Qualitätssicherungskonzept

von Vladislava Nadova

1. Dokumentationskonzept

1.1 Javadoc: Die Quellcodedokumentation wird sich an der Javadoc halten. Javadoc erstellt aus Java-Quellcode Dokumentationen in HTML-Format und parst die Quelltexte nach allen Kommentaren, die mit /** beginnen. Diese Kommentare stehen vor Beginn jeder Klasse, vor jeder Klassenvariablen und -methode. Jeder Javadoc-Kommentar wird nach darin enthaltenen Javadoc-Tags (beginnend mit `@` oder `{@}`) durchgesucht. Sie enthalten Metadaten mit dokumentativem Charakter über das jeweilige Symbol:

Tag und Parameter	Ausgabe in JavaDoc	Verwendung in
`@author` name	Beschreibt den Autor	Klasse, Interface
`@version` version	Erzeugt einen Versionseintrag. Maximal einmal pro Klasse oder Interface	Klasse, Interface
`@param` name beschreibung	Parameterbeschreibung einer Methode	Methode
`@return` beschreibung	Beschreibung des Returnwerts einer Methode	Methode
`@exception` classname beschreibung `@throws` classname beschreibung	Beschreibung einer Exception, die von dieser Methode geworfen werden kann	Methode

. . .

Verantwortliche für die Javadoc sind die Implementierer.

1.2 Java-Code-Regeln: Wichtigste Punkte, die unbedingt von den Implementierern zu beachten sind:

- Klassenamen beginnen mit einem Großbuchstaben.
- Namen von Klassen, Variablen und Methoden sollen die entsprechende Anwendung möglichst verständlich beschreiben.
- Methoden und Variablen werden klein geschrieben, in den Fällen, wo mehrere Wörter auftauchen, beginnt jedes folgende Wort mit einem Großbuchstaben.

- Die zu vergebenden Namen sind auf Englisch zu schreiben.
- Nicht-triviale Code-Zeilen (z.B. i=i+1 // i wird um 1 erhöht) sind mit Zeilenkommentaren, beginnend mit „//“ zu versehen.
- Bei Exceptions ist zu beschreiben, wann sie auftreten können.
- Umlaute werden nicht verwendet, sondern die entsprechende Umschreibung.
- Übersichtliche und einheitliche Einrückung: Zusammengehörige geschleifte Klammern stehen untereinander, in derselben Spalte.
- Eine get-Methode hat keine Parameter und gibt immer einen Wert zurück, Lesezugriff auf eine bestimmte Eigenschaft wird gestattet.Eine set-Methode hat immer einen Parameter und gibt nie einen Wert zurück, Schreibzugriff wird gestattet.

1.3 Dokumentation im SVN: Bei Dateiänderungen sind im svn Kommentare zu den jeweiligen Stellen zu vergeben. Die Dokumentation im SVN ist vom gesamten Team zu beachten.

1.4 Benutzerdokumentation: Zu einer Applikation gehört eine gut beschriebene verständliche Dokumentation, die den Nutzern, unter anderem auch Programmierern, helfen kann, sich leicht in den Applikationsanwendungen und Funktionalitäten zu orientieren. Informationen über die implementierten Module und Funktionen der Klassen, sowie Schnittstellenübersicht sollen dokumentiert werden. Benutzerdokumentation wir vom Verantwortlichen für Dokumentation gepflegt.

2. Testkonzept

JUnit ist ein Framework für Java, das ermöglicht, sog. Units (Klassen oder Methoden) auf Fehler automatisiert und gezielt zu testen. Für die Organisation der Testfälle steht das Interface junit.framework.Test zur Verfügung. Mit Assert-Methoden werden Variablen getestet, bei Fehlern werden Exceptions geworfen. Jeder geschriebene Test erbt von der Klasse junit.framework.TestCase und implementiert damit junit.framework.Test. Die sog. Testsuiten führen eine Anzahl von TestCases aus. JUnit stellt den TestRunner zur Verfügung, um alle Tests durchführen zu können.

2.1 Komponententest

Der Komponententest bezieht sich auf Klassen und Methoden, zu denen Testbeispiele und mögliche Anwendungsfälle vorgesehen werden. Die Klassen und die Methoden ihrerseits sollen dann diese Anforderungen erfüllen. Die entsprechenden Tests (TestCase) zu einzelnen Klassen werden als Testklassen erzeugt und in den Testsuiten im vorgegebenen Kontext zusammengefasst, das heißt, die Klassen eines Pakets haben entsprechende

Testklassen in einer Testsuite. Testklassennamen bzw. Testpaketnamen beschreiben die zu testenden Klassen bzw. Pakete wie folgt: TestKlassenname bzw. TestPaketname.
Jedes PP-Team ist für die Quellcodedokumentation verantwortlich, die Testorganisation wird vom Testverantwortlichen unterstützt. Für die korrekte Dokumentation und Beschreibung der Pakete ist der Verantwortliche für Qualitätssicherheit und Dokumentation zuständig. Diese Anforderungen sind wöchentlich zu erfüllen und zu kontrollieren.
Das Ergebnis der einzelnen schrittweise erstellten Implementierungsaufträge und Komponenten wird direkt über das OLAT-System angezeigt.

