Università degli Studi di Salerno
Corso di Ingegneria del Software

8-bit

SYSTEM TEST PLAN E SPECIFICA DEI CASI DI TEST

Versione 1.0

Anno 2017/2018

INDICE

1. Introduzione

Il testing riguarda l'ultima fase del sistema 8-bit consiste nel trovare le differenze tra il comportamento atteso specificato del modello di sistema e il comportamento che si ha dopo l'implementazione del sistema. Lo scopo principale del testing è quello di testare il sistema e di creare il massimo numero di errori che poi vengono corretti. Lo scopo del documento è quello di definire i test-case su cui verranno testate le funzionalità del sistema. Per ogni funzionalità saranno forniti un numero sufficienti di istanze di input in modo tale da fornire almeno un test case composto da dati corretti e quindi appartenente a classi valide ci sarà inoltre un test-case per ogni input che prevede una classe non valida.

2. Riferimenti e Relazioni

In questa fase vengono descritte le relazioni con gli altri documenti prodotti durante la fase di sviluppo del progetto software 8-bit. Tali relazioni ci permettono di produrre una fase di testing che sia coerente con tutte le specifiche descritte durante l'analisi dei requisiti e delle funzionalità che il sistema dovrà fornire. Queste relazioni si basano su:

- RAD – Documento di Analisi dei Requisiti
- SDD – Documento di System Design
- ODD – Documento di Object Design

2.1 Relazione con il RAD

Queste relazioni prendono in considerazione i requisiti funzionali e non funzionali del sistema. I test delle funzionalità terranno conto delle specifiche espresse nel RAD. I test devono tener presente dei diversi attori descritti nel RAD:

- Cliente/Utente
- Amministratore
- Giochi

Quindi per ciascun cliente e amministratore dovrà essere testate la validità della registrazione, del login e del logout.

Per i giochi dovranno essere testati l'inserimento, cancellazione, ricerca e quantità.

2.2 Relazioni con l'SDD

Tali relazioni si basano sulla pianificazione dei test delle componenti rispecchia la decomposizione in sottosistemi specificata nel SDD.

2.3 Relazioni con l'ODD

In questa fase viene effettuato Testing di unità per individuare le differenze tra il comportamento atteso del modello di sistema e quello specificato.

3. Panoramica del sistema

Questo sito 8-bit è concepito sull'idea di dare la possibilità ai clienti di acquistare giochi "retrò". Il sistema 8-bit segue il modello MVC per la suddivisione dei livelli:

- MODEL: che rappresenta i dati dell'applicazione;
- VIEW: che rappresenta la visualizzazione degli oggetti:
- CONTROLLER: che rappresenta l'insieme di regole che permettono la trasformazione degli input sulle viste in modifiche del modello.

Come descritto nell'SDD, il sistema è stato suddiviso in sottosistemi in modo da raggruppare le componenti. Ciascun sottosistema deve essere testato. Il test plan si pone i seguenti obiettivi:

- Dettagliare le attività richieste per preparare e condurre il testing;
- Definire le fonti usate per preparare la pianificazione;

4. Funzionalità testate e non testate

Le componenti prese in considerazioni nella fase di testing rappresentano la maggior parte delle funzionalità di 8-bit. In particolare saranno testate:
- gestione autenticazione (cliente e amministratore): sarà testata la funzionalità di login dei vari attori del sistema.
- gestione giochi: saranno testate le funzionalità di inserimento, "modifica", ricerca e eliminazione gioco.

- **gestione Store**: caricamento dei giochi indicando per ognuno di essi il nome, descrizione, console, softwareHouse, prezzo e una eventuale quantità desiderata.

Tutte queste funzionalità richiedono meccanismi dinamici che in base a dati persistenti permettono le diverse operazioni attraverso elaborazioni server.

4.1 View

Le funzionalità che devono essere testate per il livello di visualizzazione sono:

- Admin.jsp
- area Utente.jsp
- blog.jsp
- carrello.jsp
- footer.jsp
- inserimento.jsp
- logIn_SignIn.jsp
- navbar.jsp
- newsletter.jsp
- store.jsp
- dettagli Prodotto.jsp
- Ricerca.jsp

4.2 Controller

Le funzionalità che devono essere testate sono:

- Carrello_Control.java
- Gioco_Control.java
- Acquista_Control.java
- Login_Control.java
- Logout_Control.java
- Inserimento_Control.java
- Eliminazione_Control.java
- NewsLetter_Control.java

4.3 Model

Tutte le funzionalità del livello di controllo che devono essere testate implicano il test parallelo delle componenti dipendenti nel livello dati, che forniscono l'interazione con l'insieme dei dati persistenti presenti all'interno del sistema 8-bit.

