Qualitätssicherungskonzept

Verantwortlicher: Richard Khulusi

Daniel Alexander, Jakob Kusnick, Richard Khulusi, Christopher Schott, Tobias Turke, Nikolas Oberhäuser, Hans Dieter Pogrzeba, Felix Conrads

Inhaltsverzeichnis

1. Dokumentationskonzept 3
1.1 Allgemeines 3
1.2 Aussehen des Quellcodes (Coding Standard) 3
1.2.1 Bezeichner 3
1.2.2 Formatierungen 3
1.3 Code Dokumentation 4
1.4 Sonstige Dokumentation 4
2. Testkonzept 4
2.1 Allgemeines 4
2.2 Dokumentation 5
2.3 Komponententest 5
2.4 Integrationstest 5
2.5 Systemtest 5
2.6 Abnahmetest 5
3. Organisatorische Festlegungen 6
3.1 Treffen 6
3.2 Kommunikation 6
3.3 Dokumentenformat 6
3.4 Versionskontrolle 6
3.5 Umgang im Team 6
3.6 Fragen und Vorschläge außerhalb von Treffen 7
3.7 Externe Programme 7

Sämtliche Festlegungen aus dem Qualitätssicherungskonzept (somit auch des Dokumentationskonzepts und Testkonzepts) sind konsequent einzuhalten.
Die Einhaltung dieser ist insbesondere von den Verantwortlichen für Dokumentation (Nikolas Oberhäuser), Tests (Hans Dieter Pogrzeba) und Qualitätssicherung (Richard Khulusi) zu überwachen.

1. Dokumentationskonzept

1.1 Allgemeines

Eine saubere und einheitliche Dokumentation des Projektes ist essenziell für effektives Arbeiten. Zum Ersten wird dadurch eine schnelle Einarbeitung oder Wiedereinarbeitung in den Quellcode ermöglicht. Des Weiteren behält das Team im Verlauf des Projektes den Überblick über Zustand des Projektes und kann diesen besser kontrollieren. Außerdem können in Krisensituationen die Fehler schneller rekonstruiert und korrigiert werden.

Die Dokumentation teilt sich auf in interne und externe Dokumentation sowie Richtlinien für das Aussehen des Quellcodes.

1.2 Aussehen des Quellcodes (Coding Standard)

Im Allgemeinen richten wir uns nach http://www.oracle.com/technetwork/java/codeconv-138413.html, den Richtlinien für Javaprogrammierung. Die wichtigsten Punkte werden nun noch einmal aufgelistet.

1.2.1 Bezeichner

- Alle Bezeichner sind in Englisch zu verfassen
- Alle Bezeichner haben „sprechende Namen“ mit Ausnahme eindeutiger Laufvariablen
- Klassennamen beginnen mit einem Großbuchstaben
- Variablen-, Objekt-, Attribut- und Methodennamen beginnen mit Kleinbuchstaben
- Bei zusammengesetzten Namen beginnt jedes neue Wort mit einem Großbuchstaben
- Konstanten werden nur aus Großbuchstaben und Unterstrichen gebildet

1.2.2 Formatierungen

- In jeder Zeile steht nur ein Befehl
- In einer Zeile stehen nicht mehr als 120 Zeichen
- Code der innerhalb von Schleifen oder Bedingungen steht wird mit einem Tab eingerückt
- Zwischen Methoden, und logischen Abschnitten sowie vor Kommentaren wird eine Freizeile gelassen
- Nach Kommata sowie vor und nach zweistelligen Operatoren wie +,-,<,= etc. wird ein Leerzeichen gesetzt.
- { steht noch in derselben Zeile wie die If-Klausel oder der Schleifenkopf
- Kommentare sind über der zugehörigen Zeile oder wenn genug Platz vorhanden ist am Ende der Zeile zu schreiben (mehrzeilige Kommentare immer über der Anweisung).

