
UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Magistrale in Informatica


Corso di
INGEGNERIA, GESTIONE ED EVOLUZIONE DEL SOFTWARE

# Pre Maintenance Report


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 Obiettivi del progetto

L'obiettivo del progetto è trasformare CADOCS II da chatbot a Recommendation System (RS) andando a (1) preservare tutte le precedenti funzionalità e (2) aggiungere nuovi requisiti. Nello specifico, per ciò che concerne la trasformazione in RS si seguiranno le linee guida del libro Recommendation Systems in Software Engineering [1]. Tale RS non farà uso di tecniche di Data Mining per ottenere informazioni sulla comunità di sviluppatori, ma si affiderà a valori che saranno inseriti dall'utente.

Inoltre, si prevede di ampliare le funzionalità di CADOCS II aggiungendo un nuovo tool che calcoli la dispersione culturale e geografica all'interno di una comunità di sviluppatori e offra raccomandazioni sugli "smell" (indicatori di problemi) basati su tali dati. L'estensione delle funzionalità sarà informata (anche essa) dalla letteratura; i nuovi requisiti di CADOCS si baseranno sul lavoro di Lambiase et al. [2].

## 1.3 Struttura del documento

Tale documento di Pre-Maintenance è così strutturato:

- Capitolo 1: Introduzione;
- Capitolo 2: Reverse Engineering;
- Capitolo 3: Change Requests;

CAPITOLO 2

# Reverse Engineering di CADOCS II

## 2.1 Analisi

Prima di effettuare modifiche e passare alla stesura delle change request si è ritenuto necessario analizzare gli artefatti prodotti dagli sviluppatori precedenti, al fine di ottenere una visione generale del prodotto software.

La prima azione intrapresa è stata quella di analizzare attentamente la documentazione pregressa dello strumento in modo tale da ottenere una panoramica generale, comprendendo le sue funzionalità e il suo funzionamento. Una prima overview che abbiamo ricavato dell'architettura è quella che si può vedere nella figura 2.1.

La seconda azione intrapresa è stata l'analisi del codice sorgente di CADOCS, modulo per modulo. Poiché non era disponibile un diagramma delle classi nella documentazione degli sviluppatori precedenti che rappresentasse l'ultima versione di CADOCS, si è ritenuto essenziale crearne uno che delineasse in modo preciso le relazioni all'interno del sistema (figura 2.2).

Si possono citare le principali classi:

- ***toolGui.Cadocs***: è responsabile di rappresentare l'interfaccia utente ed è l'entry point del tool Cadocs 2.

- ***loginGui.LoginCadocs***: tale classe consente ad un utente di loggarsi con il proprio account Github.
- ***src.chatbot.intent_manager.IntentManager***: tale classe ha lo scopo di identificare l'intent che viene richiesto dall'utente tramite un prompt testuale. Si avvale al suo interno di un LanguageModel.
- ***src.intent_handling.intent_resolver.IntentResolver***: una volta identificato l'intent da parte dell'Intent Manager, tale classe chiama il tool appropriato a risorvere il corrispettivo intent.
- ***src.intent_handling.tool_selector.ToolSelector***: funge da Context per il Design Pattern Strategy, le cui Concrete Strategy sono i tool che eseguono task ben specifici, come quello di identificare Community Smell in una repository Github.
- ***src.intent_handling.tools.CsDetectorTool***: è una delle strategie concrete, questo tool viene chiamato tramite un web service, e consente di trovare un elenco di Community Smells.

A valle di quanto detto fin'ora e di come si può vedere in figura 2.1, si può dire che concettualmente il tool Cadocs II si presenta come un unico grande Adapter per altri tools, ciascuno che esegue un task specifico, che viene identificato a partire da un modulo di Natural Language Understanding.

