Fraunhofer Institut
Experimentelles
Software Engineering

MShop – Das Open Source Praktikum

OpenSource-Praktikum

Fraunhofer Institut
Experimentelles
Software Engineering

Autoren:
Marcus Ciolkowski[1]
Jens Heidrich[1]
Isabel John[2]
Wolfgang Mahnke[3]
Frank Przybilla[4]
Kristian Trottenberg[4]

1 AGSE
2 IESE
3 AGDBIS
4 maxess

IESE-Report Nr. 035.02/D
Version 1.0
01. Juli 2002

Eine Publikation des Fraunhofer IESE

Das Fraunhofer IESE ist ein Institut der Fraunhofer-Gesellschaft.
Das Institut überträgt innovative Software-Entwicklungstechniken, -Methoden und -Werkzeuge in die industrielle Praxis. Es hilft Unternehmen, bedarfsgerechte Software-Kompetenzen aufzubauen und eine wettbewerbsfähige Marktposition zu erlangen.

Das Fraunhofer IESE steht unter der Leitung von
Prof. Dr. Dieter Rombach
Sauerwiesen 6
67661 Kaiserslautern

Abstract

Das folgende Dokument beschreibt den Kontext des Open Source Praktikums im Bereich M-Commerce, welches im WS 2001/2002 an der Universität Kaiserslautern in Zusammenarbeit mit der ortsansässigen Firma „maxess systemhaus gmbh“ durchgeführt wurde.

Das Praktikum hat zwei Schwerpunkte: Die Aufgabenstellung des Praktikums im Bereich der mobilen Auftragsabwicklung (M-Commerce), und die Entwicklung des Systems als Open Source. Dies impliziert einerseits, dass das entwickelte System unter einer Open Source Lizenz freigegeben wird und von außen einsehbar ist. Zum anderen bedingt es, dass von Seiten „maxess“ kein Entwicklungsprozess vorgegeben werden kann und auch deswegen keine Annahmen über den zugrunde liegenden Prozess gemacht werden können.

Dieser Report beschreibt die verschiedenen Aspekte des Praktikums: Software Engineering, Open Source, realer Kunde maxess und M-Commerce Entwicklung. Er beschreibt die Anforderungen an die Studierenden und wie diese realisiert wurden und beschreibt den Ablauf des Praktikums. Des weiteren wird auf die Zufriedenheit und Erfahrungen der Teilnehmer, die durch eine Umfrage nach dem Praktikum erfragt wurden und die Erfahrungen der Betreuer in Form von Lessons Learnt eingegangen.

Schlagworte: Software Engineering Education; Open Source; M-Commerce; E-Commerce

Inhaltsverzeichnis

1 Einleitung

Das folgende Dokument beschreibt den Kontext des Open Source Praktikums im Bereich M-Commerce, welches im WS 2001/2002 an der Universität Kaiserslautern in Zusammenarbeit mit der ortsansässigen Firma „maxess systemhaus gmbh“ durchgeführt wurde.

Das Praktikum hat zwei Schwerpunkte. Dies ist zum einen die Aufgabenstellung des Praktikums im Bereich der mobilen Auftragsabwicklung (M-Commerce), d.h. ein Kunde soll später mit dem zu erstellenden System in der Lage sein, Einkäufe diverser Waren mit einem Mobilgerät (z.B. Palm Top oder Handy) zu erledigen.

Der zweite Schwerpunkt liegt in der Entwicklung des Systems als Open Source. Dies impliziert einerseits, dass das entwickelte System unter einer Open Source Lizenz freigegeben wird und von Außen einsehbar ist. Und zum anderen bedingt es, dass von Seiten „maxess“ kein Entwicklungsprozess vorgegeben werden kann und auch deswegen keine Annahmen über den zugrunde liegenden Prozess gemacht werden können. „maxess“ kann lediglich über Kriterien festlegen, welche Deliverables für sie akzeptabel sind und welche nicht. Auch von Seiten der Universität wird kein fester Prozess vorgegeben, es werden jedoch Richtlinien vorgegeben, um die Entwicklung nach Software Engineering Methoden sicherzustellen.

Die Organisation des Praktikums (also quasi die interne Realisierung) wird von den AGs „Software Engineering (SE)“ und „Datenbanken und Informationssysteme (DBIS)“ durchgeführt, d.h. sie stehen den Studenten beratend zur Seite. Der Kundenkontakt (die externe Realisierung des Praktikums) wird durch einen Domänenspezialisten von „maxess“ hergestellt, der das Praktikum mitbetreut. Das Fraunhofer IESE übernimmt Koordinationsaufgaben die Open Source Aspekte.

Für die Durchführung der Aufgabe wird den Studenten von „maxess“ ein Open Source Labor zur Verfügung gestellt. Hierin soll mit modernen PCs (auf Basis von Windows NT und Windows 2000) das System entwickelt und begleitend auf Palm Tops und wenn möglich Handys (oder entsprechenden Emulatoren) getestet werden.

Die Abbildung 1: zeigt den Praktikumsaushang.

Arbeitsgruppe Software Engineering
Prof. Dr. H.D. Rombach
Gebäude 57 / 5. Stock

Arbeitsgruppe Datenbanken und Informationssysteme
Prof. Dr. Theo Härder
Gebäude 36 / 3. Stock

4.9.2001

ACHTUNG!

Neues Praktikum:

Open Source Entwicklung

im Bereich M-Commerce

ACHTUNG!

- **Kooperation mit lokaler Firma (maxess systemhaus gmbh)**
- **Anwendung im Bereich Mobile Commerce mit Palm OS und Windows CE Geräten**
- **Softwareentwicklung: Technische Rollen & Qualitätssicherung**
- **Sofort anmelden! Beginn im WS 2001/2002**

Die AG Software Engineering und die AG Datenbanken und Informationssysteme bieten im Wintersemester 2001/2002 in Kooperation mit einer lokalen Firma (maxess systemhaus gmbh) ein **Open Source Praktikum im Bereich des Mobile Commerce** an.

In diesem Praktikum werden industrierelevante Aufgabenstellungen aus dem Bereich Mobile Commerce gelöst. Dazu wird ein **Labor mit modernster Technologie** bereitgestellt. Dies beinhaltet z.B. verschiedene Handhelds auf Basis von Palm OS und Windows CE.

In diesem Praktikum werden Kenntnisse aus der Vorlesung SE II (technische Rollen und Qualitätssicherung) vertieft. Das Praktikum kann als Alternative zum traditionellen SE II Praktikum der AG Software Engineering anerkannt werden.

Studenten der Angewandten Informatik können dieses Praktikum im Vertiefungsfach BI einbringen.

Anmeldung geschieht durch Eintrag in die Liste neben 57-534.

Vorbesprechung: 30.10.2001, 10 Uhr, in 57-528

Fragen? Dann meldet euch bei Marcus Ciolkowski
(57-525, Tel. 205 3339, Email: ciolkows@informatik.uni-kl.de)

Abbildung 1: Der Praktikumsaushang

2 Beteiligte

2.1 AGSE

Die AG "Software Engineering" existiert seit dem 01. Januar 1992. Sie beschäftigt sich in Lehre und Forschung mit der Untersuchung grundlegender Prinzipien des Software Engineering, der Entwicklung verbesserter Methoden und Werkzeuge zur effektiven Unterstützung einzelner Aspekte der Software-Entwicklung sowie der experimentellen Erprobung existierender Methoden und Werkzeuge. Im Rahmen der Erforschung grundlegender Prinzipien des Software Engineering konzentrieren sich die Arbeiten auf die Bereiche "Qualitative Ansätze zum Messen und Bewerten von Softwareprodukten und -prozessen", "Explizite Modellierung und Analyse von Softwareprozessen", "Wiederverwendung von Softwarewissen" sowie "Prozessbasierte Architekturen für Software-Entwicklungsumgebungen".

Im Rahmen der Entwicklung neuer Methoden und Werkzeuge zur Software-Entwicklung konzentrieren sich die Arbeiten auf "Planbare Softwareentwicklungsmethoden (z.B. Cleanroom)", "Formale Anforderungssprachen" sowie "Systematische Review- und Testverfahren". Im Rahmen der experimentellen Arbeiten werden die Stärken und Schwächen der vorgenannten Methoden und Werkzeuge in Form kontrollierter Experimente und industrieller Fallstudien untersucht.

Das Mshop Open Source Praktikum fand unter der Leitung der AGSE statt und wurde in den Räumen der AG durchgeführt.

2.2 AG DBIS

Der Forschungsschwerpunkt der Arbeitsgruppe Datenbanken und Informationssysteme (AG DBIS) liegt bei Entwurfs- und Implementierungsfragen von Informations- und Datenbanksystemen – speziell bei Modellierungs-, Architektur- und Realisierungskonzepten zur Unterstützung von komplexen Anwendungsbereichen. In den letzten Jahren werden dabei verstärkt Anforderungen verschiedener Ingenieuranwendungen berücksichtigt (CAD, CASE, etc.) und entsprechende Datenbank-Konzepte erarbeitet. Betrachtet werden dabei in erster Linie Fragen der Datenmodellierung und der Systemarchitektur, der Unterstützung von Prozessabläufen sowie die Lösung von Problemen, die im Zuge der Heterogenität und der zugehörigen Interoperabilität auftreten.

Als übergreifendes Forschungsthema werden zur Zeit objekt-relationale Datenbankkonzepte entwickelt bzw. ausgewertet. Weiterhin wird an Fragestellun-

gen im Zusammenhang mit dem World Wide Web gearbeitet – wie die Anbindung von Datenbanksystemen und die Datenbank-gestützte dynamische Aktualisierung von HTML- und XML-Dokumenten.

2.3 Fraunhofer IESE

Das Fraunhofer Institut Experimentelles Software Engineering (IESE) hat den Auftrag Ergebnisse aus der Grundlagenforschung zu sondieren, für den industriellen Einsatz aufzubereiten und letztendlich dauerhaft in die industrielle Praxis zu transferieren und zu etablieren. Experimentelles Software Engineering steht dabei für folgendes Vorgehen. Zunächst werden in Abhängigkeit von der spezifischen Umgebung eines industriellen Partners Technologien ausgewählt, die in dieser Umgebung optimalen Nutzen (d.h., Verbesserungen) bringen können. Danach werden die ausgewählten Technologien in Pilotprojekten beim Industriepartner eingeführt. Diese Einführung wird durch ein Messprogramm begleitet, das eine objektive Überprüfung des möglichen bzw. erbrachten Nutzens der eingeführten Technologien nach dem Pilotprojekt erlaubt.

Das Fraunhofer IESE koordinierte im Rahmen des Praktikums die Kontakte mit maxess und arbeitete die Open Source Aspekte aus.

2.4 Maxess

Die maxess systemhaus gmbh wurde 1995 als MARKANT-SÜDWEST Software und Dienstleistungs GmbH gegründet. Aus der Kooperation mit dem Fraunhofer Institut für Experimentelles Software Engineering IESE entstand 1998 das Warenwirtschaftssystem WWS 2000 – das heutige @x-trade. Ein integriertes System, welches spezielle Funktionen für den Lebensmittelhandel bietet, alle warenwirtschaftlich relevanten Prozesse von Warenbeschaffung über Lagerhaltung bis hin zum Vertrieb effizient abbildet und in der Branche als „State of the Art“ gilt.

Ergänzend dazu ging im Jahre 1999 mit der E-Business-Lösung @x-commerce bundesweit eine der ersten Anwendungen in Echtbetrieb, die den Zugang zum elektronischen Handel mit einem kompletten Lebensmittelangebot realisierte.@x-commerce ist eine kombinierte Internetshop- und Call Center-Lösung mit der Option der Anbindung einer mobilen Auftragserfassung. Abgerundet wird das Produktportfolio von maxess durch die Data Warehouse- und MIS-Lösung @x-decision, welche der Entscheidungsunterstützung auf allen Hierarchieebenen eines Unternehmens und der Führungsinformation für das Management dient.Selbstredend lassen sich @x-trade, @x-commerce und @x-decision zu einer ganzheitlichen, voll aufeinander abgestimmten Komplettlösung zusammenführen.

Im Oktober 2000 wechselte das Unternehmen Namen und Erscheinungsbild. Als maxess systemhaus gmbh beschäftigt der IT-Dienstleister heute nahezu 85 Mitarbeiter und positioniert sich aufgrund seiner aus dieser Branche erwachsenen Kompetenz und Projekterfahrung als spezialisierter Lösungsanbieter für den deutschen Lebensmittelhandel. Seit dem 1. April 2002 beteiligt sich der österreichische Logistiksystemspezialist Salomon Automation, Graz, zu 50 % an der maxess systemhaus gmbh. Zusammen mit der jungen Mannschaft will man die gemeinsame Vision verwirklichen: Die intensive Vermarktung der praxiserprobten Lösungen, um letztlich europaweit führender Anbieter für Warenwirtschaftslösungen im Lebensmittelhandel zu werden.

2.5 Die Studierenden

Das Praktikum wurde als Hauptstudiumspraktikum in Fachbereich Informatik durchgeführt. Es hatte 10 Teilnehmer, davon drei Studierende der Fachrichtung Diplom-Informatik und sieben Studierende der Fachrichtung Angewandte Informatik. Es nahmen Studierende aus den Semestern 7 bis 11 teil, dabei waren zum Zeitpunkt der Durchführung acht Studierende im 7. und zwei im 11. Fachsemester. Die Angewandten Informatiker benötigten zusätzlich zum Teilnahmeschein eine Benotung der im Praktikum erbrachten Leistung.

3 Die Aspekte des Praktikums

Das Praktikum, als Hauptstudiumspraktikum der Informatik, soll verschiedene Inhalte vermitteln. Im Unterschied zu anderen Praktika, die sich meist eng an den Lehrinhalten der veranstaltenden Arbeitsgruppe orientieren gibt es beim Mshop-Praktikum verschiedene Aspekte, die abgedeckt werden. Das Praktikum wird als Software Engineering Praktikum abgehalten (und geprüft), der Hauptaspekt des Praktikums ist also Software Engineering. Ein weiteres Hauptthema des Praktikums ist Open Source, also die freie Entwicklung von freier Software. Ein weiterer Aspekt, der hier eine Rolle spielt, ist Outsourcing: Man kann Open Source Projekte als extremes Outsourcing betrachten, wobei keinerlei Annahmen über den Entwicklungsprozess gemacht werden können. Relevant ist hier die Fragestellung, welche Vorgaben ein Unternehmen machen muss, um von einer solch extremen Form des Outsourcing Gebrauch machen zu können.

Das Anwendungsgebiet, in dem entwickelt wird, ist dem Mobile Commerce zuzuordnen, also dem Abwickeln von Geschäftsverkehr über mobile Endgeräte. Ein weiter Aspekt des Praktikums sind Datenbanken, die bei Geschäftsanwendungen eine große Rolle spielen. Das ganze Praktikum findet mit Unterstützung eines realen Kunden mit realen Anforderungen an das zu entwickelnde System statt.

3.1 Organisation

Dieses Praktikum unterscheidet sich von früheren SE II Praktika durch die neue Themenstellung, den externen Partner und die Ausrichtung auf Open Source Entwicklung. Deshalb sieht die Organisation des Praktikums auch etwas anders aus als in früheren Praktika. Traditionelle SE II Praktika haben einen Schwerpunkt auf organisatorischen Rollen wie Projektmanagement. Das heißt, dass die Studenten selbst für die Planung und Durchführung des Praktikums verantwortlich sind. Dabei erhalten sie Planungsunterstützung in Form von vorhandenen Prozess- und Qualitätsmodellen; das heißt, ein Prozess ist vorgegeben und muss von den Studenten an die Anforderungen angepasst und anschließend überwacht werden.

Die Organisation des Mshop Praktikums ist zweigeteilt, mit einer definierten Schnittstelle zwischen den Teilen (siehe Abbildung).

Durch die Zusammenarbeit mit „maxess“ und durch die Entwicklung der Software Komponenten als Open Source Kode ist die Organisation zweigeteilt:

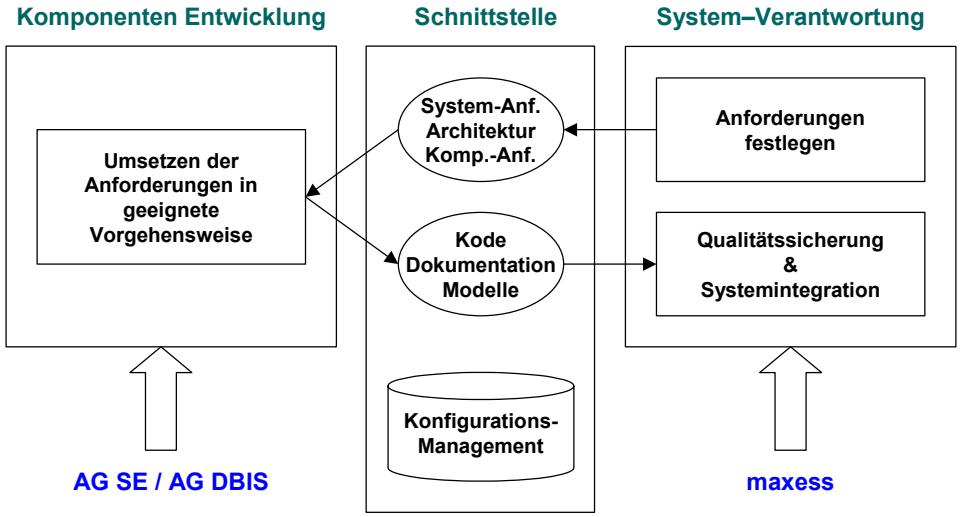

Abbildung 2: Organisation während der OS Entwicklung.

