Abschlussbesprechung Software-Projekt II (Projekt-Realisierung) bei Prof. Knabe im WS 2010

Hier die in der Abschlussbesprechung am 09.02.2010 geäußerten Meinungen der Student(inn)en, gegliedert nach Themengebieten und Meilensteinen.
Legende: + positiv, – negativ bewertet, ! Wunsch, ~ teils/teils

Vorsemester Analysephase/Durchstiche

-	Passt nicht bei untypischen Projekten wie 3D-Game
+	Ansonsten OK
!	Exemplarische Definition von Testfällen (auf Papier) schon hier vornehmen.

Meilensteine

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

-	Entwurf musste noch stark verändert werden im Laufe der Implementierung
+	Rich Domain Model (Medienbibliothek): Klappte gut nach Verstehen. Verständnis war bei Entscheidung noch nicht vorhanden. Mehr erklären, warum!
~	Anemic Domain Model (PestControl): Verständnis war bei Entscheidung noch nicht vorhanden. Technikentscheidung EJB erzwang dieses jedoch, daher musste nicht lange überlegt werden.

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

~	PestControl: Durchstich-Idee gut, parallele Entwicklung von UI und Lg klappte nicht gut.
!	AnaphylacticEngagement: KD Gesamtsystem sinnvoll, aber auf Papier zu gross. Tools dafür nötig. Altova UModel war gut (350€).
!	Knabe: Kompaktere Architekturdiagramme wahrscheinlich sinnvoller.
!	PestControl: Für JUnit-Tests zuvor das Round-Trip-Prinzip erläutern.

7 2. Durchstich: Objektverbindungsverwaltung + Suchen Objektmenge + Testsuite

9 3. Durchstich: Objektverwaltung 3 Klassen + Verbindungen + Testtreiber

12 Integrationstest

+	Zeitpunkt OK

14/15 Präsentation Gesamtsystem öffentlich

+	Gut wegen Eindrucks über andere Teams
+	Spornt einen auch selbst an.

15 Build-Prozess, Zentrales Ausnahmemelden

+	Wegen des Aufwands ist eigener Meilenstein angemessen.
!	Früher (nach dem 1. Durchstich) würde größeren Nutzen für das weitere Projekt entfalten.
!	Diskussion Ausnahme vs. Returncode sinnvoll.
!	Für viel eingesetzte Technik wie Wicket sollten Studenten recherchieren und vortragen, wie sie Zentrales Ausnahmemelden erreichen können.

17/18 Abnahme

!	Querschnittsaufgaben wie Diagramme, Dokumentation sind keinem Codebereich zugeordnet, sollten aber individuell gewürdigt werden.

Unterricht (aus Übungszeit herausgeschnitten)

+	Sinnvoll? Ja

+	Stoffauswahl: Was interessant? Generic DAO Pattern
~	Stoffauswahl: Was überflüssig? Scala war interessanter Exkurs, aber nicht relevant.
	Stoffauswahl: Was fehlte? -
!	Ersatz-Frameworks für andere Programmiersprachen als Java aus Vorsemestern vorstellen, z.B. NUnit oder MS Testing Tools für C#-Tests, MS Build statt Maven.
	Unterrichtsstil?
!	Materialien auf meiner SWP-II-Seite? Gewöhnungsbedürftig strukturiert. Aufräumen!

Rücksprachen

+	Meilensteinrhythmus alle 2 Wochen? OK
+	Anwesenheitspflicht bei Rücksprachen? OK, auch wegen Teamarbeit.
!	Regel für Anwesenheitspflicht klarstellen: Max. einmal fehlen ohne Nachweis erlaubt.

Gruppenarbeit

+	Selbstauswahl der Teammitglieder war gut.
+	Gruppengröße 4-5 war OK: Zu zweit wäre es nicht schaffbar, mehr als 5 würde Mitläufer erleichtern.
+	Regelmäßige Kommunikation (z.B. Pair Programming mittels Skype) war sehr gut.
~	Diesmal gab es keine Gruppe mit Projektleiter.
-	MonsterMail: Möchte Projektleiter verpflichtend, könnte auch rotieren, sollte sich positiv auf Benotung auswirken.
+	PestControl: War gut, dass sich Gruppen selbst entscheiden konnten, ob sie einen Projektleiter möchten.
	Notendifferenzierung durch Gruppe bei einem anderen Professor klappte i.A. gut, aber in Einzelfällen nicht.
!	Konsens-Notendifferenzierung sinnvoll. Wenn einer aus dem Team nicht einverstanden: Einzelbenotung.

Techniken

Einsatz von Subversion vs. Assembla (Versionierung+Ticketing)

+	Versionsverwaltungssystem sinnvoll. Auch wegen Eingewöhnung für die Praxis.
+	Lehrkraftnews: Projekthosting bei Assembla mit Ticketsystem war gut.
!	MonsterMail: Ticketsystem Eventum (PHP, Oracle) war OK. Sollte Uni zur Verfügung stellen.
	Oder RedMine

Einsatz von Hibernate (OR-Mapper)

Einsatz von Wicket

!	Wicket-Bücher sollten in Bibliothek sein.
	Wenn man drin war, ging es gut, z.B. AJAX. Doku grauenhaft. Beispiele im Netz nicht ausreichend. Anpassen der Komponenten schwer.

Einsatz von MulTEx

	Medienbibliothek: Vorteile erkannt, aber noch nicht so produktiv eingesetzt.
	MonsterMail: Nach Verständnis ließ es sich gut umsetzen.

Einsatz von JUnit-Testtreibersuite

+	Nützlich bei AnaphylacticEngagement, MonsterMail
~	Wäre nützlich, aber zu wenig eingesetzt bei Medienbibliothek
!	MonsterMail: Selbsttest gut, mit Nachkontrolle durch anderen Tester besser.
+	Ein Tester für Alles (Medienbibliothek): Gute Erfahrung, Methodenzweck muss gut kommuniziert werden (Doku-Kommentare).
-	Modell „Kein Commit ohne Testsuiteerfolg“ wurde bei keiner Gruppe konsequent

	eingesetzt.
!	Build-Server seitens der Hochschule wäre sinnvoll, z.B. für konsequenten Einsatz der Testsuite.

Einsatz von

	Lift/Scala? Interessant. Bei einfachen Dingen kompliziert, bei komplizierten Dingen einfach.
+	AspectJ? Nützlich, aber sparsam einsetzen. Debugging schwieriger.
+	C#/XNA? Sehr zufrieden. Vergleich zu C++/DirectX unbekannt.

Studiengang

	JUnit-Testtreibererstellung sollte schon in Pr 1-2 gelehrt werden, damit sie auch in anderen Fächern wie Computergrafik verwendet werden können.