Softwaretechnikpraktikum

Gruppe: swp14-pruef

Qualitätssicherungskonzept

Verfasser:
Steven Giesel, Mirko Schulze,
Daniel Heinze

Abgabe der Lösung zu Aufgabenblatt 3:
20.01.2014

Inhaltsverzeichnis

1 Dokumentationskonzept

Eine gute Dokumentation über alle Bestandteile des Projektes ist ein essenzieller Bestandteil. Dies hilft das Produkt auch später besser zu warten, und zu pflegen. Daher finden sich in diesem Dokument Richtlinien zur Qualitätssicherung wieder.

1.1 Quellcode und Dokumentierung

Im Rahmen der Implementierung des Programms gelten Regeln und Maßnahmen um einen einheitlichen Stil zu erhalten. Dies erhöht die Wartbarkeit und Pflege des Codes.

1.2 Richtlinien für den Quellcode

Im Allgemeinen ist es wichtig, dass durchgängig ein einheitlicher Stil bei allen Teilnehmenden Personen vorhanden ist. So ist es für jede Person gleichermaßen leicht sich in anderen Code einzulesen.
Innerhalb von Funktionen, Kontrollstrukturen und Schleifen wird jeweils um 1 Tabulator eingerückt. Insgesamt sieht man so direkt, welcher Teil wohin gehört. Sind Funktionen, Kontrollstrukturen und Schleifen länger als eine Zeile, so wird eine geschweifte Klammer darum gemacht. Sind Anweisung nach Kontrollstrukturen oder Schleifen nur eine Zeile lang, werden diese eine Zeile darunter mit Tabulator eingerückt ohne geschweifte Klammern aufgeschrieben.
Nach dem Aufruf einer Funktion sollte danach bei einer neuen Zeile angefangen werden. Dies erhöht die Lesbarkeit. Des weiteren ist zu beachten, dass zwischen logischen Abschnitten innerhalb einer Funktion auch eine Leerzeile zu lassen ist. Somit wird sofort ersichtlich, welche Teile innerhalb der Funktion zusammenhängen (zum Beispiel beim Initialisieren von Variablen und den weiteren Abschnitten).
Besitzt eine Funktion eine lange Parameterliste oder sind die Werte für Parameter sehr lang, sollen diese untereinander, statt nebeneinander geschrieben werden. So wird verhindert, dass eine Funktion weit über die sichtbare Zeile hinausgehen.
Da die Standardsprache in der Softwareentwicklung Englisch ist, werden Bezeichner von Variablen und Funktionen auch in Englischer Sprache bezeichnet. Außerdem sollte Bezeichner in Abhängigkeit ihrer Funktion auch solche Namen bekommen.
Dem gängigen Standard vieler Programmiersprachen sind Variablen und Funktionen klein zuschreiben und Konstanten groß. Klassen, sowie auch Interfaces, beginnen mit einem Großbuchstaben, gefolgt von Kleinbuchstaben.
Insgesamt halten wir uns an die Quellcode-Standard des Zend-Frameworks[1]

1.3 Richtlinien zur Dokumentation des Quellcodes

Eine gute Dokumentation ist der wesentliche Bestandteil um sich in Code einzulesen und vertraut zu machen. Dementsprechend sollte hier eine hohe Qualität vorhanden sein. Um die Einarbeitung zu erleichtern, verwenden wir das Software-Dokumentationswerkzeug Doxygen[2]. Dieses erstellt aus dem vorhandenen Code, basierend auf speziellen Kommentaren, Abhängigkeiten und Struktur, eine graphische Dokumentation (HTML, LaTex-Dateien, welche sich in eine PDF konvertieren lassen, und viele weitere) mit UML-

[1] http://framework.zend.com/manual/1.12/de/coding-standard.html
[2] http://www.stack.nl/~dimitri/doxygen/