„maxess“ hat die System- und Integrationsverantwortung, d.h. „maxess“:

- definiert die Anwendung, die entwickelt werden soll
- legt Qualitätssicherungsmaßnahmen wie Abnahmekriterien für Komponenten, zu erstellende Dokumentation, Testfälle usw. fest
- ist verantwortlich für die Systemintegration, d.h. die funktionsfähige Integration der neu entwickelten Komponenten in das existierende Basissystem,
- stellt die fachliche Kompetenz und ist Ansprechpartner bei Fragen zu Anforderungen, zum System und zur Anwendung.

Die AG Software Engineering und die AG Datenbanken und Informationssysteme kümmern sich um die Betreuung des Praktikums und die Entwicklung, d.h. die beiden Arbeitsgruppen und das Fraunhofer IESE:

- sind erster Ansprechpartner für alle Belange
- kümmern sich um die Organisation des Praktikums (Zugang zu den Räumen und den Rechnern, Umgang mit den Tools, etc.)
- liefern Unterstützung für die Durchführung der Entwicklung, das Erstellen von Modellen und Dokumentationen, das Konfigurationsmanagement, Tools, Datenbankintegration usw.

4 Software Engineering

Das Ziel von Software Engineering ist im Allgemeinen die systematische Entwicklung eines Programms oder Software-Systems.

Um Systematik im Erstellungsprozess zu gewährleisten müssen einige Grundvoraussetzungen gegeben sein. Ausgangspunkt für die systematische Entwicklung eines Software-Systems sind explizite Anforderungen an das zu erstellende System. Diese dienen im Späteren (außer als Abkommensgrundlage) einerseits dazu, systematisch einen Entwurf abzuleiten, der den Anforderungen gerecht wird, und andererseits dazu, das Endprodukt (sprich das laufende System) gegen diese Anforderungen zu testen.

Um Software-Entwicklungsprojekte systematisch durchführen zu können, benötigen wir explizite Planung vor Beginn eines Projektes. Dies umfasst zuerst die Definition des Projektzieles (d.h., was mit dem Projekt erreicht werden soll) und der Projektcharakteristika (d.h., unumstößlicher Rahmenbedingungen für das Projekt). Weiterhin umfasst dies die Festlegung eines geeigneten Entwicklungsprozesses, also einer ingenieurmäßigen Vorgehensweise für die Entwicklung des Software-Systems und die Auswahl passender Techniken, Methoden und Werkzeuge gemäß der spezifischen Aufgabenstellung und deren Anpassung für den Einsatz in einem konkreten Projekt.

Während der Durchführung des Projektes ist es wichtig, dass die Einhaltung der in der Planungsphase festgelegten Rahmenwerte überwacht wird und ggf. umgeplant werden kann. Hierfür sind das Projektmanagement und die Qualitätssicherung verantwortlich.

Kernaspekte der ingenieurmäßigen Erstellung von Software sind explizit definierte Produkte, Prozesse und Rollen. Als Produkte bezeichnet man alle Artefakte, die während der Software Entwicklung produziert oder konsumiert werden. Ein Produkt ist also nicht nur der ausführbare Code, sondern auch die Dokumentation, Modelle, Anforderungsbeschreibungen usw. Produkte werden durch Prozesse erstellt. Zur Ausführung der Prozesse werden verschiedenen Rollen verschiedene Aufgaben zugewiesen.

Beim Mshop Praktikum war es notwendig, aufgrund der speziellen Ausrichtung, spezielle Produkte zu erstellen. Durch die Ausrichtung auf Open Source Entwicklung konnten nur wenige grundlegende Prozesse vorgegeben werden. Im Unterschied zu traditionellen SE II Praktika sollten auch nur wenige Rollen vorgegeben werden, da ja der Open Source Entwicklungsprozess nicht durch ein starres Rollengerüst eingezwängt werden soll.

4.1 Produkte

Als Produkte (synonym: Artefakte) werden alle Ergebnisse von Software-Entwicklungsaktivitäten sowie Informationen bezeichnet, die für deren Durchführung benötigt werden. Beispiele für Produkte sind ein Anforderungsdokument, eine Entwurfsdokument, ein UML-Klassendiagramm, ein Kodestück oder eine ausführbare Kodekomponente. Produkte können als Verfeinerung eine Menge von Teilprodukten besitzen, die wiederum verfeinert sein kann. So kann beispielsweise ein UML-Entwurf aus einer Menge von UML-Diagrammen unterschiedlicher Typen bestehen.

Die von maxess geforderten Produkte finden sich in Abschnitt 7.5.1.

4.2 Prozesse

Ein Prozess (synonym: Aktivität) beschreibt eine Menge partiell geordneter Schritte, um ein Ziel zu erreichen. Beispiele für Prozesse sind das Erstellen der System-Anforderungen, die Bearbeitung des Entwurfs, die Verifikation von Kodekomponenten, die Integration von Teilsystemen oder ein Akzeptanztest. Prozesse können als Verfeinerung eine Menge von Teilprozessen besitzen, die wiederum verfeinert sein können. Prozesse sind gegenstandslose Phänomene der realen Welt.

4.3 Rollen

Software-Entwicklungsprozesse werden von Agenten (Personen oder Werkzeugen) durchgeführt. Diese spielen bei der Durchführung eines Prozesses bestimmte Rollen. Eine Rolle trägt die Verantwortung für die Erledigung der mit dem Prozess verbundenen Aufgaben. Beispiele für Rollen sind: Projektplaner, Projektmanager, Qualitätssicherer, Konfigurationsmanager, Anforderungsingenieur, der Entwurfsingenieur, der Validierer oder beispielsweise der Kunde. Eine Rolle kann von mehreren Agenten ausgeführt werden; ein Agent kann aber auch mehrere Rollen ausführen. In manchen Fällen sind Rollen direkt für Produkte verantwortlich.

Die folgenden Rollenbeschreibungen sind speziell auf das OS Praktikum zugeschnitten. Es handelt sich dabei *nicht* um eine allgemeine Beschreibung der Aktivitäten der einzelnen Rollen in einem Software-Entwicklungsprojekt. Wegen der Gestaltung des Praktikums als Open Source kommen einige Aufgaben hinzu, wie beispielsweise die freie Prozesswahl von Seiten der Entwicklungsteams, auf der anderen Seite fallen aber auch Aufgaben weg, die z.B. mit der expliziten Modellierung eines Projektplans (wie in MVP-L, [BLRV1995]) und dessen Einhaltung zu tun haben.

Wenn im folgenden vom Projekt-Manager (PM), Qualitätssicherer (QS), Konfigurations-Manager (KM) oder Entwickler die Rede ist, so deutet dies ausdrücklich *nicht* auf eine 1:1-Beziehung zwischen Rolle und Praktikant hin. Eine Rolle kann durchaus von mehreren Personen wahrgenommen werden. Umgekehrt kann aber eine Person auch mehrere Rollen ausfüllen.

Die managementorientierten Rollen (PM, QS und KM) tragen die Verantwortung für ihren abgesteckten Aufgabenbereich. Dies bedeutet aber *nicht*, dass sie alle Arbeiten selbst ausführen müssen. Sie können - soweit sinnvoll – auch einzelne Aufgaben delegieren. Die Besetzung und Wahrnehmung der Rollen geschieht in Eigenverantwortung der Praktikumsteilnehmer.

4.3.1 Der Projekt-Manager

Der Projekt-Manager ist für die Koordination des Teams, die Überwachung der Entwicklungsaktivitäten und die Kommunikation mit dem Kunden verantwortlich.

Koordination der Teams

Diese Aufgabe beinhaltet die Teamkoordination und die Koordination der Teams untereinander - soweit erforderlich [Jal1997]. Zur Teamkoordination gehören das Kümmern um die sozialen Aspekte des Teams (*Wer bin ich? Wer bist Du? Was tun wir hier? Wie tun wir das?*), das Absprechen von Zuständigkeiten für bestimmte Aufgaben (*Wer tut was?* basierend auf: *Wer kennt sich womit am besten aus*? oder *Wer ist woran am meisten interessiert?*), das Festlegen von Regeln zur Arbeit im Team (*Wie wollen wir zusammen arbeiten?*), die Motivation der einzelnen Teammitglieder (*Welche Gründe gibt es für eventuelle Demotivation? Was können wir dagegen tun?*).

Die Koordination mit anderen Teams ergibt sich beispielsweise daraus, dass die Teams ihre Dokumente gegenseitig inspizieren (falls Inspektionen durchgeführt werden) und die Systeme (oder Komponenten) der Teams gegen Projektende zu einem Gesamtsystem integriert werden. D.h., Termine müssen abgesprochen werden, und die Bearbeitung gemeinsamer Dokumente muss abgestimmt werden. Abhängig von den am System durchgeführten Änderungen ist bei einigen Teams zusätzlich eine inhaltliche Zusammenarbeit erforderlich. In diesem Fall wird von jedem Team eine Person als Koordinationsperson benannt, die für die Koordination der Arbeiten der beiden Teams verantwortlich ist.

Während des Projektes ist es wichtig, die zu Beginn vereinbarten Koordinationsregeln zu überprüfen und für ihre Einhaltung zu sorgen. Bei anhaltenden Koordinations- oder Kommunikationsschwierigkeiten muss eingegriffen werden, um das Projektziel nicht zu gefährden. Ein Großteil der Kommunikation des Teams

soll - neben den wöchentlichen Treffen - über ein Diskussionsforum (in diesem Fall Smartgroups http://www.smartgroups.com/groups/praktikum-se2) erfolgen. Der PM sollte die Diskussion in den Smartgroups überwachen und moderieren, sowie gegebenenfalls aktuelle Ankündigungen darin machen.

Überwachung der Entwicklungs-Aktivitäten

Der PM kontrolliert die Aktivitäten der Entwickler und behält den Überblick über begonnene und beendete Arbeiten, also den Status des Gesamtprojektes (in Zusammenarbeit mit dem QS und KM). Der PM sollte den Überblick über die Aufwände der Aktivitäten der Entwickler behalten. Dazu ermittelt er aus den Daten auf den Fragebögen für jede Aktivität und jede daran beteiligten Person, wieviel Aufwand in die Bearbeitung der Aktivität geflossen ist (die Daten sollten von der Qualitätssicherung bereitgestellt werden).

Während der Durchführung eines SW-Entwicklungsprojektes ist es wichtig, dass im Vorhinein festgelegte Meilensteine eingehalten werden. Falls abzusehen ist, dass wichtige Meilensteine nicht mehr einzuhalten sind, sollten gemeinsam eventuelle Abweichungen und Gründe dafür diskutiert werden. Das weitere Vorgehen ist daraufhin erneut festzulegen.

Kunden-Kommunikation

Der PM dient als Hauptansprechpartner für den Kunden (maxess systemhaus gmbh). Falls Fragen zum aktuellen Status des Projektes auftauchen, oder sonstige Fragen die Entwicklung betreffend, so ist der PM die Hauptanlaufstelle. Er kann dann entweder die benötigten Informationen von den Teammitgliedern in Erfahrung bringen oder direkt an zuständige Entwickler verweisen.

4.3.2 Der Qualitätssicherer

Der Qualitätssicherer ist für die Überwachung der Qualitätskriterien und die Erfassung von Messdaten verantwortlich.

Überwachung der Qualitätskriterien

Der QS ist für die Einhaltung der geforderten Qualitätskriterien zuständig. Zur Überprüfung dieser Kriterien kann er einerseits die erstellten Produkte direkt überprüfen (z.B. einen Kode-Abschnitt im Hinblick auf die Einhaltung der Kodierungs-Richtlinien) oder er kann Fragebögen austeilen und auswerten, um Kriterien zu überprüfen, die nicht direkt am erstellten Produkt abgelesen werden können (z.B. Aufwand der einzelnen Entwickler).

Der QS trägt die zentrale Verantwortung für die Qualitätssicherung. Er kann und sollte Aufgaben soweit sinnvoll an Team-Mitglieder verteilen. Beispielsweise kann die Einhaltung von Dokumentations-Richtlinien einzelner Artefakte auf

verschiedene Gruppen, die möglichst ungleich der Gruppe der Erzeuger des Artefaktes ist, verteilt werden.

Die Qualitätskriterien sind mit dem Kunden (maxess systemhaus gmbh) abzugleichen, d.h. wenn der Kunde neue Kriterien hinzufügen möchte, so ist der QS dafür verantwortlich, dass diese Kriterien entsprechend umgesetzt werden. Lassen sich umgekehrt Kriterien aus entwicklungstechnischen Gründen nicht einhalten, so ist der Kunde ebenfalls zu benachrichtigen und es muss zusammen mit dem PM und dem Kunden über eine Abschwächung der Kriterien diskutiert werden.

Werden Inspektionen oder Tests durchgeführt, so ist der QS der Ansprechpartner, um Inspektions- und Testaktivitäten zu koordinieren und aufeinander abzustimmen. Beispielsweise sollte ein Entwicklungsteam, welches eine Komponente erstellt hat, diese nicht selbst inspizieren oder Testfälle durchspielen. Dies sollte von einem außenstehenden (die Komponente nicht betreffenden) Team durchgeführt werden.

Erfassen von Messdaten

Der QS sammelt praktikumsbegleitend festgelegte Messdaten des Projektes (wie z.B. Aufwand der Teammitglieder) und verwaltet die Informationen beispielsweise in MS Excel. Er bereitet die Datenauswertung vor, indem er sich überlegt, welche Charts er wie mit Excel erzeugen wird. Die Daten sollen so abgelegt werden, dass die geplanten Auswertungen möglichst einfach zu erstellen sind. Die Auswertungen werden jeweils zum Abschluss eines Meilensteins erstellt und in der Projektbesprechung diskutiert.

Der QS muss sich für eine Darstellungen der Daten entscheiden, die er in Bezug auf das Projektteam für besonders nützlich hält und die erhobenen Daten aufbereiten. Weitere interessante Fragestellungen aus Sicht des QS können hinzugefügt werden. Die Auswertungen werden nach Bedarf bei jeder Teambesprechung, mindestens aber bei jedem Meilenstein präsentiert und es wird gemeinsam mit dem Team diskutiert, welche Schlüsse aus den Daten gezogen werden können. Dies gilt sowohl für die Entwickler als auch für die Vorgehensweisen nach denen sie entwickeln. Was bedeutet es zum Beispiel, wenn besonders viele Fehler einer bestimmten Art gefunden wurden? Verschiedene Gründe dafür sind möglich, z.B. falsch verstandene Entwicklungsrichtlinien, nicht genug Zeit bei der Entwicklung, usw. Die beteiligten Entwickler haben sicher Vermutungen, welche Gründe wahrscheinlicher sind als andere. Eventuelle Verbesserungsvorschläge sollten notiert werden.

Während des Praktikums streben wir die Erfassung von mindestens zweierlei Messdaten an. Dies umfasst zum einen den Aufwand der Entwickler (*Wie lange wurde täglich an welchem Diagramm, welcher Komponente, etc. gearbeitet?*) und zum anderen das Fehlermodell (*Wann wurde welcher Fehler gefunden?*

Wo wurde er gefunden? Wie lange dauerte seine Behebung?). Wie diese Messdaten erfasst werden ist Sache des QS. Vorstellbar wären ausgeteilte Fragebögen für den Aufwand oder eine Dokumentation des Fehlermodells über das CVS-System (z.B. bei einer Versionsänderung die Fehler, die behoben wurden, als Kommentar hinzufügen).

Wichtig ist eine Anonymisierung der Aufwandsdaten der einzelnen Entwickler. Diese Daten werden nicht zur Benotung des Praktikums herangezogen, sondern dienen lediglich der Dokumentation des Projektes und zur Planung für zukünftige Projekte. Damit dies sichergestellt ist, hat nur der QS Zugriff auf diese Aufwandsdaten. Alle Daten, die bei Team-Besprechungen vorgestellt werden, sind entsprechend zu anonymisieren.

4.3.3 Der Konfigurations-Manager

Der Konfigurations-Manager ist für die Versionsverwaltung, die Projekt-Status Verwaltung und die Release Verwaltung verantwortlich.

Versionsverwaltung organisieren

Der KM verwaltet die entwickelten Produkte, und damit alle Versionen und Konfigurationen, die bei den Entwicklungstätigkeiten erstellt werden. Um einen Überblick über die zu verwaltenden Produkte zu erhalten, muss er sich ein Bild vom Produktmodell machen (siehe Beschreibung der Beispiel-System-Dokumentation). Das Produktmodell ist üblicherweise Bestandteil eines Projektplans und formal beschrieben (z.B. in der Prozess-Modellierungssprache MVP-L [BLRV1995]). Da es im OS-Praktikum keinen explizit beschriebenen Projektplan gibt, müssen die Beispiel-System-Dokumentation und die Maxess-Abnahmekriterien für die Produkte als Grundlage ausreichen.

Die Versionskontrolle der Produkte erfolgt über CVS. Eine entsprechende Dokumentation findet sich im Praktikumsordner im OS-Labor oder online unter www.cvshome.org. Der KM ist für die Versionierungsstrategie verantwortlich. Während der Entwicklung muss bei allen Dokumenttypen auf die Versionierung geachtet werden. Dazu ist mit dem Team ein Vorgehen abzustimmen, das bei hinreichender Disziplin aller Beteiligten zu ausreichender Versionsverwaltung aller Dokumente führt. Das heißt, es ist eine Strategie festzulegen, *wer*, *wann*, *welche* Dokumente (oder Teile davon) ein- und auschecken darf und *was* bei Zugriffskonflikten passiert (falls parallele Änderungen an Dokumenten möglich sind). Die gewählte Strategie sollte gut dokumentiert allen zugänglich gemacht werden. Ihre Einhaltung ist zu überprüfen und bei Abweichung ist in den Team-Besprechungen auf Versäumnisse hinzuweisen.

