UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica

Corso di
Ingegneria, Gestione ed Evoluzione del Software

Modification Report

DOCENTE

Prof. Andrea De Lucia

AUTORI

Alfonso Cannavale

Matricola: 0522501597

Davide La Gamba

Matricola: 0522501464

Kevin Pacifico

Matricola: 0522501501

LINK REPOSITORY - BRANCH "DEV"

CADOCS: `https://github.com/alfcan/CADOCS/tree/dev`

CADOCS_NLU: `https://github.com/alfcan/CADOCS_NLU_Model/tree/dev`

csDetector: `https://github.com/alfcan/csDetector/tree/dev`

Anno Accademico 2022-2023

Indice

Capitolo 1

Introduzione

1.1 Change Requests

La lista delle change requests approvate è riportata nella tabella 1.1.

La priorità è relativa all'urgenza della modifica da effettuare in base alle esigenze del mercato, degli utenti e degli sviluppatori. L'impatto si riferisce alle conseguenze che la modifica da apportare potrà avere sull'intero sistema. La difficoltà è relativa alla complessità di realizzazione della modifica.

1.2 Struttura documento

Il documento sarà strutturato in 3 capitoli: **Impact Analysis (2)**, **Implementazione delle modifiche (3)** e **Conclusioni (4)**.

Nei primi due capitoli sarà presente una Sezione per ogni change requests, che documenterà quanto è stato pianificato e successivamente svolto per ogni change request indicata. In particolare, nel capitolo di Impact Analysis saranno presenti sia informazioni relative alla pianificazione della modifica da attuare, sia relative ai risultati delle modifiche attuate, come ad esempio i Discovered Impact Set.

Infine, nel capitolo di Conclusione sono stati riportati: l'esito finale delle modifiche, le deviazioni da quanto pianificato e le lezioni apprese durante lo svolgimento di questo progetto.

CR_ID	Titolo	Descrizione	Tipo	Priorità	Impatto	Difficoltà
CR_1	Refactoring	Effettuare un refactoring della struttura del software per migliorarne la manutenibilità.	Perfettiva	Alta	Medio	Bassa
CR_2	Italian conversation	Permettere l'interazione con gli utenti utilizzando anche la lingua italiana, aggiungendo un nuovo modello e migliorando il modello NLP esistente.	Perfettiva	Alta	Alto	Alta
CR_3	Docker Porting	Effettuare il porting del tool csDetector sulla piattaforma Docker, per migliorare la portabilità del tool che è propedeutica per l'installazione di CADOCS.	Adattiva	Media	Basso	Alta
CR_4	Discord Porting	Effettuare il porting del tool sulla piattaforma Discord.	Adattiva	Bassa	Alto	Alta

Tabella 1.1: La tabella delle change requests

Capitolo 2

Impact Analysis

2.1 CR_1 - Refactoring

2.1.1 Obiettivi della CR

L'implementazione di questa CR punta ad ottenere i seguenti obiettivi:

- **Ottimizzazione della struttura del progetto**: il refactoring mira a rendere il codice più chiaro e comprensibile; questo obiettivo viene raggiunto mediante l'adozione di un approccio sistematico per ristrutturare la disposizione dei package all'interno del progetto. Questo processo viene avviato con l'individuazione di sottosistemi basandosi sul tipo di funzionalità implementate dai moduli e dalle loro dipendenze.
- **Diminuire l'accoppiamento e aumentare la coesione**: analizzando le dipendenze tra i moduli e le componenti, le operazioni di refactoring da attuare dovranno diminuire l'accoppiamento e aumentare la coesione tra i moduli.
- **Semplificazione**: un altro obiettivo è quello di semplificare alcune parti del progetto CADOCS, sostituendo parti di metodi con codice di minore dimensione e maggior leggibilità ed interpretabilità.
- **Mantenere le funzionalità**: a seguito delle operazioni di refactoring, sarà verificato tramite testing di regressione che non saranno state apportate modifiche alle funzionalità implementate dal sistema.

- **Aumentare il valore del software:** tramite le modifiche da apportare, l'obiettivo è di aumentare sia il valore interno del software, riducendo i costi di manutenzione, sia quello esterno, fornendo una migliore esperienza all'utente.

2.1.2 Starting Impact Set (SIS)

In quanto una delle operazioni di refactoring principali da effettuare sarà quella di partizionare i file del progetto in package, è prevista la modifica di tutte le componenti del sistema, almeno per quanto riguarda la loro locazione nella struttura del progetto. In aggiunta, è prevista un'operazione di refactoring mirata a ridurre la dimensione di alcune componenti poiché ritenute troppo complesse da manutenere e analizzare allo stato attuale. Inoltre, nel modulo *utils* sono state individuate due regex utilizzate per validare i link e le date. La complessità della regex per la validazione dei link verrà ridotta, in quanto sostituita con una regex che valida solo i link della piattaforma GitHub, unica piattaforma sulla quale è possibile effettuare l'analisi con il tool csDetector; in questo modo si migliorerà la manutenibilità del sistema. Per la regex della validazione delle date, questa sarà invece modificata per validare anche le date nei seguenti formati: mm.dd.YYYY e mm-dd-YYYY, in modo da migliorare la robustezza di CADOCS, evitando il fallimento del processo di identificazione degli smell richiesto in caso di date fornite con "." e "-". In questo modo sarà migliorato sia il valore interno del software, riducendo l'effort per la manutenzione, nel caso della semplificazione alla regex per la validazione del link; sia il valore esterno, riducendo il numero di errori presentati agli utenti in caso di una data fornita con separatori diversi da "/". Per questi motivi, è stata prodotta la seguente tabella descrivente le componenti di cui è prevista la modifica, con relative operazioni e conteggio delle stesse.

Nella tabella 2.1 sono stati riportati il numero di modifiche che si stima di attuare per ogni classe e modulo presente nel sistema. Oltre alle modifiche previste riportate nella Tabella, si prevede anche di effettuare le dovute modifiche ai test di unità e integrazione. Per la componente ToolSelector, si prevede di modificare l'implementazione dello Strategy pattern poiché attualmente presenta un riferimento a un tipo di ritorno non dichiarato.

	Componente coinvolta	Operazioni
Cadocs	X	-Rilocazione in package -Extract Class
CadocsIntents	X	-Rilocazione in package
cadocs_messages	X	-Rilocazione in package
CsDetectorTool	X	-Rilocazione in package
IntentManager	X	-Rilocazione in package
IntentResolver	X	-Rilocazione in package
slack_api_connection	X	-Rilocazione in package
ToolSelector	X	-Rilocazione in package -Modifica Strategy pattern
tool_strategy	X	-Rilocazione in package
utils	X	-Rilocazione in package -Modifica regex
Totale	**10**	

Tabella 2.1: Starting Impact Set - CR1

Si prevede l'impatto di tutti i test di unità e di integrazione delle componenti presenti nello Starting Impact Set.

