Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

Software Engineering moderner Anwendungen

Jürgen Rückert · Barbara Paech

Universität Heidelberg, Fakultät für Mathematik und Informatik, AG Software Engineering
Im Neuenheimer Feld 326, 69120 Heidelberg
{rueckert | paech}@informatik.uni-heidelberg.de

Zusammenfassung

An der Universität Heidelberg wurde die Veranstaltung »Software Engineering moderner Anwendungen: komponentenbasierte, serviceorientierte oder mobile Systeme« erstmals im Wintersemester 2005/2006 angeboten. Die innovative Idee zur Lehre war dabei die Entwicklung eines verteilten Software-Systems in vier Architekturen – mit Hilfe von vier verschiedenen Technologien. Einerseits konnten die Studierenden die Vor- und Nachteile der Architekturen durch Bewertung von nicht funktionalen Anforderungen diskutieren. Andererseits konnten sie praktische Erfahrung mit neuen Technologien sammeln, die für die Entwicklung von modernen Geschäftsanwendungen nützlich sind.

In diesem Beitrag stellen wir das Konzept und dessen Umsetzung, das begleitende Anwendungsbeispiel, die verwendeten Technologien und Werkzeuge sowie abschließend unsere Erfahrungen vor.

1 Einleitung

Die Vorlesung und Übung SWE 2B »Software Engineering moderner Anwendungen: komponentenbasierte, serviceorientierte oder mobile Systeme« wurde an der Universität Heidelberg erstmals im Wintersemester 2005/2006 angeboten. Studierende der anwendungsorientierten Informatik mit Abschluss Bachelor nehmen im 4. Fachsemester daran teil, Studierende mit Abschluss Master im 1. oder 2. Fachsemester. Die Wahlpflichtveranstaltung SWE 2B umfasst 6 Semesterwochenstunden (SWS), davon 3 SWS Vorlesung und 3 SWS Übungen. SWE 2B erfordert die Kenntnisse aus SWE 1 [PBR+05] [Häf05].

Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

Ziel der Veranstaltung ist einerseits der Wissensaufbau über Architektursichten und -muster sowie die Bewertung von funktionalen und nicht funktionalen Anforderungen (NFR) zur Auswahl und Anpassung geeigneter Architekturen. Andererseits sollen die ausgewählten Architekturen umgesetzt werden, um praktische Kenntnisse über komponentenbasierte und serviceorientierte Systeme aufzubauen.

Kernidee unseres Vorgehens ist die Realisierung einer verteilten Anwendung mit Hilfe von vier Architekturen. Die Anwendung wird einmalig funktional spezifiziert. Die Architektur wird durch die jeweils eingesetzte(n) Technologie(n) bestimmt und durch die zu unterstützenden NFRs. Die Anwendung wird für AnwenderInnen zugänglich durch einen Web Client, der weitestgehend konstant gehalten wird, d.h., nur dessen Dialogkern, eine Komponente, die den Web Client mit einer Server-Komponente oder einem Web Service verbindet, muss an die jeweilige Technologie angepasst werden.

Schwerpunkte in den praktischen Übungen bilden Requirements Engineering, Entwurf (inklusive Architekturentwurf), Implementierung und Qualitätssicherung (Systemtest und Rationales).

Auf die Ausbildung von Soft Skills wird dadurch Wert gelegt, dass die Studierenden im Team arbeiten und die verschiedenen Ergebnisse präsentieren. Der überraschende Tausch von Ergebnissen zwischen den Teams, inklusive der anschließenden Weiterentwicklung von fremdem Quellcode, hat das Bewusstsein zur Entwicklung von qualitativ hochwertiger Software geschärft.

In Abschnitt 2 präsentieren wir Vorlesung und Übung im Detail: Die funktionalen Anforderungen werden in 2.1 vorgestellt, die vier verschiedenen Anwendungen kategorisieren wir in 2.2, den Ablauf der Vorlesung und Übung erklären wir in 2.3, nicht funktionale Anforderungen und Architekturen stellen wir in 2.4 vor, und die verwendeten Technologien und Werkzeuge nennen wir in 2.5. In Abschnitt 3 beschreiben wir unsere Erfahrungen. Abschließend geben wir in Abschnitt 4 eine Zusammenfassung und einen Ausblick für die Wiederholung der Veranstaltung im kommenden Wintersemester 2006/2007.

