QSE-ASE-08
Tenez
Testplan

Alexander Bohn
Manuel Djalili

11. November 2009

Inhaltsverzeichnis

Kapitel 1

Einleitung

Hinweis: Das vorliegende Dokument wurde in Anlehnung an das von Dipl.-Ing. Denis Frast am Institut QSE an der TU - Wien bereitgestellte Template erstellt. Dieses beruht wiederum auf dem im Softwareentwicklungsmodell stdSEM der Firma Siemens PSE vorgeschriebenen Testplan-Template.

1.1 Zweck des Dokuments

Im vorliegenden Dokument sind die Testmaßnahmen für das Projekt Tenez dokumentiert. Festgelegt sind sowohl die Vorgehensweisen bei der Organisation und Planung, als auch die Art und Weise der Testdurchführung.

1.2 Begriffsbestimmungen und Abkürzungen

Testobjekt
Unter dem Begriff versteht man die Einheit, die dem Test unterzogen wird. Das kann ein beliebiger Produktteil, eine vorintegrierte Einheit oder das gesamte zu testende Produkt sein. Mit dem Begriff Öbjektïm Sinne der objektorientierten Entwicklung kann der Begriff Testobjekt somit nicht gleichgesetzt werden (sehr wohl kann aber ein Objekt aus der Welt der objektorientierten Entwicklung auch Testobjekt sein).

Testfall

- Ein Testfall muss eindeutig identifizierbar sein und muss folgende Angaben enthalten:
- Testtyp (NF = Normalfall, FF = Fehlerfall, SF = Sonderfall)
- Vorbedingungen, Bedingungen die das Testobjekt vor Anwendung des Testfalls erfüllen muss
- Beschreibung Testfall (Test-Script bei automatisierten Tests)
- Äquivalenzklassen (Angabe der verwendeten Klassen für den Testfall)
- Erwaretetes Ergebnis (welche Ergebnisse werden als Reaktion auf die im Testverlauf eingegebenen Daten erwartet?)

Black Box vs White Box Tests
Black-Box-Methoden konzentrieren sich auf die Benutzeranforderungen und ignorieren die interne Programmstruktur. Reine White-Box-Methoden benutzen die Programm-Struktur und vernachlässigen die Anforderungen. Meist findet man eine Mischform: Testfälle können beispielsweise auf die Anforderungen basierend definiert werden, und dann kann ihre Anzahl durch Wissen über die interne Programm-Struktur optimiert werden.

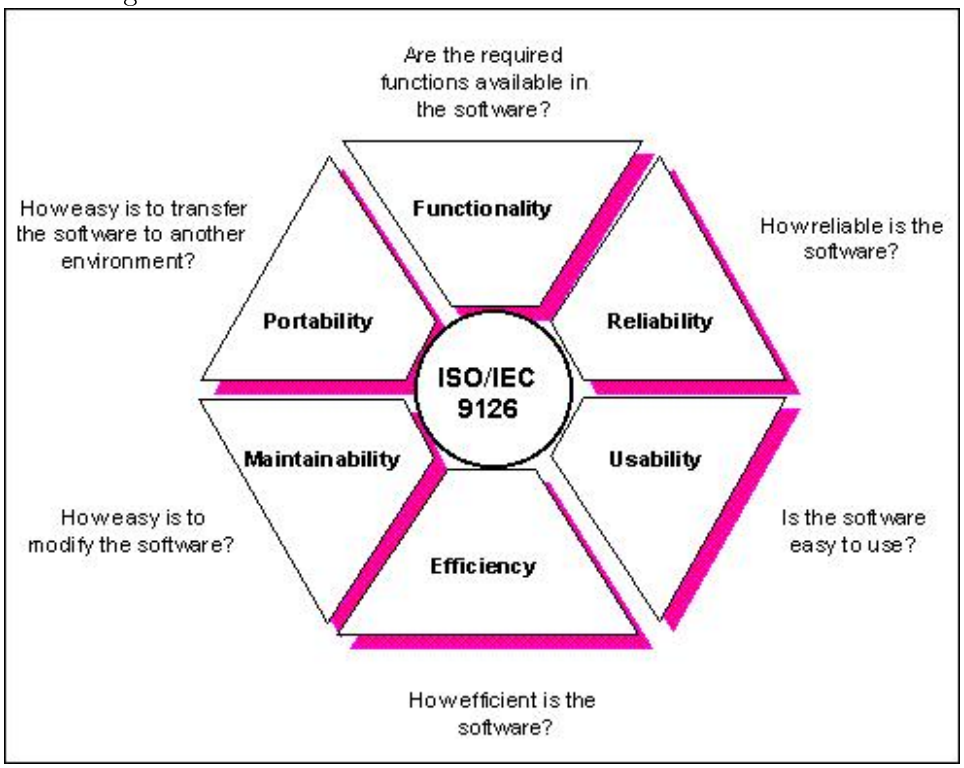
Usability-Test
Diese Tests sollen überprüfen ob die zukunftige Benutzergruppe mit dem System effizient und effiktiv umgehen kann und damit zufrieden ist. Eine Möglichkeit das zu testen wäre es, einer Menge von Benutzergruppen eine Vorabversion des Systems ausprobieren zu lassen. Eine Andere Benutzer und Entwickler/Tester erstellen gemeinsam Szenarien. Auch das führen von Checklisten kann zur Überprüfung der Benutzbarkeit eines Systems dienen.