2.1.3 Analisi delle dipendenze

L'approccio selezionato per la stima del Candidate Impact Set è basato sull'analisi delle dipendenze del sistema. Per questo motivo, si è scelto di sviluppare una matrice di dipendenze, la quale si propone di offrire una rappresentazione visiva e strutturata delle connessioni esistenti tra le varie componenti del sistema in esame.

Ogni cella (i, j) della matrice descrive il numero di chiamate ai metodi della componente della riga i dalla componente della colonna j.

L'analisi di questa matrice consente di individuare gruppi di componenti strettamente dipendenti, le quali modifiche potrebbero avere un impatto significativo su altre parti del sistema. Questo processo semplifica così l'individuazione delle componenti da inserire all'interno del Candidate Impact Set.

La matrice delle dipendenze è mostrata nella Tabella 2.2.

La matrice delle dipendenze 2.2, oltre ad essere utile per la stima del Candidate Impact Set, sarà utilizzata anche per strutturare e pianificare le operazioni di refactoring.

	Cadocs	Cadocs Intents	cadocs_ messages	CsDetector Tool	Intent Manager	Intent Resolver	slack_api _connection	Tool Selector	tool_ strategy	utils	**Totale**
Cadocs	/						7				**7**
CadocsIntents	8	/			5	10	5				**28**
cadocs_messages	1		/			3					**4**
CsDetectorTool				/		1		1	1		**3**
IntentManager	2				/						**2**
IntentResolver	3					/					**3**
slack_api_connection							/				**0**
ToolSelector						2		/			**2**
tool_strategy				2				2	/		**4**
utils	8									/	**8**
Totale	**22**	**0**	**0**	**2**	**5**	**16**	**12**	**3**	**1**	**0**	

Tabella 2.2: Matrice delle dipendenze

Per la realizzazione di questa tabella, sono state considerate sia le classi (nominate con lettera maiuscola) che i file contenenti metodi di servizio alle altre componenti. In particolare, ci si è focalizzati sull'analizzare le dipendenze tra le diverse componenti, senza valutare quelle interne alla componente stessa. Sono state calcolate come dipendenze gli usi, le chiamate ai metodi di una componente e le istanziazioni delle classi, mentre non sono state valutate le operazioni di *import*.

2.1.4 Candidate Impact Set (CIS)

A seguito delle analisi effettuate nella Sezione 2.1.3, è stato sviluppato il Candidate Impact Set descritto in Tabella 2.3. In particolare, è stato scelto di effettuare l'operazione di Move method anche per la classe IntentResolver, in modo da spostare il metodo *build_message* nella componente *cadocs_messages*.

Riguardo la componente CadocsIntents, che presentava il più alto numero di chiamate ai metodi da parte delle altre componenti, è stato scelto di non effettuare modifiche in questo senso, poiché si tratta di una classe che eredita la classe Enum di Python, fornendo quindi la lista di intent possibili del tool alle altre classi. Essa è comunque coinvolta nel CIS, in quanto sarà effettuata la rilocazione dei package.

La decisione di affrontare questi fix di bug durante la change request di refactoring è finalizzata a migliorare l'esperienza utente e a garantire una base solida per lo sviluppo futuro.

	Componente coinvolta	Operazioni
Cadocs	X	-Rilocazione in package -Extract Class
CadocsIntents	X	-Rilocazione in package
cadocs_messages	X	-Rilocazione in package
CsDetectorTool	X	-Rilocazione in package
IntentManager	X	-Rilocazione in package
IntentResolver	X	-Rilocazione in package -Move method
slack_api_connection	X	-Rilocazione in package
ToolSelector	X	-Rilocazione in package -Modifica Strategy pattern
tool_strategy	X	-Rilocazione in package
utils	X	-Rilocazione in package -Modifica regex
Totale	**10**	

Tabella 2.3: Candidate Impact Set - CR1

2.1.5 Discovered Impact Set (DIS)

A seguito dell'implementazione delle modifiche menzionate nella Sezione 3.1, è stato possibile derivare il Discovered Impact Set di questa change request. In particolare, è emersa una componente che necessitava modifiche non previste nel Candidate Impact Set, ovvero la componente *cadocs_service* del modulo CADOCS_NLU_Model, necessaria per permettere l'integrazione della nuova regex per la cattura delle date dai messaggi formulati dagli utenti.

2.1.6 Actual Impact Set (AIS)

L'AIS derivante dalle modifiche effettuate è mostrato nella tabella 2.4.

2.1.7 False Positive Impact Set (FPIS)

Dalla tabella 2.4 è possibile notare come il False Positive Impact Set sia risultato vuoto, in quanto non vi sono componenti presenti nel Candidate Impact Set che non sono presenti nell'Actual Impact Set.

	Candidate Impact Set	Actual Impact Set
Cadocs	X	X
CadocsIntents	X	X
cadocs_messages	X	X
CsDetectorTool	X	X
IntentManager	X	X
IntentResolver	X	X
slack_api_connection	X	X
ToolSelector	X	X
tool_strategy	X	X
utils	X	X
CADOCS_NLU_Model.		X
cadocs_service		
Totale	**10**	**11**

Tabella 2.4: Actual Impact Set - CR1

2.1.8 Metriche di processo

Abbiamo calcolato alcune metriche chiave per valutare l'efficacia dell'approccio di impact analysis. Queste metriche sono utili per valutare quanto l'approccio sia stato di supporto al processo di implementazione della CR. Per il calcolo delle metriche, è stato scelto di non considerare anche le componenti di test modificate e di cui era stata prevista la modifica per concentrare l'attenzione sulle componenti principali del progetto e per migliorare la comprensibilità dei risultati. Tale approccio è stato adottato anche per tutte le successive change request.

Di seguito sono riportati i risultati ottenuti:

Recall $= \frac{|CIS \bigcap AIS|}{|AIS|} = \frac{10}{11} = 0.909$

Abbiamo calcolato la metrica di Recall, che rappresenta la proporzione degli elementi correttamente identificati nel CIS che sono anche presenti nell'AIS.

Il risultato suggerisce che il 90.9% degli elementi effettivamente influenzati dalle modifiche è stato correttamente individuato nel CIS (ovvero 10 componenti su 11).

