Qualitätssicherungskonzept

SWP14-SF

- Christian Blecha -

Inhalt

1. Dokumentationskonzept

Die Dokumentation eines Projektes ist von entscheidender Bedeutung, denn erst dadurch wird es anderen Entwicklern ermöglicht, das Projekt schnell zu erfassen, zu verstehen und Änderungen bzw. Erweiterungen anzufertigen. Dazu ist die Dokumentation in verschiedene Teile unterteilt:

1. die Dokumentation des Quelltextes,
2. die Dokumentation der Änderungen am Projekt,
3. die Dokumentation der Tests und
4. eine externe Dokumentation des Projektes.

1.1 Die Quelltextdokumentation

Die Quelltextdokumentation ist dahingehend notwendig, dass andere Entwickler (aus dem Team oder von anderen Softwarehäusern) sich schnell in den Quellcode einlesen können. Um dies zu gewährleisten wird Englisch als Dokumentationssprache im Quelltext verwendet.

1.1.1 Einrückungen

Einrückungen sind ein effektives Mittel um die Lesbarkeit von Quellcode zu erhöhen. Deshalb wird jeder Befehl sowie öffnende und schließende Klammern eines jeden Funktionsrumpfes auf eine neue Zeile gesetzt und eingerückt. Die Länge einer Einrückung ist mit 4 Leerzeichen festgeschrieben („-„ steht im nachfolgenden Bsp. für ein Leerzeichen).

Bsp.:

```
1 function functionname (var x)
2 {
3 ----if(condition)
4 ----{
5 --------
6 ----}
7 }
```

1.1.2 Bezeichner

Damit keine abstrakten Bezeichnungen entstehen können, müssen folgende Regeln eingehalten werden:

- Es werden sprechende Namen verwendet.
- Sollten Bezeichnungen aus mehreren Wörtern bestehen, so ist die Bezeichnung in CamelCase zu verfassen, d.h. der erste Buchstabe aller Wörter ab dem zweiten Wort groß zu schreiben.
  Bsp.: „functionDoThis“, „thisIsAnExample“
- Funktions- und Variablennamen beginnen mit Kleinbuchstaben.
  Bsp.: „functionDoThis“
- Klassennamen beginnen mit Großbuchstaben.
  Bsp.: „TestClass“
- Bezeichnungen für Konstanten bestehen nur aus Großbuchstaben.
  Bsp.: „CONSTANT“

1.1.3 Kommentare

Es gibt verschiedene Wege, Kommentare im Quelltext zu verwenden: Man kann sie als einzeilige oder als mehrzeilige Kommentare verwenden bzw. als Kommentare, die mit einem anderen Programm ausgelesen und zu einer Quelltextdokumentation zusammengestellt werden können. Dazu erhält jede Methode einen mehrzeiligen Kommentar direkt über ihrer Definition, der Informationen über die Aufgabe der Methode, die ihr übergebenen Parameter und den Rückgabewert enthält, sowie den Autor, der diese Methode implementiert hat. Sollte die Funktion einer Quellcodezeile nicht direkt erkennbar sein, muss ein einzeiliger Kommentar direkt über dieser Zeile, welche ihren Sinn und ihre Funktion beschreibt, eingefügt werden. In PHP und JavaScript kann man Kommentare auf dieselbe Art kennzeichnen. Nur in der Verwendung von Tags für z.B. den Autor unterscheiden sie sich. Letzteres kann für PHP auf http://phpdoc.org/docs/latest/index.html und für JavaScript auf http://code.google.com/p/jsdoc-toolkit/w/list nachgelesen werden.

Bsp.:

```
1 /** Dies ist ein einzeiliger Kommentar. */
2
3 /**
4  * Dies ist ein mehrzeiliger Kommentar, welches direkt vor jeder
5  * Methode vorkommt.
6  * Es folgen Tags in PHP:
7  * @param Typ_der_übergebenen_Variable Beschreibung
8  * @return Typ_der_zurückgegebenen_Variable Beschreibung
9  * @author Name
```

```
10  * Nachfolgend Tags in Javascript:
11  * @param {Typ} Name Beschreibung
12  * @returns {Typ} Beschreibung
13  * @author Name
14  * weitere Tags bei Bedarf
15  */
```

1.1.4 Externe Dokumentation

Als Dokumentationssoftware werden phpDocumentor und JSDoc verwendet. Diese externe Dokumentation dient dazu, dass sich andere Entwickler schnell in das Projekt einarbeiten können, ohne sich erst mühsam die Informationen aus dem Quelltext suchen zu müssen.

1.2 Die Änderungsdokumentation

Die im Projekt angefertigten Änderungen müssen auch von einer Dokumentation begleitet werden. D.h. dass jeder Commit im Git-Repository eine aussagekräftige Beschreibung aufweist. Optional können Änderungen im Quelltext mit ein bis mehreren einzeiligen Kommentarzeilen am Beginn einer Funktion bzw. direkt nach der(/den) geänderten Quellcodezeilen erfolgen. Dazu ist es unabdingbar mit anzugeben, wer diese Änderungen durchgeführt hat.