Der KM ist dafür verantwortlich, dass alle Team-Mitglieder jederzeit auf die für sie benötigen Produkte zugreifen können.

Projekt-Status-Informationen verwalten

Der KM hält die restlichen Projektmitglieder über den aktuellen Produktstatus auf dem Laufenden. Dies tut er z.B. mit Hilfe einer Tabelle (z.B. Word- oder HTML-Tabelle), die allen Projektbeteiligten jederzeit zugreifbar sein sollte. Auch die Tabelle sollte einer Versionierung unterliegen, um später den Projektverlauf rekonstruieren zu können.

Der KM ist dafür verantwortlich, dass die Quelle von Dokumenten (z.B. Ablageort verschiedener Artefakte auf dem Server) stets allen kenntlich gemacht wird.

Release-Verwaltung

Der KM ist für die Zusammenstellung einer lauffähigen Version des Systems verantwortlich. D.h., dass fertige Komponenten, sofern möglich, in Form eines System-Releases zusammengestellt und verfügbar gemacht werden (z.B. für Präsentationen gegenüber dem Kunden).

4.3.4 Der Entwickler

Der Entwickler teilt dem PM und dem KM regelmäßig den aktuellen Stand seiner Aktivitäten mit, d.h. welche Aufgaben bereits erledigt sind und welche noch anstehen. Zu den anstehenden Aufgaben sollte eine Aufwands- und Dauerabschätzung angegeben werden.

Der QS informiert den Entwickler, auf welche Daten der zu erstellenden Produkte geachtet werden muss und wann diese Daten zu erfassen und weiterzumelden sind. Der Entwickler liefert diese Daten rechtzeitig und vollständig.

Der Entwickler muss die vom KM festgelegte Versionierungsstrategie (siehe oben) einhalten.

Im OS Praktikum ist der Entwickler selbst für den intern verwendeten Prozess verantwortlich (den er im Rahmen der SE Prinzipien frei gestalten kann), d.h., inwieweit beispielsweise Inspektionen durchgeführt werden oder getestet wird hängt vom Entwickler selbst ab. Im konkreten Fall arbeiten mehrere Praktikanten an einem Teil des Systems (siehe z.B. Grobstruktur des Systems aus der Problembeschreibung). Alle Personen dieser Gruppe nehmen die Rolle des Entwicklers wahr und einigen sich darauf, mit welchem Prozess der Teil des Systems - für den sie die Verantwortung tragen – entwickelt wird. Entsprechende Aktivitäten des gewählten Vorgehens müssen aber wieder mit dem PM und QS koordiniert werden.

5 Entwicklung von M-Commerce Software

Bei der Entwicklung von M-Commerce Software sind besondere Rahmenbedingungen zu berücksichtigen. Bei mobilen Anwendungen tritt das Mobilgerät (Palm, Pocket PC, Handy etc.) im Allgemeinen als Client zu einem Server auf, auf dem wichtige Software läuft (z.B. ein e-Commerce Shop). Zum einen ist beim Client, d.h. dem Mobilgerät, in der Regel nur leistungsschwache Hardware vorhanden, die zusätzlich leicht ausfallen (Batterie/Defekt) oder verloren gehen kann. Zum anderen ist die Verbindung vom Client zum Server (z.B. über Internet) oft teuer, hat nur eine geringe Bandbreite und kann leicht unterbrochen werden.

Es gibt grundsätzlich zwei Möglichkeiten, diese Rahmenbedingungen zu berücksichtigen. Die eine Möglichkeit besteht darin, den Client zu entlasten und möglichst wenig auf dem Client auszuführen. Das bedeutet, das möglichst viel auf dem Server durchgeführt werden muss. Dies hat allerdings den Nachteil, das zum einen ein hoher Kommunikationsaufwand erforderlich ist und zum anderen während der gesamten Bearbeitung die Verbindung gehalten werden muss. Die zweite Möglichkeit besteht darin, die Verbindung zu entlasten, indem benötigten Daten im Client gespeichert und dort bearbeitet werden. Die Verbindung muss nur für den Datenabgleich hergestellt werden. Hier ist der Nachteil, dass die Arbeit auf dem Client ausgeführt werden muss, dessen Leistung meist recht beschränkt ist. Ferner können bei einem Verlust oder Ausfall des Clients (d.h. des Mobilgeräts) Daten verloren gehen.

Für das Praktikum wurde die zweite Möglichkeit gewählt, um einen hohen und teuren Kommunikationsaufwand zu vermeiden. Da die meisten Daten im Mobilgerät redundant gehalten werden und lediglich die Daten der Bestellungen im Client erzeugt werden, ist ein Verlust dieser Daten annehmbar.

Damit sieht das Praktikum eine Software-Entwicklung für Mobilgeräte vor. Die verwendete Programmiersprache ist Java. Die Entwicklung für Mobilgeräte bedingt den Einsatz einer speziell für diesen Bereich angepassten Entwicklungsumgebung. Im folgenden wollen wir kurz die verwendeten Werkzeuge und ihre wesentlichen Features beschreiben.

5.1 Java 2 Micro Edition

Die *Java 2 Micro Edition* (J2ME) [Knu01] ist speziell für den Einsatz auf Geräten mit limitiertem Speicher und limitierter Prozessorleistung zugeschnitten. Um möglichst viele Geräte abzudecken besitzt sie eine dreigeteilte Struktur. Auf

Basis einer *Virtual Machine* bauen verschiedene *Konfigurationen* und *Profile* auf, welche wiederum verschiedene Geräteklassen abdecken. Eine Übersicht ist in Abbildung 3: dargestellt.

Die *Java Virtual Machine* (VM) ist für ein spezielles Betriebssystem angepasst und unterstützt eine spezielle Konfiguration. Sie ist für die Ausführung des Bytekodes auf dem Zielsystem (also den PDAs) verantwortlich. Sun stellt die KVM (kilo VM) zum Einsatz auf Palm OS und ähnlichen Geräten zur Verfügung. Es gibt aber weitere VMs anderer Hersteller, wie z.B. Jbed (kompiliert Java Bytekode in nativen Kode), Kada (Clean Room Implementierung) oder Superwaba (Open Source Implementierung).

Die *Konfiguration* ist für den Benutzer zunächst unsichtbar. Sie definiert eine Art größter gemeinsamen Teiler zwischen Geräten mit ähnlichen Hardware-Voraussetzungen in Bezug auf Features und Bibliotheken der Java Plattform. Für Palm OS und ähnliche Geräte gibt es die Connected Limited Device Configuration (CLDC). Sie ist ganz allgemein für Geräte mit 16- oder 32-bit CPU und einem Speicher von 512 KB für die Java Plattform und Anwendungen gedacht. Sie adressiert Low Level Funktionalitäten, wie Ein- und Ausgabe, Sicherheitskonzepte, Internationalisierungen, VM Features und Basis Bibliotheken (`java.lang.*` und `java.util.*`).

Auf einer Konfiguration können verschiedene *Profile* aufbauen, die jeweils eine Familie von Geräten vereinen und eine Menge an geeigneten APIs zur Verfügung stellen. Sie sind die für den Benutzer direkt sichtbare Schicht. Jede Anwendung, die für ein bestimmtes Profil geschrieben wurde, läuft auf allen Geräten, die dieses Profil unterstützen. Umgekehrt kann ein Gerät aber auch mehrere Profile bedienen. Das Mobile Information Device Profile (MIDP) baut auf der CLDC auf. MIDP fügt zu den durch die CLDC bereitgestellten Low Level Funktionalitäten noch weitere APIs bezüglich UI, DB-Zugriff und Gerätespezifischem Netzwerk-Zugriff hinzu. Die Applikationen des MIDP werden als MIDlets bezeichnet (in Anlehnung an Applets). Sie laufen z.B. auf geeigneten Mobiltelefonen und Pagern.

Im Praktikum werden also MIDlets für PDAs erstellt, welche dem CLDC/MIDP-Standard entsprechen. Dies garantiert größtmögliche Portabilität des erstellten Kodes. Sun stellt für diese Kombination das *Wireless Toolkit* (verwendete Version 1.0.3) zur Verfügung. Hiermit ist es möglich den vom Entwicklungswerkzeug erzeugten Kode in eine Form zu bringen, die auf den Endgeräten installiert und ausgeführt werden kann.

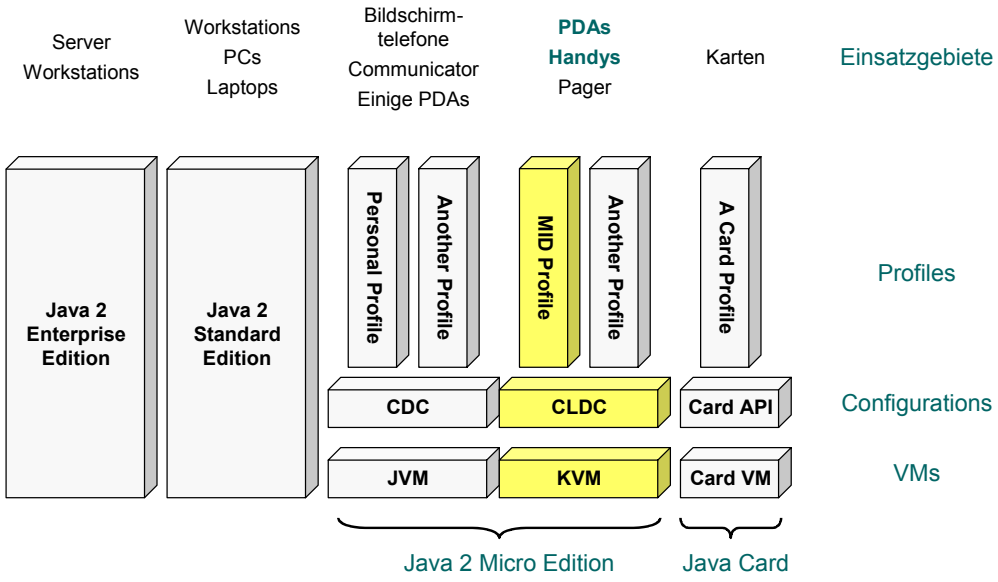

Abbildung 3: Einordnung der Java 2 Micro Edition.

5.2 Emulatoren

Es ist nicht sinnvoll, während der laufenden Entwicklung immer wieder das Programm zu Testzwecken auf dem Zielgerät zu installieren, da die Software auf dem mobilen Endgerät nicht richtig zu debuggen ist. Vielmehr möchte man das Programm aus einem Emulator heraus in Zusammenhang mit einem integrierten Debugger der Entwicklungsumgebung testen. Das Wireless Toolkit von Sun bietet zu diesem Zweck auch diverse Emulatoren an, bzw. stellt eine Anbindung für Emulatoren von Geräte-Herstellern zur Verfügung, wie Palm OS oder Windows CE Emulatoren. Diese Emulatoren können dann „remote“ über das Wireless Toolkit an den Debugger der Entwicklungsumgebung angebunden werden.

Modellierungs-/Entwicklungsumgebungen

Für die Modellierung des Praktikum-Systems und für die Bearbeitung des resultierenden Quellkodes setzen wir ein integriertes CASE-Werkzeug ein. *Together 5.5* von Togethersoft stellt eine komplette Modellierungs- und Entwicklungsumgebung dar. Der Java-Quellkode kann direkt aus den UML Diagrammen erzeugt werden. Änderungen im Kode werden konsistent in den Diagrammen mitverwaltet. Der Kode kann aus Together heraus in Verbindung mit einem der oben erwähnten Emulatoren gedebuggt werden. Es besteht also eine durchgängige Verbindung zwischen Modellen, Quellkode und ausgeführtem Kode (auf dem Emulator) innerhalb von Together.

6 Open Source Entwicklung

Das Prinzip der Open Source Entwicklung, also freie Entwicklung ohne fest definierten Prozess und Freigabe der entwickelten Source, für ein Software Engineering Praktikum einzusetzen scheint auf den ersten Blick etwas ungewöhnlich, hat aber viele Vorteile. Durch die Wahl einer geeigneten Open Source Lizenz kann sichergestellt werden, dass maxess den im Praktikum entwickelten Code verwenden kann, die Studenten jedoch nicht als „Auftragsentwickler“ missbraucht werden. Die Open Source Lizenzierung ist als ein Kompromiss zwischen dem Wunsch jedes Unternehmens, exakt auf die Unternehmensziele zugeschnittene Software zu bekommen und der Pflicht der Praktikumsbetreuenden, im Rahmen des Praktikums keine kommerzielle Software entwickeln zu lassen. Ein freier Open Source Entwicklungsprozess lässt eine Einflussnahme des Kunden, also maxess, z.B. in der Anforderungserstellung und bei der Abnahme durchaus zu, gesteht den Studenten jedoch zu, den Entwicklungsprozess im Rahmen der Software Engineering Prinzipen frei zu gestalten.

6.1 Allgemeines

Das Praktikum Software Engineering II soll als Open Source Praktikum durchgeführt werden. Die Quellen, die während des Praktikums erzeugt werden, werden nach dem Praktikum (oder bei Bedarf schon während des Praktikums) der Öffentlichkeit zugänglich gemacht. Um dies sicherzustellen, wird schon vor Beginn der Entwicklung die ausgewählte Open Source Lizenz von allen Beteiligten (Praktikumsteilnehmer, maxess systemhaus gmbh, AG SE, AG DBIS) unterzeichnet und schließlich zusammen mit dem Kode als Einheit ins Internet zum freien Download gestellt. Open Source bedeutet für uns bei der Durchführung des Praktikums:

- Der Kode ist frei verfügbar.
- Der erstellte Kode kann nach Beendigung des Praktikums elektronisch publiziert werden.
- Der Kode wird unter Konfigurationsmanagement gestellt, um die Verfolgbarkeit von konkurrierenden Versionen immer zu gewährleisten.
- Der Kode wird unter Open Source Lizenz (Mozilla Public License) gestellt und kann und darf unter Einhaltung der Lizenzbestimmungen verwendet und weiterverteilt werden.
- Die Entwicklung erfolgt aufgabenorientiert, es werden Aufgaben verteilt, die in der vorgegebenen Zeit zu lösen sind.

- Es gibt keinen vorgefertigten Entwicklungsprozess, es ist jedoch darauf zu achten, dass alle erforderlichen Produkte erstellt werden und alle Rollen ausgefüllt sind.
- Open Source Entwicklung bedeutet auch: Eigenständigkeit der Entwickler. Es existiert kein Auftrageber/Auftragnehmer Verhältnis zwischen dem Ersteller und dem Nutzer der Software. Dies bedeutet eine größere Freiheit für den Entwickler, aber auch eine größere Verantwortung, da er, teilweise ohne direkten Kontakt zum Kunden, Anforderungen implementieren muss.
- Bei gemeinsamer, verteilter Open Source Entwicklung ist vor allem auf die Schnittstellen der zu erzeugenden Komponenten zu achten.
- Eine nachträgliche Qualitätssicherung ist bei Open Source Entwicklung sehr wichtig, da während der Erstellung der Entwickler alleine für den Kode, seine Qualität und die Abbildung der Anforderungen auf den Kode verantwortlich ist.

6.2 Entwicklungsprozess

Um die Qualität der zu erzeugenden Softwareprodukte sicherzustellen, wird normalerweise im SE-Praktikum ein Entwicklungsprozess und dazugehörige Rollen vorgegeben. Dieser Prozess und die durch den Prozess zu erzeugenden Produkte sind normalerweise während der Entwicklung einzuhalten. Dies wird beim Open Source Praktikum nicht der Fall sein.
Open Source Entwicklung zeichnet sich durch ein hohes Maß an Freiheit und Eigenverantwortung aus. Jeder Teilnehmer kann seinen eigenen Entwicklungsprozess wählen oder einen existierenden Prozess modifizieren. Es sollte aber auf jeden Fall darauf geachtet werden, dass nach Software Engineering Prinzipien (z.B. Dokumentation von Entscheidungen, Verfolgbarkeit von Änderungen usw.) entwickelt wird. Des weiteren müssen grundlegende Rollen (Projektmanager, Qualitätssicherer, Konfigurationsmanager, etc.) weiterhin benannt und von den Praktikumsteilnehmern ausgefüllt werden. Dies geschieht jedoch in Eigenorganisation.

6.3 Qualitätssicherung

Um ohne einen vorgegebenen Prozess die Qualität der Produkte zu sichern, werden Abnahmekriterien definiert, welche die Software und die Softwareentwicklung erfüllen müssen. Abnahmekriterien können insbesondere in den folgenden Bereichen definiert werden:

- zu entwickelnde Produkte (Kode, Kode-Kommentare, Dokumentation, Modelle, etc.),
- Tests (definiert durch Testfälle z.B. zur Performance oder Konsistenz mit den Anforderungen , die vom fertigen Programm zu bestehen sind),

- Messpläne (bestimmte Messgrößen, wie z.B. Kopplung zwischen Modulen, die während der Entwicklung und am fertigen Produkt erhoben werden und vorgegebene Grenzwerte nicht überschreiten sollen).

Die konkreten Abnahmekriterien finden sich in einem gesonderten Dokument und werden ggf. während des Praktikums ergänzt.

