<Titel des Praktikums>

<Titel des Teilprojektes>

Softwareentwicklungspraktikum

Sommersemester 2008

Testdokumentation

Auftraggeber

Technische Universität Braunschweig

<Name des Instituts>

<Name des Institutsleiters>

<Straße und Hausnummer>

<Postleitzahl und Ort>

Betreuer: <Name>

Auftragnehmer: <überzählige Zeilen löschen>

Name	E - Mail

Braunschweig, <Abgabedatum>

Versionsübersicht

Version	Datum	Autor	Status	Kommentar

Status: aktzeptiert oder nicht aktzeptiert

Kommentar: hier eintragen, was geändert bzw. ergänzt werden musste

I n h a l t s v e r z e i c h n i s

1 Einleitung

Wie in der Vorlesung erwähnt, ist das Testen von Software unerlässlich und darf bei keinem Softwareentwicklungsprozess fehlen.

Diese Testdokumentation ist angelehnt an den IEEE 829 – Standard, der als der bekannteste – wenn nicht sogar einzige – Standard für Software – Testdokumentationen gilt. Der Standard definiert eine Menge von Testdokumenten und beschreibt deren Inhalte. Es wird nicht festgeschrieben, welche Elemente eine Testdokumentation enthalten muss, sondern es obliegt dem Anwender (in diesem Fall dem SSE) zu entscheiden, welche Dokumente angefertigt werden müssen.

2 Testplan

Der Testplan ist das zentrale Dokument und wird daher als erstes erstellt. Hier wird der Umfang und Vorgehensweise beschreiben. Außerdem werden Testgegenstände und deren zu testenden Eigenschaften bzw. Funktionen identifiziert. Ferner werden die durchzuführenden Maßnahmen und die dafür verantwortlichen Personen definiert. Falls erforderlich sollte hier auch auf allgemeine Risiken eingegangen werden.

2.1 Zu testende Komponenten

Hier sind sämtliche zu testenden Objekte einschließlich der Versionsnummer bzw. Versionsnummer aufzuführen. Ebenso ist anzugeben, auf welchem Medium die Software vorliegt, ob dies einen Einfluss auf Hardwareanforderungen hat und ob die Software vor Testbeginn in irgendeiner Weise transformiert werden muss (z.B. ob sie von Tape auf Diskette kopiert werden muss). Außerdem wird auf zum Objekt gehörende Dokumentation der Komponente (Lasten-, Pflichtenheft, Grob- bzw. Feinentwurf) referenziert.

2.2 Zu testende Funktionen

Dieser Punkt beinhaltet alle Eigenschaften bzw. Funktionen und deren Kombinationen, die zu testen sind.

Sämtliche Funktionalitäten, die getestet werden sollen, werden hier aufgeführt. Dabei sind auf die vorangegangenen Dokumentationen zu referenzieren (Pflichtenheft, Grob- und Feinentwurf) und die dortigen Funktions-IDs zu verwenden!

***Beispiel:** /F100/ : Benutzer Login*

2.3 Nicht zu testende Funktionen

(optional; auszufüllen, falls es Funktionen gibt, die nicht getestet werden sollen)

*Hier werden alle Eigenschaften bzw. Funktionen und Funktionskombinationen aufgelistet, die nicht getestet werden. **Es sollte begründet werden, warum diese nicht getestet werden**. Es versteht sich von selber, dass alle Muss-Funktionalitäten des Pflichtenheftes (Abschnitt 1.1) getestet werden müssen.*

2.4 Vorgehen

Die allgemeinen Vorgehensweisen für die einzelnen zu testenden Funktionen und Funktionskombinationen werden hier beschrieben. Die Beschreibung sollte detailliert genug sein, um die Hauptaktivitäten und deren Zeitbedarf abschätzen zu können.

Es ist zu beachten, dass für alle wichtigen Funktionalitäten das Verfahren angegeben wird. Dies gewährleistet, dass diese Funktionalitäten adäquat getestet werden.

Es ist zu dokumentieren, welche Aktivitäten, Techniken und Werkzeuge benötigt werden, damit die Funktionalitäten getestet werden können.

