Aufgabenblatt 7 - Qualitätssicherungskonzept

Inhaltsverzeichnis

1 Dokumentationskonzept

Das Dokumentationskonzept legt die Standarts fest die bei der Erstellung von Artefakten eingehalten werden müssen. Eine einheitliche Linie sorgt für leichteres Verständnis und fördert somit die Produktivität.

1.1 Code-Dokumentation

- Die Namen von Variablen, Konstanten, Methoden und Klassen, Dateien und Ordnern (im folgenden Bezeichner) sind so zu wählen das diese selbsterklärend sind.

```
  Falsch:
nou = 0;

Richtig:
numberOfUsers = 0;
```

- Als Sprache für die Bezeichner wird Englisch gewählt. Dies schafft Übereinstimmung mit den verwendeten Systemen (OntoWiki, Erfurt, Zend) und macht den Code auch für internationale Entwickler lesbar.
- Bezeichner für Klassen beginnen immer mit Großbuchstaben; besteht der Klassenname aus einem zusammengesetzten Wort, beginnt jedes weitere Wort auch wieder mit einem Großbuchstaben. Es werden keine Unterstriche zur Trennung von Wörtern verwendet.

```
class OneNewClass ...
```

- Variablennamen beginnen mit Kleinbuchstaben, ansonsten wie bei Klassen. Ist in der Variable eine Klasseninstanz enthalten, so sollte der Klassenname mit enthalten sein.

```
counter = 1; myRequestDispatcher = new RequestDispatcher();
```

- Konstanten werden vollständig in Großbuchstaben geschrieben
- Methoden-Namen werden wie Variablennamen klein geschrieben, jedes weitere Wort beginnt wieder mit einem Großbuchstaben. Spezielle Methoden zum Lesen oder Schreiben von Attribute folgen der Form `getAttribute()` bzw. `setAttribute()`. Methoden zur Abfrage Bool'scher Attribute werden mit `isAttribute()` benannt.

  Desweiteren gelten Folgende Regeln zur Strukturierung des Codes
- Öffnende und Schließende Klammern, die einen Codeblock umfassen, werden in eine eigene Zeile geschrieben. Außerdem wird der Code in diesem Block eingerückt.

```
Richtig:
function getValue()
{
     return value;
}
```

- Zur Einrückung werden (je Einrückungstiefe) 4 Leerzeichen benutzt. Bei maximal 80 Zeichen wird ein Zeilenumbruch vorgenommen.
- Jeder Befehl steht in einer eigenen Zeile.

```
 Falsch:
echo "hi"; die;
```

- Bei Algorithmen, die sich nicht selbst erklären, werden wichtige Codestücke mit Kommentaren erklärt. Bei einzeiligen Kommentar mit `//`, Bei mehreren Zeilen mit `/*` und `*/`.
- alle in phpdoc zur Verfügung stehenden Kommentare sollten auch genutzt werden.
- beim Commit über SVN ist die Kommentarfunktion zu nutzen (kurze Beschreibung was geändert wurde).

1.2 Kundendokumentation

Die Kundendokumentation wird (neben allen entstandenen Dokumenten wie Lastenheft, Glossar, Pflichtenheft und einer Entwurfsdokumention) ein gesondertes Benutzerhandbuch enthalten. Dieses schildert die Installation und Einrichtung sowie Bedienung des Produkts. Die Bedienung soll auch im Produkt über Hilfefunktionen erläutert werden.

2 Organisatorische Festlegungen

2.1 Vorgehensweisen und Verantwortlichkeiten

Alle Programmierer sind dazu angehalten, sich in die Dokumentationsrichtlinien einzuarbeiten und diese bei der Programmierung einzuhalten.

Der Verantwortliche für Dokumentation und Qualitätssicherung sichert ab, dass die Dokumentationsrichtlinien eingehalten werden. Er überprüft dies jeweils vor jedem neuen Release.

Der Verantworliche für Tests weist die Programmierer in die Einrichtung von PHPUnit Tests und Selenium ein. Er überwacht die Tests und prüft vor Releases ob alle Tests ordnungsgemäß durchgeführt wurden und Fehler behoben wurden.

2.2 Qualitätsstandards

Fehler bzw. -korrekturen im Programm werden mit einem Bugtracker festgehalten. Für die Beseitigung dieser Fehler ist der jeweilige Programmier, von dem der fehlerhafte Programmteil geschrieben wurde verantwortlich. Abweichungen hiervon müssen durch Rücksprachen mit dem verantwortlichen Programmierer abgesprochen werden.

3 Testkonzept

3.1 Allgemeines

Um möglichst robuste und fehlerfreie Software zu erstellen, ist eine umfassende Qualitätssicherung unabdingbar. Ein wichtiger Bestandteil davon ist der Softwaretest. Im Testkonzept werden dabei organisatorische, programmiertechnische und konzeptionelle Aspekte betrachtet.