Kapitel 2

Testanforderungen

Alle gegebenen Testanforderungen für das Tenez Projekt werden nun in diesem Abschnitt behandelt. Unsere Qualitätskriterien richten sich nach ISO 9126 (siehe Abb.: 1). Aufgrund dieser Kriterien wird unsere Software getestet und daher die notwendigen Testanforderungen abgeleitet. Die Intensität der Tests richtet sich nach den für das System relevanten Qualitätskriterien und dem gegebenen Zeit- und Budgetrahmen (siehe auch Kapitel 4.1 und 5.1).

Abbildung 1:



Folgende Testanforderungen für die relevanten Qualitätskriterien sind für das Projekt von Wichtigkeit(nach absteigender Relevanz):

Funktionalität - Richtigkeit
Alle implementierten Funktionen müssen zum gewünschten Ergebnis für den Enduser kommen. Daher eine An- und Abmeldung in das System muss reibungslos funktioneren. Eine

Reservierung, wenn für bestimmte Zeiteinheit möglich, soll ohne Problem durchführbar sein. Vereine sollen angelegt und editiert werden können. Plätze sollen mit Timeslots versehen und entsprechend editiert werden können. Individuelle Skins müssen für Vereine uploadbar sein und noch all die anderen Funktionen, welche in den Use Cases abgebildet sind müssen einwandfrei funktionieren. Um dies zu gewährleisten werden für alle Funktionen Testfälle nach dem Black-Box Prinzip erstellt und ausgeführt

Funktionalität - Interoperabilität
Das Hallenverwalltungssystem muss auf Servern von Windows und den bekanntesten Lnux Distributoren reibungslos funktionieren.

Funktionalität - Sicherheit
Ein unberechtigter Zugriff darf auf das System keinesfalls möglich sein. Daher muss die Abfrage von Benutzername und Passwort reibungslos funktionieren und bei Falscheingabe der Systemzutritt verweigert werden. Daten dürfen zudem nicht unbemerkt verändert werden, daher muss die Protokollierung genauestens Aufzeichnungen über Buchungsvorgänge und jedwegliche Zugriffe auf das System führen. Zum Testen werden entsprechende Testfälle generiert um das System auf Sicherheit zu überprüfen

Zuverlässigkeit - Fehlertoleranz
Bei Eingabe von nicht vorhergesehenen Daten seitens des Benutzer soll das System mit Fehlermeldungen reagieren die den Benutzer sofort Aufschluss über das notwendige Korrekturverfahen geben und nicht mit den fehlerhaften Daten weiter arbeiten sondern in den ursprünglichen Zustand vor der Eingabe zurückkehren. Zudem muss das System bei auftretenden Software Fehlern (zb.: Code nicht mehr vorhanden, Datenbank nicht mehr persistent) entsprechende Exceptions werfen und den Vorgang stoppen. Für Testzwecke werden wiederum entsprechende Testfälle definiert und ausgeführt.

Zuverlässigkeit - Robustheit
Das System soll auch unter ungünstigen Suchanfragen (zb.: Kunde an letzter Stelle in der Datenbank) zu einem schnellen Ergebnis kommen. Zudem muss das System gegenüber jedweglichen Falscheingaben seitens des Benutzer "robust"reagieren und Systemabstürtze vermeiden. Mittels relevanten Testfällen wird dieses Systemverhalten getestet.