Diagrammen.
Dies bedeutet auch, dass wir durch die Beschaffenheit des Werkzeugs, jede Funktion, Struktur oder Klasse kommentieren müssen. So sollte oberhalb der Funktion mit speziellen Anweisungen, welche doxygen bereitstellt, das Verhalten der Funktion (Funktion, Parameter, Rückgabetyp, ...) kommentiert werden.
Auch innerhalb von Funktionen sollte kommentiert werden. So können logische Abschnitte schneller erfasst und zusammengefasst werden. Arbeiten mehrere Teammitglieder an einem Abschnitt ist der ToDo-Tag hilfreich, um zu sehen, welche Arbeiten noch ausstehen (@todo).
Die Dokumentation des Quellcodes wird in Englisch erfolgen.

1.4 Benutzerhandbuch

Im späteren Verlauf des Projekts wird für den Projektträger ein Benutzerhandbuch erstellt werden. Dieses ermöglicht ein erstes Einlesen in die Funktionalität des Programms und zeigt seine Möglichkeiten auf. Damit soll das einfache Arbeiten und Verwenden des Programms gewährleistet werden.

2 Testkonzept

Das Wesen eines Softwareprojekts erfordert neben einer sorgfältigen Arbeitsweise immer auch systematisches Testen der entstehenden Bestandteile. Fehler und Unberechenbarkeiten sind während des komplexen und kreativen Prozesses der Softwareentwicklung unausweichlich. Eine gute Projektgestaltung zeichnet sich deshalb durch das frühzeitige Erkennen und Beheben dieser Mängel aus. Durch die richtige Durchführung von Tests wird sichergestellt, dass der Programmablauf in allen Fällen wie gefordert erfolgt. Dieser Aufgabe kann nur durch ein frühzeitig geschaffenes Konzept Rechnung getragen werden.

2.1 Grundzüge des Konzepts

Das Testen muss in erster Linie systematisch stattfinden, denn nur so kann man Lücken in der Qualitätssicherung wirksam vermeiden. Von Beginn an werden die Tests an klar definierten Punkten des Implementierungsprozesses durchgeführt. Dies geschieht auf mehreren Ebenen, angefangen bei der Umsetzung der einzelnen Komponenten. So erfolgen frühe Tests implementierungsnah und autonom durch die verantwortlichen Programmierer. Sind die Überprüfungen auf einer Ebene erfolgreich abgeschlossen und die Implementierung ausreichend fortgeschritten, werden umfassendere Tests veranlasst. Bei Anpassungen wird dieser Prozess erneut durchlaufen.

Entscheidend ist auch die Herangehensweise. Geeignete Werkzeuge führen zu einer partiellen Automatisierung von Testverfahren und damit zu einer merkbaren Effizienzsteigerung. Außerdem helfen sie, auch schwer zu erkennende Fehler aufzudecken. Aufgrund der Verwendung von Zend Framework nutzen wir für die Tests das zugehörige Testframework auf Basis von PHPUnit. Dabei kommt, wie beim Quelltext selbst, GitHub für die Verwaltung zum Einsatz. Zur Prüfung der Weboberfläche könnte Selenium herangezogen werden.

Ein weiterer wichtiger Punkt ist die Standardisierung der Testprozesse. Es werden hierzu gewisse Vorgaben festgelegt, die zum einen den Testenden bei der exakten Durchführung unterstützen und zum anderen eine einfachere Kommunikation gewährleisten sollen. Insbesondere die Fehlerdokumentation ist nach einem genau vorgegebenen Muster vorzunehmen. Das Team wird sich regelmäßig zur Durchführung und zu den Ergebnissen der Tests austauschen. Der Verantwortliche für Tests kontrolliert und koordiniert die auszuführenden Tests sowie die Anwendung der Richtlinien.

2.2 Komponenten- und Integrationstests

Das Testen der einzelnen Klassen in der Implementierung obliegt vollständig den jeweiligen Implementierenden. Dabei wird die Klasse auf eine korrekte Funktionsweise und auf Fehlerfreiheit untersucht. Dabei kommt das Testframework zum Zend Framework zum Einsatz. Als Rahmen für die Komponententests dienen außerdem die Spezifikationen, die den Funktionsumfang der Klasse behandeln, um mögliche Abweichungen früh zu unterbinden. Genauere Vorgaben zu den Tests werden bei Bedarf zu den Teamsitzungen besprochen.