2 Vorlesung und Übung

2.1 Funktionale Anforderungen

Als Anwendungsbeispiel haben wir versucht, ein anschauliches Beispiel zu wählen: eine Anwendung »Auctions«, mit der AnwenderInnen an Auktionen mehrerer Auktionshäuser mitbieten können. AnwenderInnen sollen in der Rolle »AuktionsteilnehmerIn« in der Lage sein, einen Auktionsgegenstand zu ersteigern. Aktoren, Aufgaben, Nutzungsfälle und Systemfunktionen sind in Abbildung 1 dargestellt. Die beiden Nutzungsfälle »Kontaktiere ein Auktionshaus« und

»Nimm an einer Auktion teil« sollen durch eine Benutzungsschnittstelle grafisch zugänglich werden. Die Aufgabe »Verwalte Auktionen« muss dagegen nicht durch eine grafische Benutzungsschnittstelle unterstützt werden.

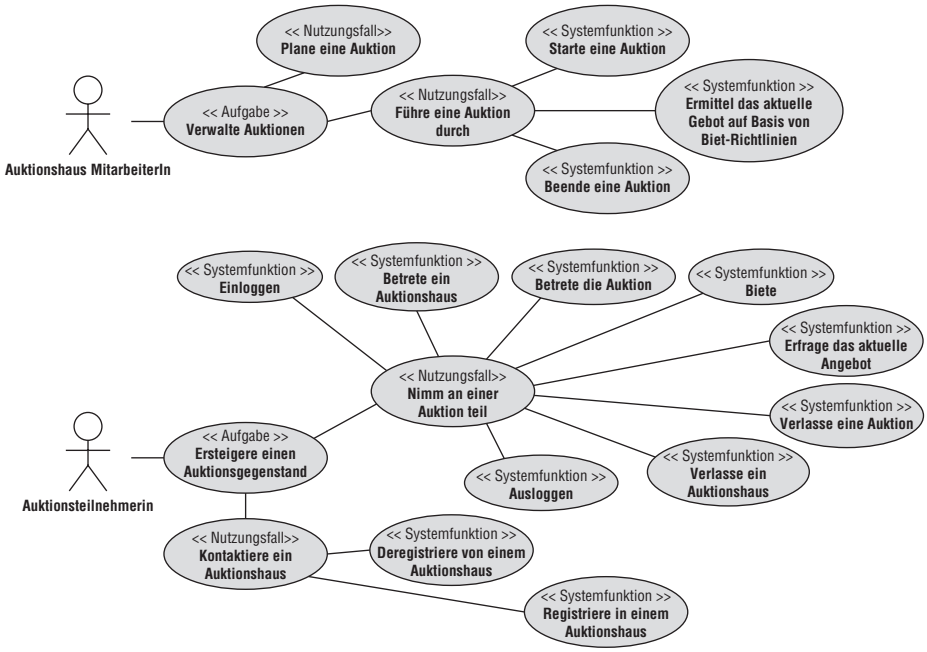

Abb. 1 *Nutzungs-Diagramm der Anwendung »Auctions«*

2.2 Anwendungen

Die in Abschnitt 2.1 genannten funktionalen Anforderungen sind fix und sollen in vier vorgegebenen Technologien umgesetzt werden. Es ist die Aufgabe der Studierenden, die unterschiedlichen NFRs herauszuarbeiten und die sich daraus ergebende Architektur abzuleiten:

Anwendung 1 mit Server-Komponente auf Basis der Java 2 Standard Edition [J2SE50]

Anwendung 2 mit Server-Komponente auf Basis der Java 2 Enterprise Edition [J2EE14]

Anwendung 3 mit Web Service auf Basis von Apache AXIS [Axis13]

Anwendung 4 mit Web Service auf Basis der Java 2 Enterprise Edition [J2EE14]

Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

2.3 Ablauf

In einer wöchentlich stattfindenden Vorlesung (an zwei Tagen: 1 SWS plus 2 SWS) wird den Studierenden das Fachwissen vermittelt, das die Basis bildet für die Übungsstunden (3 SWS). Ein durchgehender Projektkontext in den Übungen wird durch die Verwendung eines einzigen Anwendungsbeispiels (siehe 2.1) geschaffen. Die TeilnehmerInnen der Veranstaltung werden für die Übung in Teams (in diesem Fall 4 Teams à 4 Personen) eingeteilt. Alle Aufgaben werden in Teamarbeit gelöst, da neben technischen und fachlichen Erkenntnissen der Umgang mit organisatorischen und sozialen Problematiken erlernt werden soll [FSN05]. Die Studierenden lernen, miteinander zu kommunizieren und gemeinsam schwierige, komplexe Aufgaben zu verteilen und zu lösen. Der Aufbau der Entwicklungsumgebung erfolgt jedoch individuell. Tabelle 1 gibt einen Überblick über die Vorlesungs- und Übungsinhalte. Erläuterungen zu Tabelle 1 folgen im Anschluss.