Zuverlässigkeit - Wiederherstellbarkeit
Bei nichtbefugten Änderungen der Datenbank oder nicht gewollten, muss es möglich sein die gesamte Datenbank wieder in den Ursprünglichen Zustand vor der Änderzung zu versetzen. Daher Backups der Datenbank(en) müssen vom System automatisch und korrekt durchgeführt werden. Entsprechende Testfälle werden generiert und die Rollback Funktion der Datenbank auf Korrektheit überprüft

Benutzbarkeit - Verständlichkeit, Erlernbarkeit, Bedienbarkeit
Das gesamte Hallenreservierungssystem soll für den Benutzer(Systemadministrator, Vereinsverwalter und Kunde) einfach zu verstehen, erlern- und bedienbar sein. Hierfür werden regelmäßig Absprachen mit dem Auftraggeber gehalten und seine Wünsche diesbezüglich 1:1 in die Software übertragen. Entsprechende Usability-Tests sollen diese Qualitätsanforderungen gewährleisten.

Effizienz - Zeitverhalten

Das System soll auf entsprechend Anfragen und Bearbeitungen seitens des Benutzer mit möglichst geringen Zeitaufwand verarbeiten. Dies wird sichergestellt in dem das Entwicklerteam laufend die Verarbeitungszeiten des Systems testet und bei Problemenen entsprechend reagiert.

Effizienz - Verbrauchsverhalten
Die Applikation soll möglichst behutsam mit Speicher- und CPU-Auslastung umgehen, da eine Installation auf einem V-Server möglich sein muss.

Übertragbarkeit - Anpassbarkeit, Installierbarkeit
Das System soll ohne weiter Probleme auf den gängisten Serversystemen laufen können. Dafür wird getestet mit welchen Aufwand das System auf den jeweiligen Servern installiert werden kann.

Die restlichen Qualitätskriterien sind für das tenez Projekt im Vergleich zu den obigen wenig relevant und werden nicht spezifisch getestet. Die Tester sollen jedoch – vor allem bei Durchführung von explorativen Tests – auch nach Fehlern, die diese Kriterien betreffen, Ausschau halten und sie gegebenenfalls melden.

Kapitel 3

Testziele

Die im folgenden genannten Testziele beziehen sich auf die vorher definierten Testanforderungen. Zu jeder dieser Anforderungen werden Testziele definierte die eine festgelegte Mindestanforderung erfüllen müssen. Werden diese Mindestanforderungen nicht erfüllt kann kein Mindestmaß an Qualität im System erfüllt werden. Desweiteren gibt es für jedes Testziel auch eine Höhere Intensität, welche erst nach erreichen der Mindestanforderungen realisierbar ist und für alle Testziele erstrebenswet ist.

3.1 Funktionalität - Richtigkeit

Zum Testen dieses Kriteriums werden Testfälle nach der Black Box Methode erstellt und durchgeführt.

Mindestanforderung: Alle im System vorkommenden Methoden müssen auf Richtigkeit getestet werden. Dazu müssen alle möglichen Äquivalenzklassen der Eingabedaten in den Testfällen mindestens einmal zur Anwendung kommen.
Höhere Intensität: Hier wird die Überdeckung von Kombinationen von Anforderungen verlangt

3.2 Funktionalität - Interoperabilität

Zum Testen dieses Kriteriums muss das System auf verschiedenen Serversystemen zum laufen gebracht werden.

Mindestanforderung: Das Reservierungssystem muss reibungslos auf dem Server des Kunden zum laufen gebracht werden.
Höhere Intensität: Das System wird auf mehreren Windows und Linux Servern installiert und auf Funktionsfähigkeit überprüft werden.

3.3 Funktionalität - Sicherheit

Zum Testen dieses Kriteriums werden Testfälle generiert die das System auf ihrer Sicherheit weitgehend vollständig überprüfen. Da ein System immer zu 100 Prozent sicher sein muss gielt die Höhere Intensität gleichzeitig als Mindestanforderungen und wird daher ganz ausgeklammert.

Mindestanforderung/Höhere Intensität: Implementierung und Durchführung von Testfällen für alle möglichen Benutzereingaben ob diese auf Serverseite vollständig kontrolliert werden um XSS Angriffe erst gar nicht zu ermöglichen.

3.4 Zuverlässigkeit - Fehlertoleranz

Zum Testen dieses Kriteriums werden Testfälle generiert welche absichtlich zu einem Fehler führen um das Verhalten des Systems in einem Fehlerfall evaluieren zu können. Desweiteren werden absichtlich Code Teile entfernt und Datenbanken entsprechend manipuliert um das Systemverhalten in einem solch schwerwiegenden Fall zu testen.

