# Abschlußbewertung

Projekt im Rahmen der Software-Projekt Vorlesung an der TFH-Berlin in den Semestern WS02/03, SS03

**Gruppe 5 - ECDL**

Kurzbeschreibung des Projektes
Ziel dieser 2-semestrigen Veranstaltung war die Erstellung eines Systems mit dem Prüfungen für den Europäischen Computerführerschein simuliert werden können. Dazu ist es neben der Prüfungssimulation an sich auch notwendig eine Management Applikation zu erstellen die die Benutzerverwaltung und die Fachliche Verwaltung enthält. In der Benutzerverwaltung kann man Kurse mit Studenten und auch Lehrer und administrative Mitarbeiter anlegen. Kursen kann man Prüfungen zuordnen. Zur fachlichen Verwaltung gehören das Anlegen von Prüfungsfragen und die Untergliederung in verschiedenen Fragenmodule und Kapitel, sowie die Prüfungsverwaltung in der Prüfungen aus vorhandenen Fragen-Kapiteln zusammengestellt werden können.

Hauptarbeitsfelder der Gruppenmitglieder:

| | |
|---|---|
| Norbert Gocht | LgSchicht, Testreiber LG, RMI-Kommunikation |
| Claudia Getzlaff | Ui-Schicht (Management-Anwendung), I18N |
| Daniel Distelrath | DB-Schicht, Testtreiber DB |
| Wiebke Hurrelmann | UiSchicht (ExamClient+Fragenverwaltung), RMI-Kommunikation |

verwendete Technologien:

| | |
|---|---|
| OpenSource | + Software ist grundsätzlich „frei“ verfügbar<br>o Achtung, unterliegt ebenso Lizenzenbestimmungen<br>- Dokumentation ist u.U. knapp oder unfrei, Support ist meist frei (mailinglists), doch damit nicht garantiert, Ansprüche z.B. auf fehlerfreie Funktion gibt es hier nicht |
| Generatoren | + nehmen einem oft das grobe unproblematische ab<br>o irgendwo findet jede Parametrisierung (und diese gilt es oft genug vorher zu erstellen, zb UML in Together, XML Schema für source-generators, Klassen für mapping-generators, tags für javadoc, .....) ihre Grenzen und zumindest für das Feine oder Optimale muss meist selbst nachbearbeitet werden. |
| Hibernate2 | für das OR-Mapping – erst in Castor eingearbeitet und dann auf Hibernate umgeschwenkt das Castor gebrauchte Funktionalität nicht bot<br>+ Hibernate ist im Vergleich zu Castor sehr gut dokumentiert<br>+ Zur Funktion von Hibernate bedarf es keiner bidirektiv assoziierter Objekte, bidirektiv ist für Hibernate etwas Besonderes, uni ist Standard und dies ohne Nachteile für die queries!<br>+ Hibernate unterstützt zahlreiche Java-Idiome und einen fein-granulierten Klassenentwurf. Daneben ist an vielen Stellen ein Erweiterbarkeit oder Anpassung möglich. Nicht zuletzt bietet es |

| | |
|---|---|
| | verschiedenste Formen von Optimierung (versch. Caching-Mechanismen, versch. Techniken für Lazy-Loading). Unterstützt auch die Anwendung in verschidenen application servers.<br>+ Hibernate bietet eine mächtige objektorientierte Abfragesprache HQL<br>+ Hibernate bietet Tools zur Unterstützung bei Erstellung von mapping und ddls (sogar für verschiedene SQL Dialekte)<br>o Achtung, mit der Verwendung von SQL keywords als identifier in queries; und auch wo case-insensitiv draufsteht ist manchesmal case-sensitiv interpretiert bzw. dahinter („back-end“)<br>- Castors Dokumentation ist kurz, vage (Funktion explizit nur für bidirektiv asoziierte Objekte garantiert)<br>+ Castor JDO/XML integriert XML-Java data binding mit OR-Mapping Funktionalität<br>+ Tool castordoclet unterstützt bei ddls (SQL Anweisungen für Tabellen etc.)<br>- Castor JDO ist eine unglückliche Namensgebung: entspricht nicht Suns JDO<br>- Castor unterstützte keine polymorphen Listen (inzwischen steht dies nicht mehr explizit unter Rubrik about/todo auf der Castor Website) |
| PostgreSQL | als Datenbank unter Windows mit cygWin<br>- das Zusammenspiel von IPC-Daemon und verschiedenen Windows-Versionen ist „nicht-nativ“ und damit fehleranfällig<br>- Cygwin Version bietet stark begrenzte Locale Unterstützung (spielt z. B. bei order by oder case-insensitiver Suche bei ilike oder Funktionen wie upper()/lower() zusammen mit unseren dt. Umlauten eine Rolle)<br>+ Postgres bietet je nach Compilierung Unterstützung für „multibyte“ character sets, db. Zb. Unicode und damit dt. Umlaute (Cygwin Version ist multibyte-enabled, die meisten Binär-Paketversion verschiedener Linux-Distributionen ebenso)<br>+ Postgres ist schnell und bietet viele Features eines professionellen DBMS, z. B. Referentielle Integrität, stored procedures, ... und es gibt ein schönes Windows Admin Tool PgAdmin. |
| Log4J | für das Logging – zum dynamischen klassenbasierten ausgeben der Log-Meldungen mit verschiedenen Log-Leveln |
| RMI | für die Kommunikation zwischen Hauptapplikation und Exam-Client<br>o Achtung, bzgl. der java security policies mit RMI (oder auch mit JDBC) |
| I18N | mithilfe von Properties-File und konstanten Resource-Keys |
| JavaHelp | für die Onlinehilfe |
| JasperReports | für die Anzeige und den Export der Ergebnissauswertungen<br>o visueller JasperReport Editor „iReport“ setzt Kenntnis von |

