugreifen und eigene Informationen in die mit maven generierte Webseite einzufügen. Da ich die Eclipse-Plattform noch nicht kannte, war meine Einarbeitungsaufwand in die verschiedenen Eclipse Eigenschaften und Einstellungsmöglichkeiten, sowie die interne komplexe Struktur des Frameworks aufwändig.

Nach dieser Neuformierung ging es darum eine passende Spezifikation zu erstellen. Da aber auch zu diesem Zeitpunkt noch nicht alle auftretenden Fälle abgeklärt waren, gab es oft Meetings bei denen die Köpfe zum Glühen gebracht wurden und ganz neue Sichtpunkt eingenommen werden mussten. In der Spezifikationsphase machte ich mir u.A. Gedanken über eine geeignete Metadaten-Struktur und spezifizierte diese. Die Metadaten sollten im XML-Format abgelegt werden und alle nötigen Information der PL, P und CAS beinhalten.

Da ich mich gut mit VCM-System auskenne, habe ich für den Entwurf das VCM-Wrapper Plug-In entworfen. Dieses stellt die möglichen Repository-Operationen als Team-Plugin zur Verfügung. Dabei waren die groben Ablaufschritte zu spezifizieren, die vom Laden der Konfiguration vom Server, bis zum letztendlichen auschecken, gingen. Außerdem mussten die verschiedenen Repositoryoperation deklariert und die Rechteverwaltung der verschiedenen Rollen festgelegt werden.

Während der Implementierungsphase hatte ich verschiedene Aufgaben. Am Anfang setzte ich die spezifizierte Metadaten-Ablage um. Da selbst während der Implementierung noch viele Dinge umgestellt wurden, musste auch hier nachgezogen werden. Auch das Format änderte sich noch. Um die korrekte Speicherung zu verifizieren setzte ich J-Unit Testfälle ein. Dabei hatte ich auch die Erfahrung dass man nicht für alle Komponenten J-Unit Testfälle einsetzen kann, bzw. diese zuerst kapseln muss. Daraufhin beschäftigte ich mich mit dem VCM-Wrapper. Ich schrieb Skripte für alle nötigen VCM-Aktionen. Um Dateien, die sowohl unter und nicht unter Versionskontrolle stehen zu erkennen und mit diesen umzugehen, entschieden wir uns für zusätzliche Perl-Skripte, die vom VCM-Plugin angestoßen, die lokale Dateistruktur parsen und dabei Statusinformationen ausgeben. Das Schreiben dieser Skripte und das Parsing innerhalb von Java inklusive dem Erzeugen von Dateideskriptoren zur Darstellung sowie zur weiteren Verarbeitung waren meine weiteren Aufgaben. Dabei arbeitete ich eng mit Patrick zusammen. Wir mussten beide dabei feststellen, dass es viele Probleme verursacht, wenn sich unbekannte Wechselwirkungen zwischen Abhängigen Komponenten andeuten. Im letzten Schritt der Implementierung wurde es nötig, dass der Kunde eine erste Auslieferungsversion erhält. Aus diesem Grund erstellte ich eine komfortable Installations- und Dis-

tributionsroutine zur Generierung von Auslieferungs-ISO-Dateien, sowie zur Installation beim Kunden. Dabei mussten wir davon ausgehen, dass eine eigene Java, sowie Perl-Installation benötigt wird, sowie die nötigen Lib-Abhängigkeiten gelöst werden. Außerdem sollte Kobold, der Server und das Administrationsprogramm ohne vorherigen Konfigurationsaufwand sofort benutzbar sein. Auch alle bisher entstandenen Dokument wurden dabei als PDF-Dateien erzeugt. Die Teile der Distributierung und Installation fügte ich in das Handbuch ein.

In der Testphase, führten Necati und ich einen Konkurrenzprodukttest durch. Dabei fiel auf, dass deren Produkt aus einem anderen Benutzeraspekt designt wurde. Während des ganzen Projektes nahm ich an qualitätssichernden Maßnahmen teil.