6.4 Lizenzierung

Vor Beginn des Praktikums wurden einige Open Source Lizenzen auf ihre Tauglichkeit als Praktikumslizenz untersucht. Die Lizenz für ein solches Praktikum sollte folgende Eigenschaften haben:

- Echte Open Source Lizenz
  Die Lizenz sollte echten Open Source Code unterstützen, d.h. Code, der auf Dauer frei verfügbar ist und nicht nachträglich von irgendjemandem angeeignet werden kann. Modifikation sollen nur möglich sein, wenn diese explizit gemacht werden und die Modifikationen wieder unter Open Source Lizenz stehen.
- Integration in kommerzielle Software möglich
  Die Lizenz sollte es ermöglichen, den Open Source Code in kommerzielle Software zu integrieren. Lizenzen wie die Gnu GPL (General Public License) schreiben vor, dass Komponenten, die zusammen mit unter dieser Lizenz stehender Open Source Komponenten gebündelt werden, ebenfalls automatisch zu Open Source werden. Dies schließt die Integration des Code in kommerzielle Software Systeme wie die von maxess aus
- Stabile Lizenz
  Die Lizenz sollte relativ stabil sein und schon seit einiger Zeit (Monate, Jahre) existieren
- Zwang zur Offenlegung
  Die Lizenz sollte die Teilnehmer dazu zwingen, den Code und die Dokumentation offenzulegen. Kurze Lizenzen, wie die BSD-Lizenz legen nicht fest, dass der unter der Lizenz entwickelte Code veröffentlicht werden muss, d.h. der Code ist dann nicht zwangsläufig frei verfügbar
- Verständlichkeit der Lizenz
  Die Lizenz sollte nicht zu ausführlich und ohne eine Jurastudium verständlich sein
- Guter Ruf der Lizenz
  Die Lizenz sollte einen guten Ruf in Entwicklerkreisen besitzen und nicht unbedingt direkt mit einem Unternehmen assoziiert sein (wie z.B. die IBM Public License)

Es wurden im Vorfeld des Praktikum folgende Lizenzen untersucht:

- GNU GPL (General Public License, der Klassiker) aktuelle Version: 2, 1991
- GNU LGPL (Gnu Lesser Public License, Abgeschwächte GPL); aktuelle Version: 2.1, von 1999
- IBM Public License (Open Source Lizenz von IBM); aktuelle Version 1.0,
- Mozilla Public License (gebräuchliche Open Source Lizenz die vom Netscape Mozilla Team verwendet wird, aktuelle Version: 1.1, von 2000)
- QT Public License
- BSD Public License (Lizenz der Uni Berkeley); aktuelle Version: new version, 1999, ohne Werbeklausel. Keine Versionsnummer!!

Unter Beachtung der oben genannten Eigenschaften wurde sich für die „Mozilla Public License“ als Open Source Lizenz entschieden [Mozilla].. Alle Quellkode-Dateien wurden mit der Lizenz zu versehen. Eine Papierversion der Lizenz wurde allen Teilnehmern ausgehändigt. Die Mozilla Lizenz lässt echte Open Source Entwicklung zu, erlaubt jedoch die Integration der Open-Source Komponente in kommerzielle Software.

7 Der reale Kunde: maxess

7.1 Problemumfeld

Grundlage des Praktikums ist die E-Commerce-Lösung der maxess systemhaus GmbH. Diese Lösung bildet den Geschäftsprozess *Auftragsabwicklung* im Großhandelsbereich (Cash & Carry) ab [Max01]. Es handelt sich demnach um eine Business-To-Business-Lösung (B2B), da die Geschäftsbeziehungen zwischen einem Großhandelsunternehmen und Großkunden (z.B. Gaststättenbetriebe, Hotels, Tankstellen etc.), die wiederum Unternehmen sind, bestehen. Die Architektur der E-Commerce-Lösung besteht dabei aus 3 Subsystemen, die jeweils Informationen miteinander austauschen, um den Geschäftsprozess der Auftragsabwicklung zu realisieren: Warenwirtschaftssystem (Legacy-System), Internetshop und Mobiles Auftragserfassungssystem.

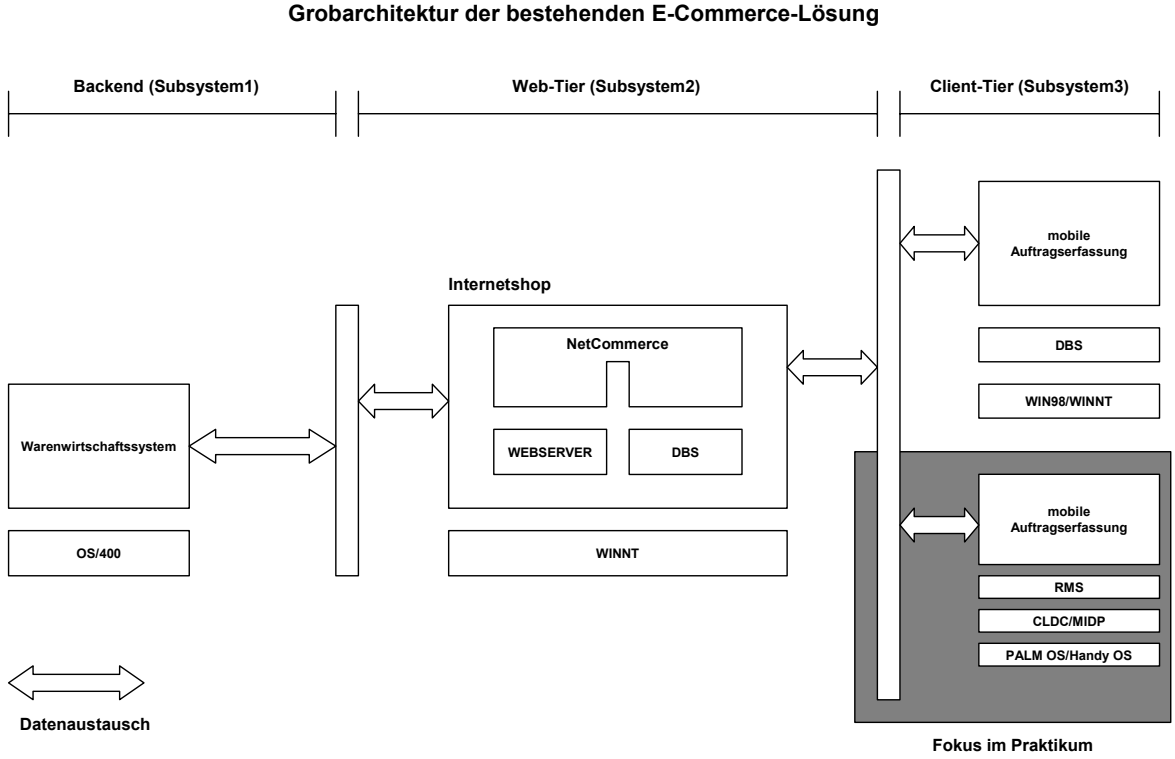

Abbildung 4: Grobarchitektur

Die Grobarchitektur dieses Mehrebenensystems ist in Abbildung1 angegeben. Prinzipiell besteht der Geschäftsprozess *Auftragsabwicklung* aus folgenden Einzelschritten:

- Auftragserfassung
- Kommissionierung
- Auslieferung
- Fakturierung

In Rahmen der Auftragserfassung wird der Warenbedarf des Kunden ermittelt. Daraus werden im Warenwirtschaftssystem konkrete Aufträge erzeugt, mit Hilfe derer Mitarbeiter des Handelsunternehmens die auszuliefernde Ware zusammenstellen. Diesen Vorgang der Warenzusammenstellung bezeichnet man auch als Kommissionierung. Anschließend wird die kommissionierte Ware an die Kunden ausgeliefert und fakturiert, also in Rechnung gestellt.

Das Praktikum setzt sich mit dem Bereich Auftragserfassung stärker auseinander. Im Rahmen der bestehenden E-Commerce-Lösung erfolgt die Auftragserfassung entweder im Subsystem2 oder im Subsystem3. Das heißt, der Großkunde kann seine Aufträge entweder selbständig als registrierter Benutzer eines Internetshops oder über den direkten Kontakt mit einem Außendienstmitarbeiter des Großhandelsunternehmens aufgeben. Die Außendienstmitarbeiter sind hierzu mit einem mobilen, auf Java basierten Auftragserfassungssystem ausgestattet, das sie in die Lage versetzt, Aufträge an Ort und Stelle zu erfassen und an den Internetshop bzw. das Legacy-System zu übertragen.

7.2 Problembeschreibung/Praktikumsziele

Die aktuelle Lösung für eine mobile Auftragserfassung ist für den Sektor Business-To-Business konzipiert worden. Ziel des Praktikums soll es nun sein, auch den Business-To-Consumer Bereich abzudecken. Dies ist mit der bestehenden Lösung wegen der verwendeten Hardware-Plattformen (Notebooks, Handheld-PCs mit Win98 bzw. WINNT, WIN2000) und der verwendeten Spracharchitektur (Java2 Standard Edition, JSE [Java]) nicht möglich.

Neben den bereits genannten Möglichkeiten der Auftragserfassung, soll es dem Geschäftskunden demnach zusätzlich ermöglicht werden, über ein entsprechendes Endgerät Aufträge eigenständig zu erfassen und an das Warenwirtschaftssystem zu übermitteln. Dabei ist an Endgeräte zu denken, die zum einen kostengünstig und platzsparend sind und sich zudem einfach bedienen lassen. Ein PC ist in diesem Anwendungsumfeld nicht geeignet, da viele Kunden entweder noch keinen PC besitzen oder zum Teil noch sehr große Berührungsängste vorhanden sind. Zudem stehen die Kosten für die Anschaffung in kei-

nem Verhältnis zu dem Nutzen und die Anforderung der Mobilität kann mit einem PC-System ebenfalls nicht erfüllt werden.

Im Fokus der zu konzipierenden Lösung steht somit Mobilität, Kundenakzeptanz und Kundenbindung (Customer Relationship).

Im Rahmen dieses Praktikums sollen die Grundbausteine eines mobilen Auftragserfassungssystem für sehr ressourcenbeschränkte Endgeräte für den Bereich Business-To-Consumer konzipiert und implementiert werden. Dabei sollen und können keine Komponenten der bestehenden mobilen Business-To-Business-Lösung verwendet werden. Eine sehr wichtige Anforderung in diesem Zusammenhang ist, dass eine ganze Klasse von Geräten unterstützt werden muss, was die Verwendung eines Sprachstandards zwingend erforderlich macht. De-facto Standard in diesem Bereich ist die Java 2 Micro Edition [Java], die über die Spezifikation spezieller Konfigurationen und Profile Plattformunabhängigkeit in bestimmten Geräteklassen erzielt.

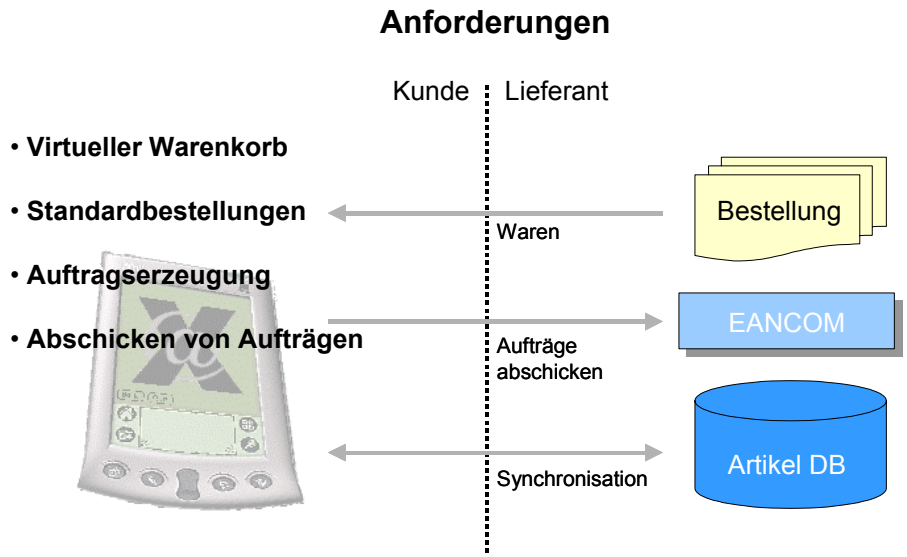

Abbildung 5: Die Anforderungen von maxess

7.3 Erwartete Funktionalität

Wie bereits erwähnt, sollte es dem Kunden in Zukunft möglich sein, seinen Warenbedarf eigenständig zu erfassen und elektronisch an das entsprechende Handelssystem zu übermitteln. Hierzu ist ihm ein geeignetes Endgerät zur Verfügung zu stellen, auf dem alle für die Auftragserfassung notwendigen Daten verfügbar sind. Für das Praktikum wurden Pocket PCs mit Palm OS und Handies

mit Javaunterstützung ausgewählt, da diese die oben geforderten Eigenschaften zum großen Teil erfüllen.

Zu den essentiellen Informationen, die die Abwicklung des Geschäftsprozesses Auftragserfassung ermöglichen sollen, gehören:

- Artikelinformationen,
- Preisinformationen.

Mit Hilfe dieser Informationen muss es möglich sein, einen Auftrag zu generieren, der durch ein nachgelagertes System verarbeitet werden kann.

Im Rahmen der Auftragserfassung sollte dem Kunden folgende Funktionalität zur Verfügung gestellt werden, die man grob in die Bereiche Produktselektion, Warenkorb und Auftragsverwaltung untergliedern kann.

Im Rahmen der Produktselektion wird der Kunde mit Hilfe geeigneter Suchmechanismen beim Auffinden von Produkten unterstützt. Diese können dann unter Angabe einer gewünschten Menge in einen virtuellen Warenkorb gelegt werden, aus dessen Inhalt sich der neue Auftrag zusammensetzt.

Nach der Speicherung des Warenkorbs, wird somit aus dessen Inhalt ein neuer, eindeutig identifizierbarer Auftrag, den der Kunde nach Bedarf innerhalb der Auftragsverwaltungskomponente elektronisch an das Handelssystem übermitteln kann, generiert. Als Austauschformat ist der EANCOM-Standard (siehe Glossar) zu verwenden.

Zudem erlaubt die Auftragsverwaltungskomponente das Bearbeiten und Wiederverwenden bereits erzeugter Aufträge, so dass man z.B. Teile eines alten Auftrages zur Generierung eines neuen Auftrages heranziehen kann.

Neben der Auftragserfassung muss auch die Möglichkeit der Datensynchronisation gegeben sein. Dies ist wichtig, da der Kunde immer aktuelle Daten auf seinem Gerät vorfinden muß, um eine reibungslose Auftragsabwicklung zu gewährleisten. Im Rahmen der Datensynchronisation (Replikation) werden über ein geeignetes Kommunikationsprotokoll und ein geeignetes Datenaustauschformat alle Artikelstammdaten- und Preisänderungen vom Handelssystem auf das Endgerät des Kunden transferiert.

Somit lässt sich die Aufgabenstellung innerhalb dieses Praktikums in zwei Teilbereiche untergliedern:

- Auftragserfassungsystem
- Replikationssystem.

In die Modellierungs- und die Designentscheidungen im Rahmen der Anwendungsentwicklung müssen vor allem die sehr eingeschränkt vorhandenen Res-

sourcen bei den verwendeten Zielplattformen einfließen. Dies betrifft insbesondere das GUI-Design sowie den Entwurf eines geeigneten Datenmodells zur Speicherung der für den Geschäftsprozess essentiellen Informationen. Es ist dabei zudem zu bedenken, dass auf den zu unterstützenden Geräten kein relationales DBS zur Verfügung steht, sondern nur ein sogenanntes RMS (Record Management System). Dieses stellt dem Anwendungsentwickler Records als Datenstruktur ohne spezielle Unterstützung durch ein spezielles Zugriffssystem zur Verfügung. Dies erscheint für die Menge der zu verarbeitenden Daten allerdings ausreichend, zudem keine komplexen Transaktionen anfallen und auch kein Mehrbenutzerbetrieb vorliegt. Dennoch sollten natürlich die ACID-Eigenschaften von Transaktionen ihre Berücksichtigung finden.

Hinsichtlich des GUI-Design, des Kontrollflusses der Applikation, des Datenmodells, der verwendeten Kommunikationsprotokolle und des Datenaustauschformats der Replikationskomponente sollen im Praktikum keine Einschränkungen gemacht werden. Hier wird auf die Kreativität der Praktikumsteilnehmer gesetzt, um aus den minimal vorhanden Ressourcen einen möglichst hohen Nutzen und Performanz zu erzielen. Als Rahmenbedingung gelten die noch weiter unten spezifizierten Anforderungen und die Randbedingungen, die sich durch die Anbindung an den Internetshop ergeben.

7.4 Systembeschreibung

Nachfolgend wird auf die einzelnen bereits erwähnten Subsysteme eingegangen, die für das Praktikum und das allgemeine Verständnis des Problemumfeldes von Bedeutung sind.

Dabei handelt es sich um die existierenden Teilsysteme der E-Commerce-Lösung der maxess systemhaus gmbh.

Warenwirtschaftssystem (Backend)

Im Warenwirtschaftssystem (WWS) wird der komplette Warenfluß des Handelsunternehmens abgebildet. Für unser Problemfeld bedeutet dies, daß dort die eigentliche Auftragsverarbeitung stattfindet. Die dazu notwendigen Informationen für die Erzeugung eines konkreten Auftrages werden innerhalb der E-Commerce-Lösung entweder vom Internetshop (Subsystem2) oder der mobilen Auftragserfassung (Subsystem3) über eine Datenaustauschbeziehung geliefert.

