Electronic version of an article published in **Zeller, A.; Deininger, M. (Hrsg) Software Engineering im Unterricht der Hochschulen SEUH 10 - Stuttgart 2007, pp. 59 - 72**

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 [Abstract]

An der Universität Heidelberg wurde die Veranstaltung "Software Engineering moderner Anwendungen: Komponentenbasierte, service-orientierte 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, service-orientierte oder mobile Systeme* wurde an der Universität Heidelberg erstmals im Wintersemester 2005/2006 angeboten. Studierende der Anwendungs-orientierten 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 Semesterwochen-stunden (SWS), davon 3 SWS Vorlesung und 3 SWS Übungen. SWE 2B erfordert die Kenntnisse aus SWE 1 [PBR+05] [Häf05].

Ziel der Veranstaltung ist einerseits der Wissensaufbau über Architektur-Sichten und –Muster, sowie die Bewertung von funktionalen und nicht-funktionalen Anforderungen zur Auswahl und Anpassung geeigneter Architekturen. Andererseits sollen die ausgewählten Architekturen umgesetzt werden, um praktische Kenntnisse über Komponentenbasierte und Service-orientierte Systeme aufzubauen.

Kernidee unseres Vorgehens dazu 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 nicht-funktionalen Anforderungen. Die Anwendung wird für AnwenderInnen zugänglich durch einen Web Client der weitestgehend konstant gehalten wird, d.h. nur dessen Dialogkern-Komponente, die den Web Client mit einer Server-Komponente oder einem Web Service verbindet, muss an deren 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 Softskills 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 fremden Quellcode hat das Bewußtsein zur Entwicklung von qualitativ hochwertiger Software geschärft.

In Abschnitt 2 präsentieren wir Vorlesung und Übung im Detail: das Anwendungsbeispiel wird in 2.1 vorgestellt, den Ablauf erklären wir in 2.2 und die verwendeten Technologien und Werkzeuge nennen wir in 2.3. 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 Anwendungsbeispiel

Als Anwendungsbeispiel haben wir versucht ein anschauliches Beispiel zu wählen: eine Anwendung mit der AnwenderInnen an Auktionen mehrerer Auktionshäuser mitbieten können. Die von den AnwenderInnen in der Rolle von „AuktionsteilnehmerInnen“ ausführbaren Aufgaben lauten:

- Registrieren und Deregistrieren
- Einloggen und Ausloggen
- „Betreten” von Auktionshäusern und teilnehmen an Auktionen
- Bieten in Auktionen

Die vom System automatisch ausgeführten Aufgaben lauten:

- Starten und Beenden von Auktionen nach Zeitvorgaben
- Ermitteln des aktuell höchsten Gebots nach Gebotsabgabe

2.2 Ablauf

In einer wöchentlich stattfindenden Vorlesung (an zwei Tagen: 1 SWS plus 2 SWS) wird den Studierenden das Fachwissen vermittelt, welches 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.

Tab 1.: Inhalt von Vorlesung und Übung pro Semesterwoche

	Vorlesung	**Übung**
1	Architektur-sichten, -modellierung, -muster	Projektmanagement: Fragebogen und Teambildung Projektmanagement: Aufbau der Entwicklungsumgebung
		Anwendung: Spezifikation der Funktionalität auf Basis einer Problembeschreibung Anwendung: Spezifikation des funktionalen Systemtest
		Web Client: Spezifikation der Funktionalität
2	Komponenten	Anwendung J2SE: Spezifikation von nicht-funktionalen Anforderungen Anwendung J2SE: Entwurf der Architektur Anwendung J2SE: Diskussion der Architektur anhand der nicht-funktionalen Anforderungen 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 Web Client: Implementierung
3	Verteilte Systeme, Middleware, Enterprise Architekturen	J2SE-Server: Implementierung
		Anwendung: Spezifikation des nicht-funktionalen Systemtests Anwendung: Implementierung des (Capture-and-Replay-) Systemtests
		Anwendung J2SE: Ausführung des Systemtests Anwendung J2SE: Aktualisierung der Spezifikation und des Entwurf nach Implementierung und Fehlerbeseitigung
4	CORBA	Anwendung J2EE: Spezifikation von nicht-funktionalen Anforderungen Anwendung J2EE: Entwurf der Architektur
		J2EE-Server: Entwurf der Server-Schnittstelle
		Web Client: Entwurf des J2EE-Dialogkerns des Web Client

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 Client
6	.NET	J2EE-Server: Implementierung 2 mit CMP
		Web Client: Implementierung des J2EE-Dialogkerns des Web Client
		Anwendung J2EE: Ausführung des Systemtests Anwendung J2EE: Aktualisierung des Entwurfs nach Implementierung
7	NFR, Rationale	Anwendung J2SE J2EE: Aktualisierung der Spezifikation und des Entwurfs
		J2EE-Server: Implementierung 3 mit BMP
		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 Client Anwendung AXIS: Ausführung des Systemtests