1.3 Code Dokumentation

Jede Klasse besitzt einen Dateikopf, in dem in einem zusammenhängenden Kommentar die Funktion der Klasse sowie Name der Autoren zu finden sind.
Klassen, Methoden und Attribute werden mit /** … */ Javadoc-Kommentaren versehen die eine Beschreibung der Funktion sowie u.U. die Informationen @param, @return enthalten. Die Kommentierung der Klasse erfolgt am Anfang der selbigen, bei Methoden und Attributen direkt über ihnen. Im Quellcode ist alles mit /* */ oder // Kommentaren zu versehen dessen Funktion oder Zweck nicht auf den ersten Blick deutlich wird. In jedem Fall sind Schleifen zu kommentieren falls die Laufvariable nicht sprechend benannt wurde. Ebenso sind Verzweigungen bei Bedingungen und Schleifen zu kommentieren.
Grundsätzlich sollen alle Kommentare in Deutsch verfasst werden.

1.4 Sonstige Dokumentation

Außer der Dokumentation im Code ist es wichtig das Projekt und den Projektverlauf gut zu dokumentieren. In den Gruppentreffen wird ein gruppeninternes Protokoll angefertigt, sodass jedes Mitglied sich im Nachhinein alles in Erinnerung rufen kann. Die Ausarbeitung eines Konzeptes sollte in einem UML-Diagramm festgehalten werden. In Textdokumenten zum Projekt ist Schriftart Calibri und Schriftgröße 12 zu verwenden (Überschriften in Größe 18, unterstrichen und Zwischenüberschriften in Größe 12, fett, unterstrichen). Des Weiteren sollte darauf geachtet werden bei den Texten unter Rechtsklick →Absatz den Abstand Vor und Nach auf 0 zu stellen sowie den Zeilenabstand auf 1.0.

2. Testkonzept

2.1 Allgemeines

Aufgrund der Größe bzw. Komplexität des Projektes ist es unerlässlich, dass die einzelnen Komponenten, wie auch das Gesamtsystem umfangreich getestet werden. Hierzu ist eine genaue Planung und die Erstellung eines durchdachten Testkonzepts, sowie die Einhaltung und Dokumentation der Umsetzung der Tests notwendig. So können Fehler frühzeitig erkannt und lokalisiert werden, um ein effektives Arbeiten zu ermöglichen.

Einerseits sollen automatische Tests mit Hilfe des JUnit Frameworks durchgeführt werden, welches sich insbesondere für den Komponententest anbietet. Andererseits sollen auch manuelle Tests z.B. die Funktion der GUI überprüfen. Für manuelle Tests, geade für unfertige, zur Zeit in der Entstehung begriffene Programmteile, kann innerhalb der einzelnen Klassen eine main Funktion geschrieben werden.

2.2 Dokumentation

Durch die Protokollierung der durchgeführten Tests soll bei eventuell erneut auftretenden ähnlichen Fehlern darauf zurückgegriffen werden können.

Es soll jeweils folgendes protokolliert werden:
Art des Tests, Liste der Fehler, Art der Fehler, Fehlerquelle (falls bekannt), Lösung (falls bekannt)

2.3 Komponententest

Beim Komponententest werden die einzelnen Klassen und Methoden auf Funktionalität, Korrektheit und Vollständigkeit geprüft. Dies ermöglicht eine Behebung der Fehler bereits bevor das Gesamtsystem fertiggestellt ist.
Mit Hilfe von JUnit können von allen Teammitgliedern Tests durchgeführt werden, wobei die Verantwortlichkeit für die jeweiligen Komponenten beim jeweiligen Umsetzungsteam liegt. Dabei soll eine möglichst vollständige Abdeckung der Codebasis erreicht werden. Nach erfolgten Tests sollen die Ergebnisse Dokumentiert werden (s.o.).

2.4 Integrationstest

Der Integrationstest überprüft die ordnungsgemäße Zusammenarbeit der einzelnen Komponenten. Nach der Fertigstellung einer Komponente soll jeweils deren Zusammenspiel mit den anderen bereits fertigen Komponenten, überprüft werden. Somit können evtl. auftretende Fehler auf einen immer noch überschaubaren Teil eingegrenzt werden und die fertigen Teile fügen sich Schritt für Schritt ins Gesamtsystem ein.