3.2 Testplan

Da es sich beim vorliegenden Projekt um eine Erweiterung eines existierenden Produktes auf Plugin-Basis handelt, ist der Auswahlrahmen des Testkonzeptes eng eingegrenzt. Da als Vorgehensweise Extreme Programming gewählt wurde, sind Unit-Tests eine feste Anforderung, diese müssen sogar vor der Erstellung einer Klasse oder eines Moduls konzeptioniert und erstellt werden. Bei OntoWiki handelt es sich um eine Anwendung auf Basis von PHP und JavaScript. Folglich ist die Erweiterung in den entsprechenden Programmiersprachen umzusetzen und auch Unit Tests können somit ausschließlich auf Basis von PHP Unit umgesetzt werden. Weiterhin wird zur Überprüfung der Funktionalität der erstellten Oberfläche Selenium als Testframework für grafische Benutzeroberflächen verwendet. Wir haben uns für eine gestaffelte Vorgehensweise, bestehend aus Unit-Tests zur Überprüfung elementarer gekapselter Funktionalitäten von Klassen und Modulen, Integrationstests zur Überprüfung des reibungslosen Zusammenspiels gekapselter Funktionseinheiten, sowie einem anschließenden Systemtest entschieden. Zusätzlich wird in der Phase von Integrations- und Systemtest Selenium eingesetzt um ein reibungsloses Funktionieren der GUI sicherzustellen. Zum Vorgehen ist weiterhin zu bemerken, dass in der Stufe der Komponententests die Testfälle nicht durch die Programmierer der Komponente designt werden dürfen, da diese mit der „erwarteten“ Funktionsweise der Unit zu vertraut sind und deshalb kritische Fälle nicht betrachten oder beachten könnten. Testdokumentation Über alle Testaktivitäten ist Buch zu führen. Dabei muss hervorgehen, in welchem Bearbeitungsstand der Einzelkomponente/des Systems welcher Test genau erfolgt ist und wer diesen durchgeführt hat. Fehler sowie deren Beseitigung sind ebenso festzuhalten.

3.3 Teststufen

Der Testablauf in unserem Projekt wird in vier Phasen gegliedert:

- Der Komponententest
- Der Integrationstest
- Der Systemtest

Der Komponententest: Komponententests werden durch uns auf Basis der Stories des Pair-Programming durchgeführt, da diese kleinste kapselbare Einheiten der Entwicklung darstellen. Für jede Story sind einzelne Funktionalitäten aufgeführt, die welche gleichzeitig die zu testenden Funktionen sind. Jeder Storybearbeiterist für seine eigenen entsprechenden Testfälle verantwortlich. Zu testen sind Eigenschaften wie: Richtigkeit, Vollständigkeit, Fehlertoleranz, Übersichtlichkeit.

3.4 Der Integrationstest

Das korrekte Zusammenspiel der oben im Einzelnen beschriebenen Komponenten ist erneut zu überprüfen durch eine aufeinander abgestimmte Reihe von Einzeltests. Das heißt nach dem Zusammenführen von Einzelkomponenten muss das Testergebnis weiterhin mit den oben erwarteten übereinstimmen. Dabei kann ein Buttom-Up-Verfahren benutzt werden wo bei schon zueinander gehörende Komponenten auf Integrationskompatibilität geprüft werden und dann im folgenden als geschlossene Komponente behandelt werden. Dadurch wird der Aufwand begrenzt. Auch diese Ergebnisse sind in der Testdokumentation nach oben beschriebenem Schema festzuhalten.

zu Testen:

- Kompatibilität mit anderen Komponenten
  - passende Datentypen bei Aufrufen von externen Funktionen
  - Race Conditions
- korrekte Darstellung im Zusammenspiel mit anderen Layoutelementen

3.5 Der Systemtest

Zum Ende der Implementierung vor Abgabe des Projekts ist durch alle Beteiligten ein Systemtest durchzuführen. Dabei müssen zwingend alle im Pflichtenheft beschriebenen Funktionalitäten bezüglich ihrer Umsetzung getestet werden. Auch über diese Ergebnisse ist Buch zu führen, so dass jede einzelne Funktionalität einzeln als funktionstüchtig bestätigt werden kann. Wichtig ist weiterhin, der Test auf ungewöhnliche Eingaben und Benutzungsweisen hin.

Bei allen Tests die JS betreffen, ist Selenium zu nutzen, bei PHP: PHPUnit. Es muss dabei dokumentiert werden, welche Pfade durch die Navigation bei einem Testfall gewählt wurden. Es sind sämtliche Bedienelemente in möglichst vielen Kombinationen zu berücksichtigen, die neu implementiert wurden.