1.3 Die Testdokumentation

Die durchgeführten Tests sollen wiederum dokumentiert werden, um zu vermeiden, dass Fehler im Quellcode nicht behoben bzw. frühzeitig erkannt werden, weil z.B. durch eine unzureichende Dokumentation der Tests einige Methoden nicht überprüft werden. Weiterhin dient dies auch zum Beleg des umfassenden Tests der Anwendung und zum Beleg der Qualität des Produktes.

1.4 Die externe Dokumentation

Die externe Dokumentation beinhaltet ihrerseits ein Benutzerhandbuch und eine Entwurfsdokumentation, die alle wichtigen Entwurfsaspekte verständlich darlegt und begründet. Die hierbei verwendete Sprache ist Deutsch.

2. Testkonzept

2.1 Modul-Tests

Für die Tests der in PHP entwickelten Module wird das Framework PHPUnit verwendet, da es sich besonders zum Testen einzelner Programmeinheiten (Units) wie Klassen oder einzelnen Methoden eignet. Für jedes einzelne Modul oder bei Bedarf auch für einzelne Klassen dieses Moduls werden Tests implementiert, um das Verhalten der Programmteile bei vorgegebenen Eingabeparametern zu prüfen. Den wesentlichen Vorteil dieser Test-Methode bildet die Möglichkeit Tests automatisch auszuführen und diese nicht manuell generieren zu müssen.

PHPUnit bietet dabei eine vollständige Trennung von Test-und Anwendungscode. Somit bleiben die Test-Module unabhängig von Änderungen im Produktionscode. Die Implementierung der Tests soll dabei vor oder spätestens während der Arbeit am Code für das zu entwickelnde Produkt stattfinden. Spezifikationen und Anforderungen werden somit vor der Entwicklung erfasst und dokumentiert. Damit wird sichergestellt, dass der zu entwickelnde Code zielführend entwickelt und Abweichungen von der Spezifikation vermieden werden.

2.2 Oberflächen-Test

Um die entstehende GUI und die zu generierenden Formulare effizient auf ihre Eignung im konkreten Anwendungsfall zu überprüfen, wird ein automatisiertes Testverfahren benötigt. Als Testumgebung wird dafür Selenium verwendet. Dabei handelt es sich um ein auf HTML und JavaScript basierendes Entwicklungstool zum automatischen Testen von Webanwendungen und -formularen.

Ein spezielles Add-on für den quell-offenen Internet-Browser Mozilla Firefox bietet die Möglichkeit, Nutzereingaben und sonstige Interaktionen mit der Web-Oberfläche aufzuzeichnen und beliebig oft wiederzugeben. Dadurch können die Oberflächen-Tests wesentlich effizienter und Flexibler gestaltet werden.

2.3 System- und Abnahmetest

Das fertige Produkt soll schließlich nicht mehr auf die Funktionalität einzelner Programmteile überprüft werden, sondern muss gegen alle Anforderungen an das lauffähige Produkt getestet werden. Der sogenannte Systemtest findet dabei in einer geeigneten Testumgebung mit anwendungsspezifischen Testdaten statt. Dabei muss gewährleistet werden, dass alle im Pflichtenheft spezifizierten Funktionalitäten und Nutzeranforderungen vom fertigen Softwareprodukt umgesetzt werden. Besonderes Augenmerk sollte dabei auf eine intuitive Bedienung der Weboberfläche und der generierten Formulare gelegt werden. Deshalb bietet sich der Test des Produkts, durch an der Entwicklung unbeteiligte Personen an, um dieses intensiv am subjektiven Empfinden der Nutzer messen zu können.

Wurden alle vorangestellten Tests erfolgreich durchgeführt, findet die Konsultation des Auftraggebers zum Abnahmetest statt. In den Abnahmetests werden alle Beteiligten der Produktentwicklung eingebunden und die im Pflichtenheft beschriebenen Funktionalitäten schrittweise überprüft und abgenommen oder u.U. neu spezifiziert. Es empfiehlt sich dabei der Testbetrieb mit Echtdaten, um einen möglichst genauen Einsatz in der Anwendungsumgebung zu simulieren.

3. Organisatorische Festlegungen

Die Projekttreffen finden wöchentlich und auch während der vorlesungsfreien Zeit statt. Dazu wird jedes Teammitglied angehalten jedem Treffen beizuwohnen. Ist dies einmal nicht möglich, muss sich das entsprechende Teammitglied abmelden. Zu diesen Treffen kommen das Team, der Betreuer und wenn nötig der Auftraggeber zusammen. Zu diesen Treffen werden neue Aufgaben verteilt bzw. das weitere Vorgehen besprochen. Davon abgesehen erfolgen die erforderlichen Absprachen im Team über E-Mail.

Die erstellten Dokumente werden im git-Repository auf dem Praktikumsserver hochgeladen, damit keine Verwirrung bezüglich der aktuellen Versionen der Dokumente entstehen kann. Weiterhin ist auch jedes Teammitglied dafür verantwortlich, seine jeweils selbstgeschriebenen Methoden zu kommentieren. Dies wird insb. durch den Verantwortlichen für Dokumentation überwacht. DesWeiteren muss jedes Teammitglied jede Woche einen Aufwandserfassungsbericht dem Projektleiter zukommen lassen, der diese dann zusammenfasst.