



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Corso di

INGEGNERIA, GESTIONE ED EVOLUZIONE DEL SOFTWARE

# Test Report 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

---

<b>Elenco delle Figure</b>	<b>iii</b>
<b>Elenco delle Tabelle</b>	<b>iv</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Contesto del Progetto . . . . .	1
1.2 Struttura del documento . . . . .	2
<b>2 Pre-Maintenance Tests</b>	<b>3</b>
2.0.1 Testing di sistema . . . . .	3
<b>3 Post-Maintenance Tests</b>	<b>7</b>
3.1 CR1: Separazione della logica di business . . . . .	7
3.1.1 Test di unità e integrazione . . . . .	7
3.1.2 Test di sistema . . . . .	7
3.2 CR2: Aggiunta di un tool per il calcolo della geodispersione . . . . .	8
3.2.1 Test di unità e di integrazione . . . . .	8
3.2.2 Test di Sistema . . . . .	8
3.3 CR3: Sviluppo di una GUI . . . . .	8
3.3.1 Test di unità e di integrazione . . . . .	8
3.3.2 Test di Sistema . . . . .	8

---

3.3.3	Test di Regressione . . . . .	9
3.4	CR4: Docker Porting . . . . .	9
3.4.1	Test di unità e di integrazione . . . . .	9
3.4.2	Test di Sistema . . . . .	9

---

## Elenco delle figure

---

2.1	Test Case Get Smells . . . . .	4
2.2	Test Case Report . . . . .	4
2.3	Test Case Get Smells By Date . . . . .	5
2.4	Test Case Info . . . . .	6

---

## **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 Struttura del documento

Il documento è così strutturato:

- Capitolo 1: Introduzione
- Capitolo 2: Pre-Maintenance Tests
- Capitolo 3: Post-Maintenance Tests

## CAPITOLO 2

---

### Pre-Maintenance Tests

---

#### 2.0.1 Testing di sistema

Nella situazione attuale del progetto, non sono stati definiti casi di test e di conseguenza non è stato effettuato nessun tipo di testing. Quindi quello che è stato fatto attualmente è effettuare un test di sistema che poi in futuro sarà utilizzato per effettuare test di regressione.

Come indicato nella sezione 2.3 del Test Plan, poiché l'applicazione è una desktop application, non è stato possibile utilizzare Selenium per automatizzare il testing di sistema.

Di conseguenza, il testing è condotto manualmente attraverso l'interazione diretta con il chatbot e la raccolta manuale dei dati risultanti.

è possibile ritrovare una completa esecuzione dei test nelle immagini 2.1, 2.3, 2.2, 2.4.



aaaaaaaaaaaaaaaaaaaaa :You

CADOCS: Hi 🐼

These are the \*community smells\* I can detect in your development communities:

-----

\*Organizational Silo Effect\* - OSE - 🧑🧑🧑

Siloed areas of the community that do not communicate, except through one or two of their respective members.

-----

(a) TC\_GS\_1

CADOCS give me community smells on this repo: ERROR\_LINK :You

(b) TC\_GS\_2

CADOCS can you give me the community smells in <https://github.com/gianwario/beehave> :You

CADOCS: Hi 🐼

These are the community smells we were able to detect in the repository <https://github.com/gianwario/beehave>:

-----

\*BCE\* Black-cloud Effect 🌧️🌧️🌧️

Information overlaid due to lack of structured communications across team

(c) TC\_GS\_3

**Figura 2.1: Test Case Get Smells**

aaaaaaaaaaaaaaaaaaaaa :You

CADOCS: Hi 🐼

These are the \*community smells\* I can detect in your development communities:

-----

\*Organizational Silo Effect\* - OSE - 🧑🧑🧑

Siloed areas of the community that do not communicate, except through one or two of their respective members.

-----

(a) TC\_RP\_1

\*TC\* Toxic Communication 🧑🗣️🧑

\_Toxic interactions and conflicting opinions among developers could push them to leave the project.\_

-----

See you soon 🐼

I need a report of the last execution :You

(b) TC\_RP\_2

**Figura 2.2: Test Case Report**

aaaaaaaaaaaaaaaaaaaa :You

CADOCS: Hi 🐼

These are the \*community smells\* I can detect in your development communities:

-----

\*Organizational Silo Effect\* - OSE - 🧑🧑🧑

Siloed areas of the community that do not communicate, except through one or two of their respective members.

-----

(a) TC\_GSD\_1

CADOCS give me the community smells on this repo: ERROR\_LINK from 09/09/2021 :You

(b) TC\_GSD\_2

CADOCS give me the community smells on this repo <https://github.com/gianwario/beehave> from 09/14/2021 :You

(c) TC\_GSD\_3

CADOCS give me the community smells on this repo <https://github.com/gianwario/beehave> from 31/02/2021 :You

(d) TC\_GSD\_4

CADOCS give me the community smells on this repo <https://github.com/gianwario/beehave> from 10/01/2023 :You

CADOCS: Hi 🐼

These are the community smells we were able to detect in the repository <https://github.com/gianwario/beehave> starting from 10/01/2023:

-----

\*BCE\* Black-cloud Effect 🌤️

\_Information overload due to lack of structured communications or cooperation.

