Abschlußbesprechung Software-Projekt II (Projekt-Realisierung) bei Prof. Knabe im SS 2005

Hier die in der Abschlußbesprechung am 11.07.2005 geäußerten Meinungen der Student(inn) en, gegliedert nach Themengebieten und Meilensteinen.
Legende: + positiv bewertet, – negativ bewertet, ! Vorschlag, ~ teils/teils

Analysephase (Software-Projekt I)

+	BugTracker, PlanetenMgr: Analysephase war gute Vorbereitung
+	LessEntropy: Analysephase war wichtig zur Gruppenfindung
-	Zu viel Zeit dafür verwendet im Vergleich zur Implementierung
-	Stofflich zu viel aus SA, SWE wiederholt.
-	Weglassen der UseCase-Spezifikationen war schlecht, OpSpecs leichter verzichtbar.
	Beispiele in der Vorlesung waren zu einfach, besser größeres als Aufgabe stellen und in der Vorlesung entwickeln! Nicht einfach etwas Fertiges präsentieren.
+	Oberflächenprototyp war in Implementierung gut verwendbar, da schon fertig.
!	Oberfläche schon im 1. Semester fertigstellen!
!	Technisches Klassendiagramm schon hier machen, um Einarbeitung abschätzen zu können
!	Programmierstil schon im 1. Semester behandeln, da über die VL-freie Zeit viel codiert wird.
!	XP-Gruppen schon früh mit Qualitätsanforderungen konfrontieren!
!	XP muss unbedingt auch mit (rollierendem) Terminplan arbeiten!
!	Weniger Plenumstermine, mehr Rücksprachezeit
!	Oder im Plenum auf konkrete Probleme einzelner Gruppen eingehen, z.B. 14-täglich

Meilensteine

2 Einarbeitungsnachweise

-	Die echten Probleme kamen erst mit dem 1. Durchstich mit echten 3 Schichten!

3 Entwurf: Lg -Klassendiagramm + Lg-Operationen

4 Erster Durchstich: Objektverwaltung 2 Klassen über 3 Schichten + Testtreiber

+	Früher Durchstich war gut!
-	Sollte nicht perfekt sein müssen.
!! !! !!	3-Schicht-Durchstich 2 Klassen mit Objektverbindungen mit Transaktionsmanagement und MulTEx besser als 1. Meilenstein statt Einarbeitungsnachweisen, zeigt schnell Architekturprobleme. Architektur muss zu Anfang des 2. Semesters vorliegen!

6 Zweiter Durchstich: Eine Objektverbindungsverwaltung 3 Schichten, mit Testtreibern

-	Nicht bei jedem Meilenstein Testtreiber fordern!

8 Dritter Durchstich: Objektverbindungsverwaltung 3 Klassen, 3 Schichten, mit Testtreibern

11 Integrationstest

12/13 Präsentation Gesamtsystem öffentlich

14 Abnahme

!	Erwartungen des Dozenten bei der Abnahme vorher besser erklären.
	„Alle Dokumente auf Papier“ vorher besser erklären.
	„Worauf sind Sie besonders stolz?“-Frage vorher besser erklären.

Vorlesung

	Stoffauswahl: Was interessant?
-	Stoffauswahl: Was überflüssig? Mapping von Assoziationen
	Stoffauswahl: Was fehlte?
	Vorlesungsstil?
!	Präsentationen einzelner Gruppen zu Techniken wären schön zur Abrundung des Wissens, z.B. Hibernate vs. JDO vs. Prevayler, kann auch am Ende des Semesters kommen. Mit Pluspunkten statt zwingend für alle Gruppen.
!	Persistenz kann, wenn gut gekapselt, nach hinten geschoben werden.
!	Im Plenum auf konkrete Probleme einzelner Gruppen eingehen (FAQ)
!	Öffentliches Forum wäre gut, einzelne Gruppen hatten sich selbst eines eingerichtet.
!	Testmethodik (Black/White Box, Testabdeckungen) interessanter als JUnit-Einführung
!	Liste interessanter Themen anfangs zur Auswahl stellen.
+	Programmierstil „Möglichst final, initialisiert, lokal“ war gut, hat Fehler aufgedeckt! Schon im Vorsemester bringen!

Rücksprachen

!	Gruppen sollen Termine, die sie nicht wahrnehmen, bekanntgeben, damit sie von anderen genutzt werden können.
!	Oder einenTermin für ungeplante Rücksprachen offenhalten!
!	Rückkopplung nach Meilenstein (verbale Bewertung, keine Note) gewünscht!
+	Fachliche Rückkopplung in den Rücksprachen war gut

Gruppenarbeit

!	Jeder Gruppe ist ein Projektleiter zu empfehlen! Bei BugTracker hat er die Logikschicht gemacht, kannte dadurch das gesamte System.
!	LessEntropy hat auch gute Erfahrung mit Terminmanagement gemacht.

Techniken

!	SourceForge-Plattform Gforge.org: Mailinglisten, Bugtracking, Versionsverwaltung, Feature-Request, Forum auf TFH-Server bereitstellen, für Folgesemester stehen lassen.

Einsatz von CVS

+	Ohne das ging es nicht.
!	Schon im Vorsemester propagieren!

Einsatz von OR-Mapping-Bibliotheken

+	Hibernate gut dokumentiert, aussagefähige Laufzeit-Exceptions (verwendet Exception-Chaining)

Einsatz von Struts

+	Idee gut
-	Dokus zu trivial, teils muss man in die Quellen schauen
-	Zu viele Wege möglich, keine Literatur zu Struts-Architektur
-	Fehlerbehandlung in Struts schlecht, manchmal keinerlei Reaktion auf Benutzeraktion
+	Ist aber Mainstream, JSF würde JavaScript verlangen!

Einsatz von MulTEx

!	Muss von Anfang an eingesetzt werden, sonst kein Nutzen. Mehr Zeit in Vorlesung dafür!
!	Empfehlungen für Logger geben!
!	Stack Traces sollten nicht an Benutzer gehen
+	sind aber für Entwickler sehr interessant

Einsatz von JUnit-Testtreibersuite

~	Testtreiber wurden oft stiefmütterlich behandelt.
!	Vielleicht eigener Meilenstein „Nur Testtreiber“ besser.