12	Orches-trierung	Projektmanagement: Tausch des Quellcodes
		Anwendung J2EEWS: Diskussion von Architektur-Optionen Anwendung J2EEWS: Entwurf der Architektur Anwendung J2EEWS: Implementierung
13	Sicher-heit	Anwendung J2EE: Diskussion der Architekturänderungen durch EJB 3.0
		Anwendung J2EEWS: Implementierung des J2EEWS-Dialogkerns des Web Client Anwendung J2EEWS: Ausführung des Systemtests

14	Semantic Web	Projektmanagement: Fragebogen und Abschlusspräsentationen der AXIS- und J2EE-basierten Web Service Anwendungen Prüfungen

In *Woche 1* wird, nach der Bildung von Teams auf Basis von Angaben über eigene Erfahrungen in einem Fragenbogen, mit der Spezifikation der Funktionalität der

Anwendung (Aktoren, Aufgaben, Use Cases, Systemfunktionen) und des Web Client (Oberfläche und Dialogkern), sowie des funktionalen Systemtests (logische und konkrete Testfälle) begonnen.

In *Woche 2* wird der Web Client entworfen (Klassen, Pakete, Dialoge und deren Zustände etc.) und implementiert. Gleichzeitig wird im *Szenario 1* (Technologie Java 2 Standard Edition [J2SE50]) begonnen die Qualität und die Architektur zu spezifizieren und zu bewerten. Der J2SE-Server, dessen Schnittstelle zum Web Client und der Dialogkern des Web Client können damit entworfen werden.

In *Woche 3* wird der Systemtest implementiert durch Aufzeichnen der Testfälle mit Hilfe eines Capture-and-Replay Werkzeugs [MaxQ098]. Das *Szenario 1* ist nun implementiert. Die Spezifikation und der Entwurf sollten in Einklang zur Implementation gebracht werden, um eine Wiederverwendung der Spezifikation zu ermöglichen. Ein Systemtest für nicht-funktionale Qualitätsanforderungen wird spezifiziert, hauptsächlich für Usability, Sicherheit und Plattform-Unabhängigkeit.

In *Woche 4* beginnt das *Szenario 2*, basierend auf der Java 2 Enterprise Edition [J2EE14]. Zunächst werden die hier speziellen, nicht-funktionalen Eigenschaften spezifiziert. Danach wird die darauf basierende Architektur entworfen und die Schnittstelle des J2EE-Servers, die sich durch das J2EE-Rahmenwerk von der J2SE-Schnittstelle technisch sehr unterscheidet, fachlich aber nur gering. Der Dialogkern wird anschließend passend zur Schnittstelle des Web Client entworfen.

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 Veränderung der nicht-funktionalen Eigenschaften und der Schnittstelle des J2EE-Server wird die Spezifikation der Systemtestfälle auf den neuesten Stand gebracht. Um die J2EE-Technologie besser zu verstehen, hat es sich bewährt zuerst einen stand-alone Application Client zu entwickeln, welcher das ausführlichste Anwendungsszenario „Erfolgreich Bieten und Ersteigern“ ohne Interaktion des Benutzers ablaufen lässt.

In *Woche 6* wird die Implementierung des J2EE-Server um Persistenz erweitert, zunächst mit der Container Managed Persistence (CMP) [EJB21, Kapitel 10] . Nach Erlernen des Aufrufs der EJBs im J2EE-Server durch den Application Client aus Woche 5, kann nun der Dialogkern des Web Client implementiert werden. Abschließend wird der Systemtest ausgeführt, Fehler behoben, sowie die Spezifikation mit der tatsächlichen Implementierung abgeglichen.

In *Woche 7* werden Entwurf und Spezifikation der beiden *Szenarien 1 (J2SE) und 2 (J2EE)* konsolidiert. Der J2EE-Server wird 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 „CodeCamp“: Spezifikation, Entwurf, Implementierung (Quellcode), Systemtest (Spezifikation und Ausführung). Hier wird ausführlich Feedback gegeben, damit Irrtümer nicht in den beiden letzten Szenarien wiederholt werden.

In *Woche 9* beginnt die Entwicklung des ersten Web Service Szenarios (*Szenario 3*) auf Basis der SOAP Engine AXIS [Axis13]. Die Schnittstelle des Web Service wird mit der Contract-first-Methode entworfen, d.h. es entsteht ein Beschreibung mit WSDL [WSDL11]. Gleichzeitig wird der im *Szenario 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 nicht-funktionalen Eigenschaften verschiedener Typen von Web Service Clients bewertet. Der allererste Web Service Client wird für einen produktiven Web Service im Netz (Google Web Service) entwickelt und ein zweiter für den bereits implementierten Java Web Service. Die nach der Contract-first-Methode entworfene Schnittstelle wird abschließend durch einen Web Service implementiert.