Weiterhin werden im WWS u.a. Artikelstammdaten, Kundenstammdaten und Preise erzeugt und gepflegt. Das WWS speist den Internetshop sowie die mobilen Auftragserfassungsysteme mit den notwendigen Stammdaten. Per Definition ist das WWS das führende System, so daß keinerlei Stammdatenänderungen in den nachfolgenden Subsystemen zugelassen werden. Einzig die erfassten Auftragsinformationen gelangen in einer Datenrückkopplung zurück ins WWS.

Internetshop (Web-Tier)

Der Internetshop dient für registrierte Kunden im wesentlichen als webbasierte GUI zur Auftragserfassung. Alle hierfür benötigten Stammdateninformationen werden als täglicher Snapshot über eine Schnittstelle vom WWS (Subsystem1) zur Verfügung gestellt und in einer relationalen Datenbank gespeichert. Als Austauschobjekte zwischen Internetshop und WWS werden Aufträge in einem standardisierten Dateiformat erzeugt (EANCOM).

Zusätzlich protokolliert auf der Shopseite ein Capture-Mechanismus alle Änderungen des Datenbestandes, die infolge des Datenabgleichs zwischen WWS und Internetshop aufgetreten sind. Diese Änderungen dienen wiederum zur Synchronisation der einzelnen lokalen mobilen Datenbanken der Aussendienstlösung (Client-Tier) mit der zentralen Shopdatenbank.

Mobile Auftragserfassung (Client-Tier)

Mit der bestehenden mobilen Auftragserfassung werden von den Außendienstmitarbeitern des Handelsunternehmens Aufträge direkt beim Kunden aufgenommen. Es handelt sich dabei um eine Java-basierte Applikation mit der im Offline-Betrieb gearbeitet wird und als eine Art "Tentakel" des Internetshops realisiert ist. D.h. einmal am Tag erfolgt eine Datensynchronisation mit dem Internetshop, so dass in der Folge kein weiterer Online-Betrieb notwendig ist. Auch hier kommt ein relationales Datenbanksystem zum Einsatz.

Eine Rückkopplung mit dem Shopsystem erfolgt ebenfalls durch auf Dateien basierende Austauschobjekte im EANCOM-Format. Die übermittelten Auftragsinformationen werden anschließend vom Shopsystem verarbeitet und an das WWS weitergeleitet, wo letztendlich konkrete Aufträge erzeugt werden. Durch diese Mehrebenenarchitektur wird die Komplexität des WWS vor dem mobilen Auftragserfassungssystem verborgen.

Im Rahmen des Praktikums soll eine neue Ausprägung einer mobilen Auftragserfassung für den Bereich Business-To-Consumer konzipiert werden, deren Anforderungen nachfolgend spezifiziert sind.Eine Grobarchitektur des zu entwickelnden Systems ist in der nachfolgenden Grafik veranschaulicht.

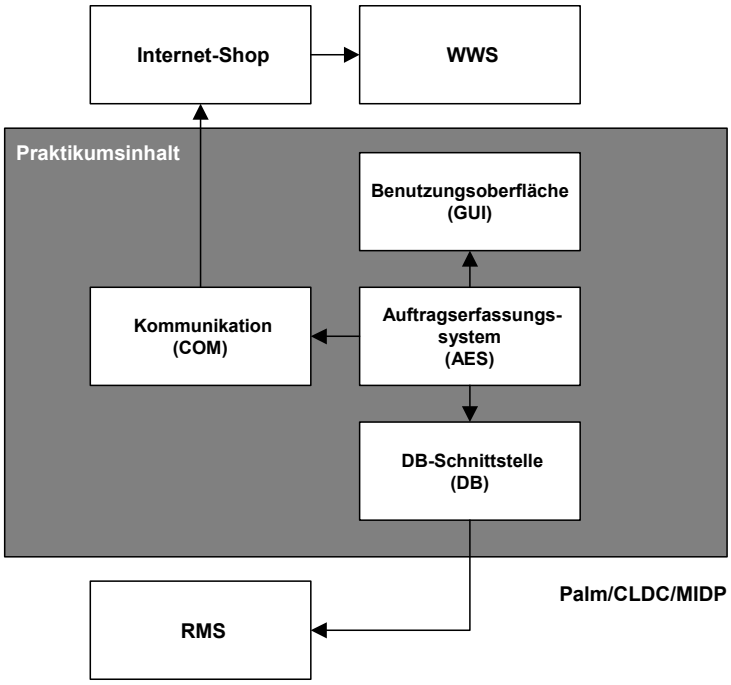

Abbildung 6: Grobarchitektur des Mshop-Systems

7.5 Abnahmekriterien

Die Abnahmekriterien der maxess systemhaus GmbH für die zu entwickelnde Software-Lösung im Rahmen des Praktikums lassen sich folgendermaßen untergliedern:

- Abzuliefernde Produkte (→ Grundlage zur Verifikation und Einbindung in den Entwicklungsprozess der maxess systemhaus gmbh),
- Abnahmetest in Form eines Akzeptanztests durch die maxess systemhaus gmbh.

Unabhängig davon sollte die Lösung nach den einschlägigen Software-Engineering-Prinzipien entwickelt werden.

7.5.1 Abzuliefernde Produkte

Folgende Produkte, gruppiert nach den einzelnen Phasen des Softwareentwicklungsprozesses, sind gewünscht:

Phase Anforderungsanalyse:

- UML USE-CASE Diagramme,

- textuelle Beschreibung der einzelnen USE CASES (vgl. Vorgaben im DO-IT-Prozess),
- Szenarien.

Nach der Erstellung dieser Produkte sollte eine Rückkopplung mit der maxess Systemhaus GmbH stattfinden. Dabei erfolgt ein Abgleich mit den spezifizierten Anforderungen der Problembeschreibung. Danach kann der Entwicklungsprozess völlig unabhängig weitergeführt werden.

Die Erzeugung dieser Produkte war auch der erste Meilenstein innerhalb dieses Projektes.

Phase Entwurf/Modellierung;

- UML Klassendiagramme,
- UML Sequenzdiagramme,
- UML Zustandsdiagramme,
- UML Aktivitätsdiagramme,
- Schemabeschreibung der Datenspeicherungskomponente (Abbildung persistenter Objekte).

Hinsichtlich der Modellierung gibt es von maxess die Vorgabe der Anwendung des MODEL-VIEW-CONTROLLER-Design Patterns.

Phase Implementierung:

- Hinsichtlich der Quellkodeerzeugung sollte sich an den Java-Kodierungsrichtlinien von Amby-Soft orientiert werden (→ Kode-Inspektion durch maxess).

Phase Qualitätssicherung:

- Zur Qualitätssicherung waren initial keine Kriterien vorgegeben.

Des weiteren sollten Angaben zur Skalierbarkeit der Applikation gemacht werden. Diese Angaben sollten sich aus durchzuführenden Lasttests ergeben und folgende Informationen liefern:

- maximal mögliche Anzahl zu speichernder Artikel- und Preisinformationen (in Abhängigkeit der Zielplattform und den dort zur Verfügung stehenden Ressourcen).
- Entwicklung der Antwortzeit für die implementierten Artikelsuchvarianten bei steigender Anzahl von Artikel- und Preisdatensätze (tabellarisch oder als graphisch).

7.5.2 Abnahmetest

Im Rahmen eines Abnahmetests der Software-Lösung wird von maxess ein Akzeptanztest auf der Grundlage der Use-Cases und Szenarien durchgeführt. Dies erfordert die Installation der produzierten Software auf den angedachten Zielgeräten durch die Teilnehmer des Praktikums. Es wird auch davon ausgegangen, dass entsprechende Testdaten bereitgestellt werden.

8 Ablauf des Praktikums

Die Durchführung des Praktikum wurde im Wesentlichen in vier Phasen unterteilt (Einarbeitung, Kommunikationsphase, Implementierung, Ergänzungen). Neben der obligatorischen Einarbeitungsphase stand die Kommunikation mit dem Kunden in der ersten Zeit des Praktikums im Vordergrund. Dies war nötig, um die Anforderungen des Kunden möglichst detailliert zu klären. Danach folgte die Entwicklung der einzelnen Komponenten im oben beschriebenen Open-Source Modus. Diese Kommunikationsphase lässt sich in Anforderungs- und Designphase trennen. Am Ende des Praktikums wurde eine Pufferzone vorgesehen, die soweit zeitlich möglich mit innovativen Ergänzungen gefüllt werden sollte. Im Folgenden werden die Phasen des Praktikums incl. der Vorbereitungsphase grob umrissen.

8.1 Vorbereitungsphase

In der Vorbereitungsphase wurden die Anforderungen an das zu entwickelnde System von den Betreuern aus allen beteiligten Organisationen entwickelt. Diese Phase war notwendig, um die Anforderungen an ein Universitätspraktikum zu erfüllen, das der Ausbildung von Studenten dient.

Die Vorbereitungsphase des Praktikums hatte die Ziele, die Erwartungshaltungen gegenüber dem Praktikum abzuklären, Anforderungen an ein zu entwickelndes System zu erstellen, Rahmenbedingungen für eine von maxess brauchbare Lösung festzulegen und die nötige Umgebung für die Durchführung des Praktikums bereitzustellen, sprich Hardware- und Software-Ausstattung.

Die Vorbereitung dauerte von Anfang August bis Mitte Oktober. In den Sitzungen trafen sich die Verantwortlichen der beteiligten Gruppen. Auf maxess Seite umfasst dies Jürgen Koch, Frank Przybilla, Holger Jungknecht, Klemens Möhlenkamp, Christian Sommer und Kristian Trottenberg. (Letzterer stand auch während des Praktikums als ständiger Ansprechpartner zur Verfügung.) Die Universitätsseite vertraten Marcus Ciolkowski, Jens Heidrich (beide AG Software Engineering) und Wolfgang Mahnke (AG Datenbanken und Informationssysteme). Die Koordination und Beratung beider Gruppen wurde durch das Fraunhofer Institut für Experimentelles Software Engineering (IESE), vertreten durch Isabel John, vorgenommen.

Zunächst war es wichtig, das Problemumfeld abzugrenzen und die von maxess vorgeschlagene Aufgabenstellung für das Praktikum zu konkretisieren. Dies umfasste folgende Gesichtspunkte:

- Das Ziel des zu erstellenden Systems muss klar definiert werden: Für das Praktikum sollte eine B2C-Lösung im Bereich M-Commerce erstellt werden. Dabei steht die Portabilität der zu erstellenden Lösung im Vordergrund. Das erzeugte System soll mit geringfügigen Anpassungen auf einer großen Menge von mobilen Zielplattformen lauffähig sein. Die Wahl fiel hierbei auf die Java 2 Micro Edition. Weitere Aspekte umfassten (a) die generelle Machbarkeit einer derartigen B2C Lösung auf ressourcenschwachen Endgeräten sowie (b) die Oberflächengestaltung und Benutzer-Kommunikation.
- Die Umgebung, in welche das zu erstellende System eingebettet werden soll, musste festgelegt werden. In unserem Falle umfasste dies den Anschluss an den von maxess betriebenen Internetshop und die Zielplattform für das System, sprich Geräte, welche die Java 2 Micro Edition unterstützen (genauer gesagt CLDC/MIDP kompatibel sind).
- Standard-Entwicklungswerkzeuge mussten festgelegt werden, um eine einheitliche Entwicklung zu garantieren, deren resultierenden Produkte von maxess weiterverwendet werden können. Hier fiel die Entscheidung auf die integrierte Software-Entwicklungsumgebung Together (Version 5.5 von TogetherSoft), welche eine Modellierung des Systems in UML erlaubt und die erzeugten Diagramme mit dem korrespondierenden Java-Code in Einklang hält.

Ferner müssen explizite Anforderungen an das zu entwickelnde System gestellt werden. Hierbei wurde auf der bestehenden Außendienstlösung von maxess aufgebaut. Sie diente als Vorlage, um Anforderungen an eine B2C-Lösung abzuleiten. Um die Aufgabenstellung des Praktikums nicht zu umfassend werden zu lassen, beschränken sich die Anforderungen im Wesentlichen auf die Client-Seite des Systems, d.h., die Bereitstellung der Daten. Die konkrete Anbindung des Systems an den maxess Internetshop wird zunächst zurückgestellt. Zusätzlich wurde ein inkrementelles Vorgehen vereinbart; das heißt, der Schwerpunkt sollte zunächst darauf liegen, ein ablauffähiges System zu erstellen. Wünschenswerte oder nicht essentielle Funktionalität sollte eventuell auf ein späteres Inkrement (d.h., ein späteres Praktikum) verschoben werden. Dies bedeutete auch, dass der automatische Datenabgleich zwischen Client und Server zunächst nicht behandelt werden sollte.

Ein weiterer wichtiger Aspekt stellt der Entwicklungsprozess dar. Hierbei haben wir uns dafür entschieden, die Software im Wesentlichen nach oben beschriebenen Open-Source Modus zu entwickeln, also (abgesehen vom inkrementellen Vorgehen) keinen konkreten Prozess vorzugeben. Dies erfordert allerdings eine genaue Spezifikation der Abnahmekriterien für im OS Modus erzeugte Produkte des Praktikums. Ihre Festlegung war ebenfalls Bestandteil der Vorbereitungs-

phase. Um eine einheitliche Beschreibung/Dokumentation des Gesamtsystems zu erhalten ist es ferner von Bedeutung die Bestandteile derselben exakt festzulegen. Hierbei wurde auf eine bereits in früheren SE-Praktika erprobte Beschreibungsweise (z.B. von Use-Cases) zurückgegriffen.

Da der Einsatz von Open-Source zur Komponentenentwicklung eine klar definierte Schnittstelle zwischen Kunde und Entwicklungsteams und eine zugrunde liegende Systemarchitektur voraussetzt, musste der eigentlichen Entwicklung im OS Modus eine Kommunikationsphase vorausgestellt werden, die sicherstellt, dass alle Schnittstellen wohl definiert sind und eine einheitliche Architektur des Gesamtsystems vorliegt bevor in den OS Modus übergegangen wird.

Zusammengefasst lieferte die Vorbesprechung also folgende Ergebnisse:

- Die Erwartungshaltungen der beteiligten Parteien wurde abgeklärt.
- Die Praktikumsziele wurden definiert und die Aufgabenstellung wurde inhaltlich eingrenzt (Schwerpunkt auf der Client-Seite des Systems).
- Anforderungen an das zu erstellende System wurden definiert.
- Die zu verwendende Entwicklungsumgebung wurde festgelegt.
- Abnahmekriterien für das zu erstellende System wurden spezifiziert.
- Die grundlegende Vorgehensweise im Praktikum (siehe 9.1) wurde festgelegt.

8.2 Start des Praktikums

Das Praktikum startete mit Beginn des Wintersemesters (d.h. Ende Oktober) 2001/2002. Zur Anfangsbesprechung wurden die Studenten mit der Aufgabenstellung vertraut gemacht, sowie das Vorgehensmodell und der Zeitplan des Praktikums erläutert. Anschließend hatten die Studenten eine Woche Einarbeitungszeit, um sich mit den Anforderungen und (zumindest teilweise) mit den verwendeten Werkzeugen vertraut zu machen. Danach startete die Anforderungsphase.

Für die Einarbeitungsphase wurde 1 Woche vorgesehen. Die Schwerpunkte lagen hierbei zunächst auf dem Verstehen der Problemstellung in den Bereichen M- und E-Commerce der Firma maxess Systemhaus GmbH. In diesem Zusammenhang wurde auch ein Besuch des „Edeka C+C“ Marktes vorgenommen, um einen Eindruck des Problemumfeldes zu gewinnen. Anschließend erfolgte das vertraut machen mit den in der Vorbereitungsphase des Praktikums festgelegten Entwicklungswerkzeugen und das Festlegen von Gruppen, die für einzelne Abschnitte des Systems zuständig sind. In unserem Falle wurden vier Gruppen gebildet, die der Grobarchitektur des Systems im Anforderungsdokument entsprachen: (1) Zwei Teilnehmer beschäftigten sich mit der Benutzungs-

oberfläche, (2) drei Teilnehmer mit dem Auftragserfassungssystem, (3) zwei Teilnehmer mit der Datenbankkomponente und schließlich (4) drei Teilnehmer mit der Kommunikationskomponente. Unter den Studenten wurden ferner management-orientierte Rollen verteilt: Projektmanager (Alexander Hilliger von Thile), Qualitätssicherer (Ove Armbrust) und Konfigurationsmanager (Torsten Lenhart).

8.3 Anforderungsphase

Wozu eine Anforderungsphase, wenn schon vom Kunden Anforderungen vorgegeben waren? Nun, die vom Kunden vorgegebenen Anforderungen waren in textueller Form gegeben. Damit hatten die Studenten in der Anforderungsphase die Aufgabe, Unvollständigkeiten und Inkonsistenzen in den Anforderungen zu entdecken und zu beheben. Dazu wurden Use-Cases (siehe Abb.) und Szenarien sowie ein GUI-Prototyp erstellt. Auch wenn dies einigen Studenten sehr mühsam erschien, konnten doch einige schwerwiegende Missverständnisse rechtzeitig entdeckt werden. Eine weitere Aufgabe der Anforderungsphase war, die Anforderungen so weit zu verfeinern, dass in der folgenden Entwurfsphase ein weitgehend unabhängiges Arbeiten der einzelnen Teams möglich war. Dazu wurden bereits Teile des Entwurfs erstellt, etwa ein grobes Klassendiagramm, das die Schnittstellen zwischen den einzelnen Systemteilen repräsentierte.