JasperReports voraus
- ausführliche Doku ist unfrei

Swing — für die Oberfläche der beiden Applikationen

ANT — zum Deployen und Kompilieren der Dateien, zum Anlegen der Datenbank, für weitere Administrative Aufgaben
+ plattform-neutral, da java-basiert
+ viele tasks zur Automatisierung verschiedenster Aufgaben

JUnit — für die Testtreiber
+ steigert das Codevertrauen
+ „test-driven" – zweckorientierte - Entwicklung, vgl. Johannes Link, dPunkt Verlag
o Achtung, bei sonderbaren Problemen, z. B. mit jakarta common-logging (die auf den ersten Blick nicht nach Classloader Problemen aussehen) hilft erst einmal ein Blick in die Junit FAQ ... und dann Einträge in excluded.properties

<u>verwendete Softwareprodukte:</u>

Together — in der Analysephase
++ Reverse Engineering, Codegenerierung, tag-basierte Metainformationen in Code
+- umfangreiche (gestufte) Konfigurationsmöglichkeiten

Eclipse — in der Implementierungsphase mit verschiedenen Plugins: jalopy, dbEdit, generate-rmi, call-hierarchy, implementors, ...
++ Refactoring, „JIT-Fehleranzeige", Code-Completion, Codegenerierung, Plugins ..... unverzichtbare IDE!!!
+ DbEdit erlaubt direktes Editieren in Db Tabellen (über JDBC)
- DbEdit mangelt es bisher manchmal etwas an „Aktualität" (dann schafft ein Blick auf den Tabelleninhalt mit PgAdmin Gewissheit) und bisher verwendet der Plugin für Tabellen und Constraints ein identisches Icon

CVS — zur Versionsverwaltung - unverzichtbar
+ sehr schön in Eclipse integriert
+ Versionskontrolle und concurrent Zugriff
o bietet keine semantischen Checks und ersetzt nicht Absprachen bzgl. der gemeinsamen konfliktfreien Bearbeitung des Codes
- pcx22 unterstützte anfangs keine Erstellung von Versionen

PHP-Nuke — Bietet eine Community-Platform und unterstützt damit Kommunikation
+ integriert Standard PHP BBS „phpBB"
+ relativ schnell installiert
+ sehr gute Erweiterungs- und Anpassungsmöglichkeiten
+ freie Software

Entwicklungsstatus:
Zusammenfassend kann gesagt werden das die Muß-Kriterien aus dem Pflichtenheft alle implementiert wurden. Aufgrund des Umfangs des Projektes war es allerdings nicht möglich die Kann-Kriterien umzusetzen.

Erfahrungen die wir gemacht haben
Im großen und ganzen funktionierte die Teamarbeit ganz gut. Einige schwierige Momente gab es aber die wurden durch die regelmäßigen Projektmeetings abgeschwächt oder ganz und gar aus dem Weg geräumt.

Kommunikation war bei uns und ist wahrscheinlich gleichermaßen bei allen anderen Teamprojekten der schwierigste Punkt den es zu meistern gilt. Nachdem unsere teilweise sehr umfangreiche Kommunikation per E-Mail schnell unübersichtlich wurde – stiegen wir auf ein Webbasiertes Forum um. Hier richteten wir eigene Themenbereiche für die verschiedenen Schichten, ToDo Listen, Verbesserungswünsche und Bugs ein. Damit erreichten wir Übersichtlichkeit in der Kommunikation – und keine Nachricht ging in der Menge der E-Mails unter.

Das Projekt war insgesamt zu groß – eine aktuelle Statistik verdeutlicht dies:

| | |
|---|---|
| Anzahl der Klassen: | ca. 200 |
| Lines of Code: | ca 30.000 |

Unser Projekt war außerdem zu unflexibel gewählt. Wir konnten keinen Teil weglassen da sonst das ganze Projekt keinen Sinn mehr gemacht hätte.

Was würden wir in Zukunft anders machen
Das Projekt muss auf jeden Fall besser skalierbar sein – die Muss-Kriterien sollten so klein wie möglich gehalten werden – alles was nicht unbedingt nötig ist sollte in die Kann-Kriterien aufgenommen werden. Das Projekt sollte von Umfang her relativ klein sein und einfach erscheinen – dann ist es während eines Semesters zu schaffen ohne in nicht enden wollende Nachtschichten auszuarten.

Auch didaktisch ist es sinnvoll eher kleine dafür durchdachte und anspruchsvolle Projekte zu wählen – und nicht welche die in reine Schreibarbeit ausarten und wo keine Zeit bleibt Sachen zu verbessern, weil alles fertig werden muss.

Die Implementierungsphase sollte früher beginnen. Vielleicht sollte im ersten Semester schon damit begonnen werden an die Implementierung zu denken und kleine Prototypen von Teilen der Applikation herzustellen.

Auf jeden Fall sollte die Kommunikation zwischen den einzelnen Komponenten des Projektes schon sehr früh stehen und ausprobiert werden – da dies im späteren Stadium der Implementierung fast einen Stillstand der Entwicklung bedeutet wenn die Kommunikation nicht funktioniert.