Beispiel für Vorgehen (unvollständige Liste):

a) Komponenten- und Integrationstests

Klassen werden mit JUnit-Testfällen geprüft. Vor Beginn der Implementierung werden bereits Blackbox-Testfälle erstellt, die dann begleitend zur Implementierung genutzt werden ("Test first"). Nach Abschluss der Implementierung einer Komponente wird diese dann durch Whitebox-Tests geprüft.

Der Integrationstest der Klassen und Komponenten erfolgt nach dem Bottom-Up-Prinzip. Anfangs muss die Integration der Datenbankanbindung und den entsprechenden Data-Access-Objects (DAO) geprüft werden, da das Mapping der Datenbank auf Objekte die unterste Schicht des Projektes bildet. Dieser Testabschnitt wird durch die Schnittstellentests abgedeckt.

Die Komponenten werden damit unter Berücksichtigung ihrer Abhängigkeiten konkret in folgender Reihenfolge integriert: …

(Hier kommt das konkrete Vorgehen bei der Integration: Welche Klassen werden zusammen getestet, welche kommen dann dazu etc. Das kann man z.B. auch schön in Form eines Baumes aufzeigen.)

b) Funktionstests

Die Anwendungsfälle aus der Anforderungsspezifikation werden über das Web-Interface geprüft. Mindestanforderung hierfür ist es, jeden Fall einmal auf seine korrekte Funktionalität zu testen.

c) …

2.5 Testumgebung

Die genutzte Testumgebung(en) bitte hier angeben und kurz beschreiben.

Beispiel: *JUnit Testsuite, lokal installierter Web Application Server, …*

3 Testdurchführung

In diesem Abschnitt werden die einzelnen Testfälle beschrieben und die Durchführung protokolliert.

Ein Testfall ist eine Kombination von Eingabedaten, Bedingungen und erwarteten Ausgaben, die einem bestimmten Zweck erfüllen. Man prüft z.B., ob Vorgaben in einem Spezifikationsdokument eingehalten werden oder ob der Programmablauf tatsächlich dem erwarteten Pfad entspricht

Es sind so viele Abschnitte einzufügen, wie es Testfälle gibt.

3.1 Testfall – ID und Bezeichnung

Jeder Testfall erhält eine eindeutige Identifikation mit Kurzbezeichnung.

Beispiel: /T100/ Lager anlegen

3.1.1 Zu testende Objekte und Methoden

Hier sind alle Testobjekte und Methoden zu beschreiben, die von diesem Testfall ausgeführt werden. Testobjekte können dabei z.B. auch separate Softwaremodule oder einzelne Webseiten sein.

3.1.2 Kriterien für erfolgreiche bzw. fehlgeschlagene Testfälle

Es sind die Kriterien anzugeben, mit denen man feststellt, dass der Testfall erfolgreich bzw. fehlgeschlagen ist.

3.1.3 Einzelschritte

Es ist zu beschreiben, was zu tun ist, um einen Testlauf vorzubereiten und ihn zu starten.

Ggf. sind erforderliche Schritte während seiner Ausführung anzugeben (z.B. Benutzerinteraktion über ein User-Interface). Ferner ist zu beschreiben, was zu tun ist, um den Testlauf ordnungsgemäß oder im Falle unvorhergesehener Ereignisse anzuhalten (falls er nicht von selbst terminiert).

Ggf. sind Aufräumarbeiten zu beschreiben, um nach den Tests den ursprünglichen Zustand wiederherzustellen (falls der Testlauf nicht seiteneffektfrei ist)

3.1.4 Beobachtungen / Log

Es sind alle speziellen Methoden oder Formate zu beschreiben, mit denen die Ergebnisse der Testläufe, die Zwischenfälle und sonstige wichtige Ereignisse aufgenommen werden sollen.

Beispiel: Logdatei eines Servers, Messung der Antwortzeit eines Remote Terminals mittels Netzwerk Simulator, …

3.1.5 Besonderheiten

(optional; auszufüllen, falls es Besonderheiten in diesem Testfall gibt)

Testfallspezifische Besonderheiten, z.B. Ausführungsvorschriften oder Abweichungen von der Testumgebung (siehe 2.5) werden hier aufgelistet.