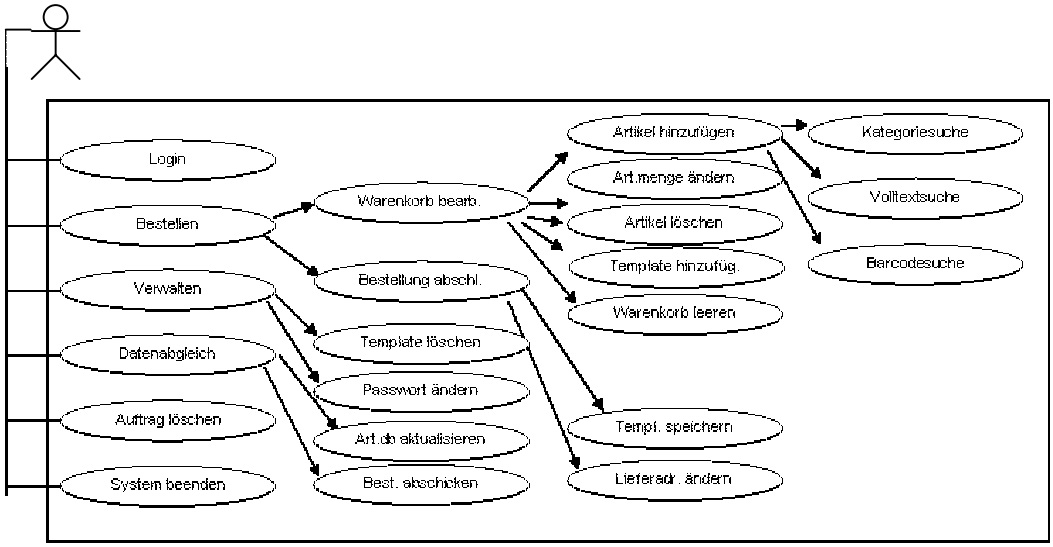

Abbildung 7: Die Use Cases des mshop

Die Kommunikation mit der maxess systemhaus gmbh umfasste nach Plan 4 Wochen des Praktikums. Im Vordergrund standen das Abklären der von maxess geforderten Anforderungen an das zu entwickelnde System. Davon ausgehend

sollte mit dem Kunden eine grundlegende, komponenten-orientierte Architektur des Systems entwickelt werden. Basierend auf dieser Architektur erfolgt dann das Ableiten von Komponentenanforderungen (z.B. Schnittstellen zwischen einzelnen Komponenten der Architektur).

Während der Durchführung ergaben sich einige Unklarheiten im Anforderungsdokument und in den benutzten domänenspezifischen Begrifflichkeiten. Es erwies sich deshalb als sinnvoll dies Kommunikationsphase auf 6 Wochen auszudehnen und sie mit der Entwicklungsphase zu überschneiden.

Am Ende dieser Phase erfolgte eine Präsentation und Diskussion der entwickelten Architektur. Daneben waren die organisatorischen Rollen in dieser Phase aktiv. Während dieser Phase wurden folgende Risiken identifiziert:

- Ursprünglich wollten die Studenten kein Konfigurationsmanagement einsetzen (Keine Einführung von CVS sondern manuelle, filebasierte Konfiguration)
- Warenkorb/Auftragstrennung (Es ist unklar, ob die Realisierung von Warenkörben / Aufträgen sich für Erweiterungen des Systems als sinnvoll herausstellt)
- Falls der Kontrollfluss bei der GUI liegt, könnte dies zu GUI-orientierter Entwicklung führen. Zusätzlich: Schnittstellen entsprechen nicht der Spezifikation (siehe Problembeschreibung), da dann GUI und Kommunikation gemeinsame Schnittstellen haben
- Kontrollfluss bei der GUI führt zu Überlastung des GUI Team
- Kontrollfluss bei der AES führt zu AES-orientierter Entwicklung und Überlastung des AES Team
- Das Team, das für die DB-Komponente zuständig war, entschied sich zu einer (reduzierten) SQL-Implementierung für RMS. Vorteil: Geringe Kopplung, native SQL-Datenbank kann mit geringem Aufwand angekoppelt werden. Risiko: Diese Vorgehen kann zu Überlastung des DB-Teams führen
- Bei allen stark Implementierungs-getriebenen Prozessen besteht das Risiko, dass die Inspektion zugunsten Testen vernachlässigt wird.
- Verlust der Übersicht durch Fehlen projektbegleitender Dokumentation (und am Projektende fehlende Zeit für Dokumentation)
- Nach Integration der Komponenten zu einem System tauchen neue Fehler (im Zusammenspiel) auf
- Die Implementierung nimmt zuviel Zeit in Anspruch

Das Eintreten eines der Risiken könnte verschiedene Folgen haben. Dies wäre z.B. Chaos in der Entwicklung; Probleme lauffähige Versionen zu erzeugen, unausgeglichenes System, Systemteile können nicht rechzeitig fertiggestellt wer-

den usw. Gegenmaßnahmen wären beispielsweise Re-Modellierung, Redesign und Reimplementierung von Systemteilen, die Aufträge betreffen.

Des weiteren könnten Fehler zu spät bemerkt werden, die u.U. große Änderungen nach sich ziehen, z.B. Rework von Komponenten und Schnittstellen.

8.4 Designphase

In der Designphase, die mit drei Wochen angesetzt war, sollten die Teams die Schnittstellen zwischen den Komponenten definieren sowie ihre Komponenten durchgehend entwerfen. Als Mittel wurden vorrangig Klassen- und Sequenzdiagramme eingesetzt. Teile der Designphase wurden bereits in der Analysephase durchgeführt; außerdem wurde parallel mit der Implementierung begonnen.

Für die Entwicklung im Open-Source Modus wurden 8 Wochen vorgesehen. In dieser Zeit erfolgte die Realisierung der in der Architektur spezifizierten Komponenten. Dies umfasst folgenden immer wiederkehrenden Entwicklungszyklus:

Zunächst werden die Komponenten nach dem in einer Gruppe gewählten, internen Prozess entwickelt. Dieser Prozess ist dem Kunden nicht bekannt, sondern liegt in der Verantwortung jeder einzelnen Gruppe. Wenn eine Komponente einen „vorzeigbaren“ Stand erreicht hat, erfolgt die Vorstellung der von maxess geforderten Deliverables. „Vorzeigbarkeit“ liegt dabei im Ermessensspielraum der einzelnen Gruppen. Wenn die Gruppen der Ansicht sind, das Gesamtsystem kann im momentanen Entwicklungsstrand präsentiert werden oder einzelne Komponenten davon können sinnvoll demonstriert werden, so werden diese dem Kunden vorgeführt. Der Kunde entscheidet dann über die Annahme oder Ablehnung der Deliverables. Dies kann prinzipiell dazu führen, dass die Abnahmekriterien verändert oder angepasst werden müssen.

Der Zyklus endet mit einer vom Kunden akzeptierten Lösung des Gesamtsystems. Den Abschluss stellt wiederum eine Präsentation des Systems dar.

8.5 Implementierungsphase

Die Implementierungsphase war mit vier Wochen angesetzt (einschließlich Testen und Integration). Dieser Zeitplan ist realistisch, wenn der Entwurf klar genug durchgeführt werden kann. Falls dies nicht möglich sein sollte, stand den Studenten die Pufferzeit als Ausgleich zur Verfügung. In der Tat traten schwerwiegende Probleme bei der Realisierung auf, unter anderem die schwache Performanz der Java-Maschine auf den mobilen Geräten. Dies hatte zur Folge, dass der Entwurf erheblich überarbeitet werde musste, um diesen Nachteil wenigs-

tens teilweise aufzufangen. Außerdem traten Kommunikationsprobleme zwischen den Teams auf sowie Missverständnisse mit dem Kunden, die in Mehrarbeit resultierten. Zusätzlich kamen studiumsbedingte Termine (wie etwa Klausuren) hinzu, was die Arbeiten am System verzögerte.

8.6 Testen und Abnahme

Der Testprozess wurde den Studenten selbst überlassen. Insgesamt kann man sagen, dass diejenigen Teile des Systems nicht getestet wurden, die schwer zu testen waren (wie etwa die GUI). Ansonsten wurde, je nach Kritikalität der Komponente, sehr ausgiebig getestet. Zum Beispiel wurde die SQL-Implementierung der Datenbank mit mehreren Methoden getestet, unter anderem mit automatisch generierten statistischen Testfällen.Zusätzlich wurden Systemtests durchgeführt, indem das System auf der Zielplattform installiert und mehrere Szenarien durchgespielt wurden. Abnahmekriterien von Maxess basierten im wesentlichen auf den im Anforderungsdokument festgelegten Szenarien.

Gewissermaßen als Pufferzone wurde eine Phase für innovative Ergänzungen vorgesehen. Diese Pufferzone wurde deshalb eingeplant, weil die Systementwicklung mit großen Risiken behaftet war, etwa der Unsicherheit, ob sich die Anforderungen mit der gewählten Technologie wirklich einfach umsetzen lassen. Für den Fall also, dass bei der Entwicklung keine größeren Probleme auftauchen sollten, wurde für die Pufferzeit das Hinzufügen von *nice-to-have* Verhalten vorgesehen, d.h., die Suche nach neuer, innovativer Funktionalität und Realisierungsmöglichkeiten (z.B. Einsatz eines Barcode-Scanners).

Da bei der Systementwicklung jedoch erhebliche Probleme auftraten (z.B. die schwache Performance der virtuellen Java-Maschine auf den Zielplattformen Palm und Handys) war es nötig, diese Phase zu verkürzen und Teile davon in die Entwicklungsphase zu verlagern und mit ihr zu verknüpfen.

9 Ergebnisse des Praktikums

9.1 Der M-Shop

Das Hauptergebnis des Praktikums war ein funktionierender Mobiler Shop um Bestellungen beim maxess-Internetshop über mobile Endgeräte abzuwickeln. Der Code für den M-Shop wurde unter Open Source Lizenz gestellt und veröffentlicht, er kann unter http://revlon.informatik.uni-kl.de/ heruntergeladen werden. In der Abschlusspräsentation wurde der Shop sowohl mit den Emulatoren als auch mit einem realen Endgerät, einem Palm demonstriert. Die folgende Abbildung zeigt die Eingangsseite des Mshops auf dem Palm und die Login-Screen des Shop auf dem Emulator (wegen besserer Lesbarkeit).

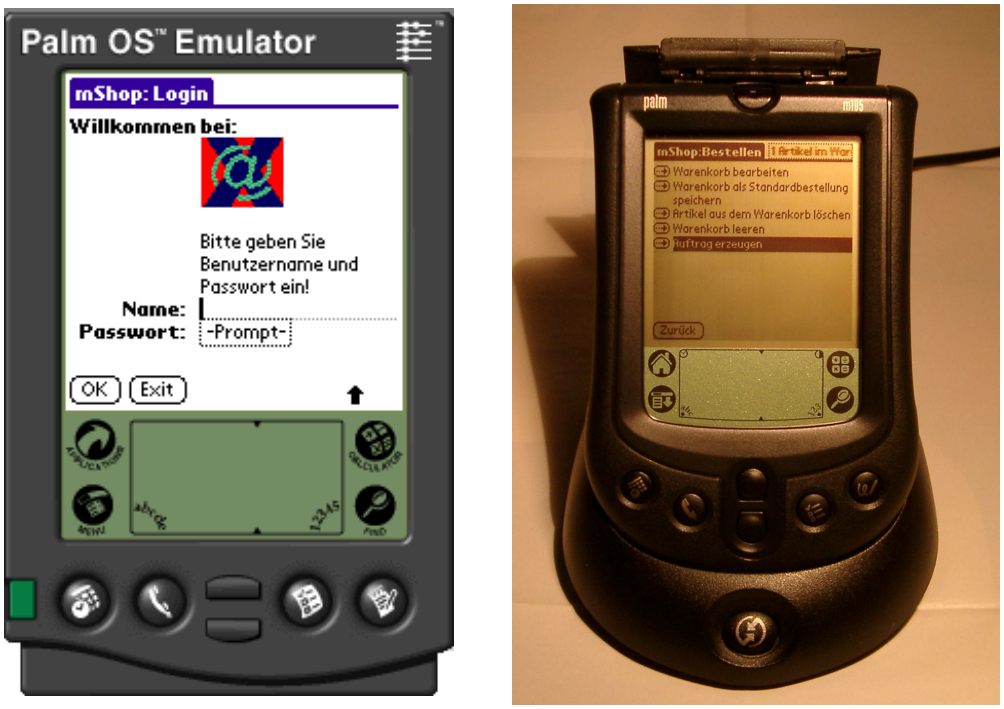

Abbildung 8: Die Login Screen mit dem Palm- Emulator und die Warenkorbübersicht des mshop im Echteinsatz

9.2 Lessons Learnt

Die folgenden Punkte waren aus Betreuersicht während des Praktikums auffällig, waren Besonderheiten bedingt aus der neuen Praktikumsstruktur oder sind allgemeine Lessons Learnt zur Praktikumsbetreuung:

- Durch das neue Themengebiet, die Abstimmung mit dem Kunden und die Thematik der Open Source Entwicklung erforderte dieses Praktikum im Vorfeld einen sehr hohen Arbeitsaufwand. Dieser Arbeitsaufwand sollte bei weiteren Praktika mit eingeplant werden
- Die Entwicklung im Open Source Modus macht es für die Praktikumsbetreuer schwierig, bei Problemen einzugreifen. Bei der Open Source Entwicklung sollten ja Entscheidungen allein von den Studentischen Teilnehmern gefällt werden, nur, was tun wenn die Betreuer dies als eine Fehlentscheidung sehen? (Wie z.B. die Entscheidung, auf Versionsverwaltung zu verzichten). In diesen Fällen offen Druck auszuüben oder die Entscheidung der Praktikumsteilnehmer rückgängig zu machen, hätte den Open Source Gedanken ad absurdum geführt. Es wurde dann zu verschiedenen Mitteln gegriffen:
  1. Aufklärung der Studenten über die Folgen von Entscheidungen und nennen von Erfahrungen aus anderen Projekten (a la „ Ohne Versionsverwaltung ist das bisher immer schief gegangen“)
  2. Explizitmachen der Risiken von Entscheidungen der Studenten durch eine Tabelle von Projektrisiken. Durch diese Tabelle konnten die Bedenken der Betreuer geäußert werden, ohne zu direkten Einfluss zu nehmen.
  3. Geduld. Oft kamen die Studenten durch eigene Erfahrung (z.B. Missverständnisse mit dem Kunden maxess, Schwierigkeiten bei Schreibrechten ohne eine Funktionierende Versionsverwaltung) noch auf den „rechten Weg“ zurück, ohne dass die Betreuer eingreifen mussten.

  Dieses Vorgehen ist für die Betreuer schwieriger als bei einem „normalen“ Praktikum, wo man direkt Einfluss nehmen kann und erfordert Geduld und gute Nerven. Es wurden jedoch durch die drei beschriebenen Maßnahmen alles einigermaßen unter Kontrolle behalten
- Durch den realen Kunden maxess konnte im Praktikum ein realistisches Projektumfeld geschaffen werden. Die Studenten konnten ein sehr gutes Bild davon gewinnen, wie Projekte in der Industrie ablaufen und wo bei Industrieprojekten typische Stolpersteine liegen (mehrdeutige Anforderungen; Entwickler implementieren Anforderungen anders als sie im Anforderungsdokument stehen; Entwicklungsumgebung ist teilweise nicht optimal auf das zu entwickelnde System abgestimmt; Zeitdruck und verschobene Deadlines; Projektmeetings mit Kunden...)
- Durch die neue Fragestellung im Praktikum (M-Commerce Bestellsystem) war vorher nicht klar, ob und wie die Anforderungen alle realisiert werden können. Während der Durchführung des Praktikums musste permanent ab-

gestimmt werden wie und ob bestimmte Anforderungen zu realisieren sind. Dies führte z.B. zu einer Streichung der Volltextsuche aus Performancegründen. Die Klärung dieser Unsicherheiten war sowohl für den Kunden maxess als auch für die Studenten hilfreich und zielführend. Im Unterschied zu einem normalen Praktikum wurde dadurch den Studenten das Gefühl gegeben, etwas neues zu machen.

- Die Dokumentation und das Testen gehen bei solch vielschichtigen Anforderungen und bei der Open Source Entwicklung etwas unter. Hier wäre zu wünschen, in einem weiteren Praktikum dieser Art Mittel und Wege zu finden, Dokumentation und Testen schon während der Entwicklung zu integrieren oder zu instrumentalisieren.
- Das Konfigurationsmanagement-Konzept wurde zwischen beginn des Praktikums und Entwicklungsphase mehrfach verworfen und geändert. Dies lag zum Teil an technischen Schwierigkeiten (Zugang über Firewall), zum Teil an Entscheidungen der Studenten. Bei einem weiteren Praktikum wäre es wünschenswert, das jetzt gewählte Konfigurationsmanagement (Zugang zum Praktikumsserver über CVS bzw WebCVS/WinCVS) beizubehalten und nicht mehr zur Diskussion zu stellen.
- Jedes System ist nur so gut wie seine Anforderungsbeschreibung, gegen schlechte Requirements hilft auch Open Source Entwicklung nichts.
- Die Studenten wurden durch das Praktikum und seine vielschichtigen und neuen Anforderungen stark gefordert, hatten aber sichtlich Spaß an der Aufgabe (es wurden z.B. Meetings ohne die Betreuer, nur mit dem Kunden abgehalten, um die Use-cases und Anforderungen zu klären). Der Open Source Modus und der Kundenkontakt scheinen also die Kreativität und das Potential der Studenten gut zu unterstützen und zu fördern.
- Das Praktikum hatte einen sehr guten Projektmanager aus den Reihen der Studenten, der die Meetings (sowohl intern als auch mit maxess) sehr gut organisierte und seine Entwicklertruppe immer gut im Griff hatte. Dies hat sicher sehr stark zu einem gelingen des Praktikums beigetragen. Mit einem Projektmanagement mit weniger Überblick wäre die Entwicklung wahrscheinlich zu sehr aus dem Ruder gelaufen
- Es wurde auch während der Durchführungszeit des Praktikums sehr viel Zeit mit der Durchführung und Vorbereitung von Meetings und Präsentationen verbracht (sehr viel mehr als in normalen Praktika). Dies wurde von den Studenten so nicht erwartet, ist aber durchaus typisch für den Industrieprojektalltag.
- Software Engineering lässt sich auch in Open Source Projekten durchführen, es erfordert jedoch Disziplin, nicht in „wildes Hacken“ zu verfallen. Open Source unterstützt per se Software Engineering zwar nicht, Open Source Entwicklung kann aber auch mit Software Engineering Prinzipien durchgeführt werden (und wird besser dadurch).