2.5 Systemtest

Beim Systemtest wird nicht mehr auf Codeebene getestet, sondern das gesamte System wird aus Sicht des späteren Anwenders anhand bestimmter Szenarien überprüft. Während zunächst nur die bereits fertiggestellten Abschnitte des Systems getestet werden können, müssen zum Schluss alle Anforderungen des Lastenhefts erfüllt werden.
Müssen aufgrund der Ergebnisse des Systemtests Programmteile angepasst werden, müssen diese erneut Komponenten- und Integrationstests durchlaufen.

2.6 Abnahmetest

Der Abnahmetest ist der abschließende Test, bei dem das System dem Auftraggeber vorgestellt wird und dieser die Funktionen der Testinstallation mit den festgelegten Anforderungen abgleicht.

3. Organisatorische Festlegungen

3.1 Treffen

Um ein ausgeglichenes Team zu schaffen, wurde vereinbart, sich jede Woche neben dem offiziellen Termin zu treffen. Hier wird über Fortschritte, Überlegungen und Probleme diskutiert. Dieser Termin wird auch benutzt um Wissenslücken bei Teammitgliedern zu füllen, die aufgrund unterschiedlicher Arbeiten und unterschiedlicher Anforderungen entstehen können.
Es gibt einen Standard Termin. Bei diesem Termin wird der nächste Termin noch einmal explizit bestätigt, um es so zu ermöglichen, dass im Idealfall immer das gesamte Team vor Ort ist. Sollte dies nicht möglich sein kann rechtzeitig ein geeigneter Ausweichtermin vereinbart werden.

Um bei den Terminen nicht ständig vom Thema abzuweichen, wird eine Art Buzzer (akustisches Signal) vereinbart mit dem die aktuelle Konversation unterbrochen werden darf um auf das eigentliche Thema zurück zuführen.

3.2 Kommunikation

Die Kommunikation zwischen den Mitgliedern wird über einen Email-Verteiler geregelt, um so zu ermöglichen, dass keiner wichtige Termine verpasst und im Vorhinein wichtiges abzusprechen. Kurzfristige Probleme werden über Skype geregelt und diskutiert.
Sollte ein Teammitglied zu einem Termin nicht (rechtzeitig) erscheinen können, oder sollte es diesem nicht möglich sein, den erteilten Arbeitsauftrag termingerecht zu erfüllen, so wird der Projektleiter oder ein anderes zuverlässiges Teammitglied kontaktiert.

3.3 Dokumentenformat

Bei der Erstellung von Dokumenten wird ein einheitliches Dateiformat vereinbart. Für Textdokumente wird DOC verwendet und als Abgabeformat PDF.

3.4 Versionskontrolle

Durch die Benutzung eines Versionskontrollsystems wie git ist es möglich eine gemeinsame Konvention über die einzelnen Commits zu treffen, welche besagt, dass möglichst zusammenhängende und größere Commits getätigt werden sollen. Dadurch kann mithilfe des Programms SourceTree eine sehr gute Übersicht über die Commits des git-Projekts geschaffen werden, um so ohne größeren Aufwand zu sehen, ob der Quellcode die Konventionen aus dem Dokumentationskonzept einhält.

3.5 Umgang im Team

Im Weiteren wird um einen freundlichen Umgang untereinander gebeten.
Zudem sollten Pausenzeiten geregelt werden, welche nach einem gegebenen Zeitintervall abgehalten werden sollten. Diese sind vorzugsweise abseits von weiter Arbeitenden abzuhalten um im Arbeitsraum das Arbeitsklima nicht zu stören.

3.6 Fragen und Vorschläge außerhalb von Treffen

Sollten außerhalb von Treffen Fragen oder Vorschläge aufkommen, so sollten diese entweder per Skype gestellt (bei größeren, dringenden Themen ist ein Skypetreffen zu vereinbaren um möglichst viele Teammitglieder zu erreichen) oder beim nächsten Treffen angesprochen werden.
Wichtige Beschlüsse oder Ergebnisse werden per E-Mail an alle Teammitglieder geschickt.

3.7 Externe Programme

Als Entwicklungsumgebung ist einheitlich auf das bereits bekannte Eclipse zu setzen.

Für das effektive bearbeiten und festhalten von Bugs wird ein externes System benutzt.