2.2 Integrationstest

Die Implementierung wird in Storys, Implementierungsaufträge, aufgeteilt, die wöchentlich lauffähige Release-Versionen der zu erstellenden Software zusammenstellen. Deswegen ist es sinnvoll von Anfang an, wichtige Zusammenhänge der Klassen zu erkennen und ihr korrektes Zusammenspiel in einem Integrationstest zu prüfen. Erst alles programmieren und dann zusammenfügen ist zu riskant, da es durchaus möglich ist, dass nichts danach nach Vorstellung zusammen funktioniert. Deswegen werden Klassen nach Geschäftsprozessfunktionalität integriert und lauffähige Release-Versionen erweitert.

2.3 Systemtest

Um zu überprüfen, ob das Gesamtsystem die im Pflichtenheft und Entwurfsbeschreibung festgelegten Anforderungen erfüllt, wird die Software einem Systemtest unterworfen. Die Rolle des Nutzers ist für diesen Test aus zentraler Bedeutung. Verschiedene Anwendungssichten müssen vorgesehen werden: Es sollen die an den Prozessen beteiligten Rollen durchgespielt werden, aus der Sicht des Studenten als Nutzer, des Professors und des Prüfungsamtes.

2.4 Abnahmetest

Am Ende der Implementierung wird einen Abnahmetest von dem Auftraggeber durchgeführt, ob die vertraglich vereinbarten Anforderungen und Funktionalitäten des Endproduktes erfüllt wurden.

3. Planung und Organisation

3.1 Dokumentation/Tests

Dokumentation und Tests sind wöchentlich zu kontrollieren. Angeforderte Versionierung wird aktualisiert und auf den Internet-Seiten der Gruppe im Rahmen des Prototypsfortschrittes präsentiert. Die Implementierung erfolgt vom 26.05.08 bis 30.06.08.

3.2 Planung der Storys

- Woche 1. Im Settingsmenü des OLAT-Portals wird das Profilformular um die zusätzlichen Felder Studiengang und Studserv-Account erweitert. Des Weiteren wird die Verbindung zwischen dem Prüfungsamt und dem Nutzer erstellt, der einen Antrag auf seine elektronische Studentenakte erstellt hat. Der Tab „Prüfungsverwaltung“ im Account des Prüfungsamtes wird um zwei Funktionen erweitert, die es ermöglichen, eine Liste mit Anträgen und eine Liste aller vorhandenen ESAs anzuzeigen. Im Account des Professors wird diesem Tab nur die Funktion hinzugefügt, die die Liste mit den ESAs zur Verfügung stellt.

  Verantwortlich M.Stuber, C.Wang
- Woche 2. Neue Datenbanktabellen erstellen, die für die Speicherung der Daten vorgesehen sind: Prüfungseinträge, Studentendaten, Noten, Ergänzungen usw. In dieser Woche soll die Studentenakteansicht mit den individuellen Studentendaten realisiert werden, wobei die unterschiedlichen Darstellungen der ESA beabsichtigt werden sollen: der Student darf sich die ESA nur ansehen, der Professor darf zusätzlich Ergänzungen zu Prüfungen und allgemeine Ergänzungen schreiben und das Prüfungsamt verfügt über alle Zugriffsrechte.

  Verantwortlich D.Kotlarz, Hung Do
- Woche 3. Detaillierte Implementierung bzgl. des UML-Diagramms, zusätzliche Funktionserweiterungen, mit denen es möglich sein soll, Prüfungseinträge und Ergänzungen hinzuzufügen. Der Fokus fällt auf Funktionen im Bereich der angezeigten ESA im Prüfungsamt- bzw. Professoraccount, speziell wird ein Funktionsmenü rechts erzeugt und entsprechend angepasst.

  Verantwortlich C.Wang, M.Stuber
- Woche 4. Editierfunktionen werden ermöglicht und ihre vorgesehene Speicherung wird angepasst.

  Verantwortlich Hung Do, V.Nadova
- Woche 5. Korrekturarbeiten, Feinheiten.

  Verantwortlich V.Nadova, D.Kotlarz