Gruppe: swp12-2 Verfasser: Oliver Schurig
Datum: 14.04.2012

Qualitätssicherungskonzept

1. Dokumentationskonzept

Die Dokumentation ist ein wesentlicher Bestandteil eines erfolgreichen Projektes. Sie leistet nicht nur Beiträge zur Sicherstellung einer guten Qualität, sondern ermöglicht auch eine sehr viel kürzere Einarbeitungszeit in den Quellcode und das Projekt an sich.

1.1. Programmcode

Ein Muss für jeden Quellcode ist Struktur, um die intuitive Lesbarkeit zu garantieren. Folglich ist es sinnvoll, den Quellcode entsprechend der Java-Konventionen (http://www.oracle.com/technetwork/java/codeconv-138413.html) anzulegen. Eine ausreichende und verständliche Kommentierung von Klassen, Methoden und Konstanten sollte hierbei als selbstverständlich betrachtet werden. Variablen die nicht nur temporär auftreten, sollten ebenso kommentiert werden. Treten längere Abschnitte von Code auf, die eventuell als unverständlich oder kompliziert erachtet werden könnten, sollten ebenso mit einer eigenen Dokumentation versehen sein. Dem Verfasser der Kommentare muss klar sein, dass Außenstehende den Quellcode oder die Kommentare eventuell nicht so leicht verstehen können, sodass er speziellen Wert auf gutes Verständnis legen muss. Wie der Code funktioniert wird aus dem Code selbst ersichtlich, deshalb sollte die Dokumentation viel eher darauf eingehen, was der Code eigentlich tun soll. Auch das Einhalten der JavaDoc-Syntax sollte eine Selbstverständlichkeit sein, um ein automatisches Ziehen einer Kommentierung aus der Dokumentation zu ermöglichen.
Eine weitere Konvention betrifft sprechende Namen. Hiermit wird zusätzlich festgelegt, dass diese im Englischen bezeichnet werden sollen. Sollte die Bezeichnung mehrere Wörter benötigen, so sind diese Wörter aneinandergereiht und neue Wörter werden durch einen führenden Großbuchstaben kenntlich gemacht. Den Java-Konventionen zufolge bilden Konstanten eine Ausnahme und sind komplett groß zu schreiben, und mehrere Wörter werden durch Unterstriche getrennt.
Eine zweite Ausnahme bilden Variablen, die nur innerhalb von Methoden angewandt werden. Sie dürfen kurz und „nicht sprechend“ sein, da anzunehmen ist, dass die Funktion selbst gut dokumentiert ist und selbstbeschreibend ist.
Die übersichtliche Gestaltung des Codes durch Leerzeichen und Einrückung, sowie Klammerung an gegebener Stelle sollte ebenso intuitiv und selbstverständlich sein. Einrückungen sind mittels Tabs vorzunehmen. Feinere Einrückungen mittels weniger Leerzeichen sind ebenso erlaubt, wenn der Programmierer das genaue Übereinanderstellen bestimmter Codeelemente aus Gründen der Übersicht für sinnvoll hält.
Da jegliche uns bekannte öffentliche Dokumentation zu LIMES, in englischer Sprache verfasst ist, wird hiermit ebenso verlangt, auch die Kommentierung ebenso in englischer Sprache vorzunehmen.

1.2. Modelldokumentation

Java-Modelle werden nach dem UML2-Standard angefertigt, und auf das notwendige begrenzt. Dies gewährleistet, dass man sich nicht in einem Meer unnötiger Informationen verliert, so dass Übersichtlichkeit und Verständlichkeit im Vordergrund stehen.

Gruppe: swp12-2 Verfasser: Oliver Schurig
Datum: 14.04.2012

1.3. Versionskontrollsystem

Um mit dem verwendeten Versionskontrollsystem (VCS) möglichst effizient umzugehen, muss darauf geachtet werden, dass man immer mit den aktuellsten Versionen arbeitet. Auch beim Einsenden von Code sollten zuvor Fehler bereinigt werden, was sich vor allem auf Syntaxfehler bezieht. Das beseitigen inhaltlicher Fehler kann nicht zu jedem Zeitpunkt vor dem Einsenden von Code geschehen, da ein Quellcode in aller Regel zu komplex ist, um sämtliche Fehler dieser Art zuvor in vollem Umfang zu erfassen. Sind solche Fehler allerdings bekannt, so sollten sie vorher so gut wie möglich beseitigt, oder Mängel zumindest dokumentiert werden, damit andere Programmierer über mögliche Fehler des Codeabschnittes aufgeklärt sind, und mit diesen umgehen, oder sie gar beseitigen können.