Precision $= \frac{|CIS \bigcap AIS|}{|CIS|} = \frac{10}{10} = 1.000$

La Precision è stata valutata per misurare la proporzione degli elementi effettivamente modificati del CIS rispetto a tutti gli elementi inclusi nel CIS. Questo risultato

indica che tutti gli elementi identificati nel CIS sono effettivamente influenzati dal cambiamento proposto.

Inclusiveness $= 0$ perché $AIS \nsubseteq CIS$

L'Inclusiveness è stata calcolata per verificare se il CIS include tutti gli elementi dell'AIS, e il risultato deriva dal non aver incluso anticipatamente la componente cadocs_service del progetto CADOCS_NLU_Model nel CIS.

2.2 CR_2 - Italian Conversation

2.2.1 Obiettivi della CR

La change request ha l'obiettivo di estendere le capacità linguistiche del sistema CADOCS al fine di supportare la lingua italiana e di migliorare il modello NLU inglese. L'implementazione di questa CR punta ad ottenere i seguenti obiettivi:

- **Aumentare il numero di possibili utenti**: introducendo il supporto alla lingua italiana sarà possibile interfacciarsi ad un pubblico più vasto includendo coloro che preferiscono utilizzare l'italiano come lingua di interazione con il sistema.
- **Miglioramento dell'esperienza utente**: l'introduzione di un modello per le richieste in italiano ed il miglioramento del modello inglese, che potrebbe risultare in una migliore esperienza utente; in quanto si mira a comprendere le richieste in entrambe le lingue e rispondere con maggiore accuratezza e coerenza.

2.2.2 Starting Impact Set (SIS)

A seguito di una analisi preventiva delle componenti relative all'interazione con l'utente in lingua inglese, è stata stilata la tabella 2.5, contenente le componente che si prevedono saranno modificate per la realizzazione della CR2, per permettere la conversazione in lingua italiana.

All'interno del progetto CADOCS main è stato preventivato il coinvolgimento della componente *slack_api_connection*, in quanto due dei sui metodi (*update_waiting _message* e *post_waiting_message*) inoltrano messaggi alla piattaforma Slack. Quindi,

	Componente Coinvolta	Operazioni
CadocsSlack	X	-Modifica new_message(), error_message(), something_wrong()
cadocs_messages	X	-Rimozione ask_confirm() -Modifica di tutti i metodi -Modifica update_waiting_message() post_waiting_message()
slack_api_connection	X	-Rimozione handle_action(), action_received() e update_dataset()
CADOCS_NLU_Model. CADOCS	X	-Aggiunta gestione modelli NLU -Modifica CADOCSModel
CADOCS_NLU_Model. cadocs_service	X	-Integrazione nuovi modelli NLU
Totale	**5**	

Tabella 2.5: Starting Impact Set - CR2

sarà necessario apportare delle modifiche per adattare le risposte di questo modulo in base alla lingua della richiesta. Inoltre, per questo modulo, è preventivata la cancellazione dei metodi *action_received* e *handle_action*, poiché i nuovi modelli NLU che verranno introdotti sono più complessi ed il loro retraining comporta un uso eccessivo di risorse, risulta quindi inutile andare ad aggiornare il dataset del modello.

All'interno di questo progetto è stato preventivato anche il coinvolgimento della classe *CadocsSlack*, infatti essa è coinvolta nell'utilizzo di metodi per la costruzione dei messaggi di risposta. Dato che verranno introdotti nuovi modelli NLU per i quali non verranno effettuate operazioni di retraining, si prevede anche l'eliminazione della funzione *ask_confirm* che gestisce questa funzionalità.

Infine, la componente che risulta avere l'impatto maggiore è *cadocs_messages*. Infatti, questa componente, si occupa della costruzione dei messaggi di risposta del chatbot ed è quindi la componente che richiederà il maggior numero di modifiche. Per concludere, nel progetto *CADOCS*, si prevede l'aggiunta di una nuova componente per l'identificazione della lingua della richiesta.

Questa modifica ha un impatto anche sul progetto CADOCS_NLU_Model, in quanto sarà necessario aggiungere un modello NLU per le richieste italiane ed è prevista la sostituzione del modello esistente con un modello più avanzato. Non è prevista la creazione dei modelli per questa CR, poiché sono già stati sviluppati precedentemente e verranno quindi solo integrati. È necessario, quindi, modificare la classe *CADOCSModel*.

Si prevede l'impatto di tutti i test di unità e di integrazione delle componenti presenti nello Starting Impact Set.

2.2.3 Analisi delle dipendenze

L'analisi delle dipendenze e la tabella delle dipendenze prodotta si riferisce alla tabella 3.1 prodotta a seguito della CR_1. Sulla base di questa tabella sarà stimato il Candidate Impact Set.

2.2.4 Candidate Impact Set (CIS)

A seguito delle analisi effettuate nella Sezione 2.2.3 è stato sviluppato il Candidate Impact Set 2.6, in cui sono state valutate solo le modifiche dirette derivanti dalle componenti presenti nello Starting Impact Set (tabella 2.5) e non quelle indirette. Questa scelta è stata presa per non sovrastimare eccessivamente le dimensioni del CIS. Per quanto riguarda il progetto CADOCS_NLU_Model, non è stata sviluppata una matrice delle dipendenze poiché esso è composto solo da due componenti.

	Componente Coinvolta	Operazioni
CadocsSlack	X	-Modifica new_message(), error_message(), something_wrong() -Rimozione ask_confirm() -Direct impact di slack_api_connection
CadocsIntent	X	-Direct impact di slack_api_connection, CadocsSlack, cadocs_messages
cadocs_messages	X	-Modifica di tutti i metodi -Direct impact di CadocsSlack
cadocs_utils	X	-Direct impact di slack_api_connection, CadocsSlack
IntentManager	X	-Direct impact di CadocsSlack
IntentResolver	X	-Direct impact di CadocsSlack
slack_api_connection	X	-Modifica update_waiting_message(), post_waiting_message(), -Rimozione handle_action(), action_received() e update_dataset()
utils	X	-Direct impact di CadocsSlack
CADOCS_NLU_Model. CADOCS	X	-Aggiunta gestione modelli NLU -Modifica CADOCSModel
CADOCS_NLU_Model. cadocs_service	X	-Integrazione nuovi modelli NLU
Totale	**10**	

Tabella 2.6: Candidate Impact Set - CR2

2.2.5 Discovered Impact Set (DIS)

Dopo aver implementato le modifiche delineate nella Sezione 3.2, è stata derivato il Discovered Impact Set. Questo processo ha rilevato che non sono state apportate modifiche alle componenti non precedentemente elencate nel CIS della Tabella 2.6. Di conseguenza, il DIS risultante è vuoto.