- Das Zusammenwirken von Software-Engineering, Open Source, M-Commerce und Industriekunde als vielschichtige Inhalte aus verschiedenen Bereichen hat das Praktikum zu etwas Besonderem gemacht. Durch Kooperationswille und Flexibilität aller Beteiligten ist das Praktikum für alle bereiche als Erfolg zu werten.

9.3 Arbeitsaufwände

Die zur Messung der Arbeitsaufwände notwendigen Daten wurden leider nicht systematisch erfasst. Allerdings dürfte der Aufwand etwas über dem ansonsten üblichen Aufwand für ein traditionelles Praktikum gelegen haben, zumal das System erst nach den Semesterferien fertig gestellt wurde. Aber die Bereitschaft der Studenten, sich überhaupt so weit zu engagieren, spricht wohl dafür, dass der höhere Arbeitsaufwand durch die Attraktivität des Praktikums gerechtfertigt ist. Dies bestätigten uns auch Aussagen der Studenten.

9.4 Erfahrungen der Studierenden

Um die Zufriedenheit der Studierenden mit dem Praktikum an sich und mit dem im Praktikum eingeführten innovativen Aspekten wie Open-Source oder M-Commerce zu erfragen wurde nach Beendigung des Praktikums eine anonyme Umfrage mit 15 Fragen unter den teilnehmenden Studenten durchgeführt. Von den zehn Teilnehmern füllten acht den Fragebogen aus, die Rücklaufquote war also 80%. Im folgenden sollen die Ergebnisse dieser Umfrage zusammengefasst werden. Es handelt sich dabei um keine empirische Auswertung, sondern eher um ein Meinungsbild quer über das Praktikum. Für die Antworten wurde keine Skala vorgegeben sondern nur soweit möglich sinnvolle Antworten vorgegeben (z.B. „Hat Dir das Praktikum Spaß gemacht ? (ja, nein, teils/teils)“) Besonders wichtig war dabei die Frage zu den innovativen Verbesserungen des M-Shop, da das Sammeln von Verbesserungsvorschlägen auch Teil der Praktikumsanforderungen war. Die von den Studenten genannten Erweiterungen /Verbesserungsvorschläge werden im ersten Abschnitt aufgezählt. Die ausgewerteten Antworten der Studenten auf die weiteren Fragen und ausgewählte Zitate folgen im zweiten Abschnitt. Abschließend werden die Beurteilungen zusammengefasst und bewertet.

9.4.1 Verbesserungen des M-Shop

Ein Ziel des Praktikums war es, innovative Ergänzungen für den existierenden M-Shop von den Studenten zu erfragen. Diese Ergänzungen, die teilweise bei der Abschlusspräsentation schon anklangen wurden in der Umfrage erfragt. Es gab folgende Verbesserungs-/ Ergänzungsvorschläge von den Studenten:

- Einführung von Sicherheit bei der Benutzung und Übertragung (verschlüsselte Übertragung, Nutzerdaten usw.)
- Im Moment ist keine Multiuserverwaltung vorgesehen, aber wenn man das Config-Objekt in den Kunden nimmt und in den Orders/Templates kleine Änderungen vornimmt, dann klappt das auch relativ problemlos.
- Vielleicht sollte man den Warenkorb persistent machen, so dass Unterbrechungen im Bestellvorgang einfacher sind.
- Integration einer schnelleren Virtual Machine. Das Gesamtsystem ist soweit ok, manchmal etwas träge, da würde eine schnellere VM helfen.
- Obwohl die Reaktionszeit des Systems sehr langsam ist, gibt es keine Möglichkeit Vorgänge abzubrechen.
- Man sollte die Möglichkeit haben, aus dem M-Shop auf dem Palm in andere Programme zu wechseln, da dies für ein Palm-Programm absolut üblich ist
- Erweiterung der DB:
  - UPDATE-Statements (für eine spätere, verbesserte Capture-Komponente mit selektiven Updates)
  - Verbesserung der Indexnutzung bei Joins
  - B*-Baum für Strings (wäre zwar eigentlich straightforward, aber sehr fehleranfällig und nur für die equals-Suche brauchbar)
  - Like-Index (unklar, wie der Index zu realisieren ist)
- kommerzielle Datenbank verwenden
- andere Programmiersprache
- Vielleicht wäre es besser, wenn der Benutzer mit dem Palm durch sein Lager geht, einen Auftrag erstellt, aber dann den Auftrag über seinen PC und das Internet abschickt. Wenn später die Geräte leistungsfähiger geworden sind, kann dies alles auf dem Palm geschehen.
- noch weitere Anforderungen einbauen
- versuchen einigen Screens ein "Schöneres" Aussehen zu verleihen
- Klasse „ObjectList“ aus dem AES rauswerfen
- bessere Lösung für die abbrechbare Suche

9.4.2 Allgemeine Zufriedenheit

Im folgenden soll ein Überblick über die Beurteilungen des Praktikums durch die Studenten gegeben werden. Dazu werden die Bewertungen der Studenten

zusammengefasst, Einzelaussagen zu den verschiedenen Fragen werden teilweise zitiert:

- Hat Dir das Praktikum Spaß gemacht ? (ja, nein, teils/teils)
  6 von 8 Studenten antworteten hier mit „Ja“ oder „ja, sehr“, ein Student mit „Insgesamt hat es Spaß gemacht“ ein Student antwortete mit „teils/teils“
- Würdest Du das Praktikum noch mal machen bzw. an Kommilitonen weiterempfehlen? (ja uneingeschränkt, vielleicht, nein)
  Alle Studenten antworteten mit „Ja“ oder „Ja, uneingeschränkt“
- Wie schätzt Du den Aufwand den Du für die Durchführung des Praktikums benötigt hast ein? (zuviel, zuwenig, genau richtig)
  Von allen Studierenden wurde der Aufwand als genau richtig oder akzeptabel eingeschätzt.
  “Es war schon viel Arbeit, aber es hat sich ja auch gelohnt, insofern also genau richtig. Man hätte mehr Features einbauen können, also zu wenig. Es hat viel Zeit gekostet, also zu viel. Für ein Praktikum war es denke ich normal“
  “Dadurch dass wir bis April letztendlich Zeit hatten, war es in Ordnung, ansonsten wäre es zuviel gewesen, aber so konnte man auch noch viel machen ohne Vorlesungen zu haben.“
  “Als Projektmanager war es schon sehr viel Arbeit – diese hat aber auch viel Spaß gemacht also: genau richtig“
- Wie beurteilst du die einzelnen Phasen des Praktikums? (lief gut, soweit ok, frustrierend)
- Einarbeitungsphase
  Drei von sieben Studenten beurteilten die Einarbeitungsphase sinngemäß mit „Lief gut“, zwei Studenten mit „ok“, zwei mit „frustrierend“
  “Enorme Einarbeitung bedingt durch neue Technologien und vorhandene Systeme (wie den Webshop). Damit war aber zu rechnen und es ist nicht anders möglich“
  “Lief gut, wenn man mal den Vergleich zur Einarbeitung in StP 2.6 (vorheriges Praktikum) zieht...“
- Anforderungsphase
  Drei von acht Studenten beurteilten die Einarbeitungsphase sinngemäß mit „Lief gut“, ein Student mit „ok“, vier mit „frustrierend“.
  “ Diese Diskussionen über immer das Gleiche, was eigentlich schon abgeklärt worden war, waren ziemlich nervig, vor allem da sie zum Teil nicht zu einem Ende gekommen sind.“
  “ Frustrierend, die Maxess-Leute haben z.T. unsere IMO guten Ansätze einfach nicht erkannt“
- Designphase
  Die Designphase wurde von zwei Studenten mit „gut“, von 5 Studenten mit

„soweit ok“ und von einem mit „frustrierend“ bewertet
“ War soweit ok. Nur das Erstellen der Use-Cases zum reinen Selbstzweck ist für mich immer noch unverständlich . Wer hat die eigentlich nachher verwendet?“
“ soweit okay, wurde aber von vielen Teams nicht ernst genommen und häufig mit den Aussagen: "Kann man sich vorher eh nicht überlegen" oder "Ändern wir nachher eh alles wieder" quittiert. Damit wurde in dieser für das Projekt wichtigen Phase sinnvolle Leistung durch die Teams verschenkt“
“Oh, sehr spaßig, weil unser Team eigentlich keine Schnittstelle nach außen hatte :-) Auf der anderen Seite aber auch recht frustrierend, weil die ersten Einschränkungen des Palm-Java sichtbar wurden und die Stimmung teilweise ziemlich down war. Es gab teilweise auch recht gereizte Reaktionen, weil innerhalb von sehr kurzer Zeit praktisch ein komplettes System entworfen werden sollte und wir die Möglichkeiten der Plattform noch gar nicht richtig einschätzen konnten.“

- Zwischenpräsentation
  Die Zwischenpräsentation wurde von 7 der 8 Studenten mit „Lief gut“ bewertet, für einen Studenten war die Zwischenpräsentation „ok“.
  “ Lief gut, es kam das erste Mal richtiges Feedback und wir bekamen auch mal den Eindruck, dass sich wirklich jemand für unsere Arbeit interessiert hat -> Motivationsschub“

- Implementierungsphase
  Drei Studenten beurteilten die Implementierungsphase mit „Lief gut“ , drei mit „ok“, einer mit „frustrierend“
  “ Am Anfang recht schleppend, aber als das System nach und nach an Funktionalität gewonnen hat, war das schon ein Super-Erlebnis, vor allem, als es am Ende dann wirklich funktioniert hat.“
  “ Lief gut, ich konnte alles so realisieren, wie ich es wollte und konnte mal etwas implementieren, was ich eh schon immer mal machen wollte.“
  “ frustrierend, da bereits in der Designphase das "Hacken" begonnen hatte bzw. abzusehen war.“

- Abschlusspräsentation/Prüfung
  Die Abschlusspräsentation wurde von sieben Studenten mit „Lief gut“ beurteilt, von einem Studenten gab es keine Wertung
  “ Mit der Note würde ich sagen: Rundum gelungen! ;-)„
  “.. denke ich, dass es sinnvoll ist, sein System einem Kunden zupräsentieren, man lernt dabei, sich richtig zu verkaufen. “

- Wie zufrieden warst Du mit der Durchführung des Praktikums als Open Source Praktikum? (Open Source war gut, hab nicht viel von den Open Source Aspekten mitgekriegt, Open Source hat eher gehindert)
  Zwei der Studenten fanden die Open Source Aspekte im Praktikum gut, sechs Studenten fanden die Open Source Aspekte eher sekundär oder haben nicht viel davon mitgekriegt.
  “ Open Source? Habe ich nun nicht wirklich was davon mitbekommen, da

waren die Rollen wohl doch sehr genau verteilt und das hat sich nun nicht wirklich vom Team-Work bei anderen Praktika unterschieden..."
"Leider hat man vom Open Source nicht viel mitbekommen, es hätte vielleicht von den Betreuern mehr dafür sensibilisiert werden müssen."
" Open Source kommt eigentlich erst zum Tragen, wenn jetzt mehrere andere Teams am Projekt weiterarbeiten würden bzw. wenn einige Komponenten von fremden Teams entwickelt worden werden. Ansonsten ist der Unterschied nur die Lizenz unter welcher das Produkt am Ende geliefert wird"

- Wie zufrieden warst Du mit der Betreuung durch AGSE, AGDBIS und IESE ?(zufrieden, unzufrieden, teils/teils)
  Alle acht Studenten waren mit der Betreuung zufrieden
  " Zufrieden. Selten so einen Betreuer-Overkill erlebt. ;-) Normalerweise hat man bei einem Praktikum mit 20 und mehr Leuten vielleicht zwei Assistenten und ein oder zwei Hiwis."
  " Ich war zufrieden, habe auch nicht so viel bemerkt - das war vielleicht gut? Aber unsere Komponente hat auch eher wenig technische Probleme gehabt, da haben wir IESE und AGSE nicht so dringend gebraucht. Sehr gut war unser Technik-Hiwi, der echt ständig da rumgebastelt hat, um uns ein halbwegs reibungsloses Arbeiten zu gewährleisten! Insgesamt war immer jemand da, der sich gekümmert hat, und den man fragen konnte, aber keiner hat rumgenervt ("und, wie weit bist du")
  "zufrieden, negativ war nur die Art der Präsentation der Projektrisiken, denn sie hat in etwa wie eine Absicherung der Betreuer bei einem Misslingen des Projekts gewirkt. Hier hätte man die Risiken etwas besser präsentieren können und mehr diskutieren müssen, damit diese Problemfälle vermieden werden können. Glücklicherweise sind diese Probleme dann doch nicht aufgetreten."
- Hättest Du dir teilweise mehr Betreuung/Eingriffe durch die Betreuer gewünscht? (ja mehr, war genau richtig so, weniger Betreuung)
  Sieben der 8 Studenten fanden die Betreuung „genau richtig so", ein Student hätte sich teilweise mehr Eingriffe gewünscht.
  " Mehr Eingriffe hätten nur der eigenen Kreativität geschadet."
  " Die Betreuer hätten vielleicht etwas mehr darauf achten sollen, dass alle Leute(Studenten) ihren Job machen. Der Projektmanager hat da als Student einfach zu wenig Handhabe."
- Wie zufrieden warst Du mit dem „Kunden" maxess? (zufrieden, unzufrieden, mittel)
- Betreuung durch den Kunden
  Fünf Studenten waren mit der Betreuung durch maxess zufrieden, ein Student war unzufrieden, ein Student fand die Betreuung „na ja".
  "zufrieden, hohe Bereitschaft zu Investitionen falls nötig, daraus resultierte unter anderem das Gefühl, das man etwas wirklich sinnvolles entwickelt."

"Die Büffets waren lecker :-) Die Maxess-interne Kommunikation ist sicher verbesserungsfähig. Aber dass immer einer dabei war, war gut"

- Kundenanforderungen
  Fünf Studenten waren mit den Kundenanforderungen zufrieden, ein Student war unzufrieden, ein Student fand die Anforderungen mittel.
  "Im Großen und Ganzen sehr gut und präzise"
  "Waren ok, auch am Anfang ziemlich unklar, mehrdeutig, also vermutlich ziemlich realistisch. Es war gut, dass sie was weggenommen haben, sonst wäre es wohl ziemlich heftig geworden"
- Kommunikation mit dem Kunden
  Fünf Studenten fanden die Kommunikation mit dem Kunden gut, drei Studenten mittel.
  " Gut, war aber auch nicht so oft notwendig"
  " Sicher besser als in der Realität, da der Kunde sehr oft da war"
- Wie zufrieden warst du mit der Entwicklungsumgebung? (zufrieden, unzufrieden, mittel)
- PCs im Praktikumsraum
  Vier Studenten waren mit den PCs zufrieden, vier Unzufrieden.
  "Gut, für Together allerdings zu langsam (was aber Together Schuld ist)"
  "Viel zu lahm, insbesondere wenn man bedenkt, dass es P4s sind"
- Palms
  Sechs Studenten waren mit dem Palm zufrieden, einer unzufrieden.
  " Hm, waren zu wenige, da hätte wenigstens für jede Gruppe einer da sein sollen!"
  " sehr zufrieden, denn das ganze hätte in einem "normalen" Praktikum wohl nur auf dem Emulator stattgefunden (Motivationsschub)"
- Together
  Ein Student war mit Together zufrieden, drei Studenten waren unzufrieden und vier Studenten hatten das eine oder andere an Together auszusetzen, waren also nur teils/teils zufrieden.
  " Na ja, es ist halt Java-basiert, also superlangsam. Wenn das in C geschrieben wäre dann könnte man es vermutlich richtig benutzen. Und wenn wir es von vorne bis hinten benutzt hätten, dann hätte das auch mit der konsistenten Doku geklappt, aber als Editor ist es halt leider unbrauchbar"
  "Sobald es erst mal gestartet war, war es etwas Schönes um damit den Entwurf zu machen, aber zum implementieren, absolut ungeeignet."
- J2ME
  Vier Studenten war mit der J2MicroEdition zufrieden, zwei Studenten waren unzufrieden und zwei Studenten waren teils/teils zufrieden.
  " Uh, sehr primitiv, aber wenigstens gibt es so was. So sind wir halbwegs plattformunabhängig, aber lahm."
  "Die Möglichkeiten sind wirklich sehr beschränkt, vor allem hat es einige

sehr "interessante" Effekte, die man erst mal verstehen muss.“

- Emulatoren
  Die Emulatoren wurden von sechs Studenten mit „gut“ bewertet, ein Student war unzufrieden , ein Student war zum teil zufrieden.
  “ Meistens besser, als die Originale :-)“
  “ Gut. Bisschen dumm nur das die nicht wirklich realistische Arbeitsspeicherbedingungen vorweisen (und das auch nirgendwo steht) und dadurch ein gravierendes Problem erst spät bemerkt wurde“
  “Die Handies des J2MEWTK sind etwas buggy,...“
- CVS
  Mit CVS waren sechs Studenten zufrieden, zwei der Studenten waren mit CVS nur zum Teil zufrieden.
  “ Außer, dass es gegen Ende erlaubte, zu alte Versionen wieder einzuchecken, schöne Sache, vor allem die Möglichkeit sich über Änderungen informieren zu lassen.“
  “ Anfangs habe ich es abgelehnt, aber für derartige Projekte ist es sehr sinnvoll.“
- sonstiges
  Unter dem Punkt Sonstiges gab es Anmerkungen zur Betreuung und zu weiterer Software:
  “ sehr gute Betreuung seitens des Systemhiwi“
  “ Sehr großer Freiraum bei der Realisierung + motivierte Teilnehmer -> innovatives Ergebnis“
  “ Software: Bitte einen gescheiten Editor installieren (z.B. UltraEdit), Together ist zum flüssigen Arbeiten zu lahm.“
- Was lief gut im Praktikum?
  Hier wurden von den Studenten sehr viele Aspekte genannt. Zweimal wurde die Betreuung durch den Systemhiwi genannt, zweimal die gute Teamarbeit.
  “ Gute Betreuung. Professionelle Hilfe des Systemadministrators. Das Ziel eines lauffähigen Systems wurde erreicht.“
  “ Motivation, Spaß, Zusammenhalt, Team-Work“
- Was lief schlecht im Praktikum?
  Die Aspekte die hier von den Studenten genannt wurden sind ebenfalls breit gefächert. Zweimal wurde bemängelt, dass die Designphase nicht ernst genug genommen wurde, zweimal wurde bemängelt, dass das Praktikum bis ins nächste Semester angedauert hat, dreimal wurde die fehlende Mitarbeit einer der vier Gruppen als negativer Aspekt genannt..
  “ Designphase wurde zu "locker" genommen, festgelegte Grundsätze (z.B. nicht Code andere Teams ändern) wurden ignoriert, Praktikum wurde während der Semesterferien verschleppt“
  “Eigentlich nichts. Alles was störte, wurde doch bestmöglich verbessert.“