5. Pass/Fail Criteria

La fase di testing necessita di criteri formali per la determinazione del successo o dell'insuccesso di un determinato test. I dati di ogni input relativi ai test verranno divisi in classi di equivalenza: un input appartenente ad una classe specifica supera il test se l'output ottenuto corrisponde ai risultati attesi specificati nell'oracolo.

5.1 Pass Criteria

I pass criteria determinano l'insuccesso del test e quindi la correttezza del comportamento della componente testata. Essi sono stati raggruppati in due categorie principali:

- Comportamento atteso:
    - L'oracolo rileva che il comportamento della componente rispetti il comportamento atteso.
- Nessun errore rilevato dalla componente:
    - La componente non genera alcun comportamento non previsto.

5.2 Fail Criteria

I fail criteria determinano il successo del test e quindi la presenza di errori nel comportamento della componente testata. Essi sono stati raggruppati in due categorie principali:

- Errore rilevato dalla componente:
    - La componente genera un comportamento non previsto.
- Comportamento non atteso:

- L’oracolo rileva che il comportamento della componente è diverso dal comportamento atteso rispetto agli input.

6. Approcci

L’approccio che utilizziamo nel sistema 8-bit Nella sezione di testing è di tipo “BLACK BOX”, il quale prevede che i test siano effettuati ad un livello di astrazione più alto, in modo da non rientrare nei dettagli del codice, ma basandosi sulle specifiche delle funzionalità da testare.

L’approccio si compone in 3 fasi:

- testing dell’unita: che controlla i singoli componenti(classi, metodi);

- testing di integrazione: che testa l’integrazione dei vari sottosistemi;

- testing funzionale: che verifica la funzionalità dell’intero sistema assemblato.

6.1 Testing delle Unità

Con il testing di unità sarà effettuato un controllo sulle classi e metodi del sistema, quindi saranno ricercate le condizioni di fallimento evidenziando gli errori. Il testing di unità sarà sviluppato utilizzando il framework JUnit. In particolare, per ogni classe che esegue operazioni complesse sarà sviluppata la relativa classe JUnit.

6.2 Testing di Integrazione

Con il testing di integrazione effettuiamo un controllo sull’integrazione delle varie componenti del sistema. Utilizziamo la strategia “Bottom-up” utilizzando dei driver, dato che integriamo man mano i sottosistemi partendo da quello situato a livello più basso.

7. Sospensione e Ripristino

In queste due fasi il testing del sistema 8-bit può essere interrotta e ripresa più volte.

7.1 Criterio di Sospensione

La fase di sospensione deve avvenire quando un test abbia esito positivo ovvero si è riscontrato un errore all'interno di una componente. Esempi di questi criteri sono:Questi criteri il crash del database, il crash del sistema operativo.

7.2 Criterio di Ripristino

Il ripristino del test viene fatto quando sono stati risolti tutti i problemi relativi all sospensione del test e riprenderà dall'ultimo test che ha causato la sospensione.

8. Strumenti per il Testing

Gli strumenti per il testing verranno divisi in 2 differenti tipi:
- strumenti hardware
- strumenti software

8.1 Strumenti Hardware

Il sistema 8-bit sarà testato su un personal computer con la necessaria dotazione software.

8.2 Strumenti Software

Per i test di unità e di integrazione si utilizzerà JUnit. JUnit è un framework interamente sviluppato in Java open source. Esso fornisce un insieme di Api che crea in modo semplice e automatico dei test per il proprio software. Fornisce un'interfaccia grafica elementare per la valutazione dei risultati dei test:
- Barra verde:
I test Case in esecuzione falliscono.
- Barra rossa:
I test Case della test Suite falliscono.
Junit consente di definire:

- test case: singolo test di una specifica funzionalità(un metodo, una classe);
- test suite: una collezione di test case raggruppati in base a caratteristiche omogenee e possono essere eseguite in blocco. In particolare i software che si utilizzeranno per l'attività di testing sono i seguenti:
- JUnit;
Apache Tomcat;
Java Runtime Enviroment(jre);
Java Development kit (jdk);
Selenium IDE: per l'esecuzione dei test Case;
Browser: Google Chrome, Mozilla Firefox.

9. Test Case

I test case ci permettono di scoprire eventuali malfunzionamenti o comportamenti errati del sistema. Per fare ciò è necessario testare il sistema su diverse istanze di input, ognuna diretta a testare comportamenti del sistema in determinate condizioni, un motodo o classe di input piuttosto di un'altra.