3.1.6 Abhängigkeiten

(optional; auszufüllen, falls es Abhängigkeiten in diesem Testfall gibt)

Ist dieser Testfall von der Ausführung anderer Testfälle abhängig, so werden diese Testfälle hier aufgelistet und kurz beschrieben, worin die Abhängigkeit besteht.

3.1.7 Testlauf 1

Nachfolgend wird beschrieben, wie der Testfall ausgeführt wurde und welches Ergebnis er geliefert hat. Da es bei Korrektur von Softwarefehlern oder anderen Gegebenheiten notwendig ist, einen Test mehrfach durchzuführen (Testläufe), ist jede Testdurchführung zu dokumentieren. Daher ist dieser Abschnitt für ***jeden Testlauf*** *zu erstellen und* ***fortlaufend zu nummerieren****.*

3.1.7.1 Eingaben

Es sind alle Eingabedaten bzw. andere Aktionen aufzuführen, die für die Ausführung des Testfalls notwendig sind.

Diese können sowohl als Wert angegeben werden (ggf. mit Toleranzen) als auch als Name, falls es sich um konstante Tabellen oder um Dateien handelt. Außerdem sind alle betroffenen Datenbanken, Dateien, Terminal Meldungen, etc. anzugeben.

Hinweis: Es sind nicht noch mal die Einzelschritte aus 3.1.3 zu wiederholen. Während jene allgemeiner sind (z.B. „Einloggen über das Login-Formular“) sind hier die konkreten eingegebenen Testdaten aufzuführen (z.B. „Loginname: test; Passwort: xxxtest“).

3.1.7.2 Soll - Reaktion

Hier ist anzugeben, welches Ergebnis bzw. Ausgabe der Test haben soll.

Hinweis*: Es sind nicht noch mal die Erfolgskriterien aus 3.1.2 zu wiederholen. Während jene allgemeiner sind (z.B. „Testnachricht wird über Netzwerkkanal empfangen“) sind hier die konkreten erhaltenen Testdaten aufzuführen (z.B. „Konsole zeigt Meldung: ‚Testnachricht 123 erhalten“).*

3.1.7.3 Ist – Reaktion

Hier ist anzugeben, welches Ergebnisdaten bzw. Ausgaben dieser Testlauf geliefert hat.

3.1.7.4 Ergebnis

Für jeden Testlauf ist zu vermerken, ob der Test erfolgreich durchgeführt werden konnte oder nicht. Einen missglückten Test bitte begründen, sofern der Grund des Fehlschlags bekannt oder offensichtlich ist.

3.1.7.5 Unvorhergesehene Ereignisse während des Testlaufs *(optional; nur anzugeben, falls es unvorhergesehene Ereignisse gab)*

3.1.7.6 Nacharbeiten

Ist ein Testlauf nicht erfolgreich durchgeführt worden, so werden hier die erforderlichen Nacharbeiten aufgeführt (z.B. Bugfixes).

4 Zusammenfassung

Hier wird eine Zusammenfassung der Testergebnisse aufgelistet. Dabei sind alle behandelten Probleme aufzulisten und darzustellen, wie ihre Lösungen erreicht wurden.

Dabei kann auf folgendes eingegangen werden:

- *Zusammenfassung aller Hauptaktivitäten und der wichtigsten Ereignisse und Ergebnisse der Tests*
- *Abweichungen der vorliegenden Software von der Aufgabenstellung. Aber auch Abweichung vom Testplan oder den Testfällen während des Testens (z.B. aufgrund von veränderter Funktionalität während der Nacharbeiten). Die Abweichungen sollten begründet werden.*
- *Umfang der Testverlaufs (Vollständigkeit) im Vergleich zu Umfangskriterien des Testplans. Auflistung der Funktionalitäten, die nicht getestet wurden. Selbstverständlich mit Begründung.*
- *Bewertung der Softwarequalität*