Università degli Studi di Salerno

Corso di Ingegneria del Software

Football League Manager

Test Plan

Versione 1.5

Data: 30/01/2017

Partecipanti

Nome	Matricola
Nocerino Christian Giuseppe	0512103250
Manzo Gerardo	0512102974
Sarubbi Raffaele	0512102934
Capaldo Giovanni	0512102980
Iannuzzi Serena	0512103216

Scritto da	Raffaele Sarubbi

Revision History

Data	Versione	Descrizione	Autore
02/01/2017	1.0	Aggiunte sezioni: 1. Introduction 2. Relationship to other documents (2.1, 2.2, 2.3)	1. Gerardo Manzo 2. Serena Iannuzzi 3. Raffaele Sarubbi
05/01/2017	1.1	Aggiunta sezione: System Overview	1. Raffaele Sarubbi
11/01/2017	1.2	Aggiunte sezioni: 1. System Overview 2. Features to be tested/not to be tested (4.1, 4.2)	1. Serena Iannuzzi 2. Raffaele Sarubbi 3. Christian Giuseppe Nocerino
17/01/2017	1.3	Aggiunte sezioni: 1. Pass/Fail criteria (5.1, 5.2) 2. Approach (6.1, 6.2, 6.3)	1. Gerardo Manzo 2. Christian Giuseppe Nocerino 3. Raffaele Sarubbi
24/01/2017	1.4	Aggiunte sezioni: 1. Suspension and Resumption (7.1, 7.2) 2. Testing materials	1. Gerardo Manzo 2. Christian Giuseppe Nocerino 3. Raffaele Sarubbi

30/01/2017	1.5	Aggiunte sezioni: 1. Testing Materials 2. Testing Schedule	1. Gerardo Manzo 2. Christian Giuseppe Nocerino 3. Serena Iannuzzi

Indice

1. Introduction

La realizzazione di un buon sistema software richiede che al centro dei processi di sviluppo, manutenzione ed evoluzione del software stesso vi sia il concetto di qualità. Il Testing rappresenta una delle più importanti tecniche per verificare la qualità del software, in quanto consente di analizzare, valutare e promuovere il miglioramento della correttezza dell'implementazione con riferimento alle caratteristiche definite in particolare dal modello dei requisiti software. Lo scopo di tale attività è quello di provare il sistema e rilevare problemi; si intende dunque massimizzare il numero dei test effettuati, in modo tale da poter correggere un numero cospicuo di errori. E' importante definire i casi d'uso su cui verranno testate le funzionalità del sistema. Tale attività va in contrasto con quelle svolte in precedenza: analisi, design, implementazione sono attività costruttive mentre il testing tenta di "rompere il sistema". Questo documento si basa sulla descrizione della pianificazione del testing, delle tecniche utilizzate e dei risultati ottenuti in fase di esecuzione. In questo senso, il documento mira a valutare l'aspetto qualitativo del software sviluppato attuando tecniche di analisi dinamica, cioè tecniche utilizzate per osservare il comportamento del sistema in fase di esecuzione sulla base di particolari dati di ingresso.

2. Relationship to other documents

Le relazioni tre il Test Plan e gli altri documenti sviluppati per la costruzione del sistema FLM hanno una grande importanza. Sono necessarie al fine di produrre una fase di testing coerente con tutte le specifiche dedotte durante l'analisi e sia le decisioni che gli obiettivi che caratterizzano il System Design e la fase di Object Design. Inoltre, sono importanti anche per il raffinamento dei requisiti e delle funzionalità che il sistema dovrà fornire.

2.1. Relations with the Requirements Analysis Document

I test dovranno tenere conto delle specifiche espresse nel RAD. Si darà rilevanza ai requisiti non funzionali ed ai vari modelli prodotti in fase di analisi dei requisiti.

2.2. Relations with the System Design Document

Il testing dovrà garantire la coerenza tra il software e gli obiettivi di design definiti in fase di System Design, specificati nel SDD. Si analizzeranno, quindi, i conflitti e le inconsistenze presenti tra le componenti del sistema testate e tali obiettivi. Si analizzerà, inoltre, la struttura del sistema al fine di scoprire le differenze presenti tra essa e quella prevista durante la fase di System Design.

2.3. Relations with the Object Design Document

La fase di testing dovrà considerare il contenuto del documento di Object Design, in quanto quest'ultimo rappresenta la base per la realizzazione dell'implementazione, fondamentale per il testing. Pertanto, sarà necessario effettuare il testing delle unità per individuare le differenze tra ciò che è stato stabilito in fase di Object Design ed il sistema effettivo.

3. System Overview

FLM è un sistema web-based che permette la creazione e la relativa gestione di un campionato. Tra i suoi sottosistemi è presente quello volto ad ospitare le funzionalità relative alla gestione del database. Inoltre, sono presenti anche un sottosistema per le funzionalità offerte all'amministratore del sistema ed un sottosistema per le funzionalità offerte agli utenti registrati e non.