2.2.6 Actual Impact Set (AIS)

L'AIS ottenuto a seguito delle modifiche effettuate è riportato in tabella 2.7. Nella Tabella sono state omesse le componenti di test relative ai moduli modificati. In particolare, per tutte le componenti incluse nell'Actual Impact Set, sono stati apportati aggiornamenti ai test di unità e di integrazione. Tuttavia, vi è un'eccezione per la componente *utils*, in quanto in essa sono stati modificati solo i test di unità, dato che la componente non presentava test di integrazione, e per la componente *IntentResolver*, la quale non ha necessitato di modifiche alle sue classi di test. Le componenti del progetto CADOCS_NLU non presentano test implementati.

	Candidate Impact Set	Actual Impact Set
CadocsSlack	X	X
CadocsIntent	X	
cadocs_messages	X	X
cadocs_utils	X	
IntentManager	X	
IntentResolver	X	X
slack_api_connection	X	X
utils	X	X
CADOCS_NLU_Model. CADOCS	X	X
CADOCS_NLU_Model. cadocs_service	X	X
Totale	**10**	**7**

Tabella 2.7: Actual Impact Set - CR2

2.2.7 False Positive Impact Set (FPIS)

Dalla tabella 2.7, è possibile estrarre il False Positive Impact Set. Questo contiene le componenti *CadocsIntent*, *cadocs_utils* e *IntentManager*.

Questo deriva dall'aver inserito nel Candidate Impact Set (tabella 2.6) anche le componenti impattate direttamente da quelle oggetto di modifica.

2.2.8 Metriche di processo

Recall $= \frac{|CIS \bigcap AIS|}{|AIS|} = \frac{7}{7} = 1$
Precision $= \frac{|CIS \bigcap AIS|}{|CIS|} = \frac{7}{10} = 0.7000$
Inclusiveness $= 1$ perché $AIS \subseteq CIS$

L'approccio di impact analysis ha prodotto risultati migliori rispetto all'approccio della CR_1, come indicato dalle metriche ottenute. La recall pari a 1 indica che l'approccio è efficace nel catturare tutte le modifiche rilevanti. Una precision pari a 0.7 suggerisce che le modifiche indicate nel CIS sono state in buona parte effettivamente impattate. Inoltre, l'inclusiveness pari a 1 afferma la capacità dell'approccio di identificare tutti gli elementi interessati da modifiche.

2.3 CR_3 - Docker Porting csDetector

2.3.1 Obiettivi della CR

Analizzando i vari moduli necessari al corretto funzionamento del chatbot CADOCS, è stato notato che l'installazione del tool csDetector risulta essere un punto critico. L'installazione di questo modulo consiste in una serie di passaggi complessi ed il suo funzionamento riscontra problemi di compatibilità con sistemi operativi MacOS, limitando quindi anche i possibili utilizzi di CADOCS. Per questi motivi, la CR_3 ha come obiettivo quello di facilitare l'installazione di questo modulo necessario al funzionamento di CADOCS, tramite una ***containerization*** di csDetector. Per quanto riguarda invece i moduli CADOCS e CADOCS_NLU, questi non necessitano di tale attenzione in quanto la loro installazione non presenta particolari complicazioni.

2.3.2 Starting Impact Set (SIS)

Eseguendo un'analisi per la creazione di un'immagine Docker per il modulo csDetector, è stato identificato un impatto iniziale che richiede di modificare i file *devNetwork* e *csDetectorWebService*. Queste modifiche sono necessarie per supportare l'integrazione dell'immagine Docker basata su Ubuntu, con il modulo csDetector.

Nel dettaglio, la modifica coinvolge l'aggiornamento del parametro *FILEBROWSER_PATH* nella componente *devNetwork*. La sua correzione è necessaria per garantire che l'immagine possa accedere ai file e alle risorse. Inoltre, è necessario apportare una modifica al web service di csDetector per consentire l'accesso anche quando è in esecuzione all'interno di un container Docker, affinché sia possibile l'accesso da varie interfacce sulla macchina host.

Nella tabella 2.8 sono riportate le modifiche che si stima di attuare.

	Componente coinvolta	Operazioni
csDetectorWebService	X	-Modifica per il funzionamento con Docker
devNetwork	X	-Modifica FILEBROWSER_PATH
Totale	2	

Tabella 2.8: Starting Impact Set - CR3

2.3.3 Analisi delle dipendenze

Per comprendere meglio la struttura del modulo csDetector è stata realizzata la matrice delle dipendenze, mostrata in Tabella 2.9. Per la realizzazione della tabella sono state considerate solo le dipendenze tra file diversi e non quelle presenti tra elementi dello stesso file. Sono state calcolate come dipendenze gli usi, le chiamate ai metodi di una componente e le istanziazioni delle classi, mentre non sono state valutate le operazioni di import. Inoltre, per la realizzazione della tabella si è scelto di non considerare i file all'interno della directory *graphhqlAnalysis*, in quanto queste componenti si occupano della rappresentazione grafica dei community smells, pertanto si stima non verranno impattati nell'implementazione della CR_3.

	alias Worker	authorAlias Extractor	centrality Analysis	commit Analysis	configuration	csDetector	csDetector Adapter	csDetector WebService	dev Analysis	dev Network	perspective Analysis	politeness Analysis	repo Loader	smell Detection	stat Analysis	tag Analysis	utils	**Totale**
aliasWorker	/									1								**1**
authorAliasExtractor		/																**0**
centralityAnalysis			/							2								**2**
commitAnalysis				/						1								**1**
configuration	1	2	4	2	/				1	1	1	1	1	1		2		**17**
csDetector						/	1											**1**
csDetectorAdapter							/	3										**3**
csDetectorWebService								/										**0**
devAnalysis									/	1								**1**
devNetwork						1				/								**1**
perspectiveAnalysis											/							**0**
politenessAnalysis										1		/						**1**
repoLoader		1								1			/					**2**
smellDetection										1				/				**1**
statsAnalysis			5	7								2			/	1		**15**
tagAnalysis										1						/		**1**
utils	1	2	3	1													/	**7**
Totale	**2**	**5**	**12**	**10**	**0**	**1**	**1**	**3**	**1**	**10**	**1**	**3**	**1**	**1**	**0**	**3**	**0**	

Tabella 2.9: Matrice delle dipendenze del modulo csDetector

2.3.4 Candidate Impact Set (CIS)

L'approccio per la realizzazione del Candidate Impact Set utilizzato per l'Impact Analysis della CR_2, ovvero analizzare tutti i possibili impatti diretti dei moduli coinvolti derivanti dalla tabella 2.9, avrebbe generato un CIS contenente un numero elevato di moduli che probabilmente non sarebbero stati impattati. Per questo motivo, è stato scelto di utilizzare un approccio differente per questa change request. In particolare, sono state prima individuate le singole modifiche da effettuare ai moduli tramite un approccio manuale, ed in seguito sono state analizzate le possibili componenti impattate dalla modifica. Con questo approccio è stato cercato in particolare di individuare tutte le modifiche necessarie per il corretto funzionamento del tool su un sistema operativo Ubuntu.