	Vorlesung	**Übung**
1	Architektur-sichten, -modellierung, -muster	**Projektmanagement**: Fragebogen und Teambildung **Projektmanagement**: Aufbau der Entwicklungsumgebung
		Anwendung: Spezifikation der Funktionalität (Aktoren, Aufgaben, Nutzungsfälle, Systemfunktionen) auf Basis einer vorgegebenen Problembeschreibung **Anwendung**: Spezifikation des funktionalen Systemtests (logische und konkrete Testfälle)
		Web Client: Spezifikation der Funktionalität (Arbeitsbereiche, Sichten, Sichten-Übergänge)
2	Komponenten	**Anwendung J2SE**: Spezifikation von NFRs **Anwendung J2SE**: Entwurf der Architektur **Anwendung J2SE**: Diskussion der Architektur anhand der NFRs **Anwendung J2SE**: Diskussion der Architektur anhand von Architektursichten
		J2SE-Server: Entwurf der Server-Schnittstelle **J2SE-Server**: Entwurf des Servers
		Web Client: Entwurf der Architektur und des J2SE-Dialogkerns (Komponenten, Klassen, Pakete) **Web Client**: Implementierung
3	Verteilte Systeme, Middleware, Enterprise-Architekturen	**J2SE-Server**: Implementierung
		Anwendung: Spezifikation des nicht funktionalen Systemtests (hauptsächlich für Usability, Sicherheit und Plattform-Unabhängigkeit) **Anwendung**: Implementierung (Aufzeichnung) des Systemtests mit Hilfe des Capture-and-Replay-Werkzeugs [MaxQ098]
		Anwendung J2SE: Ausführung des Systemtests **Anwendung J2SE**: Aktualisierung der Spezifikation und des Entwurfs nach Implementierung und Fehlerbeseitigung

	Vorlesung	**Übung**
4	CORBA	**Anwendung J2EE**: Spezifikation von NFRs **Anwendung J2EE**: Entwurf der Architektur
		J2EE-Server: Entwurf der Server-Schnittstelle
		Web Client: Entwurf des J2EE-Dialogkerns des Web Clients
5	J2EE	**J2EE-Server**: Entwurf des Servers **J2EE-Server**: Implementierung 1 (ohne Persistenz)
		Anwendung: Aktualisierung der Spezifikation des funktionalen Systemtests **Anwendung**: Aktualisierung der Spezifikation des nicht funktionalen Systemtests
		Anwendung J2EE: Implementation eines Application Clients
6	.NET	**J2EE-Server**: Implementierung 2 (mit Container-Managed Persistenz)
		Web Client: Implementierung des J2EE-Dialogkerns des Web Clients
		Anwendung J2EE: Ausführung des Systemtests auf Basis CMP **Anwendung J2EE**: Aktualisierung des Entwurfs nach Implementierung
7	NFR, Rationale	**Anwendungen J2SE und J2EE**: Aktualisierung der Spezifikation und des Entwurfs
		J2EE-Server: Implementierung 3 (mit Bean-Managed Persistenz)
		Anwendung J2EE: Ausführung des Systemtests auf Basis BMP
		Projektmanagement: Präsentation der J2SE- und J2EE-Anwendungen

8	CodeCamp

9	Web Services	**Anwendung AXIS**: Entwurf der Architektur
		AXIS-Service: Entwurf der Server-Schnittstelle nach der Contract-first-Methode **AXIS-Service**: Implementation eines Java Web Service (JWS) nach der Implement-first-Methode
10	XML, SOAP	**Web Service Client**: Diskussion von Client-Entwicklungsarten **Web Service Client**: Implementation eines Google Web Service Client **Web Service Client**: Implementierung eines Dynamic Invocation Client für JWS
		AXIS-Service: Implementierung des Web Service nach der Contract-first-Methode
11	WSDL, UDDI	**UDDI Client**: Entwurf **UDDI Client**: Implementierung
		Anwendung AXIS: Implementierung des AXIS-Dialogkerns des Web Clients **Anwendung AXIS**: Ausführung des Systemtests

