Qualitätssicherungskonzept

SWP12-10

Alexander Richter
Mark Hennig
Peggy Lucke

16. April 2012

Inhaltsverzeichnis

1 Dokumentationskonzept

Das Dokumentationskonzept bezieht sich auf allgemein anerkannte Richtlinien für die Softwareentwicklung im Softwarenentwicklungsprozess mit Java bzw. Java-Script.

1.1 Bezeichner

- Bei Bezeichnerrichtlinien wird zwischen Variablen, Methoden, Konstanten und Klassen unterschieden.
- Als Bezeichner sind Entsprechende Namen nach Aufgabe zu wählen. Diese Bezeichner sollten dem Kontext entsprechen und aus dem Namen sollte hervorgehen für was dieser Bezeichner gebraucht wird.
- Standardbezeichnersprache ist Englisch.
- Klassenbezeichner beginnen mit einem Großbuchstaben. Wenn mehrere Worte in einem Bezeichner verbunden werden, sollten die Anfangsbuchstaben groß geschrieben werden. z.B. `CreateNewMapping(...)`
- Variablen- und Methodennamen beginnen mit kleinem Buchstaben, allerdings wird auch jedes weitere Wort im Namen groß geschrieben. z.B. `tempÜbergabeparameter = ...;`
- Konstantenbezeichner bestehen nur aus Großbuchstaben. Mehrere Wörter werden mit einem Unterstrich getrennt. z.B. `TEMP_MAX_SIZE = ...;`

1.2 Einrückungen

Einrückungen dienen der besseren Lesbarkeit des Quelltextes. Es ist zu Empfehlen die festgelegten Richtlinien sofern möglich in der Entwicklungsumgebung zu definieren. Dann ist es möglich den Quelltext automatisch zu formatieren. Jeder Unterbefehl ist innerhalb einer Methode um ein Tab einzurücken. Es steht genau ein Befehl in einer Zeile. Öffnende und schließende geschweifte Klammern des Funktionsrumpfes stehen immer in neuen Zeilen

1.3 Kommentare

Kommentare dienen der Nachvollziehbarkeit des Quelltextes. Die Dokumentationssprache ist Englisch. Verantwortlich für die Programmierung sind die jeweiligen Programmierer selbst. Die Kommentare sollten während der Programmierung geschrieben werden, damit wichtige Details nicht verloren gehen.
Bei Variablen und Konstanten sollte immer angegeben werden wofür diese gebraucht werden. Funktionen und Methoden sind immer zu dokumentieren. Bei eigenen Algorithmen ist deren Funktionsweise und Zweck zu dokumentieren.

- Einzeilige Kommentare beginnen mit `//`

- Mehrzeilige Kommentare werden mit `/* ... */` umschlossen

1.4 Schlüsselwort `this` für globale Variablen

Zur Verbesserung der Verständlichkeit des Quellcodes ist bei der Verwendung von definierten globalen Variablen das Schlüsselwort `this`. Voranzustellen. Dadurch können lokale von globalen Variablen unterschieden werden. z.B. `this.tempVariable` für globale Variablen Allerdings sollte man am besten überhaupt keine globalen Variablen benutzen, damit der Quelltext verständlicher wird.

1.5 Modelldokumentation

Modelle werden nach UML2 Standard angefertigt. Sie sind auf notwendiges begrenzt, um Übersichtlichkeit und Verständlichkeit zu gewährleisten. Zur Modellierung von UML Diagrammen wird das Programm Dia verwendet.

1.6 Benutzer Dokumentation

Für die Benutzer wird ein Benutzerhandbuch erstellt, welches ausführlich und leicht verständlich ist. Dies steht dem Nutzer zur Verfügung und soll alle Funktionalitäten verständlich und umfassend beschreiben.

2 Testkonzept

Fehler aus einem kompletten Softwaresystem zu entfernen, ist zeitintensiv und zum Teil sehr schwierig, daher lohnt es sich, jedes Teilprojekt bis hin zur einzelnen Klasse separat zu überprüfen. Somit werden Fehler, die durch Irrtümer und Schnitzer entstanden sind, früh erkannt und behoben. Mit dieser Herangehensweise ergeben sich vier Untergliederungen des Testkonzepts:

- Komponententest
- Integrationstest
- Systemtest
- Abnahmetest

2.1 Komponententest

Im Komponententest wird jede Klasse/Objekt auf funktionale Richtigkeit innerhalb des eigenen Universums getestet. Damit wird sichergestellt, dass Fehler auf Klassen- /Modulebene behoben sind, bevor diese zu einem Teilsystem kombiniert werden. Jedes Programmiererpaar schreibt zu jeder seiner programmierten Klassen/Modulen eigene Testklassen und prüft dann mit diesen die fertige Klasse/Modul. Diese Testklasse wird dann benannt nach dem Klassen/Modul-Namen und dahinter Test. Damit wird sichergestellt dass bei alphabetischer Ordnung Testklassen leicht zugeordnet werden können.