(e) TC\_GSD\_5

**Figura 2.3:** Test Case Get Smells By Date

```

aaaaaaaaaaaaaaaaaaaa :You
CADOCS: Hi 🐼
These are the *community smells* I can detect in your
development communities:
-----
*Organizational Silo Effect* - OSE - 🧑🧑🧑
Siloed areas of the community that do not communica
te, except through one or two of their respective mem
bers.
-----

```

(a) TC\_IN\_1

```

Hey CADOCS can you tell me more about the community smells you can d
etect? :You
CADOCS: Hi 🐼
These are the *community smells* I can detect in your development comm
unities:
-----
*Organizational Silo Effect* - OSE - 🧑🧑🧑
Siloed areas of the community that do not communicate, except through on
e or two of their respective members.
-----
*Black cloud Effect* - BCE

```

(b) TC\_IN\_2

**Figura 2.4:** Test Case Info

L'output dei casi di test TC\_GS\_1, TC\_GS\_2, TC\_GSD\_1, TC\_GSD\_2, TC\_GSD\_3, TC\_GSD\_4, TC\_RP\_1, TC\_RP\_2 e TC\_IN\_1 non è conforme all'oracolo. Alcuni di essi sono classificati con intent errati, mentre altri semplicemente non funzionano. Nel sistema mancano i controlli necessari per garantire il corretto funzionamento di tali test.

Questi controlli, essenziali come la validità della data o la presenza del link, rappresentano una grave lacuna nel sistema in esame. Tuttavia, in questa fase del report, verranno ignorati e non verranno corretti.

Si intende invece affrontare tali lacune tramite le change request che saranno implementate successivamente.

### Post-Maintenance Tests

---

## 3.1 CR1: Separazione della logica di business

### 3.1.1 Test di unità e integrazione

Dopo aver completato le modifiche relative al refactoring del codice, è stata avviata una fase di testing per garantire la correttezza e l'affidabilità del sistema. Questa fase ha incluso test di unità, mirati a verificare il comportamento di singoli componenti. Come oracolo per il processo di testing, è stato utilizzato il system test. Questo ha permesso di confrontare i risultati ottenuti dai test di unità e di integrazione con quelli attesi, determinando la correttezza delle modifiche.

### 3.1.2 Test di sistema

Al termine delle modifiche relative a questa Change Request (CR), è stato necessario eseguire nuovamente tutti i System Tests per assicurarsi che la CR non avesse compromesso le funzionalità del sistema. Questo passaggio è stato fondamentale per assicurarsi che il refactoring non avesse compromesso il sistema.

## **3.2 CR2: Aggiunta di un tool per il calcolo della geodispersione**

### **3.2.1 Test di unità e di integrazione**

Al termine dell'implementazione della CR2 è stato necessario sviluppare prima 3 test di unità al fine di assicurarsi che la nuova concrete strategy lavorasse come ci si aspetta in un ambiente isolato, sia 3 test di integrazione.

### **3.2.2 Test di Sistema**

Per assicurarsi il corretto funzionamento sia del nuovo requisito che quello dei vecchi requisiti è stato necessario eseguire nuovamente i test di sistema, i quali hanno funzionato anche come test di regressione. Tutti i test risultano essere passati.

## **3.3 CR3: Sviluppo di una GUI**

### **3.3.1 Test di unità e di integrazione**

Al termine dell'implementazione della Change Request (CR3), non è stato necessario modificare la test suite esistente. I test di unità e di integrazione sono stati eseguiti come previsto, e tutti hanno dato esito positivo. Questo ha confermato che le modifiche apportate non hanno introdotto regressioni o malfunzionamenti, mantenendo l'integrità e la stabilità del sistema.

### **3.3.2 Test di Sistema**

Al termine dell'implementazione della Change Request (CR3), è stato necessario riprogettare e rieseguire tutti i test di sistema. Questo passaggio si è reso indispensabile poiché la modifica ha comportato un cambiamento significativo nell'interfaccia grafica utente (GUI), rendendo necessario ritestare tutte le componenti del sistema per garantirne il corretto funzionamento. La nuova GUI ha richiesto una verifica approfondita per assicurarsi che tutte le interazioni e funzionalità fossero state mantenute e che non ci fossero stati impatti negativi sul sistema. Come oracolo sono stati utilizzati i test di sistema che erano presenti in precedenza.

### 3.3.3 Test di Regressione

Al termine di Tale CR è stata anche generata una test suite di regressione. Si tratta di test case che originariamente facevano parte del progetto originale: CADOCS, tuttavia un sottoinsieme di questi risulta essere compatibile anche con il sistema attuale a discapito di qualche modifica nel codice sorgente.

## 3.4 CR4: Docker Porting

### 3.4.1 Test di unità e di integrazione

Questa CR non ha richiesto la modifica della test suite. Al termine della dockerizzazione dell'intero sistema sono stati eseguiti tutti i test di unità ed integrazione, i quali sono passati tutti, a discapito di un test case facente parte del regression testing. Tale test case faceva riferimento al requisito "Report" che era implementato nella versione originale di CADOCS, ma non è stato portato nella versione Desktop application: CADOCS II. A questo punto emerge che la coverage totale risulta essere dell' 80%, quindi un valore ampiamente superiore al 75% che era il livello minimo per l'accettazione della test suite.

### 3.4.2 Test di Sistema

Al termine di tale CR è stato necessario eseguire di nuovo tutti i System Tests al fine di essere sicuri che la dockerizzazione non avesse compromesso le funzionalità del sistema.

Possiamo dire che anche dopo l'operazione di dockerizzazione, tutti i system tests sono passati.