In particolare, per la componente *csDetectorWebService* si prevede di modificare esclusivamente un parametro di avvio e l'interazione con il sistema operativo per la creazione di directory nel metodo *getSmells()*, non recando modifiche ad altre componenti. Anche per la componente *devNetwork* non si prevedono altri moduli impattati, in quanto la variabile FILEBROWSER_PATH è utilizzata solo in questo modulo.

Durante questa fase si è ritenuto necessario andare ad analizzare anche le librerie utilizzate da csDetector, in quanto potrebbero essere richiesti aggiornamenti all'utilizzo delle librerie per garantire il corretto funzionamento del tool csDetector su sistemi operativi diversi da Windows. Da questa analisi è emersa la necessità di modificare

le componenti coinvolte nella creazione dei grafi. Esaminando il file delle specifiche delle librerie (*requirements.txt*), è stato individuato che la libreria coinvolta nella creazione dei grafi era *networkx*. Pertanto, è stato necessario identificare i moduli e le classi che facevano uso di questa libreria per valutare gli impatti potenziali. L'analisi ha rivelato che il modulo *centralityAnalysis* era l'unico a utilizzare la libreria *networkx* per la creazione dei grafi. Di conseguenza, questo modulo è stato inserito nel CIS.

È stato analizzato manualmente l'impatto che le modifiche apportate al modulo csDetector avrebbero potuto avere sul modulo CADOCS, mentre per il modulo CADOCS_NLU non si prevedono modifiche in quanto non interagisce con il modulo csDetector. In particolare, si è previsto di modificare le componenti *CsDetectorTool* e *slack_api_connection*, poiché esse si occupano dell'interazione con csDetector. Date queste modifiche, non è prevista la modifica anche delle componenti richiamate da *slack_api_connection*, poiché esso si occupa solo di ricevere le richieste da parte di csDetector per poi gestirle ed inoltrarle, mentre sono previste possibili modifiche alle componenti *intent_resolver* e *tool_strategy*, poiché dalla tabella 3.2 risultano avere dipendenze da e verso *CsDetectorTool*.

Nella tabella 2.10 sono quindi riportate le modifiche descritte in questo paragrafo.

	Componente coinvolta	Operazioni
centralityAnalysis	X	-Modifica per aggiornamento libreria
csDetectorWebService	X	-Modifica per il funzionamento con Docker
devNetwork	X	-Modifica FILEBROWSER_PATH -Direct impact di centralityAnalysis
CADOCS.api.slack_api_connection	X	-Modifica diretta di csDetectorWebService
CADOCS.intent_handling.CsDetectorTool	X	-Modifica diretta di csDetectorWebService
CADOCS.intent_handling.intent_resolver	X	-Modifica diretta di CsDetectorTool
CADOCS.intent_handling.tool_strategy	X	-Modifica diretta di CsDetectorTool
Totale	7	

Tabella 2.10: Candidate Impact Set - CR3

Per tutte le componenti di CADOCS elencate nel CIS si prevede la modifica dei test ad esse associati.

Non sono previste modifiche ai test di csDetector, poiché apportare tali modifiche richiederebbe l'aggiunta di nuovi test oltre alle eventuali modifiche ai test esistenti. L'implementazione di queste modifiche richiederebbe uno sforzo significativo, e pertanto non è stata inclusa nell'attuale fase di evoluzione. I test per il tool csDetector

verranno quindi eseguiti esclusivamente durante la fase di test di sistema. Questa scelta è motivata dalla volontà di concentrarsi solo sull'interazione di csDetector con il tool CADOCS.

2.3.5 Discovered Impact Set (DIS)

A seguito delle modifiche descritte nella Sezione 3.3, è stato possibile realizzare l'Actual Impact Set, mostrato in Tabella 2.11. Come è possibile notare, non sono state individuate componenti che hanno necessitato modifiche non precedentemente incluse nel Candidate Impact Set. Di conseguenza, il Discovered Impact Set risultante è vuoto.

2.3.6 Actual Impact Set (AIS)

Nella Tabella 2.11 sono riportate le componenti modificate. Come per la CR_2, non sono stati riportate anche le componenti di test soggette a modifiche. In particolare, è risultato necessario modificare esclusivamente i test di unità della componente *slack_api_connection*.

	Candidate Impact Set	Actual Impact Set
centralityAnalysis	X	X
csDetectorWebService	X	X
devNetwork	X	X
CADOCS.api.slack_api_connection	X	X
CADOCS.intent_handling.CsDetectorTool	X	X
CADOCS.intent_handling.intent_resolver	X	
CADOCS.intent_handling.tool_strategy	X	
Totale	7	5

Tabella 2.11: Actual Impact Set - CR3

2.3.7 False Positive Impact Set (FPIS)

Dall'analisi della Tabella 2.11, risultano esclusivamente le componenti *intent_resolver* e *tool_strategy* nel FPIS.

2.3.8 Metriche di processo

Le metriche di processo sono riportate di seguito:

Recall $= \frac{|CIS \bigcap AIS|}{|AIS|} = \frac{5}{5} = 1$

Precision $= \frac{|CIS \bigcap AIS|}{|CIS|} = \frac{5}{7} = 0.7142$

Inclusiveness $= 1$ perché $AIS \subseteq CIS$

Questo approccio di impact analysis ha prodotto risultati molto simili a quelli ottenuti per la CR_2, pur avendo utilizzato un approccio diverso.

Capitolo 3

Implementazione delle modifiche

3.1 Implementazione CR_1

Nel corso dell'implementazione della CR_1, sono state apportate diverse modifiche al fine di migliorare la struttura del codice, aumentare la modularità e consentire una gestione più efficiente delle funzionalità. Le modifiche eseguite sono riportate nelle sezioni successive.