	Vorlesung	Übung
12	Orchestrierung	**Projektmanagement**: Tausch des Quellcodes
		Anwendung J2EEWS: Diskussion von Architekturoptionen **Anwendung J2EEWS**: Entwurf der Architektur **Anwendung J2EEWS**: Implementierung
13	Sicherheit	**Anwendung J2EE**: Diskussion der Architekturänderungen durch EJB 3.0
		Anwendung J2EEWS: Implementierung des J2EEWS-Dialogkerns des Web Clients **Anwendung J2EEWS**: Ausführung des Systemtests
14	Semantic Web	**Projektmanagement**: Abschlussfragebogen **Projektmanagement**: Abschlusspräsentationen der AXIS- und J2EE-basierten Web-Service-Anwendungen **Prüfungen**

Tab. 1 *Inhalt von Vorlesung und Übung pro Semesterwoche*

- In *Woche 1* beginnt die Spezifikation der Funktionalität der Anwendung.
- In *Woche 3* ist die *Anwendung 1* fertig implementiert.
- In *Woche 4* beginnt die Entwicklung der *Anwendung 2*. Basierend auf NFRs wird die Architektur entworfen, wie auch die Schnittstelle des J2EE-Servers, die sich aufgrund des J2EE-Rahmenwerks von der J2SE-Schnittstelle technisch sehr unterscheidet (fachlich dagegen nicht).
- In *Woche 5* wird der J2EE-Server entworfen und implementiert. Die einzelnen (prinzipiell verteilbaren) Enterprise JavaBeans (EJB) [EJB21] werden von einem lokal installierten JBoss Application Server [JBossAS403] instanziiert. Aufgrund der Veränderung der NFRs gegenüber der *Anwendung 1* und der Schnittstelle des J2EE-Servers wird die Spezifikation der Systemtestfälle auf den neuesten Stand gebracht. Um die J2EE-Technologie besser zu erlernen, d.h. konkret den Aufruf von EJBs, hat es sich bewährt, zuerst einen »standalone Application Client« zu entwickeln, der den gesamten Nutzungsfall »Nimm an einer Auktion teil« ohne Interaktion des Benutzers ablaufen lässt.
- In *Woche 6* wird die Implementierung des J2EE-Servers um Persistenz erweitert, zunächst mit der Container Managed Persistence (CMP) [EJB21, Kapitel 10].
- In *Woche 7* wird der J2EE-Server mit Hilfe der Bean-Managed Persistence (BMP) [EJB21, Kapitel 12] implementiert und getestet, um auch diese Technologie kennenzulernen.
- In *Woche 8* präsentieren die Teams ihre beiden Lösungen in je einer halben Stunde im Rahmen eines »CodeCamps«: Spezifikation, Entwurf, Implementierung (Quellcode), Systemtest (Spezifikation und Ausführung). Hier wird ausführlich Feedback gegeben, damit Irrtümer bei den beiden letzten Anwendungen nicht wiederholt werden. Ab Woche 12 werden die Teams auf Basis

des Quellcodes eines anderen Teams arbeiten. Das CodeCamp dient somit zum gezielten Auffrischen der Kenntnisse über die Implementierung des »neuen« Systems.

- In *Woche 9* beginnt die Entwicklung der ersten Web-Service-Anwendung (*Anwendung 3*). Die Schnittstelle des Web Service wird mit der Contract-first-Methode entworfen, d.h., es entsteht eine Beschreibung mit WSDL [WSDL11]. Gleichzeitig wird der für *Anwendung 1* entwickelte J2SE-Server als Java Web Service (JWS) [Axis13, User Guide] gekapselt. Dies dient zum Kennenlernen der Implement-first-Methode und der AXIS SOAP Engine selbst.
- In *Woche 10* werden die NFRs verschiedener Typen von Web Service Clients (static stub, dynamic invocation) bewertet. Der allererste Web Service Client wird für einen produktiven Web Service im Internet (Google Web Service) entwickelt, und nicht für einen selbst entwickelten Web Service, um die tatsächliche Anwendbarkeit von Schnittstellenbeschreibungen aufzuzeigen.
- In *Woche 11* wird ein UDDI-Client [UDDI2] entworfen und implementiert. Dabei werden öffentlich zugängliche UDDI-Registraturen verwendet.
- In *Woche 12* tauschen die Teams überraschend die entwickelten Spezifikationen sowie die entwickelte Software und beginnen, die *Anwendung 4* zu realisieren.
- In *Woche 13* wird die *Anwendung 4* vervollständigt und die Vor- und Nachteile der neuen EJB 3.0-Technologie [EJB30] diskutiert, um zumindest einen kurzen Blick auf diese neue Komponententechnologie zu werfen.
- In *Woche 14* präsentieren die Teams die beiden *Anwendungen 3* und *4*, d.h. Spezifikation, Entwurf, Quellcode und Systemtest (Spezifikation und Ausführung).

