Abschlussbesprechung Software-Projekt II (Projekt-Realisierung) bei Prof. Knabe im WS 2011

Hier die in der Abschlussbesprechung am 08.02.2012 geäußerten Meinungen der Student(inn)en, gegliedert nach Themengebieten und Meilensteinen.
Legende: + positiv, – negativ bewertet, ! Wunsch, ~ teils/teils

Vorsemester Analysephase/Durchstiche

?	Wäre Extreme Programming (XP) oder Scrum besser? Warte auf Erfahrung von Hr. Haschemi. Meine eigenen Erfahrungen mit XP für einzelne Teams liefen auf zu langsamen Projektfortschritt und zu geringe Qualität hinaus.

Meilensteine

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

	Rich Domain Model?
	Anemic Domain Model?
~	TMS: Mischkonzept: Geschäftsregeln mit Prüfungen in Model-Klassen, aber Persistenzzugriffe nur über GenericDAO. Teilweise umständlich, weil man sich immer das DAO holen musste.
~	RatioApocha: Mischkonzept: Geschäftsregeln mit Prüfungen in Model-Klassen, aber Persistenzzugriffe nur über Generic Gateway. Probleme: Man musste sich um mehr kümmern, als versprochen war.
+	Belegsystem: Mischkonzept: Möglichst reiche Model-Klassen, aber ohne Zugriff aus Persistenz. Waren zufrieden damit.

6 1. Durchstich OV + JUnit-Tests + PD Gesamtsystem + Zuständigkeitsverteilung

-	Knabe: Zuständigkeitsverteilung übersehen.
+	Belegsystem: Sinnvoll
!	Testsuite würde als eigener Meilenstein mehr gewürdigt werden.

7 Build-Prozess mit Testsuite-Ausführung mit Maven

!	RatioApocha: Wäre noch früher besser gewesen, da es umständlich war, eine bestehende Lösung nach Maven umzubauen. Möglichkeiten nicht ausgenutzt.
+	Belegsystem: Maven wegen Tapestry-Archetype von Anfang an eingesetzt. Sinnvoll zur JAR-Besorgung.
+	TMS: Wicket-Maven-Archetyp eingesetzt, keine großen Probleme. Source-Download genutzt.

8 Zentrales Ausnahmemelden

+	RatioApocha: Nach dem Verständnis ließ es sich gut und umfangreich einsetzen, auch in der einfachsten Variante Template Method Pattern.
+	Belegsystem: Nach der Einrichtung und Anpassung hat es trotz anfänglicher Skepsis gut funktioniert.

10 2. Durchstich: Objektverbindungsverwaltung + Suchen Objektmenge + Testsuite

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

~	Belegsystem: Bringt nicht Neues gegenüber vorigem Meilenstein. Knabe: Dient dem abrechenbaren Projektfortschritt für alle Teams.

14 Integrationstest

15 Präsentation Gesamtsystem öffentlich

+	TMS: Notwendig
-	Belegsystem: Zu kurz, für Technikwürdigung wären 30 Minuten nötig.
+	RatioApocha (JSF): 15 Minuten haben ausgereicht.

17 Abnahme

Unterricht (aus Übungszeit herausgeschnitten)

+	Sinnvoll? Inhaltlich hätte es man sich aneignen können, aber die Dozentenerwartungen wurden dadurch klarer.
	Stoffauswahl: Was interessant?
-	Stoffauswahl: Was überflüssig? OR-Mapping bekannt aus SWE
!	Stoffauswahl: Was fehlte? Mehr praktische Beispiele, z.B. wie Rich Domain Model aussieht. Wie Maven-Projekt aussieht.
-	Unterrichtsstil? Materialien abgelesen.
!	Materialien auf meiner SWP-II-Seite? Wäre gut, die Lücken des Lückenskripts danach zur Verfügung zu stellen.
!	Stärker auf die zentralen Themen konzentrieren: Transaktionen, Zentrales Ausnahmemelden, Build-Prozess, Testsuite.

Rücksprachen

+	Meilensteinrhythmus alle 2 Wochen? OK
~	Anwesenheitspflicht bei Rücksprachen? TMS: Sinnvoll, danach wurde gemeinsam am Projekt gearbeitet. RatioApocha: Es müssen nicht immer alle aus dem Team da sein, aber es dürfen nicht immer dieselben fehlen.

Gruppenarbeit

+	Selbstauswahl der Teammitglieder? Gut
+	Gruppengröße 4-5? Gute Größe.
+	Projektleiter? Galapagos: Nicht nötig, da Erfahrene automatisch in diese Rolle rutschen.

Techniken

Einsatz von Subversion vs. Assembla (Versionierung+Ticketing) vs. RedMine

~	Belegsystem: Subversion, eigene Tasklisten per Mail ausgetauscht, war noch handhabbar. Ticketingsystem wäre besser.
+	TMS: Versionierung und Ticketing von Assembla benutzt. War in der Endphase sehr gut. Gut sichtbar, was noch offen ist.
-	Galapagos: RedMine und externes Subversion-Repository arbeiteten nicht zusammen, daher nur Subversion benutzt.
~	RatioApocha: Nur Subversion auf Assembla benutzt, aber Ticketing nicht ausgenutzt. Visualisierung war gut, benötigte aber zu viel Zeit.

Einsatz von Hibernate (OR-Mapper)

+	Keinerlei Probleme.

Einsatz eines Web-Frameworks

~	Wicket? TMS: Obwohl Neuland, ganz OK. Dokumentation unterirdisch. Community zu klein.
~	JSF/PrimeFaces? RatioApocha: Sieht gut aus, aber sehr trickreich. Einarbeitungszeit nötig. Dokumention und Community OK.
+	Tapestry? Belegsystem: Tutorials mit guter Dokumentation, viele Beispiele auf Jumpstart, viele Problemlösungen schon vorhanden, aber man muss sie auch finden.

Einsatz von MulTEx

+	Belegsystem: Mit eigener Erweiterung sehr gut. Geprüfte Ausnahmen wären besser gewesen.

Einsatz von JUnit-Testtreibersuite

+	Galapagos: Sinnvoll, besonders bei Test Driven Development
~	TMS: Durch späte Erstellung der Testfälle haben sie kaum zur Produktivität beigetragen. Waren aber konzeptionell interessant.
~	Belegsystem: Auch Tests zu spät entwickelt, daher auch zu wenig Nutzen.
-	RatioApocha: Testsuite erst für größere Systeme sinnvoll.
+	Ein Tester für Alles (RatioApocha)? "Getestete" Programmierer fanden das gut.
~	Jeder testet eigene Units? Belegsystem: Nachträgliche Tests besser durch andere Person! TMS: Eigentest kann besser Schwachstellen aufdecken. Unabhängiger Tester geht unbefangener ran.

Einsatz von

Studiengang

-	Abnahme während der Klausuren problematisch.