Qualitätssicherungskonzept

Thema: Inspirata
Gruppe: swp13-insp
Verantwortlicher: Tobias Kosmalla
Letzte Änderung: 02.06.2013
Version: 2

Inhaltsverzeichnis

1. Dokumentationskonzept

Eine umfassende Dokumentation ist von entscheidender Bedeutung für die Produktentwicklung und Wartung. Sie dient zum einen der Sicherung der Qualität der internen sowie der externen Kommunikation (Arbeitgeber, Außenstehende). Zum andern wird dadurch projektfremden Entwicklern die Möglichkeit gegeben, sich in Aufbau und Funktionalität des Programms einzuarbeiten. Dies ist ein für uns besonders wichtiger Punkt, da die Wartung und Pflege des Buchungssystems nach diesem Praktikum nicht abgeschlossen ist und somit in fremde Hände gegeben wird.
In unserer Gruppe haben wir uns dabei auf die folgenden Standards geeinigt.

1.1 Quelltextkonventionen

Ein einheitliches Design des Quelltextes dient dem besseren Verständnis des Codes und hilft Fehler zu vermeiden.
Folgende Regeln wurden aufgestellt:

- Kommentare sind in Englisch zu verfassen .
- Keine Umlaute im Quellcode.
- Komplexe Codeblöcke sind detailliert zu kommentieren.
- Eine Klasse wird mit Autor und Beschreibung der Klasse eingeleitet.

1.2 Namens-/ Benennungskonventionen

Die Namensvergabe soll nach einheitlichem Muster geschehen:

- Klassen werden stets mit Großbuchstaben begonnen.
- Methoden,Funktionen[1] und Variablen werden klein geschrieben.
- Konstanten werden komplett groß geschrieben.
- Falls eine Konstante aus mehreren Wörtern besteht, werden die Wörter mit einem „_“ getrennt
- Bei zusammengesetzten Worten wird jedes Teilwort mit Großbuchstaben begonnen.

1.3 Umbrüche und Einrückung

Der Code soll nach folgendem Schema strukturiert werden:

- Anweisungsblöcke werden in geschweifte Klammern gesetzt.
- Es folgt nach jeder geschweiften Klammer ein Zeilenumbruch.
- Eine Zeile darf höchstens 150 Zeichen lang sein.
- Unterschiedliche Codeblöcke werden mit einen Tabulator getrennt.
- Bei Instanziierungen wird vor und nach '=' ein Leerzeichen gesetzt.
- Kommentare für eine Funktion oder Klasse stehen immer in der vorherigen Zeile
- Bei jeder Funktion werden die Parameter und die return-Variable extra kommentiert
- Falls Zeile kommentiert wird folgt der Kommentar in vorheriger Zeile.

1.4 Änderungsdokumentation

Um den Überblick über Änderungen am Quellcode nicht zu verlieren, muss jeder commits angelegt und kommentiert werden. GIT übernimmt dabei die Versionsnummer sowie das Datum der Bearbeitung. Außerdem werden wir das Versionsprotokollsystem für den internen Austausch von Dokumenten benutzen. Dadurch wird eine bessere Übersicht über den Projektstatus erzeugt. Des Weiteren wird dadurch eine konfortablere Übersicht über die Dokumente erreicht.

---

1 Als Ausnahme gelten hier die Zustandsübergangsfunktionen, die zur besseren Übersicht mit „_“ getrennt werden (Beispiel: buchung_stornieren())

1.5 externe Dokumentation

Den Anwendern soll eine möglichst Benutzerfreundliche Bedienung über ein Screencast, in Form einer Schritt-für-Schritt Anleitung, ermöglicht werden. Außerdem wird es eine README geben, welche Installationsanweisungen bereithält.

2. Testkonzept

Um stabile und funktionale Software garantieren zu können, ist ein umfangreiches und durchdachtes Testkonzept einzelner Komponenten bis hin zum fertigen System unerlässlich. Dies ermöglicht uns ein frühzeitiges Aufspüren eventueller Fehler in den einzelnen Modulen bis hin zur fertigen Software.
Unsere Tests sind in 3 Ebenen gegliedert:

- Komponententest
- Integrationstest
- Systemtest

Man muss dabei zwischen mauellen und automatisierten Tests unterscheiden. Bei unserem Projekt spielen die manuellen Tests eine viel größere Rolle als die automatischen. Deswegen werden die drei Arten des Testen durch manuelle Tests realisiert. Die manuellen Tests werden von den jeweiligen Entwicklern vorgenommen. Dabei wird manuell mit bestimmten Werten die Funktionalität überprüft. Diese Tests sollen vorallem Grenzwerte und unvorhersehbare Eingaben testen.

2.1 Komponententest

Hier werden frühzeitig die einzelnen Komponenten wie Methoden, Funktionen, Klassen und Skripte getestet. Dies geschieht unabhängig von dem komplexen Gesamtprojekt. Es werden zum Beispiel die Eingabeparameter verschiedener Funktionen variiert. Mit solchen Mitteln soll nahezu jede Codezeile getestet werden.
Durch diese frühzeitige Fehlererkennung und deren Elimination haben haben wir eine sehr gute Ausgangssituation für ein erfolgreiches Projekt geschaffen. Auf dieser Ebene wird die Integration der einzelnen Komponenten in das Gesamtprojekt getestet. Besonderes Augenmerk werden wir auf die Schnittstellen der zu testenden Elemente sowie auf die Übergabe der verschiedenen Parameter legen.
Sobald eine Komponente, welche mit einer anderen interagiert, fertiggestellt ist, wird ein Integrationstest erstellt und ausgeführt.

2.2 Integrationstest

Auf dieser Ebene wird die Integration der einzelnen Komponenten in das Gesamtprojekt getestet. Besonderes Augenmerk werden wir auf die Schnittstellen der zu testenden Elemente sowie auf die Übergabe der verschiedenen Parameter legen.
Sobald eine Komponente, welche mit einer anderen interagiert, fertiggestellt ist, wird ein Integrationstest erstellt und ausgeführt.

2.3 Systemtest

Hier steht das Gesamtbild unserer Software aus Anwendersicht im Vordergrund. Die Software wird auf Funktionalität, Bedienbarkeit und ein einheitliches Konzept getestet. Im Projektvertrag vorgeschriebene Funktionen werden hier noch einmal getestet.
Da das System wöchentlich um weitere Funktionen wachsen soll, ist ein wöchentlicher Systemtest mit den dazugehörigen Tests bei jedem Releasebündel angestrebt.