1.4. Weitere Dokumentation

Neben der Dokumentation des Quelltexts wird eine technische Dokumentation erstellt, die in einem Wiki, das in einem VCS verwaltet wird, abgelegt wird. Sie beinhaltet eine genaue Dokumentation der Konzepte des Projektes aus technischer Sicht und vermittelt so einen Überblick über die Software, so dass die Orientierung erleichtert wird und Änderungen durchgeführt werden können. Die technische Dokumentation zusammen mit der Schnittstellendokumentation ist sehr wichtig, um die Einstiegshürde zum Mitwirken an dem Projekt zu senken. Im Gegensatz zur Schnittstellendokumentation, die zumeist in den Quelltext eingebettet ist, muss die technische Dokumentation sorgfältig gepflegt und aktualisiert werden. So erfüllt sie einen Zweck und verursacht nicht nur Wartungsaufwand.

Gruppe: swp12-2 Verfasser: Oliver Schurig
Datum: 14.04.2012

2. Organisatorisches in der Gruppe

Um die Kommunikation in der Gruppe zu erhalten wurde mindestens ein wöchentliches Treffen eingeführt, das in aller Regel möglichst bald nach dem Tutorentreffen angesetzt ist. Zu diesen Treffen ist die gesamte Projektgruppe anwesend und es wird das Vorgehen für die folgende Woche besprochen, es werden Aufgaben verteilt, Verantwortliche benannt, sowie Unklarheiten aus der Welt geschafft. Sollte dieses Treffen für diese Aufgaben nicht reichen, oder ist eine Verschiebung nötig, so wird auch ein weiteres Treffen der Gruppe beanschlagt, an dem jeder, soweit möglich, teilnehmen soll. Auf diesem Wege ist garantiert, dass jeder zu jedem Zeitpunkt über die aktuellen Probleme, Aufgaben und die allgemeine Situation informiert ist und das weitere Vorgehen für sich selbst und die anderen ersichtlich ist.
Zur weiteren Kommunikation zwischen den Projektteilnehmern besteht zum einen eine E-Mail-Kontaktliste über die alle nötigen Informationen übermittelt werden können, sowie ein gemeinsamer Jabber-Chatroom, über den Instant Messaging ermöglicht wird, um Absprachen in der Gruppe auch von zuhause, vor allem zu abendlichen Tageszeiten, ermöglicht, und so die Erreichbarkeit weiter verbessert wird. Dies ist sowohl für kurzfristige Absprachen, sowie zum Besprechen von projektbedingten Problemen praktisch.
Zum Bereitstellen der technischen Dokumentation steht ein gruppeneigenes Wiki, sowie ein Mercurial-Repository zum Hochladen von Quellcode zur Verfügung.

Gruppe: swp12-2 Verfasser: Oliver Schurig
Datum: 14.04.2012

3. Testkonzept

Auch Testkonzepte gehören zum Prozess der Entwicklung von Software, um eine weitestgehende Fehlerfreiheit anzustreben. Vor allem auf den Ebenen der Belastbarkeit und Erweiterbarkeit, führt ein solches Konzept zur Qualitätssteigerung.
Das Hauptziel liegt im Ausmachen von Fehlern die erst beim Compiliervorgang, oder sogar erst zur Laufzeit des Programms ausfindig gemacht werden. Um Fehler erster Art festzustellen reicht auch ein Compiler, wie er von Java-Entwicklungsumgebungen zur Verfügung gestellt wird. Für Tests der zweiten Art bieten sich Frameworks an, die auch das Grundgerüst für das Projekt bereitstellen. Das Play!-Framework bietet ganz eigene Testklassen an, so dass Programme wie das bekannte JUnit nicht einmal nötig sind, aber dennoch als zweite Idee nicht vergessen werden sollten. An dieser Stelle ist Play!'s Testumgebung Selenium zu nennen, mit der Komponenten, die mittels dem PLAY!-Framework erstellt wurden, effektiv getestet werden können.
Nennenswert ist des Weiteren, dass nicht alle Tests durch den Programmierer stattfinden, sondern auch Tests aus Anwendersicht durchgeführt werden müssen. Zu diesen Tests werden der abschließende Systemtest gezählt, sowie der Abnahmetest durch den Kunden, zu welchem das Projekt zuvor fertiggestellt sein muss.