2.2 Integrationstest

Während des Integrationstests wird das Zusammenspiel einzelner Klassen, welche untereinander abhängig sind, überprüft. Damit wird erreicht, dass Probleme und Fehler, die bei der Kommunikation zwischen diesen Komponenten auftreten, frühzeitig erkannt und behoben werden können. Wird eine Klasse von einem Programmiererpaar fertiggestellt, die mit einer schon fertigen Klasse zusammenarbeiten wird, und ist der Komponententest dazu wie erwünscht verlaufen, wird von diesem Paar der Komponententest der eigenen Klasse und der schon vorhandenen Klasse durchgeführt. Werden Fehler bemerkt, muss die eigene Klasse noch einmal überprüft werden und falls das erfolglos bleibt auch noch einmal die andere, schon fertige Klasse.
Die GUI ist ein Teil des Clients, welche aus einer Vielzahl von Komponenten besteht, daher wird der Test der GUI im Integrationstest vollzogen.
Da wir die GUI nicht automatisiert testen können, werden wir dafür einen manuellen Test nutzen. Dabei werden wir per Hand die funktionelle Richtigkeit der GUI testen und sicherstellen das alle Komponenten der GUI wie gewünscht agieren.

2.3 Systemtest

Der Systemtest bietet die Möglichkeit, das Softwaresystem im Ganzen auf Fehler und Probleme zu testen, als Basis dient dabei das Pflichtenheft. In dieser Phase wird also aus der Sicht des Anwenders geprüft, ob alle Anforderungen zufriedenstellend umgesetzt wurden. Es wird in einer Testumgebung ausgeführt, und es gibt eine überschaubare Menge an Daten in der Datenbank, welche speziell zum Testen aller Funktionen erstellt wurden.

2.4 Abnahmetest

Nach dem Systemtest wird das Softwaresystem unter realen Bedingungen und im Beisein des Auftraggebers getestet, mit der uns zur Verfügung gestellten Datenbank mit Kopien von Echtdaten. Damit wird einerseits erreicht, dass unter realen Bedingungen alles zufriedenstellend funktioniert und auch die erwünschte Performance eingehalten wird, und andererseits der Auftraggeber sieht, dass das Programm auch das macht, was er möchte.

2.5 JUnit

Dieses Test-Framework bietet die Möglichkeit Java Programme zu testen, sowohl einzelne Klassen als auch deren Interaktion. Weiterhin ist es möglich das ganze Projekt damit zu testen. Junit spezialisiert sich auf automatische Unit-Tests (für Klassen und Methoden). Solch ein Test kennt nur zwei mögliche Ergebnisse, das Gelingen bzw. das Misslingen des Tests. Ein Misslingen liegt bei einem Fehler oder bei einem falschen Ergebnis vor, dessen Ursache durch eine Exception signalisiert wurde. Beim Auftreten von keinem Fehler gilt der Test als gelungen.

2.6 UnitTesting

Mit diesen Test-Framework kann man Javascript-Code systematisch clientseitig auf Fehler und Probleme testen.

2.7 Testdokumentation

Wenn Fehler gefunden und behoben wurden, werden diese durch den zuständigen Programmierer im Quelltext durch Kommentare angemerkt, und zwar in der Form, dass Fehler und Lösungsweg vermerkt werden. Dies bietet die Möglichkeit, bei erneutem Auftreten derselben oder ähnlicher Fehler entsprechend zu handeln.

3 Organisatorische Festlegungen

An den wöchentlichen Teamtreffen wird das weitere Vorgehen sowie die Arbeitsteilung besprochen sowie über vorliegende Ergebnisse diskutiert. Dieses Treffen findet in der Regel direkt nach dem Tutorengespräch statt. Hier wird der Kurs für die nächste Woche gemeinsam beraten: Die zu erledigenden Aufgaben werden diskutiert, verteilt und mit Verantwortlichen versehen, ggf. werden gruppeninterne Deadlines gesetzt und weitere Arbeitstreffen vereinbart. Somit ist gewährleistet, dass alle Teammitglieder stets über die aktuelle Situation innerhalb des Projektes informiert sind und ihre nächsten Arbeitsaufgaben kennen.
Die Ergebnisse sind spätestens drei Tage vor dem Abgabetermin mittels Mercurial hochzuladen oder dem technischen Assistenten per Email zu schicken. Die Kommunikation außerhalb der Teamtreffen erfolgt hauptsächlich per Mail und eventuell Skype. Die Zusammenführung der einzelnen Ergebnisse obliegt dem jeweiligen Phasenverantwortlichen. Ihm werden alle Einzelergebnisse zu einer internen Deadline zugeschickt, der diese zusammensetzt.
Alle Gruppenmitglieder machen sich vor Beginn der Implementierung mit den festgelegten Dokumentationsrichtlinien vertraut. Die Umsetzung oben genannter Dokumentationsrichtlinien obliegt jedem Programmierer selbst und wird durch den Verantwortlichen für Dokumentation und Qualitätssicherung überprüft, d.h. der Programmierer dokumentiert und kommentiert seinen Quelltext selbst. Falls dabei die Dokumentationsrichtlinien nicht eingehalten wurden, wird der Programmierer darauf hingewiesen und muss diese Mängel beseitigen. Die komplette Dokumentation wird am Ende zu einer Gesamtdokumentation zusammengefügt, so dass man sich später leicht in das Projekt einarbeiten kann bzw. das Produkt einfach benutzen kann.
Ausserdem ist der Verantwortliche für Dokumentation und Qualitätssicherung für die Erstellung des Benutzerhandbuches zuständig.
Die Verwaltung der Quelltexte findet über Mercurial statt. Treten während der Implementierung Fehler auf, werden diese mit korrekter nachvollziehbarer Beschreibung dokumentiert.