6.8.2. Beurteilung

Das Projekt hat meine Erfahrungen im Umgang innerhalb eines Teams erweitert. Unser verwendetes Entwicklungsmodell passte gut zu unserem Projekt, aber auch die Schattenseiten wurden deutlich. Der große Aufwand der anfänglichen Einarbeitung wurde auch von mir unterschätzt, was zu kleineren Problemen führte, da das komplexe Eclipse-Framework viele Fehlerquellen bereithält. Dank Patricks großer Starthilfe und der anderen Teilnehmer konnte ich jedoch schnell die Starthürden überwinden. Vielen Dank an das ganze Team und den sehr fähigen Projektleiter für eine gute Atmosphäre während des Projektes, die Offenheit und das Engagement die den Zusammenhalt der Gruppe ausmachen.

6.9. Patrick Schneider

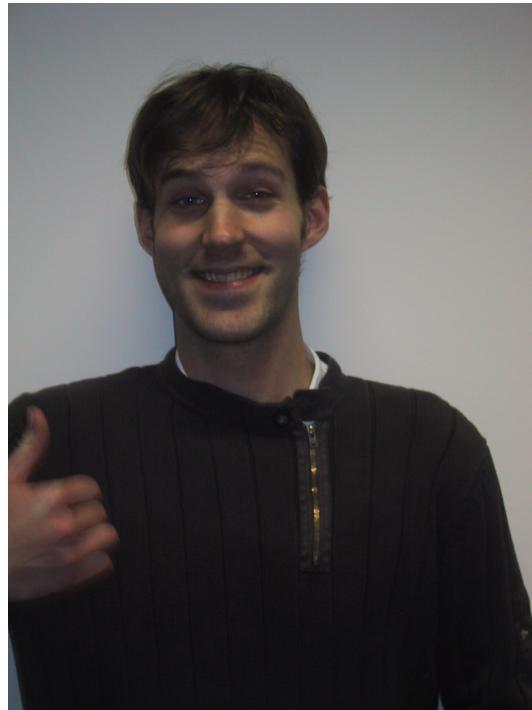


Abbildung 6.9.: Patrick Schneider

In diesem Projekt sammelte ich meine ersten Erfahrungen mit der eigenständigen Entwicklung von Software in einem größeren Team von Entwicklern. Ich arbeitete durch meine vorherige Tätigkeit im Rahmen meines Werkstudenten Jobs schon mehrfach in sehr viel größeren Projekten mit, nie jedoch so eingebunden in den Entwicklungsprozess, von der Analyse bis zur Auslieferung. Die Seminarvorträge boten für mich eine wichtige Wissensgrundlage für die Entwicklung des Tools.

6.9.1. Tätigkeiten

Im Laufe des Projektes variierten meine Aufgaben ständig. So verhalf ich durch meine Vorherigen Erfahrung in der Plug-In Entwicklung für Eclipse, anhand von Hilfestellung und einem Vortrag den anderen Team Mitgliedern zu einem Überblick über die Grundsätzlichen Eclipse Eigenschaften und der Architektur des Systems. Danach wurde ich auch noch häufiger bei Fragen und Problemen im Bezug auf Eclipse herangezogen und versuchte zu helfen wo es möglich war. Nach dieser Anfangsphase erstellte ich zusammen mit Bettina Druckenmüller die Spezifikation auf Basis der im Team und extern zusammengetragenen Anforderungen. Dies stellte sich als eine Herausforderung heraus,