Le caratteristiche da testare per il controllo del corretto funzionamento di ciascuna funzionalità saranno:

- Robustezza: capacità del sistema di reagire fornendo input non valido per il dominio applicativo.
- Usabilità: analisi di ogni forma di interazione corrisposta da messaggi di aiuto (in caso di errore) o di notifica (in caso di operazioni eseguite con successo) .
- Sicurezza: verrà testato che un utente non possa intaccare dati non inerenti alla specifica dell'operazione o accedere a dati non consentiti.
- Correttezza: verrà testato che le operazioni vengano eseguite correttamente così come dalla specifica dei requisiti sono stati designati.

4. Features to be tested/not to be tested

4.1. Features to be tested

Tutte le funzionalità il sistema FLM offre e che sono state espresse nelle fasi di Raccolta e di Analisi dei Requisiti sono state implementate e saranno oggetto di testing.

4.2. Features not to be tested

Non saranno invece oggetto di testing:

- Durante le fasi Unit Testing si eviterà in genere di testare tutto ciò che l'IDE genererà in automatico (metodi getter e setter), concentrandosi sui metodi più rilevanti
- Componenti off-the-shelf.

5. Pass/Fail criteria

5.1. Pass criteria

Se nella fase di testing non si nota un comportamento diverso da quello richiesto dal cliente e previsto dagli sviluppatori del sistema, allora il test interessato ha insuccesso.
I criteri di successo del sistema software previsti per la fase di testing sono i seguenti:

- Uguaglianza tra comportamento atteso e comportamento previsto dall'oracolo
- Mancanza di eccezioni

5.2. Fail criteria

Se nella fase di testing si nota un comportamento diverso da quello richiesto dal cliente e previsto dagli sviluppatori del sistema, allora il test interessato ha successo.
I criteri di insuccesso del sistema software previsti per la fase di testing sono i seguenti:

- Differenza tra comportamento atteso e comportamento previsto dall'oracolo
- Presenza di eccezioni

6. Approach

Le tipologie di testing che si svolgeranno per testare il software FLM saranno tre:

- Unit Testing
- Integration Testing
- System Testing

6.1. Unit Testing

Sarà effettuato un testing sulle singole classi di ogni sottosistema. Per ogni componente verranno costruiti uno o più test case, che saranno di due differenti tipologie:

- Black Box: Si focalizza sul comportamento di I/O, senza preoccuparsi della struttura interna della componente. Se ogni dato input, siamo in grado di prevedere l'output, allora il modulo supera il test.
- White Box: si focalizza sulla completezza. Ogni statement nella componente è eseguita almeno una volta.Indipendentemente dall'input, ogni stato nel modello dinamico dell'oggetto e ogni iterazione tra gli oggetti viene testata

6.2. Integration Testing

Seguendo una metodologia bottom-up, a partire dai sottosistemi indipendenti, si integrano tutte le classi ad esso relativo e si procede con il testing. Si continua aggiungendo i sottosistemi che presentano dipendenze fino a raggiungere l'integrazione dell'intero sistema.

6.3. System Testing

Viene eseguito sull'intero sistema, con l'obiettivo di determinare se il sistema rispecchia i requisiti (funzionali e globali).

7. Suspension and Resumption

Nel testare il sistema FLM, potranno esserci casi in cui bisogna sospendere il testing e casi in cui bisogna ripristinarlo. Dato che saranno svolte tre tipologie differenti di testing, che hanno strategie diverse e coinvolgono i segmenti del software in maniera diversa l'una dall'altra, per ognuna di queste tre diverse tipologie ci saranno differenti scelte di sospensione e ripristino del testing.

7.1. Suspension

Le attività di testing saranno sospese ogni qualvolta si osservano differenze tra comportamento osservato e comportamento richiesto. La sospensione riguarderà soltanto il testing di tale unità. In tale caso, si procederà alla correzione dei difetti riscontrati.

7.2. Resumption

Le attività di testing saranno riprese soltanto quando i difetti riscontrati saranno corretti. Inoltre, ad ogni ripristino di testing, saranno effettuati nuovi test su ciò che si è appena corretto, al fine di verificarne il nuovo comportamento.

8. Testing materials

Le risorse software necessarie per effettuare le tipologie di testing sono le seguenti:

- Eclipse, IDE per l'implementazione delle varie componenti e delle classi di test.
- JUnit, framework per l'automazione di unit testing.
- Google Docs per la stesura dei vari documenti relativi alle attività di testing.

Le risorse hardware necessarie per effettuare il testing sono:

- Computer per analizzare il sistema e lavorare sui vari documenti del testing e la loro integrazione.

9. Testing schedule

Ogni test verrà eseguito appena vengono completate le classi oggetto di test e sarà condotto dal componente che avrà scritto tali classi.