2.4 Nicht funktionale Anforderungen und Architekturen

Neben den Technologien sollen die Studierenden vor allem die Auswirkungen von Architekturentscheidungen verstehen lernen.

Dies wird zum einen durch die Variation der Architektur erreicht. Die vom System automatisch ausgeführten Aufgaben, d.h. Anwendungslogik und Datenhaltung, ermöglichen entweder Server-Komponenten (*Anwendungen 1* und *2*) oder Web Services (*Anwendungen 3* und *4*). Der Web Client wird zuerst entwickelt und bleibt weitestgehend konstant. Weitestgehend deshalb, da der Web Client an die jeweilige Technologie angepasst werden muss. Dabei wird aber nur die Dialogkern-Komponente ausgetauscht, die den Web Client an eine Server-Komponente oder an einen Web Service anbindet. Die Architektur der Web-Oberfläche basiert auf [Sie05]. Die Server-Komponenten und Web Services werden in der jeweiligen Technologie nach und nach entwickelt.

Andreas Zeller · Marcus Deininger (Hrsg.) (2007): Software Engineering im Unterricht der Hochschulen
SEUH 10, Stuttgart, dpunkt.verlag, Heidelberg

Zum anderen werden die Auswirkungen der Architekturentscheidungen durch ständige Diskussion der NFRs verdeutlicht. Tabelle 2 zeigt, welche NFRs bei welchen architektonischen Komponenten eine Rolle spielen. Lernziel hierbei ist die Erkenntnis, in welchem Maße sich eine bestimmte Technologie eignet zur Erfüllung bestimmter NFRs. Wird »Persistenz« gefordert, so hat man die Möglichkeit, einen J2SE-Server auf Basis JDBC zu implementieren, oder einen J2EE-Server auf Basis von EJBs. Zur Auswahl kann das NFR »Transparente Einbindung von Persistenz (kein SQL)« dienen, das dann den Ausschlag für den J2EE-Server gibt. Die gleichzeitige Forderung von »Verteilung« und »Interoperabiliät« schließt die Verwendung der J2SE- und J2EE-Technologie (d.h. RMI und EJB) aus und resultiert in einer der Web-Service-Technologien (d.h. SOAP). In Tabelle 2 sind die NFRs der obersten Abstraktionsstufe dargestellt, diese werden von den Studierenden zu einer Vielzahl von NFRs auf Ebene von u.a. Nutzungsfall, Systemfunktion und Benutzungsschnittstelle verfeinert.

	J2SE-Server (Anw. 1)	J2EE-Server (Anw. 2)	AXIS Web Service (Anw. 3)	J2EE Web Service (Anw. 4)
Interoperabilität			✓	✓
Logging	✓	✓	✓	✓
Multi-User	✓	✓	✓	✓
Persistenz	✓	✓		✓
Sicherheit		✓	✓	✓
Verteilung	✓	✓	✓	✓

Tab. 2 *Zusammenhang zwischen nicht funktionalen Anforderungen und architektonischen Komponenten*

2.5 Technologien und Werkzeuge

Wir unterscheiden die Technologien und Werkzeuge, die zur Realisierung der vier Architekturen dienen, von denen, die zur ständigen Qualitätssicherung eingesetzt werden. Die verwendeten Technologien und Werkzeuge sind in Tabelle 3 dargestellt.

	Technologien	Werkzeuge
Web Client	■ HTML oder XHTML ■ Java Servlets [JST24] oder Java Server Pages [JSP20]	■ Apache Tomcat [Tomcat55] ■ MAXQ für Capture-Replay-Systemtest [MaxQ098] ■ NVU als HTML-Editor [NVU10]
J2SE-Server	■ Java 2 Standard Edition (J2SE) 5.0 und Dokumentation [J2SE50]	■ Eclipse 3.1 [Eclipse31]
J2EE-Server	■ Java 2 Enterprise Edition (J2EE) 1.4 und J2EE Tutorial [J2EE14] ■ JBoss 4.0 [JBossAS403]	■ Eclipse JBoss IDE 1.5 [JBossIDE15]
AXIS Web Service	■ Java 2 Standard Edition 5.0 (J2SE) [J2SE50] ■ AXIS 1.3 [Axis13]	■ Eclipse 3.1 WTP [Eclipse31] ■ AXIS SOAP Monitor [Axis13, User Guide, Appendix]
J2EE Web Service	■ Java 2 Enterprise Edition 1.4 [J2EE14] ■ JBoss 4.0 [JBossAS403]	■ Eclipse JBoss IDE 1.5 [JBossIDE15]