Beim Integrationstest wird überprüft wie sich eine Klasse im Zusammenspiel mit anderen Komponenten verhält. Besonders entscheidend ist dabei, ob die Schnittstellen harmonieren und die Komponenten korrekt miteinander kommunizieren. Im Fehlerfall müssen Klassen angepasst und die Komponententests erneut durchgeführt werden. Integrations-

tests sollten möglichst direkt nach den Komponententests eigenverantwortlich mit den schon vorhandenen Klassen durchgeführt werden.

2.3 Systemtests

Ist die Implementierung soweit fortgeschritten, dass eine autonome Vorabversion der Software erstellt werden kann, wird ein Systemtest vorgenommen. Dies könnte vorzugsweise jeweils zu einem Release-Bundle erfolgen. Beim Systemtest wird erstmals aus Anwendersicht getestet, so dass alle Aspekte des Softwareprodukts geprüft werden müssen. Grundlage dafür bilden die in den Projektdokumenten festgeschriebenen Anforderungen. Besonderer Stellenwert ist auch den Bedingungen des Anwenders und eventuellen Kompatibilitätsproblemen zuzumessen. Ferner wird die Benutzeroberfläche an dieser Stelle zusammenhängend geprüft. Aufgrund des Umfangs dieser Tests erfolgt hierzu eine gezielte Koordinierung im Team. Dabei könnten Checklisten und andere Planungsdokumente von besonderem Nutzen sein.

Als finale Überprüfung des Gesamtprodukts wird dann der sogenannte Abnahmetest durchgeführt. Er stellt eine erweiterte Form zu obigem Systemtests dar. Ziel ist es, die entstandene Software in allen Aspekten umfassend zu erproben und zu bewerten. Dazu sollte eine möglichst realistische Testumgebung geschaffen werden. Hinzu tritt die Prüfung der Dokumentation.

2.4 Dokumentationsrichtlinien

Sämtliche Fehler, die nicht unverzüglich ausgebessert werden können (Syntaxfehler), sind in einem Log-File zu dokumentieren. Zu jedem Test sind zunächst folgende Daten aufzuführen:

- Art bzw. Anlass des Tests
- Testgegenstand
- Name des Testenden und Zeitpunkt

Anschließend sind zu jedem auftretenden Fehler folgende Informationen zu erfassen:

- Fehlerbeschreibung
- Klassifizierung des Fehlers
- Fehlerquelle
- Lösungsmaßnahmen
- Status

Diese Aufzeichnungen sind mit dem Quellcode zusammen zu speichern.

3 Organisatorische Richtlinien

Das Dokumentations- und Testkonzept ist für alle Mitglieder des Projektes verbindlich und ist damit einzuhalten. Bei Verletzungen dieser Regeln ist hauptsächlich der Verantworltliche für Qualitätssicherung dafür zuständig, das jeweilige Gruppenmitglied auf den Verstoß aufmerksam zu machen.
Es werden jede Woche Treffen abgehalten, in denen der aktuelle Stand des Projekts ausgewertet und mögliche auftretende Probleme besprochen werden. Außerdem wird die weitere Verteilung der nächsten Aufgaben festgelegt.
Die Kommunikation innerhalb der Gruppe erfolgt über mehrere Kanäle, wie eine Facebook-Gruppe, Skype, E-Mail und ein Google Forum, mit denen alle Gruppenmitglieder erreicht werden können. Sollten kurzfristig Probleme auftreten, so können diese über E-Mail oder Facebook besprochen werden. Verbindliche Informationen werden generell über Facebook versendet. Außerdem steht das OLAT (genauer das Wiki im OLAT) als Informationssammlung zum Thema jedem offen zum Lesen und Bearbeiten.