3.1.1 Operazioni di Extract Class e Move Method

Per snellire le componenti che presentavano troppe dipendenze a seguito della realizzazione della tabella 2.1.3, è stato scelto di effettuare diverse operazioni di Extract Class, cercando di mantenere una coerenza in termini di funzionalità fornite dalle componenti risultanti. In particolare, la componente Cadocs è stata snellita andando ad estrarre i metodi *get_last_execution* e *save_execution* creando la nuova componente *cadocs_utils*, poiché i metodi citati erano poco coesi con il resto della classe. In questa operazione, è stato anche deciso di rinominare la componente *Cadocs* in *CadocsSlack*, poiché ritenuto più opportuno in quanto specifica dell'interazione con Slack, in modo da poter separare le implementazioni per una futura integrazione con altri tool. Inoltre, la classe *CadocsIntents* è stata estratta dal file contenente anche la componente *utils*, in modo da rendere la struttura del progetto più aderente alla vera suddivisione in componenti del progetto. Infine, il metodo *build_message* della componente *IntentResolver* è stato rilocato nella componente *cadocs_messages* con

una operazione di Move method, in modo da diminuire le dipendenze della classe IntentResolver e rendere più coerenti i metodi in essa presenti. Le modifiche effettuate rendono quindi il progetto più manutenibile.

3.1.2 Rilocazione delle classi e file in package

Per la rilocazione delle classi in package sono stati creati quattro package: *api*, *chatbot*, *intent_handling* e *service*; ognuno dei quali raggruppa classi e moduli con compiti affini. In particolare, l'approccio utilizzato per la suddivisione in package del sistema è basato sull'analisi delle dipendenze tra i moduli mostrate nella tabella 2.1.3 e analizzando il tipo di funzionalità offerte dalle componenti. L'obiettivo è stato quello di bilanciare un basso accoppiamento tra le componenti in package diversi mantenendo un'alta coesione interna e raggruppando le componenti che si occupano di parti simili del sistema nello stesso package. Tale strategia ha portato alla creazione dei seguenti package:

- **Package *api***. Questo package ospita il modulo *slack_api_connection*, dedicato all'interfacciamento con la piattaforma Slack. In questo package, sarà possibile includere futuri moduli che si occuperanno dell'interfacciamento con altre piattaforme. La creazione di questo package favorisce la separazione delle interazioni esterne dal resto del sistema, consentendo un'implementazione più flessibile e adattabile.

- **Package *chatbot***. Qui risiede il cuore del chatbot stesso. Le classi e i moduli in questo package sono responsabili della gestione delle conversazioni, infatti sono presenti le seguenti componenti: *cadocs_slack*, *cadocs_utils* e *intent_manager*. In particolare, queste si interessano all'interazione con il chatbot per la piattaforma Slack e per l'individuazione dell'intent.

- **Package *intent_handling***. Questo package contiene le classi che si occupano della gestione del processo di risposta all'utente in base all'intent identificato, comunicando con il tool csDetector. Sono qui presenti le classi *CadocsIntent*, *IntentResolver*, *ToolSelector*, *Tool* e *CsDetectorTool*

- **Package *service***. In questo package si trovano i file di supporto alle funzionalità di CADOCS, in particolare le componenti *cadocs_messages*, per la costruzione dei messaggi di risposta all'utente, e *utils*, per la validazione delle date e dei link forniti.

3.1.3 Modifica delle regex

Come già descritto nella Sezione 2.1.2, uno degli obiettivi di questa fase di refactoring è stato quello di modificare le regex per il riconoscimento e la validazione di link e date. In particolare, per la validazione delle date, è stato necessario modificare la regex utilizzata dal progetto CADOCS_NLU_Model. Per permettere una corretta interazione tra i vari moduli di CADOCS, in CADOCS_NLU_Model è stato necessario implementare il replace delle date dai nuovi formati al formato standard con il separatore "/".

Per quanto riguarda invece la regex per restringere i link accettati dalla validazione ai soli progetti GitHub, è stato deciso di non modificare la regex del progetto CADOCS_NLU_Model, in quanto si sarebbe limitato il campo dei link consentiti e di conseguenza si sarebbero limitati i possibili utilizzi futuri del modello. Inoltre, questa scelta permette l'estensione del tool CADOCS ad altri tool non basati esclusivamente sull'identificazione di community smell tramite l'analisi di repository GitHub, senza richidere una ulteriore modifica al modulo CADOCS_NLU_Model.

La modifica alla regex per la validazione delle date ha comportato la modifica del messaggio di errore fornito all'utente indicando anche i nuovi formati accettati dal tool per le date.

3.1.4 Analisi delle dipendenze

Come già fatto nella Sezione 2.1.3 è stata creata una nuova matrice delle dipendenze in modo da analizzare le dipendenze prima e dopo le modifiche apportate. In questo modo è possibile notare se si è verificata una riduzione del grado di accoppiamento tra le varie componenti.

	Cadocs Slack	Cadocs Intents	cadocs_ messages	cadocs_ utils	CSDetector Tool	Intent Manager	Intent Resolver	slack_api _connection	Tool Selector	tool_ strategy	utils	**Total**
CadocsSlack	/							5				**5**
CadocsIntents	7	/	4			4	5	4				**24**
cadocs_messages	2		/									**2**
cadocs_utils	1			/				2				**3**
CsDetectorTool					/		2					**2**
IntentManager	2					/						**2**
IntentResolver	2						/					**2**
slack_api_connection								/				**0**
ToolSelector							2		/			**2**
tool_strategy					1				3	/		**4**
utils	7										/	**7**
Total	**21**	**0**	**4**	**0**	**1**	**4**	**9**	**11**	**3**	**0**	**0**	

Tabella 3.1: Matrice delle dipendenze dopo le modifiche della CR_1

Come è possibile notare dalla tabella 3.1 c'è stata una riduzione delle dipendenze tra le classi ed i file. Questo indica che tramite le modifiche di refactoring si è verificata una riduzione dell'accoppiamento tra le componenti.

3.2 Implementazione CR_2

3.2.1 Modifiche dei modelli NLU

Al momento di implementare le modifiche della CR_2, i nuovi modelli di NLU per la conversazione in italiano e inglese erano già stati da noi precedentemente implementati all'interno del progetto CADOCS_NLU, in particolare nel branch *dev*. Per questo motivo, le modifiche da effettuare sono state basate su questa versione aggiornata del progetto NLU. Tuttavia, l'implementazione effettuata ha comunque necessitato operazioni di refactoring, riportate nella Sezione 3.2.2.

3.2.2 Modifica della classe CADOCSModel

Per la selezione dei modelli è stato utilizzato uno strategy pattern al fine di migliorare l'efficienza del modello di NLU nel selezionare la lingua corretta. Lo strategy è stato implementato nella classe *ModelSelector*. Di conseguenza, è stata modificata la classe *LanguageModel* in una classe astratta per definire una struttura

di base con comportamento condiviso tra sottoclassi. La classe *PredictionService* è stata estratta dal file *CADOCS* ed inserita nel file *prediction_service*, inoltre sono state apportate delle modifiche in modo che la classe implementi lo strategy pattern. Queste modifiche mirano a migliorare la manutenibilità e la flessibilità del nostro sistema di NLU. L'adozione dello strategy pattern, la trasformazione in classe astratta e la separazione delle responsabilità contribuiscono quindi alla creazione di codice più comprensibile e manutenibile.