Tab. 3 *Zur Realisierung der Anwendungen und zur Qualitätssicherung verwendete Technologien und Werkzeuge*

Als Entwicklungsumgebung dienen zwei Eclipse-Plattformen (in der Ausprägung 3.1 WTP und in der Ausprägung JBoss IDE 1.5), die beide den Vorteil haben, die Entwicklung von J2EE-Anwendungen und Web Services durch spezielle Plug-ins zu erleichtern. Zur Versionskontrolle der kollaborativ entwickelten Software wird CVS der Vorzug gegenüber Subversion gegeben, da das Subversion Eclipse Plug-in »Subclipse« nicht stabil genug zur Verfügung steht. Zur Erstellung von UML-Diagrammen dient das UML-Modellierungswerkzeug JUDE [JUDE251]. Anforderungen werden mit Hilfe des Werkzeugs Sysiphus [Sysiphus] dokumentiert, genauer gesagt unter Verwendung dessen Web-Oberfläche »REQuest«.

3 Erfahrungen

Kenntnisstand:
TeilnehmerInnen waren Studierende der anwendungsorientierten Informatik (Bachelor und Master), Mathematik, Computerlinguistik, Physik, Geografie und des wissenschaftlichen Rechnens. Darunter erfahrene Software Engineering 1 [PBR+05] Teilnehmer und Studienortwechsler von anderen Universitäten, Fachhochschulen und Berufsakademien, bei denen Wissen über Software Engineering teilweise nur gering vorhanden war – diese Lücke musste durch eine geeignete Teameinteilung ausgeglichen werden. Ebenso verhielt es sich mit Programmierkenntnissen – im Vorfeld wurden die Erfahrungen und Kenntnisse durch einen Fragebogen erfasst, damit die Teams gleichmäßig eingeteilt werden konnten. Programmierkenntnisse wurden dennoch in den ersten zwei Wochen des Semesters nachgeschult in zwei Vorlesungen zu Java und Java-Web-Anwendungen und in

ergänzenden Hausaufgaben, die individuell korrigiert und besprochen wurden. Im Laufe des Semesters konzentrierten sich die TeilnehmerInnen mit geringeren Programmierkenntnissen auf die Spezifikation mit Hilfe von [Sysiphus] und die Synchronisation von Entwurf und Quellcode.

Ablauf der Übungen:
Die Übung bestand jeweils aus drei Teilen: Im ersten Teil wurde in einem Vortrag des Übungsleiters die jeweils neuen Technologien und Werkzeuge vorgestellt (Java, Java-Web-Anwendungen, Java 5 Neuheiten, Eclipse, J2EE-Details, JBoss, AXIS, Build und Deployment). Im zweiten Teil wurde das neue Aufgabenblatt erklärt. Im dritten Teil haben die Teams mit der besten Lösung ihre Ergebnisse präsentiert.

Anwendung und Web Client:
Die Anforderungsspezifikation der Anwendung und die Entwicklung des Web Clients stellten einen angenehmen Einstieg zu Beginn des Semesters dar, da die Studierenden die vorausgesetzten Kenntnisse von SWE 1 entweder auffrischen oder punktuell neu aufbauen konnten. Gleichzeitig konnte die Vorlesung in der Zeit neues Wissen vermitteln, das ja erst bekannt sein muss, bevor es in den Übungen vertieft werden kann.

J2SE-Server:
Die Entwicklung eines nicht persistierenden J2SE-Servers hat sich aufgrund der frühzeitigen Modellierung der Web-Client-Schnittstelle bewährt. Diese Schnittstelle kann bei den folgenden Anwendungen als Startpunkt dienen. Allerdings kann diese Schnittstelle nicht vollständig wiederverwendet werden, da AXIS Web Services keine Java-Kollektionen unterstützen. Die gesamte Anwendung ist zudem schnell lauffähig. Der Systemtest ist damit frühzeitig implementier- und ausführbar und kann bei nachfolgenden Anwendungen als Regressionstest wiederverwendet werden. Eine frühzeitige, genaue Kontrolle und Korrektur der Studierenden-Ergebnisse ist sehr zu empfehlen, da falsche Entwurfsentscheidungen aufwendige Änderungen der Schnittstelle bei Verwendung anderer Technologien nach sich ziehen.

