UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Corso di
INGEGNERIA, GESTIONE ED EVOLUZIONE DEL SOFTWARE

Test Plan Document

DOCENTE
Prof. Andrea De Lucia
Università degli Studi di Salerno

REVIEWER
Dott. Stefano Lambiase
Università degli Studi di Salerno

AUTORI
Benedetto Scala
Mat: 0522501794
Leopoldo Todisco
Mat: 0522501795
Carlo Venditto
Mat: 0522501796

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto del Progetto

Negli studi recenti di Ingegneria del Software, la comunità ha cominciato a preoccuparsi dell'impatto degli aspetti umani nello sviluppo del software. I lavori esistenti hanno analizzato come gli sviluppatori e le sotto-comunità interagiscono, con l'obiettivo di individuare schemi di comunicazione e collaborazione, portando al concetto di "Community Smells". I community smells riflettono schemi organizzativi e socio-tecnici sub-ottimali nella struttura della comunità del software.

Allo stato dell'arte, siamo a conoscenza di un solo tool, CADOCS, che propone il rilevamento di community smells. Tale tool, è presente in due varianti:

- CADOCS che è un Conversational Agent integrato in Slack;
- CADOCS II che è la corrispettiva desktop application standalone;

Inoltre, data l'introduzione recente di questi concetti, molti manager non sono a conoscenza della loro esistenza e dell'importanza della loro individuazione.

1.2 Funzionalità da testare

Come già riportato nel documento di Pre-Maintenance, i requisiti funzionali di Cadocs II sono 4, ma di queste solo 3 sono state implementate del tutto come si può vedere nella tabella 1.1.

Requisito Funzionale	Descrizione
RF1: Get Smells	Tale requisito fa riferimento alla capacità del sistema di poter rilevare Community Smells in una repository Github.
RF2: Get Smells By Date	Tale requisito fa riferimento alla capacità del sistema di poter rilevare Community Smells in una repository Github a partire da una determinata data.
RF3: Info	Tale requisito fa riferimento alla capacità del sistema di spiegare cosa è in grado di fare e di spiegare i community smells agli utenti

Tabella 1.1: Tabella dei requisiti funzionali del sistema Cadocs II

1.3 Criteri di Adeguatezza

Un test viene considerato passato quando il comportamento ottenuto è uguale al comportamento atteso dall'oracolo, altrimenti il test si considera fallito.

Nel contesto del nostro progetto saranno testati tutti i requisiti funzionali, sebbene nel momento di inizio del progetto non vi è alcun test da cui partire, e addizionalmente, si testerà anche tutto ciò che si implementerà per realizzare le Change Requests.

Per quanto riguarda i test di unità e integrazione si fa affidamento a tecniche white-box e il requisito minimo è quello del raggiungimento del 75% di branch coverage.

1.4 Struttura del documento

- Capitolo 1: Introduzione;

- Capitolo 2: Strategie di Testing;

CAPITOLO 2

Strategie di Testing

Per garantire che il processo di testing soddisfi i criteri di adeguatezza elencati nel precedente capitolo, in questo capitolo delineeremo le strategie di testing appropriate e i relativi tool. Saranno eseguiti vari livelli di testing, inclusi quelli a livello di singola unità, di integrazione e di sistema, oltre a test di regressione e pratiche di revisione del codice per individuare potenziali errori introdotti.

2.1 Testing di Unità

Il testing di unità consiste nel testare le singole componenti in isolazione. Ciò sarà effettuato usando una strategia di tipo white-box con l'utilizzo delle libreria *pytest* e *unittest*. Il criterio di coverage di riferimento sarà la branch-coverage, per tale motivo si usa il tool *pytest-cov*.

Per quanto riguarda le funzionalità già implementate in CADOCS, saranno aggiunti dei casi di test sulle classi e metodi già presenti poichè allo stato iniziale del progetto, non vi sono casi di test.

2.2 Testing di Integrazione

Il testing di integrazione consiste nel verificare come le unita del sistema software si integrino e collaborino all'interno dell'applicazione nel suo complesso. Nel nostro caso applicheremo una strategia di testing bottom-up, utilizzando **pytest** e testando soltanto le classi presenti nel sistema CADOCS II.

L'architettura di CADOCS II subirà modifiche attraverso l'implementazione delle richieste di modifica, in particolare con l'attuazione della CR 1 che prevede il disaccoppiamento del componente resolver dalle dipendenze precedenti, nella CR2 con lo sviluppo e l'integrazione del nuovo tool per la geodispersione culturale, e nella CR3 per la rielaborazione dell'interfaccia grafica.

Per la CR 1 verranno sviluppati test che verifichino l'interazione del componente resolver con le nuove dipendenze o interfaccia, se necessario. Sarà fondamentale assicurarsi che il componente resolver funzioni correttamente con le nuove implementazioni, senza compromettere la funzionalità esistente del sistema.

I test riguardanti l'interazione con il sottosistema csDetector e il nuovo strumento sviluppato nella CR2 saranno eseguiti durante la fase di test di sistema, poiché la complessità nell'emulare tali interazioni è considerevole.

Il processo di testing dell'integrazione della GUI reingegnerizzata, presente nella CR 3, sarà condotto mediante l'utilizzo degli strumenti di testing forniti dal framework scelto. Questo processo avrà come obiettivo verificare l'interazione tra la GUI e le classi esistenti in CADOCS II.

2.3 Testing di Sistema

Per la fase di testing di sistema, l'obiettivo è garantire l'integrità del comporta- mento del sistema implementato rispetto alle aspettative dell'utente. In questa fase sarà utilizzata una strategia black-box per creare i test case; nello specifico per garantire la completezza della test suite si userà la strategia *Category Partition*. Per la natura del progetto, ossia una web application scritta in Python, sarà utilizzato Selenium, che è uno dei tool più diffusi per fare System Testing.

2.4 Testing di Regressione

Nello stato iniziale del progetto, non erano stati definiti casi di test, determinando l'assenza di qualsiasi forma di testing strutturato. Attualmente, è stato condotto un test di sistema preliminare che fungerà da base per i futuri test di regressione. Questo test preliminare rimarrà valido almeno fino all'implementazione della Change Request (CR1), assicurando che le funzionalità fondamentali del sistema siano correttamente verificate e offrendo un punto di riferimento per valutare l'impatto delle modifiche successive.