"Am Anfang ein bisschen lang mit Use-Cases etc. aufgehalten. Am Ende etwas knapp geworden und ins nächste Semester angedauert."

- Was sind die drei wichtigsten Dinge die Du in diesem Praktikum gelernt hast?
  Auch hier gab es sehr viele verschiedene Aspekte. Dreimal wurde (sinngemäß) „Teamwork & Teamkommunikation" genannt, zweimal wurden jeweils folgende Aspekte genannt: Die Umsetzung von Software Engineering Methoden in die Praxis; die Durchführung eines realen Projekts mit einem realen Kunden; die Umsetzung von Anforderungen in ein Design.
  "Ein gutes (und vollständiges!) Design am Anfang erspart Kummer und Sorgen und Hacken"
  "Das absolut wichtigste das ich gelernt habe, war die Bedeutung der Software Engineering Methoden, die in den Vorlesungen zu theoretisch sind. Hier hat man zumindest im Ansatz das Gefühl für die Ziele von SE verstehen gelernt."
  "ohne Zusammenarbeit geht nichts"
  "für manche Sachen, muss man sich einfach mal treffen, denn plötzlich geschieht in 15 Minuten, an denen man gemeinsam dran arbeitet mehr, als in 2 Wochen, in denen man darüber redet, dass es getan werden muss. Dies betrifft vor allem gruppenübergreifende Dinge"
  "Frei nach SE 1, Kapitel 2: planning, organizing, enacting"
  "Eine gute Präsentation ist die halbe Kundenzufriedenheit!"
- Hast Du Verbesserungsvorschläge zur Durchführung des Praktikums oder zu einzelnen Phasen?
  Hier wurden folgende Verbesserungsvorschläge gemacht:
  "Man könnte vielleicht die Schnittstellen am Ende des Designs festlegen (also die, die die Gruppen entwickelt haben, zementieren), um alle dazu zu bewegen, sich auch dazu mal Gedanken zu machen und "unbefugte" Code-Änderungen technisch unterbinden"
  "Wichtigste Verbesserung ist die Konzentration auf eine sinnvolldurchgeführte Designphase und ein später korrekt umgesetztes Design, Designänderungen müssen explizit begründet werden"
  "Der Projektmanager muss zumindest ein bisschen Weisungsbefugnis haben, da er sonst nicht genug Druck auf "seine" Teams ausüben kann. Möglicherweise muss er auch etwas aus der Entwicklungsarbeit herausgehalten werden, damit er sich mehr auf die Koordinationstätigkeit konzentrieren kann."
  "Anforderungsphase straffen/beschleunigen, dafür mehr Zeit für Features und Implementierung."
  "ausführlichere Designphase Besser wäre gewesen 4 Leute für den Design zu nehmen und andere Teams zu bilden, die sich mit der neuen Technologie vertraut machen, so dass schneller ein vollständiger Prototyp entstehen kann".
  "Ohne "Prüfung" wäre es auch gegangen, die Arbeit wurde eh nicht wegen

der Prüfung hineingesteckt."
"Auch in der Vorlesungsfreien Zeit sind regelmäßige Treffen vielleicht nicht so verkehrt, da man dann mal wieder alle Leute zusammen hat und offene Dinge abklären kann, geht direkt oft besser"
"Mehr Freiheit in der Anforderungsphase. Eigene Ideen sollten gefördert werden. Maxess-Anforderung zu restriktiv."
"Insgesamt war alles in Ordnung und man sollte eigentlich nichts ändern. Man kann halt nicht immer nur Sachen machen die einem Spaß machen."

9.4.3 Zusammenfassung der Beurteilungen und Bewertung

Vergleicht man die einzelnen Phasen des Praktikums, so fällt auf, dass die Zwischen- und Abschlusspräsentationen eine sehr positive Bewertung erhalten haben. Dies ist möglicherweise zum Teil auf die angenehme Stimmung bei den Präsentationen und die guten Noten zurückzuführen. Durch die Präsentationen wurden auch Meilensteine gesetzt, bei denen die Studenten zeigen konnten, was sie geschafft hatten. Dass dies immer gut lief und von allen Parteien sehr positiv aufgenommen wurde, trugen wahrscheinlich auch zur positiven Bewertung der Präsentationen bei. Am negativsten wurde die Anforderungsphase bewertet, jedoch gab es auch hier nur 4 von 8 negativen Beurteilungen.

Mit der Betreuung durch die internen Betreuer und durch den Kunden waren die Studenten im großen und ganzen zufrieden. Das Konzept der gemeinsamen Betreuung durch Kunde und interne Betreuer wurde angenommen und hat sich im Praktikum bewährt. Die Open Source Aspekte kamen in den Augen der Studenten wohl etwas zu kurz. Man sollte also bei weiteren Praktika darauf achten, dass die Open Source Aspekte der Entwicklung und vor allem des Entwicklungsprozess klar kommuniziert und auch während des Praktikums gelebt werden

Bezüglich der Entwicklungsumgebung waren die Studenten mit dem Palm, mit den Emulatoren und mit CVS sehr zufrieden, eher unzufrieden waren Sie mit Together und den PCs. CVS hat sich also trotz anfänglicher Probleme als sinnvolles Tool für dieses Praktikum herausgestellt. Für ein weiteres Praktikum sollte vielleicht nach einer Alternative zu Together gesucht werden.

Es wurden von den Studenten ausführliche Verbesserungsvorschläge gemacht, die sicher in die Planung eines weiteren Praktikums eingehen sollten. Die Studenten haben (wahrscheinlich in höherem Maße als bei einem „normalen" Praktikum") Teamwork und Kommunikation miteinander und mit einem Kunden gelernt, haben gelernt wie Kundenprojekte ablaufen und haben gesehen, wie sich Software Engineering Prinzipien von selbst, ohne vorgegebenen Prozess in die Praxis umsetzen lassen.

10 Glossar

ACID-Eigenschaften:

Damit sind die generell für Transaktionen geforderten Eigenschaften gemeint:

A=Atomicity: Zustandsänderungen werden nach dem Alles-oder-Nichts-Prinzip eingebracht.

C=Consistency: Nach Beendigung einer Transaktion müssen die zuvor definierten Integritätsbedingungen erfüllt sein.

I=Isolation: Diese Eigenschaft bezieht sich auf Synchronisationsmechanismen, die den Eindruck eines Einbenutzer-Betriebes vermitteln (→ Schutz vor dem Einfluss konkurrierender Transaktionen).

Durability: Zustandsänderung müssen dauerhaft eingebracht werden.

Artikel:

Ein Artikel ist die konkrete Ausprägung eines Produktes, den der Kunde bestellen kann. Ein Artikel kann eindeutig über eine Artikelnummer identifiziert werden.

Artikelstamm:

Unter Artikelstamm versteht man die Gesamtheit der Informationen über die im Sortiment befindlichen Artikel. Dazu zählen vorwiegend Informationen wie Artikelnummer, Artikeltext etc., die im Laufe der Zeit nur wenig Änderungen unterworfen sind. Davon abzugrenzen sind demnach Preisinformationen.

Apply-Komponente:

Hierbei handelt es sich um einen Prozess, der die von einem Capture-Prozess bereitgestellten Daten in ein Datenhaltungssystem einbringt.

Capture-Komponente:

Eine Capture-Komponente protokolliert permanent oder zu festgelegten Zeitpunkten Änderungen an Datenobjekten.

CLDC:

CLDC (Connected Limited Device Configuration) ist eine Konfiguration der J2ME. Das Herz der CLDC ist Suns K virtual machine (KVM). CLDC ist geeignet für Geräte mit 16/32-bit RISC/CISC Mikroprozessoren und benötigt nur 160 KB Speicher, wovon 128 KB zum Ablegen der Virtuellen Maschine und der Bibliotheken verwendet werden.

EANCOM-Standard:

Hierbei handelt es sich um ein international standardisiertes Nachrichtenaustauschformat speziell für den Bereich Handel. Im Praktikumsumfeld ist insbesondere EANCOM-Orders, der den Bestelldatenaustausch standardisiert, relevant.

EPOC:

EPOC ist ein Betriebssystem für PDAs, das bspw. bei den PDAs von Psion, Nokia und Ericsson eingesetzt wird. EPOC wird von Symbian weiterentwickelt, ein Firmenkonsortium aus Psion, Ericsson, Nokia und Motorola. Es steht damit in Konkurrenz zu Windows CE.

Handheld:

Ein „handheld computer" ist ein Computer, der bequem in der Tasche aufbewahrt werden kann und zur Benutzung n der Hand gehalten wird. Handheld ist ein Synonym für PDA.

J2ME:

Die J2ME (Java 2 Micro Edition) ist speziell für den Einsatz auf Geräten mit limitiertem Speicher und limitierter Prozessorleistung zugeschnitten. Um möglichst viele Geräte abzudecken besitzt sie eine dreigeteilte Struktur. Auf Basis einer *Virtual Machine* bauen verschiedene *Konfigurationen* und *Profile* auf, welche wiederum verschiedene Geräteklassen abdecken.

Kategorie:

In einer Kategorie werden Artikel mit gleichen oder zumindest ähnlichen Eigenschaften zusammengefasst. Das Kategorienkonzept erlaubt es, somit die Menge der verschiedenen Artikel zu strukturieren. Da eine Kategorie wieder aus mehreren Unterkategorien bestehen kann, ergibt sich eine Kategorienhierarchie, die vor allem für die Navigation innerhalb der Artikelsuche verwendet werden kann.

Konfiguration:

Eine Konfiguration bietet bei der J2ME eine Menge von Bibliotheken und eine Virtuelle Maschine für eine bestimmte Kategorie von mobilen Geräten.

KVM:

Die KVM (K virtual machine) ist eine Virtuelle Maschine für Java, die speziell für Geräte mit stark eingeschränkten Ressourcen (bspw. PDAs) entwickelt wurde.

M-Commerce:

M-Commercce (Mobile Commerce) bezieht sich auf den Handel über das Internet, der von einem mobilen Client ausgeführt wird. Dies kann ein beispielweise ein PDA oder ein Handy sein.

MIDP:

MIDP (Mobile Information Device Profile) ist eine Menge von Java APIs, die zusammen mit dem CLDC eine komplette J2ME Laufzeitumgebung bereitstellen, die auf mobile Geräte wie Handys und PDAs ausgelegt ist. Das MIDP adressiert Belange wie Benutzerschnittstelle, persistente Speicherung und Netzwerkverbindungen.

Palm:

Palm ist der Markennahme eines populären PDAs. Ursprünglich wurde der Palm PalmPilot genannt. Er wurde bereits 1996 eingeführt. Der Palm hat sein eigenes, proprietäres Betriebssystem, das Palm OS.

Palm OS:

Das Palm OS ist ein Betriebssystem für die PalmPilots. Das Palm OS wurde speziell entworfen, um in einen Palm zu passen und die spezielle Oberfläche des Palms zu unterstützen.

PDA:

PDA (personal digital assistant) ist ein anderer Ausdruck für ein Handheld. Damit ist ein kleines, mobiles Gerät gemeint, das Dienste wie Berechnungen, Informationsspeicherung und Suchen für den persönlichen Einsatz anbietet. Die meisten PDAs haben eine kleine Tastatur oder einen mit Schrifterkennung ausgestatteten Touchscreen. Als Betriebssystem kommen auf einem PDA in der Regel EPOC, Windows CE oder ausschließlich für Palms das Palm OS zum Einsatz.

Preise:

Jedem Artikel im Internetshop können n verschiedenen Preise zugeordnet werden. Für das Praktikum ist allerdings nur der Verkaufspreis relevant, als der Preis, den der Kunde für den entsprechenden Artikel zu bezahlen hat. Daneben gibt es noch andere Preisvarianten, auf die hier nicht weiter eingegangen werden soll.

Profile:

Ein Profile ist in der J2ME eine Spezifikation von einer Menge von Java APIs für eine bestimmte Kategorie von Geräten (zum Beispiel PDAs, Handys etc.) zusammen mit der Spezifikation einer J2ME Konfiguration.

Produkt:

Ein Produkt ist eine Art Repräsentant einer bestimmten Ware im Internetshop. D.h. ein Produkt kann nicht bestellt werden, sondern nur dessen Ausprägungen, nämlich die Artikel. Somit gruppiert ein Produkt einen oder mehrerer Artikel, die sich im wesentlichen in der Verpackungseinheit (z.B. 1 Stück, 3 Stück etc.) unterscheiden.

Replikation:

Replikation beschreibt ganz allgemein einen Prozess, in dem der Datenbestand einer Quellsystems mit einem oder mehreren Zielsystem synchronisiert wird. Subprozesse innerhalb der Replikation sind Capture auf dem Quellsystem und Apply auf dem Zielsystem.

Warenkorb:

Unter Warenkorb verstehen wir die visuelle Repräsentation eines noch nicht abgeschlossen Auftrages. In einen solchen Warenkorb, können selektierte Artikel eingefügt und auch wieder gelöscht werden. Nach der Speicherung, wird der Warenkorb geleert und aus den darin enthaltenen Informationen ein neuer Auftrag generiert. Der Warenkorb ist somit nichts weiter als ein.

Windows CE:

Windows CE basiert auf Microsofts Windows, wurde aber speziell für mobile Geräte entworfen. Obwohl Microsoft offiziell das „CE“ nicht erläutert, wurde berichtet, dass es für „Consumer Electronics“ steht. Obwohl Windows CE von Grund auf neu entwickelt wurde, wurden laut Microsoft die Vorteile von Microsofts Windows Schnittstellen und die Konzepte der Architektur übernommen. Windows CE steht in Konkurrenz zu EPOC.

11 Literatur

[Kne00] Antje von Knethen, Jürgen Münch. Entwicklung eingebetteter Software mit UML. Der Do-it-Prozess V 1.0. SFB 501 Bericht 05/00, Fachbereich Informatik, Universität Kaiserslautern, 2000.

[Knu01] J. Knudsen. Wireless Java: Developing with Java 2, Micro Edition. ISBN 1-893115-50-X, Springer-Verlag, New York, 2001.

[Mozilla] Mozilla Public License. http://www.mozilla.org/MPL/MPL-1.1.html

[OS-O'Reilly] O'Reilly. Open Source, kurz und gut. http://www.oreilly.de/german/freebooks/os_tb/os_tb_1.htm

[Ray-en] E. Raymond. The Cathedral and the Bazaar. http://www.tuxedo.org/~esr/writings/cathedral-bazaar/

[BLRV1995] A. Bröckers, C.M. Lott, D. Rombach, M. Verlage. MVP-L Lanuage Report Version 2. SFB-Bericht 265/95, Universität Kaiserslautern 1995.

[Jal1997] P. Jalote. An Integrated Approach to Software Engineering. 2. Auflage, Springer-Verlag, 1997.

[Java] http://java.sun.com

[Max00] Handbuch der mobilen Auftragserfassung; Maxess, interne Dokumentation

Dokumenten Information

Titel:	MShop – Das Open Source Praktikum
Datum:	01. Juli 2002
Report:	IESE-035.02/D
Status:	Final
Klassifikation:	Public