J2EE-Server:
Diese Technologie hat sich als sehr umfangreich und schwer zu erlernen dargestellt, insbesondere die Verwendung der Bean Managed Persistence, die der Vollständigkeit halber mit einbezogen wurde. Hierbei ist eine individuelle, sehr zeitaufwendige Betreuung notwendig.

Web Services:
Die Verwendung der einstufigen Implement-first-Methode und der zweistufigen Contract-first-Methode hat sich in dieser Reihenfolge nicht bewährt, da die Studierenden mit Hilfe von Werkzeugen den Contract aus der Implementierung

generierten und diesen nicht mehr unbefangen modellierten, insbesondere die Datenstrukturen. In der Zukunft ist der Vorrang eindeutig der Contract-first-Methode zu geben, damit die Modellierung in WSDL [WSDL11] gründlich erlernt wird.

Technologien:
Die Entwicklung einer verteilten Anwendung in mehreren Technologien mit mehreren Werkzeugen erfordert eine genaue Abstimmung der Versionen aller eingesetzten Software-Produkte. Diese Abstimmung muss im Vorfeld herausgefunden werden und sollte den Studierenden als Gesamt-Installationspaket (in Form einer CD) zur Verfügung gestellt werden, damit durch Verwendung eigener Downloads der Lernerfolg durch inkompatible Versionen nicht verhindert wird. Die xdoclet-Technologie [XDoclet], die zur Entwicklung aller vier Anwendungen verwendet werden kann, zur Entwicklung von *Anwendung 3* sogar unentbehrlich ist, hat sich als schwer zu erlernend herausgestellt, da wenig gute Dokumentation vorhanden ist.

Betreuungsaufwand:
Der Betreuungsaufwand ist außerordentlich hoch und steigt linear mit der Anzahl Teams, da pro Team verschieden granulare Spezifikationen und vor allem sehr unterschiedliche Entwürfe und Implementierungen entstehen. Aufgrund der neuen Technologien und Werkzeuge ist individuelle Unterstützung, per E-Mail oder durch Vorführungen, unumgänglich. Die 16 TeilnehmerInnen wurden durch den Übungsleiter und zwei wissenschaftliche Hilfskräfte betreut.

Feedback:
Die Studierenden zeigten sich bis zum Schluss sehr motiviert. In einem Abschlussfragenbogen wurde als Ursache ermittelt, dass sie umfangreiche theoretische Kenntnisse erhalten haben und gleichzeitig Erfahrung mit vielen aktuellen Technologien sammeln bzw. vertiefen konnten. Die meisten neuen Erkenntnisse haben die Studierenden bei Entwicklung der J2EE-Anwendung (aufgrund Enterprise JavaBeans) und der AXIS-Anwendung (aufgrund Web Services) erhalten. Der theoretische Vorlesungsanteil in Verbindung mit praktischen Übungen wurde als sehr lehrreich beurteilt. Insbesondere die Reihenfolge der Vorlesungsinhalte, die Abfolge der Anwendungen und die Synchronisation von Vorlesung und Übung wurden gelobt. Die Einarbeitung in den Quellcode des anderen Teams hat viel Arbeit verursacht. Ebenso viel Zeit hat die Erstellung von Konfigurationen zum Build und Deployment jeder Anwendung gekostet. Durch die Entwicklung mehrerer Anwendungen hatten die Studierenden mehrere Erfolgserlebnisse.

4 Zusammenfassung und Ausblick

In diesem Beitrag haben wir unser Vorgehen zur Lehre von modernen Technologien und Architekturen vorgestellt. Innerhalb eines Semesters haben wir Konzepte über verteilte Systeme und Architekturen vermittelt, diese in Zusammenhang mit Software-Engineering-Methoden gebracht und in einem Forward-Engineering praktisches Entwicklungswissen über J2EE- und Web Service basierte Geschäftsanwendungen weitergegeben.

Die Lehrveranstaltung wird im Wintersemester 2006/2007 wiederholt. Da sich in der Zwischenzeit die Technologien weiterentwickelt haben, werden wir die neue Komponententechnologie Enterprise JavaBeans 3.0 [EJB30] einsetzen und die Orchestrierung von Web Services mit BPEL [BPEL11] einüben. Damit der Entwicklungsaufwand zur Anpassung des Web Client an die unterschiedlichen Technologien von Server-Komponenten und Services entfällt, kann ein durch Konfiguration adaptierbarer Dialogkern eingesetzt werden, der in [RHN+06] vorgestellt wurde. Weiterhin wollen wir nicht funktionale Anforderungen verstärkt modellieren und zur Bewertung der Architekturoptionen einsetzen.