da nicht immer ganz klar war was sich genau aus den Anforderungen in die Spezifikation übernehmen lies. Auch waren einige Anforderungen bis kurz vor Schluss noch weitgehend ungeklärt. An die Spezifikation schloss sich dann relativ schnell die erste Implementierungsphase an in der ich zuerst verschiedene Funktionalitäten des PLAM Plug-Ins realisierte. Als nächstes fiel dann das VCM Plug-In meinen Aufgabenbereich. Ich erstellte einen Abstraktionslayer zur Eclipseinternen Versionskontrollinterface und implementierte ein Grobgerüst zu den vom VCM Plug-In zur Verfügung gestellten VCM Aktionen. Dazu gehörten auch die Integration der Actions in die Menüstruktur des Gesamtsystems. Im Laufe der Entwicklung wurden die meisten Aktionen jedoch aus dem Benutzerobermenü entfernt und automatisiert, so dass der Benutzer die einzelnen Menüeinträge nicht mehr sieht. Um die Benutzerangaben die für das abarbeiten der VCM Aktionen nötig waren implementierte ich User Preferences und ein Preference Page zur Speicherung bzw. zur Änderung der Benutzerinformationen. Nach der Grobimplementierung schritt ich zur Ausarbeitung der einzelnen Tasks fort und Oliver erstellte mit mir zusammen die für die Funktionalität nötigen Schnittstellen für die extern aufgerufenen Skripte. Einige Schwierigkeiten bereiteten mir die Abarbeitung der verschiedenen externen Prozesse die für die Ausführung der Skripte benötigt wurden. Speziell die Kapselung in einzelnen Threads und Prozesse wies sich als sehr komplex und Fehleranfällig auf. Durch ständige Kurztests der Funktionalität anhand einer Runtime-Workbench fielen mir schon nach der 1. Implementierung viele Fehler auf die ich sogleich korrigierte. Da ich aber unter Windows programmierte und testete blieben mir viele Fehler verborgen die dann erst von Teamkollegen im Laufe der Zeit entdeckt wurden. Da sich das Datenmodell leider öfters änderte war ich dementsprechend oft und Zeitintensiv mit Änderungen und Anpassungen beschäftigt die nur den Zweck hatten Funktionalität wieder bzw. überhaupt erst herzustellen.

6.9.2. Fazit

Es traten wie in jedem größeren Projekt zu erwarten einige Probleme auf, die zu Erwarten waren. So wurden meines Erachtens von uns allen die frühen Phasen unterschätzt und führten später bei der Implementation zu einem Mehraufwand und Problemen. Was jedoch das gravierendste Problem dieses Projektes war, war der von Teammitglied zu Teammitglied sehr Unterschiedliche Arbeitseifer. Obwohl dies aus keinen der erhobenen Statistiken herauszulesen ist, war die Bereitschaft zur Einarbeitung und selbstständigen Problemlösung doch sehr unterschiedlich ausgeprägt. Dies führte dazu, dass einige Teammitglieder sehr viel mehr leisteten als andere, und das man manchen Leuten regelrecht ihre Aufgaben vorkauen musste. Relativiert wird dieses Argument durch den Fakt, dass alle Teammitglieder einen sehr unterschiedlichen Wissensstand im Bezug auf die eingesetzten Technologien hatten, und Anfangs bei einigen eher Frust über die relativ komplexe Struktur des Projektes und seines Frameworks vorherrschte. Ich bin sehr erfreut über die gute Atmosphäre, die trotz der technischen Schwierigkeiten im Team herrschte und bin überzeugt, dass jedes einzelne Teammitglied seine Erkenntnis-

se und Lehren aus diesem Projekt ziehen konnte. Leider konnten wir nicht alles umsetzen was wir uns zuerst vorgestellt haben, die Grundfunktionalität und einige nette Details sind letztlich doch zu meiner Zufriedenheit verwirklicht worden. Meiner Meinung nach bietet das Tool sehr hohe Benutzerfreundlichkeit sowie eine zeitgemäßes User Interface und integriert sich auch Problemlos in bestehende Entwicklungsprozesse. Ein weiterer wichtiger Pluspunkt des Projektes ist die Plattformunabhängigkeit die durch die Entwicklung in Java gewährleistet ist. Da die Sourcen für die Plattform und das Programm frei verfügbar sind, sollte eine unter Umständen gewünschte Anpassung keine größeren Schwierigkeiten bereiten.

A. Erklärung gemäß Prüfungsordnung

Ich erkläre hiermit, dass ich im Rahmen meiner Mitarbeit am Studienprojekt A "Produktlinien" außer von den Betreuern vorgesehenen bzw. genehmigten Hilfsmitteln keine unzulässige Hilfe in Anspruch genommen habe.

Stuttgart, den 26. November 2004

Necati Aydin

Anselm Garbe

Michael Grosse

Martin Plies

Oliver Rendgen

Patrick Schneider

Armin Cont

Bettina Druckenmüller

Tammo van Lessen