**Master Test Plan Outline**

1. **Introduzione**
   1. **Identificatore documento**

ID: MTP cASpER 1

Data del problema: 23/05/2022

Organizzazione: sesalab

Autore (i): Angelo Afeltra, Antonio Trovato, Walter D’Ambrosio

Status: Draft

Firma (e):

Immagine che contiene testo

Descrizione generata automaticamente





* 1. **Scope**

cASpER è un plug-in di IntelliJ per la detection e il refactoring di CodeSmell.

L’obiettivo del test è verificare che le modifiche apportate non portino alla distruzione del sistema.

Le modifiche da noi apportate vanno a incidere sulla gestione delle interfacce grafiche e sulla procedura di detection dei CodeSmell. Andremo a testare le componenti interessate dall’intervento di manutenzione per verificare che, quest’ultimo, non abbia introdotto nuove anomalie nel sistema. Infine, eseguiremo dei test di regressione relativi alle strategie di identificazione dei CodeSmell e al refactoring di essi in quanto tali test erano stati sviluppati già nella iniziale versione del software.

La metodologia di testing seguirà un approccio iterativo in quanto effettueremo prima test di unità poi di integrazione per tutte le componenti appartenenti a diverse aree di cASpER. Terminate le iterazioni effettueremo un unico test di sistema.

Il test di regressione verrà effettuato prima e dopo l’intervento di manutenzione.

1. **Details of Master Test Plan**

**2.1 Process: Maintenance**

**2.1.1 Activity: Maintenance test**

|  |  |
| --- | --- |
| **Task** | **Stesura component test plan** |
| Metodi | Stesura del component test plan |
| Input | Modification request |
| Output | Component test plan |
| Programmazione | Da ultimare durante la fase di design della manutenzione. |
| Risorse | Modification request |
| Ruoli e responsabilità | Walter D’Ambrosio, Angelo Afeltra, Antonio Trovato |

|  |  |
| --- | --- |
| **Task** | **Individuazione component test** |
| Metodi | Individuazione dei test case |
| Input | Modification request |
| Output | I test case |
| Programmazione | Da ultimare durante la fase di design della manutenzione. |
| Risorse | Modification request |
| Ruoli e responsabilità | Walter D’Ambrosio, Angelo Afeltra, Antonio Trovato |

|  |  |
| --- | --- |
| **Task** | **Implementazione component test** |
| Metodi | Implementazione dei casi di test precedentemente descritti tramite JUnit |
| Input | I test case |
| Output | Componenti software |
| Programmazione | Da ultimare durante la fase di design della manutenzione. |
| Risorse | IntelliJ IDEA, JUnit |
| Rischi e assunzioni | Il rischio principale è quello di generare flaky test |
| Ruoli e responsabilità | Walter D’Ambrosio, Angelo Afeltra, Antonio Trovato |

|  |  |
| --- | --- |
| **Task** | **Esecuzione component test** |
| Metodi | Esecuzione dei casi di test precedentemente implementati tramite JUnit |
| Input | Le componenti di test implementate |
| Output | Report di esecuzione |
| Programmazione | Da ultimare durante la fase di design della manutenzione. |
| Risorse | IntelliJ IDEA, JUnit |
| Rischi e assunzioni | Si assume che non vi siano comportamenti anomali durante l’esecuzione |
| Ruoli e responsabilità | Walter D’Ambrosio, Angelo Afeltra, Antonio Trovato |

|  |  |
| --- | --- |
| **Task** | **Report component test** |
| Metodi | Generazione di report per tutti i test eseguiti tramite JUnit |
| Input | Risultati dell’esecuzione dei test |
| Output | Test report log |
| Programmazione | Da ultimare durante la fase di design della manutenzione. |
| Risorse | IntelliJ IDEA, JUnit |
| Ruoli e responsabilità | Walter D’Ambrosio, Angelo Afeltra, Antonio Trovato |