ITP Integration Testing Plan

Riferimento	
Versione	1.0
Data	13/12/2018
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Mario De Cicco, Mario Greco, Giovanni Di Nocera, Anna Maria Raffaela Riccio

Revision History

Data	Versione	Descrizione	Autori
13/12/2018	Draft 1.0	Strutturazione Documento	Tutti

Sommario

1. Introduzione

Il testing di integrazione rappresenta una delle fasi di testing più importanti, in quanto lo scopo è di verificare le interazioni tra due o più componenti. Questo testing rileva bug che non sono stati determinati durante lo Unit Testing, focalizzandosi su un insieme di componenti che sono integrate.

Infatti, quando i bug in ogni componente sono stati rilevati e riparati, le componenti sono pronte per essere integrate in sottosistemi più grandi. Due o più componenti sono integrate e analizzate, e quando dei bug sono rilevati nuove componenti possono essere aggiunte per riparare tali bug. Quindi l'obiettivo del testing consiste nella verifica della corretta interazione tra le componenti e il rispetto delle interfacce, secondo quanto stabilito nelle specifiche di interazione.

2. Riferimenti

Per verificare la corretta integrazione dei sottosistemi del sistema TF sono stati predisposti dei test case basati sulla divisione in sottosistemi proposta in base di system design (TF_SDD_v2.0).

3. Test di integrazione

3.1 Approccio di Integration Testing

L'ordine in cui i sottosistemi sono selezionati per il testing e l'integrazione determina la strategia di testing. Possiamo selezionare diverse startegie per l'integration testing, tra cui:

- Big Bang integration
- Bottom Up integration
- Top down integration
- Sandwich testing

Ognuna di queste strategie è nata per la decomposizione gerarchica del sistema, successivamente utilizzate in fase di testing.

La strategia adottata è quella di tipo "Bottom Up", in cui i sottosistemi dei layer del livello più basso della gerarchia sono testati individualmente; poi i prossimi sottosistemi ad essere testati sono quelli che "chiamano" i sottosistemi testati in precedenza, cioè vengono testati i layer dell'Application Logic.

Si ripete questo passo finché non vengo testati tutti i sottosistemi, ovvero finchè non sono testati sia il layer Storage, sia l'Application Logic layer e sia il Interface layer.

La scelta dell'approccio di testing Bottom Up non comporta l'utilizzo di test stub, inoltre data la natura del sistema non sarà necessario l'utilizzo dei test driver.

Infatti, per testare lo Storage layer è sufficiente l'esecuzione delle query e la visualizzazione dei risultati; per testare l'Application Logic layer è sufficiente l'esecuzione dei metodi implementati al fine di visualizzare la misura calcolata o il json generato dalla richiesta di creazione da un grafico. La strategia d tipo Bottom Up non è buona per sistemi decomposti funzionalmente poiché test i sottosistemi più importanti alla fine, ma è utile nel nostro caso per integrare i sistemi object-oriented.

3.2 Componenti da testare

La scelta delle componenti da testare segue la decisione di eseguire la strategia di testing Bottom Up.

Per quanto riguarda lo Storage layer, quindi, la componente da testare è:

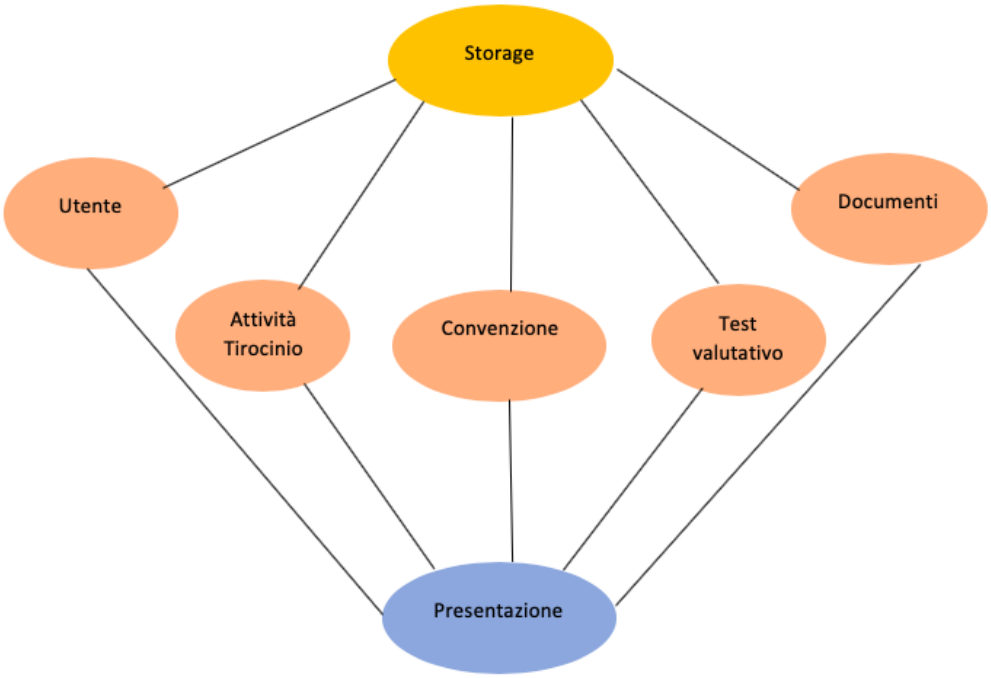
- Storage

Per quanto riguarda l'Application Logic layer, le componenti da testare sono:

- Gestione utente
- Gestione attività tirocinio
- Gestione convenzione
- Gestione test valutativo
- Gestione documenti

Per quanto riguarda l'Interface layer, la componente da testare è:

- Presentazione



4. Pass/Fail Criteri

Il testing avrà successo se l'output osservato è diverso dall'output atteso: questo sta a significare che se stiamo parlando di SUCCESSO, il test ha individuato una failure.

Quindi nel caso in cui si verifichi una failure, questa verrà analizzata e, se legata ad un fault, si procederà alla sua correzione.

Sarà infine iterata la fase di testing per verificare che la modifica non abbia avuto effetti su altri componenti del sistema.

In caso contrario parliamo di FALLIMENTO se il test non riesce ad individuare alcun errore.