3.2.3 Gestione della lingua in CADOCS

In risposta alla CR_2 per l'aggiunta della conversazione in italiano, nel modulo CADOCS sono state apportate le seguenti modifiche:

- **Rimozione delle funzioni di retraining**. Sono state rimosse le seguenti funzioni: *handle_action*, *action_received*, *update_dataset* e *ask_confirm*. Queste funzioni inizialmente sono state progettate per interagire con il modulo NLU per la funzione di retraining. Tuttavia, in seguito alla sostituzione del modello NLU con uno più complesso che non richiede il retraining, queste funzioni sono diventate obsolete. La rimozione di queste funzioni semplifica il codice e lo rende più manutenibile, eliminando componenti inutilizzate.

- **Aggiunta di un componente per la gestione della lingua**. È stato utilizzato il design pattern *singleton* per gestire la rilevazione della lingua del messaggio. Questo significa che una volta impostata la lingua alla ricezione del messaggio da parte dell'utente, è possibile accedere alla stessa istanza della componente in qualsiasi punto del sistema utilizzando il singleton. Per implementare questa funzionalità, è stata aggiunta la classe *LanguageHandler*.

- **Aggiunta dei controlli per la risposta nella lingua impostata dalla classe *LanguageHandler***. È stato esteso l'uso della classe *LanguageHandler* per gestire la lingua delle risposte del chatbot in tutte le componenti coinvolte nella costruzione di un messaggio di risposta. Questo assicura che le risposte del bot siano coerenti con la lingua con cui l'utente ha inviato la richiesta.

- **Modifica regex di validazione link**. È stato deciso di modificare nuovamente la regex per la validazione dei link in seguito alla modifica effettuata durante la CR_1. In particolare è stato scelto di non accettare i link che non iniziano con "https://", a causa di problemi di compatibilità con il tool csDetector non emersi precedentemente in fase di test.

- **Modifica gestione della data**. Durante lo sviluppo di questa CR è emersa una problematica legata al passaggio della data dal progetto CADOCS al tool csDetector per l'analisi. In particolare, è stato necessario invertire il giorno ed il mese della data, ove questa fosse presente nel messaggio. Tale problematica non era sorta durante i test di sistema per la CR_1 a causa della scelta delle date per gli input validi.

In conclusione, le modifiche apportate a CADOCS hanno permesso l'integrazione della conversazione in italiano, migliorando la gestione della lingua e semplificando il codice grazie alla rimozione di funzioni obsolete. Questi aggiornamenti contribuiscono a rendere il chatbot più versatile e manutenibile, garantendo una migliore esperienza utente. Al termine dell'implementazione della CR_2, è stata anche effettuata una modifica alla descrizione del community smell Toxic Communication, sia nella versione inglese che italiana.

3.2.4 Analisi delle dipendenze

Di seguito, viene riportata la matrice delle dipendenze realizzata dopo il completamento della CR_2.

	Cadocs Slack	Cadocs Intents	cadocs_ messages	cadocs_ utils	CSDetector Tool	Intent Manager	Intent Resolver	language_ handler	slack_api _connection	Tool Selector	tool_ strategy	utils	**Total**
CadocsSlack	/								3				**3**
CadocsIntents	3	/	4			4	5		2				**18**
cadocs_messages	2		/										**2**
cadocs_utils	1			/					1				**2**
CSDetectorTool					/		1						**1**
IntentManager	2					/							**2**
IntentResolver	2						/						**2**
language_handler	4		8					/	6				**18**
slack_api_connection									/				**0**
ToolSelector							2			/			**2**
tool_strategy					1					3	/		**4**
utils	7											/	**7**
Total	**21**	**0**	**12**	**0**	**1**	**4**	**8**	**0**	**12**	**3**	**0**	**0**	

Tabella 3.2: Matrice delle dipendenze dopo le modifiche della CR_2

3.3 Implementazione CR_3

Nel processo di creazione di un container Docker per ospitare il web service di csDetector, sono state apportate diverse modifiche al codice ed è stato aggiunto il Dockerfile. Di seguito, vengono descritte le modifiche principali effettuate:

- **Aggiunta Dockerfile per la build del container**. È stato creato un Dockerfile per definire l'ambiente di esecuzione del web service di csDetector, che include tutte le dipendenze necessarie, le configurazioni preliminari e le istruzioni per l'esecuzione del web service.

- **Modifiche al codice del web service**. Il codice del web service, basato su Flask, è stato modificato per garantire la corretta esecuzione all'interno del Docker container. È stato aggiunto il parametro *host='0.0.0.0'* per permette al web service di essere accessibile anche quando è in esecuzione all'interno del container.

- **Compatibilità con più sistemi operativi**. Al fine di garantire il corretto funzionamento del tool sul container, lasciando invariato il funzionamento per piattaforme di esecuzione basate su sistema operativo Windows, sono stati aggiunti controlli sul tipo di piattaforma di esecuzione utilizzando la libreria *platform* di Python. In questo modo, è stato possibile adattare il codice per la creazione delle directory di salvataggio e per l'apertura delle directory necessarie per l'esecuzione dell'analisi di csDetector, utilizzando comandi compatibili con il sistema operativo specifico.

- **Aggiornamento della funzione per la creazione dei grafi**. La funzione *prepareGraph* è stata aggiornata, in quanto sono state modificate delle dipendenze della libreria *networkx* utilizzata da quest'ultima. Questa modifica permette la corretta creazione dei grafi anche su piattaforme di esecuzione basate su sistemi operativi diversi da Windows.

Nel modulo CADOCS, sono state apportate due modifiche significative per garantire un corretto funzionamento del chatbot:

- **Sostituzione di "localhost" con "127.0.0.1"**. Nelle richieste di indirizzi, la stringa "localhost" è stata sostituita con "127.0.0.1". Questa modifica è stata effettuata per consentire la corretta interazione tra il modulo CADOCS ed il container Docker.

- **Modifica dell'operazione di split del path dei file**. Nella classe *CsDetectorTool*, è stata apportata una modifica all'operazione di split del percorso dei file ricevuti dal web service di csDetector. Poiché i percorsi dei file possono variare tra sistemi operativi, è stata aggiunto un controllo per determinare il separatore di percorso appropriato in base alla presenza di "/" o "\". Questa modifica garantisce la gestione corretta dei percorsi dei file in base alla piattaforma di esecuzione.