Registrazione

Le funzionalità da testare riguardano una componente del sottosistema ACCESSO, in particolare la funzionalità di registrazione.

9.1.1 Classi Di Equivalenza

Input	e-mail	
Classi Valide	C_0001	stringa alfanumerica con format "nomeUtente@provider"
Classi non valide	C_0002	Stringa alfanumerica discord dal formato

Input	nickname	
Classi Valide	C_0003	Stringa alfanumerica con lunghezza compresa tra 6 e 20 caratteri
Classi non valide	C_0004	Stringa alfanumerica discorde dal formato

Input	password	
Classi Valide	C_0005	Stringa alfanumerica con lunghezza compresa tra 6 e 20 caratteri e contenente una lettera maiscola e almeno una cifra
Classi non valide	C_0006	Stringa alfanumerica discorde dal formato
	C_0007	Stringa alfanumerica maggiore di 20 caratteri
	C_0008	Stringa alfanumerica che non contiene una cifra

Input	Ripetipassword	
Classi Valide	C_0009	Stringa alfanumerica che deve essere uguale alla password
Classi non valide	C_0010	Stringa alfanumerica discorde dal formato

9.1.2 Formal Test Specification

Input	e-mail
ID_Codice	Specifica formale
C_0001	[propertyemailOK]
C_0002	[errore]

Input	nickname
ID_Codice	Specifica formale
C_0003	[propertynicknameOK]
C_0004	[errore]

Input	password
ID_Codice	Specifica formale

C_0005	[propertypasswordOK]
C_0006	[errore]
C_0007	[errore]
C_0008	[errore]

Input	RipetiPassword
ID_Codice	Specifica formale
C_0009	[propertyripetiPasswordOK]
C_0010	[errore]

9.1.3 Test Case

Test Case	TC_Registrazione_01	
Parametro	Scelta	Valore
e-mail	C_0001	giaco@libero.it
Nickname	C_0003	giacomo
password	C_0005	Giacomo95
ripetiPassword	C_0009	Giacomo95

Test Case	TC_Registrazione_02	
Parametro	Scelta	Valore
e-mail	C_0001	nello@libero.it
Nickname	C_0003	Aniello
password	C_0005	Aniello95
ripetiPassword	C_0009	Aniello95

9.2 Login

Le funzionalità da testare riguardano una component del sottosistema ACCESSO, in particolare la funzionalità di login.

9.2.1 Classe di Equivalenza

Input	nickname	
Classi Valide	C_0011	Stringa alfanumerica con lunghezza compresa tra 6 e 20 caratteri
Classi non valide	C_0012	Stringa alfanumerica discorde dal formato

Input	password	
Classi Valide	C_0013	Stringa alfanumerica con lunghezza compresa tra 6 e 20 caratteri e contenente una lettera maiscola e almeno una cifra
Classi non valide	C_0014	Stringa alfanumerica discorde dal formato
	C_0015	Stringa alfanumerica maggiore di 20 caratteri
	C_0016	Stringa alfanumerica che non contiene una cifra

9.2.2 Formal Test Specification

Input	nickname
ID_Codice	Specifica formale
C_0011	[propertynicknameOK]
C_0012	[errore]

Input	password
ID_Codice	Specifica formale
C_0013	[propertypasswordOK]
C_0014	[errore]
C_0015	[errore]
C_016	[errore]

9.2.3 Test Case

Test Case	TC_Login_01	
Parametro	Scelta	Valore
Nickname	C_0011	giacomo
password	C_0013	Giacomo95

Test Case	TC_Login_02	
Parametro	Scelta	Valore
Nickname	C_0011	aniello
password	C_0013	Giacomo95

9.3 Inserimento

Le funzionalità da testare riguardano una component del sottosistema INSERIMENTO, in particolare la funzionalità Di Inserimento di un prodotto da parte del gestore.