Mindestanforderung: Für jede Methode muss es einen Testfall geben der diese auf alle möglichen in Äquivalenzklassen zusammengefassten Falscheingaben testet. Desweiteren dürfen Testfälle nach einer Manipulation von Code/Datenbanken nicht mehr erfolgreich durchführbar sein.
Höhere Intensität: Das System muss auch gegenüber einer bestimmten Anzahl von kombinierten Fehlerzuständen tolerant sein.

3.5 Zuverlässigkeit - Robustheit

Zum Testen dieses Kriteriums werden Suchanfragen und Falscheingaben aus Benutzersicht durchgeführt.

Mindestanforderung Ungünstige Suchanfragen werden zeitlich gemessen und dürfen eine vorher festgelegte Dauer nicht überschreiten. Falscheingaben aus Benutzerischt werden durchgeführt um das System auf Stabilität auch nach eines Fehlerzustandes zu testen.
Höhere Intensität: Auch bei einer größeren Anzahl gleichtzeitiger Userzugriffen soll das System reibungslos funktionieren. Zum Testen müssen mindestens 10 Personen gleichzeitg eine Anfrage auf das System durchführen.

3.6 Zuverlässigkeit - Wiederherstellbarkeit

Datenbanken sollen jederzeit wieder in einen vorherigen Zustand gebracht werden können.

Mindestanforderung Eine fehlerhafte oder korrupte Datenbank soll mittels einer einfachen Anweisung wieder in einen fehlerlosen Zustand gebracht werden können.
Höhere Intensität Alle Änderungen ab einem bestimmten Zeitpunkt x müssen durch einen bestimmten Methodenaufruf wieder rückgängig gemacht werden.

3.7 Benutzbarkeit - Verständlichkeit, Erlernbarkeit, Bedienbarkeit

Einer Gruppe von Endbenutzer muss das Programm zur Verfügung gestellt werden. Die Benuter sollen dann vorher definierte Aufgaben mit dem Programm lösen(zb.: Verein anlegen, Reservierung vornehmen). Die benötigte Zeit für die zu erledigenden Aufgaben wird gemessen und mit einer vorher gestgelegten Zeit verglichen. Sind mehr als 25 Prozent der Benutzer nicht in der Lage die Aufgaben innerhalb der vorher festgelegten Zeiten zu lösen gielt der Test als nicht bestanden.

Mindestanforderungen 5 Personen aus dem Umfeld des potentiellen Kundenkreises sollen für diese Test herangezogen werden und vorher festgelegte Aufgaben in der entsprechenden Zeit ausführen.
Höhere Intensität Neben dem Zeitrahmen sollen die Benutzer selbst Noten von 1-5 für die Bedienbarkeit vergeben. Ergiebt das Mittel der Noten einen höheren Wert als 2 gielt dieser Test als nicht bestanden. 10 Benutzer sollen für diesen Test herngezogen werden.

3.8 Effizienz - Zeitverhalten

Die Ausführzeiten auf bestimmte Anfragen gegenüber dem Systems werden getestet.

Mindestanforderungen: Das Entwicklerteam testet nach erfolgreicher Implementierung eines jeden Features die benötigte Verarbeitungszeit für das System, bei vorher definierter Anfrage. Benötigt eine Anfrage eine zu große Menge an Zeitressourcen, schlägt der Test fehl und der Enwickler muss den code entsprechend umgestalten und eventuell efektivier Algorithmen zur Abwicklung einer Anfrage implementieren.
Höhere Intensität: Das Zeitverhalten einzelner Funktionen soll bei einer größeren Anzahl an Aufrufen getestet werden.

3.9 Effizienz - Verbrauchsverhalten

Die Applikation soll möglichst behutsam mit Speicher- und CPU-Auslastung umgehen, da eine Installation auf einem V-Server möglich sein muss.

Mindestanforderungen: In der finalen Konfiguration sollte das System im Betrieb nie mehr als 96 Megabyte Speicher pro Vereinsinstanz belegen.
Höhere Intensität: Speicherverbrauch unter 96 Megabyte.

3.10 Übertragbarkeit - Anpassbarkeit, Installierbarkeit

Unerfahrene Benutzer sollen das System ohne größere Probleme installieren können.

Mindestanforderungen: 5 programmiererfahrene Benutzer, jedoch ohne Helma Kenntnisse und Kenntnisse über das Reservierungssystem, sollen das System auf den Server installieren.
Höhere Intensität: Noten zwischen 1 bis 5 sollen von den Benutern über die Einfachheit der Installation vergeben werden. Ergibt das Mittel mehr als 2 schlägt der Test fehl.