3.1. Modultest

Im Modultest wird ein Modul, also eine Gruppe von Klassen, die zusammenspielen, getestet. Da bei der Arbeit an einer WebGUI in der Regel die Anzahl der programmeigenen Funktionen und Methoden vergleichsweise gering zu anderen Softwarekomponenten ist, und da eine Vielzahl von Funktionalitäten in HTML sowie Javascript beschrieben sein werden, bezieht sich dieser Test vor allem darauf, dass die Module auch im Rahmen unterschiedlicher Browser ihre volle Funktionalität bieten und ansehnlich bleiben. Dieser Test erfolgt durch uns oder Dritte, indem wir eine Menge unterschiedlicher Browser anwenden und gewisse (vorgegebene sowie zufällige) Eingaben durchlaufen um grenzwertige Fälle ausfindig zu machen. So kann geprüft werden, ob es Browser gibt, bei denen unerwartete Fehler funktionaler oder optischer Art auftreten.

3.2. Integrationstest

Im Integrationstest hingegen wird eine Reihe von Tests durchgeführt, welche dem Zweck dienen, Funktionen und Schnittstellen, sowie das Zusammenspiel einzelner Komponenten des Programms zu testen.
Ein Integrationstest ist ratsam wenn die Arbeit an einem Modul fertiggestellt ist, um so die Bausteine des Projektes nach und nach zusammenzufügen, anstatt sie erst herzustellen, und am Ende alle Module in ihrer Gesamtheit miteinander zu Testen. Sinnvoll ist dieser Test im Anschluss an die Fertigstellung des Vorprojektes, mit dem Hinzukommen neuer Funktionen durch die verschiedenen Release-Bündel. Auf diese Weise ist sichergestellt, dass man immer nur eine vergleichsweise kleine Anzahl von Fehlern beheben muss, und somit eine bereits funktionierende Menge von Modulen hat, die nur mit einzelnen neu hinzukommenden Modulen einhergehen müssen. Sollten im Nachhinein Fehler auftreten, so ist der Fehler damit leicht auf eine vergleichsweise kleine Codemenge einzugrenzen.

3.3. Systemtest

Der Systemtest beinhaltet das Testen der Software in ihrem gesamten Umfang, mit einem wesentlichen Unterschied zu den bisher genannten Tests: Der Test wird nicht mehr vom Entwicklerteam durchgeführt, sondern stattdessen von Anwendersicht. Die Ergebnisse der Entwicklungsarbeit stellen eine Blackbox dar, Aufbau von Methoden, Programmstruktur und ähnliches steht nicht mehr im Mittelpunkt, sondern das Ergebnis. Es wird geprüft, ob die Anforderungen des Pflichtenheftes erfüllt wurden, sowie ob alle Funktionen auch so funktionieren wie sie eben dort beschrieben sind.

3.4. Abnahmetest

Dieser Test erfolgt nicht, bis das Projekt fertiggestellt wurde. Der Tester ist der Auftrag gebende Kunde selbst. Bei diesem Test steht nicht nur im Vordergrund, ob alles was implementiert wurde auch reibungslos läuft, sondern der Kunde soll ein Bild der Zufriedenheit mit dem Ergebnis entwickeln. Sind alle Anforderungen enthalten? Ist die Software übersichtlich und verständlich gehalten? Gibt es „Störfaktoren“, die dem Anwender auf Dauer unangenehm werden können? Fragen dieser Art sind das Zentrum dieses Tests.