Queste modifiche sono state apportate per garantire che il modulo CADOCS sia in grado di operare senza problemi all'interno del container Docker e su sistemi

operativi diversi, garantendo la compatibilità e il corretto funzionamento dell'applicazione. Non sono state aggiunte o modificate dipendenze tra le componenti dei moduli CADOCS e csDetector, per cui non risulta necessaria un'ulteriore analisi.

Sono state realizzate due immagini Docker:

- ***latest***: tag che contiene l'immagine Docker di cui è stata fatta la build su processore con architettura di tipo **amd64**. Questa immagine dovrebbe risultare funzionante per la maggior parte delle macchine host, fatta eccezione per macchine host basate su processore Apple Silicon.

- ***macOS***: tag che contiene l'immagine Docker di cui è stata fatta la build su processore con architettura di tipo **arm64/v8**. Questa immagine risulta compatibile solo con macchine host basate su processore Apple Silicon.

Le immagini Docker prodotte sono presenti al seguente link: `https://hub.docker.com/r/acannavale/cs-detector-webservice`.

Capitolo 4

Conclusioni

4.1 Esito delle modifiche

Al termine delle modifiche descritte nel Capitolo 3, il tool CADOCS risulta essere evoluto rispetto al suo stato iniziale. In particolare, il modulo principale di CADOCS presenta adesso una strutturazione in package che prima non era presente, ed è adesso corredato da una serie di test di unità e integrazione che risulteranno di supporto per le successive modifiche al tool. A seguito delle modifiche per la CR_2, è stata aggiunta la possibilità di interagire con il chatbot anche utilizzando la lingua italiana, oltre ad aver migliorato il modello NLU per la lingua inglese. Questo ha causato una modifica nel comportamento del chatbot alle richieste prodotte dagli utenti, come è stato possibile notare durante l'esecuzione dei test di sistema, producendo in ogni caso risultati accettabili. Un'altra modifica importante apportata è stata la realizzazione della CR_3, che ha comportato la **containerization** di csDetector, su cui si basa attualmente CADOCS. Questa modifica ha semplificato notevolmente il processo di installazione dell'intero tool. Grazie a queste ultime modifiche implementate, si auspica che il tool diventi molto più accessibile ed utilizzato.

4.2 Deviazioni da quanto pianificato

Durante il corso dell'implementazione del progetto, ci sono state alcune deviazioni rispetto alla pianificazione iniziale che hanno richiesto una gestione diversa. Di seguito, un riepilogo delle principali deviazioni effettuate:

- **Problemi con l'installazione iniziale**. L'installazione iniziale dell'intero sistema CADOCS ci ha fatto riscontrare alcune difficoltà impreviste, che hanno richiesto più tempo del previsto. Questi problemi includevano la risoluzione di dipendenze e la configurazione dell'ambiente di esecuzione.

- **Problemi con il tool Selenium**. L'automazione dei test di sistema tramite tool Selenium non è stata possibile, in quanto le richieste degli utenti consistono in messaggi scritti in una chat Slack. L'esecuzione del tool registrava anche l'attesa alla risposta del chatbot ed inoltre non era in grado di verificare l'output prodotto con l'oracolo. Questo ha comportato una deviazione rispetto a quanto pianificato rispetto al Test Plan, oltre a richiedere maggior effort per l'esecuzione manuale dei test.

- **Creazione della test suite iniziale**. La fase di creazione della test suite iniziale per il modulo CADOCS ha richiesto un effort maggiore rispetto a quello preventivato, principalmente a causa dei problemi riscontrati con il tool *pytest* e della difficoltà nella realizzazione di stub. Inoltre, anche la struttura di alcune parti del sistema, basate su interazioni http, ha reso difficile questa fase.

- **Implementazione della CR_4**. La change request 4, inizialmente pianificata, non è stata implementata come previsto a causa di un esaurimento dell'effort disponibile. Questa decisione è stata presa dopo una valutazione della complessità dell'implementazione e delle priorità del progetto.

- **Aggiornamento e aggiunta dei test per il modulo csDetector**. Come precedentemente specificato, non è stato possibile aggiornare o aggiungere ulteriori test per csDetector a causa dell'effort richiesto, che è risultato essere più elevato del previsto. Questa limitazione ha reso necessario concentrarsi sulla valutazione delle funzionalità durante il testing di sistema.

Inoltre, un problema significativo che è emerso durante il progetto è la disponibilità limitata di strumenti di supporto per le operazioni di reverse engineering specifici per i progetti Python e di strumenti di supporto per il testing in Python. Questa carenza ha richiesto un effort maggiore per l'esecuzione di queste fasi, come già parzialmente indicato.

4.3 Lessons Learned

Durante la realizzazione di questo progetto, le lezioni apprese sono le seguenti:

- **Utilizzo strutturato di GitHub**. Abbiamo imparato a sfruttare alcune funzionalità di GitHub, tra cui la gestione dei branch, l'uso efficace di issues e pull request, e l'implementazione di GitHub Actions per l'automazione dei processi di test del software. Questo ha permesso di gestire il progetto avendo una maggiore visibilità dello stato delle attività e permettendo una pianificazione dei task.

- **Importanza del pair programming**. Abbiamo sperimentato i vantaggi del pair programming, riconoscendo la sua importanza nella risoluzione efficiente dei problemi, nell'apprendimento reciproco e nella condivisione delle conoscenze tra i membri del team di sviluppo.

- **Utilizzo dei design pattern**. Abbiamo compreso l'importanza dell'utilizzo dei design pattern nell'architettura software. L'adozione di design pattern ben consolidati ha reso più semplice e chiara l'implementazione di alcune modifiche effettuate.

- **Importanza del Reverse Engineering**. Abbiamo potuto notare come la realizzazione di diagrammi di alto livello, in particolare Sequence Diagrams, sia stata fondamentale per la comprensione dei processi di un progetto già sviluppato. Questo ha sicuramente velocizzato le successive fasi del progetto, avendo consolidato la nostra conoscenza del tool.

- **Rilevanza dei test e dei test di regressione**. Abbiamo potuto notare l'importanza dei test per garantire la qualità del software. I test di regressione hanno dimostrato di essere strumenti essenziali per identificare eventuali errori introdotti dalle modifiche, contribuendo così a mantenere l'affidabilità del sistema nel tempo.

Queste lezioni apprese hanno arricchito la nostra esperienza di sviluppo e di lavoro in team. Siamo riusciti a consolidare le nostre capacità nella gestione di progetti software complessi e siamo determinati a mettere in pratica queste lezioni in futuri progetti.