1 Dokumentationskonzept

Eine gute Dokumentation trägt nicht nur wesentlich zur Qualitätssicherung bei, sondern erleichtert auch die Wartung der Software und verkürzt Einarbeitungszeiten.

1.1 Programmcode

Damit der Programmcode verständlich ist, muss er lesbar und strukturiert angelegt sein. Deshalb wird der Programmcode entsprechend der Java-Konventionen gestaltet. Besonders wichtig für verständlichen Programmcode sind ausreichend viele und verständliche Kommentare. Diese werden zu jeder Methode, Klasse, Konstante und nicht temporären Variable verfasst, sowie in längeren oder komplizierteren Abschnitten von Programmcode. Kommentare sollen vor allem das WAS beschreiben, da das WIE durch den Code ersichtlich sein sollte. Beim Verfassen von Kommentaren ist darauf zu achten, dass auch Außenstehenden ein gutes Verständnis ermöglicht wird. Um aus den Kommentaren direkt (z.B. mit JSDoc) eine Dokumentation generieren zu können, entsprechen diese der JavaDoc Syntax.

Für den Programmcode selbst gilt, dass sprechende Namen in englischer Sprache für Bezeichner (von Klassen, Methoden, Variablen, Konstanten, ...) verwendet werden, wobei einzelne Wörter in Camelcase aneinander gehängt werden. Den Java Konventionen folgend bilden Konstanten eine Ausnahme. Sie sind komplett groß zu schreiben, Wörter werden durch Unterstriche zusammengefügt. Eine weitere Ausnahme sind Variablen, die nur innerhalb einer Funktion Gültigkeit besitzen. Sie dürfen kurz gehalten werden, weil davon ausgegangen wird, dass die Funktion selbst gut dokumentiert und der Code verständlich ist.

Dass man Programmcode durch Einrückungen, Klammern und Leerzeichen entsprechend übersichtlich gestaltet, gilt als Standardpraxis und sollte nicht explizit erwähnt werden müssen. Zusätzlich sind maximal 4 Ebenen der Verschachtelung innerhalb einer Funktion zulässig.

Zusätzlich erachten wir es als eine sinnvolle Metrik, festzulegen, dass keine Zeile länger als 80 Zeichen ist. Einrückungen werden mit Tabs vorgenommen und Bezeichner und Kommentare sind in englischer Sprache zu verfassen.

1.2 Dokumentation im VCS

Um möglichst effizient mit dem VCS umzugehen, wird darauf geachtet, mit einer möglichst aktuellen Version zu arbeiten, Commits ordentlich zu kommentieren und möglichst fehlerfreien Code einzusenden. Das bedeutet insbesondere, dass eingesendeter Code frei von Syntaxfehlern ist, und bestehende Tests nicht gebrochen werden. Im Zweifelsfall müssen existierende Tests an den geänderten Codebestand angepasst werden.

1.3 Modelldokumentation

Modelle werden nach UML2 Standard angefertigt. Sie sind auf notwendiges begrenzt, um Übersichtlichkeit und Verständlichkeit zu gewährleisten.

1.4 Weitere Dokumentation

Um die Dokumentation zu vervollständigen, und unabhängig von Programmcode und VCS ein Verständnis der Software zu ermöglichen, werden eine Benutzerdokumentation und eine Designbeschreibung erstellt, die leicht eine grobe Vorstellung vermitteln.

2 Organisatorische Festlegungen

Für teaminternen Absprachen findet ein wöchentliches Treffen des gesamten Teams statt, in der Regel kurz nach dem Tutorengespräch. 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.

Außerhalb dieser wöchentlichen Treffen erfolgt der Großteil der teaminternen Kommunikation über eine Email-Liste; für die Dokumentation der Ergebnisse stehen ein teaminternes Wiki (für Dokumentation und Texte), sowie ein Mercurial-Repository (für Quelltext) zur Verfügung. Für intensivere Absprachen wurde darüber hinaus ein XMPP-Chatraum eingerichtet, der bei Bedarf für kurzfristige Absprachen genutzt werden kann.

3 Testkonzept

3.1 Einleitung

Testkonzepte sind ein essentieller Bestandteil bei der Enstehung von Software. Ein durchdachtes Testkonzept sichert (weitgehende) Fehlerfreiheit, sowie die Qualität der Software in Hinsicht auf Belastungsfähigkeit, Wartbarkeit, etc. Fehler beim Compilieren, sowie Fehler, die erst zur Laufzeit auftreten können, gilt es auszumachen. Für Ersteres reicht ein einfacher Compiler, für letzteres bietet sich die Verwendung eines Programmes an, welches eine große Anzahl an Tests zeitgleich durchführt und dessen Ergebnisse verwaltet (wie

etwa JUnit). Desweiteren gibt es Tests, die nicht aus Programmierer- sondern aus Anwendersicht stattfinden. Dazu zählt der abschließende Systemtest, sowie der Abnahmetest, der dann allerdings durch den Kunden stattfindet und die komplette Fertigstellung des Projekts voraussetzt.

3.2 Modultest

Der Modultest beinhaltet das Testen eines Moduls, also einer zusammengehörigen Gruppe von Klassen. Er erfolgt automatisiert durch ein Testautomatisierungstool, sodass eine große Anzahl an Tests in kürzester Zeit durchgeführt werden kann. Dieses Tool gibt für jeden der einzelnen Tests aus, ob er erfolgreich war oder nicht, und kann somit Fehler feststellen, die zur Laufzeit auftreten würden (bei Eingabe der verschiedensten Parameter). Solche Tests werden, sowohl innerhalb einer einzelnen Klasse, als auch in einer größeren Menge an Klassen durchgeführt.

3.3 Integrationstest

Der Integrationstest ist eine Reihe einzelner Tests, wie etwa Funktions- und Schnittstellentests, die das Zusammenspiel der einzelnen Komponenten des Programmes zum Gegenstand hat. In unserem kleinen Programmierteam ist es ratsam, jeden Integrationstest im Anschluss an die Fertigstellung eines Moduls durchzuführen, um sicherzustellen, dass dieses mit dem bereits vorhandenen Code harmoniert.

3.4 Systemtest

Der Systemtest beinhaltet das Testen des Projekts in seinem gesamten Umfang, diesmal nicht aus der Sicht der Entwickler, sondern aus der des An-

wenders. Nach dem Black-Box-Prinzip stehen die einzelnen Methoden und Module nicht mehr im Vordergrund, sondern nur das Gesamtergebnis. Die Funktionen des Programmes aus dem Pflichtenheft stehen im Vordergrund; es wird getestet, ob sie wirklich funktionieren, wie dort beschrieben.

3.5 Abnahmetest

Der Abnahmetest erfolgt nach der Fertigstellung des Projekts durch den Kunden. Er testet, ob das Programm zu seiner Zufriedenheit fertiggestellt wurde und ob seine Vorgaben eingehalten wurden.