Abschlussbesprechung Software-Projekt II (Projekt-Realisierung) bei Prof. Knabe im WS 2009

Hier die in der Abschlussbesprechung am 16.12.2009 geäußerten Meinungen der Student(inn)en, gegliedert nach Themengebieten und Meilensteinen.
Legende: + positiv bewertet, **–** negativ bewertet, ~ teils/teils, ! soll geändert werden.

Vorsemester Analysephase/Durchstiche

+	Durchstiche waren gut als Vorbereitung
!	Umfang der Projekte geriet bei der Implementierung zu groß, bitte Studenten auf Erfahrungen der Vorgänger hinweisen!
!	Eigene Meilensteine zu Maven (Build-Prozess), Zentralem Ausnahmemelden, Testsuite

Meilensteine

3 Entwurf: Lg -Klassendiagramm + Lg-Operations-Spezifikationen, DB-Entwurf

+	War nützlich, da Fehler früh erkannt
-	Nicht als großes Klassendiagramm, sondern besser als Architekturskizze

5 1. Durchstich OV + JUnit-Tests + KD Gesamtsystem + Zuständigkeitsverteilung

+	Testsuite war nützlich, wenn früh eingesetzt. Muss sich mit Code mitentwickeln.
-	4 Gruppen hatten einen Nur-Tester.. Erfahrungen weniger positiv, weil durch Codeänderungen am Produktcode die Testsuite zerbrochen wurde.
+	Durchstich Objektverwaltung sinnvoll.
!	Zuständigkeitsverteilung soll auch Projektmanagement, Qualitätssicherung, UI-Design berücksichtigen.
!	Hinweis auf Projektleiter geben, der die Arbeitsverbindlichkeit organisieren soll.
~	Gruppe CE-HTML kam gut ohne formalen Projektmanager aus.

7 2. Durchstich: Objektverbindungsverwaltung + Suchen Objektmenge + Testsuite

+	sinnvoll

9 Integrationstest

+	notwendig

10 Präsentation Gesamtsystem im Tagesplenum

+	Von Vorteil, weil es Druck Richtung Fertigstellung entfaltet.
-	Benotungskriterien für Präsentation waren nicht klar: einige Folien zu Techniken zusätzlich zur Vorführung.

11 Abnahme

-	War zu kurz, konnte die Arbeit nicht ausreichend würdigen.
!	Mehr Zeit für die Gesamtvorführung und Robustheitstests: 20 min zu wenig!
!	Sollte Teammitglieder über das befragen, was sie im Projektverlauf gearbeitet haben.
+	Codeinspektion während Abnahme war gut. Besser als nur Oberfläche anschauen.

Unterricht (aus Übungszeit herausgeschnitten)

+	Sinnvoll? Ja, wegen Möglichkeit zum Fragen.
-	Anwesenheitspflicht? Hat nicht viel gebracht.
!	Anwesenheitspflicht wäre bei den Rücksprachen sinnvoller.
!	Katalog von Themen anbieten, Wunschthemen auswählen lassen.
!	Wichtigen Stoff am Anfang vorstellen: Daraus auswählen lassen gemäß Vorkenntnissen.
+	Tagesplenum/Gesamtplenum besser? In diesem Kurzsemester war Tagesplenum gut. Gesamtplenum im Stundenplansystem zuvor ausweisen!
+	Stoffauswahl: Was interessant? AspectJ! (leider zu spät), JUnit

-	Stoffauswahl: Was überflüssig? OR-Mapping (macht man nicht mehr selbst)
!	Stoffauswahl: Was fehlte? Mehr JUnit-Muster (Exception expected, diagnosestarke Assertions), JPA-Beispiele. Genaue Abstimmung der Stoffauswahl mit Software Engineering nötig! Kniffe in den IDEs zeigen!
!	Einführung in Debugger!
-	Vorlesungsstil? Einschläfernd. Mehr konkrete Codebeispiele wären gut.
-	Skript/Folien: Was verbesserungswürdig? Wirkt zusammengewürfelt. Nicht klar erkennbar, welcher Stoff wichtig ist.
+	Was gut? Begründung bei Programmierrichtlinien, Kommentierte Literaturhinweise.

Rücksprachen

+	Meilensteinrhythmus alle 2 Wochen? War gut.
!	Dazwischen freiwillige Rücksprachen? Besser Pflicht.
!	Es sollten alle Teammitglieder zu ihrem Arbeitsstand befragt werden.
+	Viel über Eclipse gelernt.

Gruppenarbeit

+	Sehr teamfördernd.
!	Es gibt manchmal Mitläufer ohne eigenen Arbeitsbeitrag. Auf Möglichkeit der Teamänderung nach dem 1. Semester hinweisen.
!	Selbstevaluierung des Teams anfordern.
~	Gruppengröße diskutiert. Pro und Contra-Meinungen zur Vorgabe 4-5.

Techniken

Einsatz von Subversion

+	Essentiell. Überlebenswichtig.

Einsatz von OR-Mappern

	Oracle-Toplink? Hat funktioniert.
	Hibernate? Im allgemeinen JPA auf Hibernate sehr gut.

Einsatz von UI-Frameworks

	JSF manuell?
+	Visual JSF? Erstaunlich gut. Lesbarer Code generiert. Positiv überrascht.
	Flex?
+	Seam? Sehr gut, aber viel Lernaufwand.
+	Swing? Hervorragend dokumentiert. Viel Anleitungen dafür verfügbar. Sieht auf Windows/Mac sehr verschieden aus, wenn man nicht alle Properties setzt.

Einsatz von

+	Netbeans: Gut in VisualJSF, alle Techniken vorab eingebunden.
-	Netbeans: Sehr langsam, Refactoring sehr unzuverlässig, Glassfish-Server läuft sehr langsam. Zu starke Kopplung an Datenbank, App-Server, Editoren. Sobald man irgend etwas ändert, klappt das Gesamtsystem nicht mehr.

Einsatz von MulTEx 8

+	Früher wäre besser gewesen. Sinn erkannt.
-	Pflicht zur Spezifikation der Ausnahmen im Methodenkopf vermisst.

Einsatz von JUnit-Testtreibersuite

~	War nützlich. Besser jeder testet seinen Code statt Testerrolle. Jedoch nicht alles wegen Umfang!