9.3.1 Classi Di Equivalenza

Input	Titolo	
Classi valide	C_0014	Stringa alfanumerica compresa tra 1 e 45 caratteri

Classi non valide	C_0015	Stringa alfanumerica minore di 1 carattere
	C_0016	Stringa alfanumerica maggiore di 20 caratteri

Input	Prezzo	
Classi valide	C_0017	Numero decimale con due numeri dopo la virgola
Classi non valide	C_0018	Carattere alfabetico all'interno del prezzo

Input	Anno	
Classi valide	C_0019	Numero intero costituito da 4 cifre
Classi non valide	C_0020	Stringa alfanumerica minore di 1 carattere

Input	Console	
Classi valide	C_0021	Scegliere tra quelle standard
Classi non valide	C_0022	Non viene selezionata alcuna azione

Input	Categoria	
Classi valide	C_0023	Scegliere tra quelle standard
Classi non valide	C_0024	Non viene selezionata alcuna azione

Input	softwareHouse	
Classi valide	C_0025	Scegliere tra quelle standard

Classi non valide	C_0026	Non viene selezionata alcuna azione

9.3.2 Formal Test Specification

INPUT	Titolo
ID_CODICE	SPECIFICA FORMALE
C_0014	[propertycodice_titoloOK]
C_0015	[errore]
C_0016	[errore]

INPUT	Prezzo
ID_CODICE	SPECIFICA FORMALE
C_0017	[propertycodice_pezzoOK]
C_0018	[errore]

INPUT	Anno
ID_CODICE	SPECIFICA FORMALE
C_0019	[propertycodice_annoOK]
C_0020	[errore]

INPUT	Console
ID_CODICE	SPECIFICA FORMALE
C_0021	[propertycodice_consoleOK]
C_0022	[errore]

INPUT	Categoria
ID_CODICE	SPECIFICA FORMALE
C_0023	[propertycodice_categoriaOK]
C_0024	[errore]

INPUT	softwareHouse
ID_CODICE	SPECIFICA FORMALE
C_0025	[propertycodice_pezzoOK]
C_0026	[errore]

9.3.3 Test Case

TEST CASE	TC_Inserimento_01	
PARAMETRO	SCELTA	VALORE
Titolo	C_0014	Donkey Kong
Prezzo	C_0017	6.8
Anno	C_0019	1999
Console	C_0021	Super Nintendo
software House	C_0025	Sega

TEST CASE	TC_Inserimento_02	
PARAMETRO	SCELTA	VALORE
Titolo	C_0027	Metal Slug
Prezzo	C_0028	5.4
Anno	C_0022	1989

Console	C_0023	Arcade
software House	C_0029	Namco

TEST CASE	TC_Inserimento_03	
PARAMETRO	SCELTA	VALORE
Titolo	C_0028	Pac Man
Prezzo	C_0029	3.90
Anno	C_0023	1980
Console	C_0024	Arcade
software House	C_0029	Namco

9.4 Quantità

Le funzionalità da testare riguardano una componente del sottosistema Gioco, in particolare la funzionalità di inserire la quantità di un gioco che si vuole acquistare.

9.4.1 Classi Di Equivalenza

INPUT	Quantità	
CLASSI VALIDE	C_0027	Numero intero maggiore o uguale a 0.
CLASSI NON VALIDE	C_0028	Numero intero minore di 0.
	C_0029	Carattere alfabetico all'interno della quantità.

9.4.2 Formal Test Specification

INPUT	Quantità
ID_CODICE	SPECIFICA FORMALE
C_0027	[propertycodice_quantitàOK]
C_0028	[errore]
C_0029	[errore]

9.4.3 Test Case

TEST CASE	TC_Quantità_01	
PARAMETRO	SCELTA	VALORE
Quantità	C_0027	10

TEST CASE	TC_Quantità_02	
PARAMETRO	SCELTA	VALORE
Quantità	C_0028	-5

TEST CASE	TC_Quantità_03	
PARAMETRO	SCELTA	VALORE
Quantità	C_0029	10b

10. Testing Schedule

Di seguito sono elencate la gestione dei rischi che occorre seguire durante la fase di testing e l'organizzazione delle attività di testig.

10.1 Gestione dei rischi

I possibili rischi generate dalle attività di testing sono stati minimizzati diminuendo le componenti del sistema da implementare e quindi testare. D'altro canto il testing di funzionalità rallenta l'individuazione di errori qualora un caso di test avesse esito positivo, poichè l'utilizzo di più componenti per il test di una singola funzionalità viene eseguita ma non in modo completamente corretto. Qualora la fase di testing evidenziasse un numero di errori maggiore rispetto alla media attesa, viene pianificato un impegno maggiore dei membri del team sulle attività di testing ed in casi estremi l'abbandono delle altre attività finchè errori gravi (funzionalità non corretta, risultati errati, modifiche apportate in modo errato) non vengano risolti.

10.2 Organizzazione delle attività

Le attività di testing devono svolgersi sulle singole funzionalità divise nei livelli di suddivisione del sistema, rispettando le direttive indicate dal documento di system design.