In *Woche 11* wird ein UDDI-Client [UDDI2] entworfen und implementiert. Es werden öffentlich zugängliche UDDI-Registraturen verwendet. Der nach der Contract-first-Methode entwickelte Web Service wird mit dem Web Service Client verbunden, dazu ist der Dialogkern anzupassen. Abschließend wird gesamte Anwendung (das *Szenario 3*) getestet.

In *Woche 12* tauschen die Teams überraschend die entwickelten Spezifikationen und die entwickelte Software und beginnen das *Szenario 4* (Web Services auf Basis von J2EE und EJB 2.1) zu realisieren.

In *Woche 13* wird das *Szenario 4* komplettiert und die Vor- und Nachteile der neuen EJB 3.0-Technologie [EJB30] diskutiert.

In *Woche 14* präsentieren die Teams die letzten Web Service Lösungen von *Szenario 3 (AXIS WS)* und *4 (J2EE WS)*, d.h. Spezifikation, Entwurf, Quellcode, Systemtest (Spezifikation und Ausführung).

Neben der Erfahrung mit den Technologien sollen die Studierenden vor allem die Auswirkungen von Architekturentscheidungen verstehen lernen. Dies wird zum einen durch die ständige Veränderung der Architektur erreicht. Die vom System automatisch ausgeführten Aufgaben, d.h. Anwendungslogik und Datenhaltung, ermöglichen entweder Server-Komponenten (*Szenario 1* und *2*) oder Web Services (*Szenario 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 und Architektur nach und nach entwickelt. Zum anderen werden die Auswirkungen der Architekturentscheidungen durch ständige Diskussion der nicht-funktionalen Eigenschaften verdeutlicht. Tabelle 2 zeigt, welche nicht-funktionalen Eigenschaften bei welchen architektonischen Komponenten eine Rolle spielen.

Tab 2.: Zusammenhang zwischen nicht-funktionalen Eigenschaften und architektonischen Komponenten

	Web Client	Szenario 1: J2SE Server	Szenario 2: J2EE Server	Szenario 3: AXIS Web Service	Szenario 4: J2EE Web Service
Browser-Unabhängigkeit	✓				
Interoperabilität	✓				
Logging		✓	✓	✓	✓
Multi-User	✓	✓	✓	✓	✓
Persistenz			✓		✓
Sicherheit	✓		✓	✓	✓
Usability	✓				
Verteilung			✓	✓	✓

2.3 Technologien und Werkzeuge

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

Tab 3.: Zur Entwicklung der Komponenten verwendete Technologien und Werkzeuge

	Technologien	Werkzeuge
WebClient	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]

Als Entwicklungsumgebung dienen zwei Eclipse-Plattformen (in der Ausprägung 3.1 WTP und in der Ausprägung JBoss IDE 1.5), welche 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-Modellierungs-Werkzeug 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 Anwendungs-orientierten Informatik (Bachelor und Master), Mathematik, Computerlinguistik, Physik, und Geographie und des wissenschaftlichen Rechnens (Mathematik). Wissen über Software Engineering war teilweise nur gering vorhanden, 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 entsprechend ausgeglichen eingeteilt werden konnten.

Anwendung und Web Client: Die Anforderungs-Spezifikation der Anwendung und die Entwicklung des Web Client 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, welches 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 Schnittstelle der Web-Oberfläche bewährt. Auf dieser Schnittstelle kann bei den folgenden Szenarien aufgebaut werden, allerdings kann diese nicht vollständig wiederverwendet werden. Zudem ist die gesamte Anwendung lauffähig, was es erlaubt einen Systemtest frühzeitig zu implementieren und auszuführen. Der Systemtest kann bei nachfolgenden Szenarien als Regressionstest wiederverwendet werden. Eine frühzeitige, genaue Kontrolle und Korrektur der Studierenden-Ergebnisse ist sehr zu empfehlen, da falsche Entwurfsentscheidungen aufwändige 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 zeitaufwändige 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.

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 Erfahrung mit aktuellen Technologien sammeln konnten. Der theoretische Vorlesungsanteil in Verbindung mit praktischen Übungen wurde darin als weiterqualifizierend beurteilt. Durch die Entwicklung von mehreren Anwendungen hatten die Studierenden zudem 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 Engineerung 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 Herrn Jerko Horvat für die Unterstützung der Übung und allen TeilnehmerInnen für die konstruktive Mitarbeit.

References

[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, 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. Rueckert, 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. 1. Auflage, 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