Kapitel 4

Testdurchführung

4.1 Zu testende Komponenten

Das gesamte Tenez Reservierungssystem wird getestet

4.2 Zu testende Funktionen

Zu testen sind alle vom Team erstellen oder veränderten Methoden des Projekts. Um die Funktionalität zu überprüfen werden Black Box Tests durchgeführt und bei Bedarf durch White Box Tests ergänzt.

4.3 Nicht zu testende Funktionen

Keine

4.4 Testprotokollierung

Zweck der Testprotokolle ist es festzustellen wer, wann, welche Tests durchgeführt hat und mit welchen Ergebnis. Hierzu wird ein einfaches Latex Template verwendet in dem folgende Felder für jeden durchgeführten Test unbedingt ausgefüllt werden müssen:

- Name des Testers
- Datum des Tests
- Programmversion
- Zu testende Funktion
- Zu testende Methode
- Testergebns (ok/nicht ok)
- Begründung Testergebnis (Warum nicht ok!)

4.5 Testberichte

Testberichte sind von jedem Gruppenmitglied regelmäßig zu verfassen und beinhalten im Anhang alle durchgeführten Testfälle inklusiver Protdokollierung. In einem Testbericht müssen alle auftretende Fehler genauestens beschrieben und die Ergebnisse der Testdurchläufe interpretiert und bewertet werden. Daher, folgende Daten müssen in einem Testbericht abgedeckt werden:

- Anzahl der Testfälle
- Interpretation der Ergebnisse
- Gesamter Testaufwand
- Weiteres Vorgehen(Soll - Ist Vergleich)
- Abschließendes Resümee

Kapitel 5

Testmanagement

5.1 Zeitplan

Testdurchläufe werden nach Fertigstellung jeder Methode durchgeführt. Es werden alle Artefakte die laut Projektplan zu einem bestimmten Zeitpunkt fertig gestellt sein müssen laufend getestet.

Vorgang	Beginn	Testgegenstand	Ende
Implementierung 0.1	09.11.2009	Alle Methoden	08.12.2009
Implementierung 1.0	10.12.2009	Alle Methoden	09.01.2010
System Tests u. Usability Tests	11.01.2010	Fertiges Programm	16.01.2010

5.2 Personal

Jede Methode wird nach Implementierung sofort vom jeweiligen Programmierer selbst getestet und ein Testprotokoll erstellt.

5.3 Risiken und Risikomanagement

Im folgenden werden alle relevanten Risiken die im Zuge der Testabwicklung auftreten können dargestellt.

5.3.1 Nicht testen einer Methode

Annahme: Der Programmierer hat die entsprechenden Tests für eine Methode nicht erstellt.
Maßnahme: Programmierer muss den Test sofort nach implementieren, notfalls muss ein anderers Gruppenmitglied die Tests nachträglich erstellen.

5.3.2 Testfälle sind unvollständig

Annahme: Nicht alle möglichen Varianten um bestimmte Funktionen zu testen sind in den Testfällen abgebildet
Maßnahme: Ensprechend die Testfälle erweitern.

5.3.3 Testfehler kann nicht behoben werden

Annahme: Der Programmierer findet den Lösungsweg auch nach mehreren Stunden nicht, zb.: aufgrund von Unkenntnissen mit Helma.
Maßnahme: Bevor mehrere Stunden für unsinniges Suchen verloren gehen, muss dies der Gruppe mitgeteilt werden um dann gemeinsam zu versuchen den Fehler zu beheben.

5.4 CM-Regelungen für den Test

Bei einem Fehlerfall wird dieser sofort in der Testprotokollierung begründet und im nachfolgenden natürlich im Testbericht erfasst. Da der Programmierer gleichzeitg der Tester ist, muss dieser auch für die Behebung des Fehlers sorgen.

5.5 Testabschluss

Für den Testabschluss müssen alle Mindestanforderungen der Testziele aus Kapitel 3 erfüllt werden. Die vorher definierten Testfälle müssen komplett abgearbeitet sein sowie alle Testprotokolle und Testberichte vollständig vorliegen.

Kapitel 6

Anhang

6.1 Version

Version 0.3 Alexander Bohn - 25.11.2009 - Testplan vollständig
Version 0.2 Alexander Bohn - 22.11.2009 - Kapitel 4 und 5 vervollständigt(Zeitplan fehlt noch)
Version 0.1 Alexander Bohn - 13.11.2009 -Testplan bis Kapitel 4