Wir danken Jerko Horvat und Nick Meier für die Unterstützung der Übung und allen TeilnehmerInnen für die konstruktive Mitarbeit.

Literatur

[Axis13] SOAP Engine Apache AXIS 1.3 Final. Apache Web Service Project, 5 October 2005. *http://ws.apache.org/axis*

[BPEL11] Business Process Execution Language (BPEL) 1.1. IBM, May 2003. *http://www-128.ibm.com/developerworks/library/specification/ws-bpel*

[Eclipse31] Eclipse Platform 3.1 inklusive Web Tools Project (WTP) Plug-in. *http://www.eclipse.org* und *http://www.eclipse.org/webtools*

[EJB21] Enterprise JavaBeans Technologie 2.1 Final Release. Sun Microsystems, 24 November 2003. *http://java.sun.com/products/ejb*

[EJB30] Enterprise JavaBeans Technologie 3.0 Proposed Final Draft. Sun Microsystems, December 2005. *http://java.sun.com/products/ejb*

[FSN05] A. Fleischmann, K. Spies, K. Neumeyer: Teamtraining für Software-Ingenieure. Software Engineering im Unterricht der Hochschulen. Aachen, 2005.

[Häf05] P. Häfele: Softwareentwicklung mit dem TRAIN-Prozess. Bachelor-Arbeit. Universität Heidelberg, AG Software Engineering, 2005. *http://www-swe.informatik.uni-heidelberg.de/research/publications/TRAIN.pdf*

[J2EE14] Java Platform Enterprise Edition (J2EE). Version 1.4. Sun Microsystems. *http://java.sun.com/javaee*

[J2SE50] Java Platform Standard Edition (J2SE). Version 5.0. Sun Microsystems. *http://java.sun.com/javase*

[JBossAS403] JBoss Application Server 4.0.3, Service Package 1. JBoss, 24 October 2005. *http://labs.jboss.com/portal/jbossas*

[JBossIDE15] JBoss IDE für die Eclipse Plattform 1.5.0. JBoss, October 2005. *http://labs.jboss.com/portal/jbosside*

[JSP20] JavaServer Pages Technologie. Version 2.0. Sun Microsystems. *http://java.sun.com/products/jsp*

JST24] Java Servlet Technologie. Final Release 2.4. Sun Microsystems, 24 November 2003. *http://java.sun.com/products/servlet*

[JUDE251] UML Modellierungswerkzeug JUDE/Community 2.5.1. Change Vision, October 2005. *http://jude.change-vision.com/jude-web*

[MaxQ098] MaxQ Capture-and-Replay Testwerkzeug für HTTP-Clienten. Tigris.org (Open Source Software Engineering Tools), October 2005. *http://maxq.tigris.org*

[NVU10] NVU HTML-Editor 1.0. Linspire Inc., 28 June 2005. *http://www.nvu-composer.de*

[PBR+05] B. Paech, L. Borner, J. Rückert, A.H. Dutoit, T. Wolf: Vom Kode zu den Anforderungen und zurück: Software Engineering in 6 Semesterwochenstunden. Software Engineering im Unterricht der Hochschulen. Aachen, 2005.

[RHN+06] J. Rückert, J. Horvat, D.-K. Nguyen, S. Becker, B. Paech: Modell zur Adaption eines Dialogkerns. Modellierung 2006, Workshop »Qualität von Modellen«. Innsbruck, 2006.

[Sie05] J. Siedersleben. Moderne Software-Architektur. Umsichtig planen, robust bauen mit Quasar. dpunkt.verlag, August 2004.

[Sysiphus] Sysiphus Requirements-Engineering-Werkzeug. Build October 2005. *http://wwwbruegge.in.tum.de/Sysiphus*

[Tomcat55] Apache Tomcat 5.5.12. Apache, October 2005. *http://tomcat.apache.org*

[UDDI2] Universal Description, Discovery and Integration Specification (UDDI) Version 2. Organization for the Advancement of Structured Information Standards (OASIS). *http://www.oasis-open.org*

[WSDL11] Web Services Definition Language (WSDL) 1.1. W3C Note 15 March 2001. *http://www.w3.org/TR/wsdl*

[XDoclet] XDoclet Open Source Generation Engine. *http://xdoclet.sourceforge.net/xdoclet*