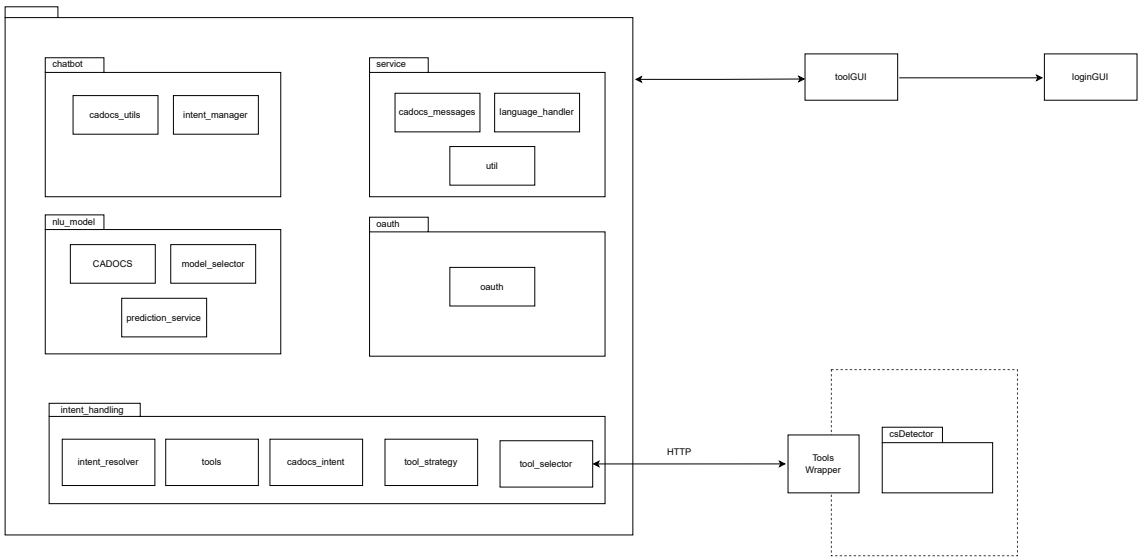


**Figura 2.1:** Overview dell'architettura di CADOCS II

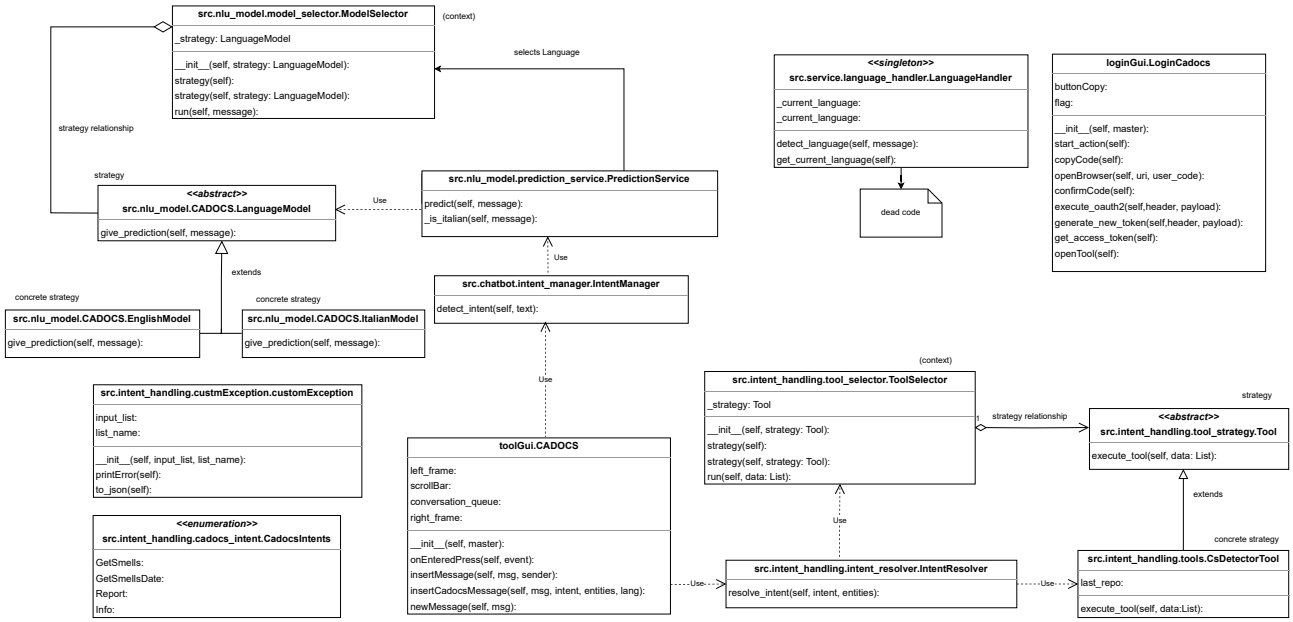


**Figura 2.2:** Class Diagram CADOCS II

### 2.1.1 Estrazione delle dipendenze fra moduli

In questa sezione vengono mostrate le dipendenze fra i vari moduli che compongono Cadocs II. Tali dipendenze sono state trovate analizzando il codice staticamente.

Emerge così la matrice 2.1, in cui il numero in corrispondenza di una cella (i, j) indica il numero di chiamate che il modulo j ha nel modulo i. Come si può vedere, il livello di accoppiamento risulta essere piuttosto basso, tranne per il modulo *toolGui* che effettua ben 6 chiamate al modulo *intent_manager*.

Emerge, inoltre, anche che i moduli *oauth*, *utils*, *language_handler* non vengono mai utilizzati, aprendo la strada a una prima fase di refactoring in cui si va ad eliminare il codice morto. La classe LanguageHandler nel package service è stata sostituita dal COTS *langdetect*. Il package OAuth (figura 2.1) non viene mai impiegato nell'applicazione poiché le funzionalità correlate sono già implementate all'interno della classe LoginCadocs.

### 2.1.2 Estrazione dei requisiti funzionali

Analizzando il codice sorgente, e leggendo la documentazione del tool Cadocs II, emergono i seguenti requisiti funzionali 2.2

| Modulo | Id | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **toolGui** | 0 | 0 | **1** | 0 | **2** | **6** | 0 | **2** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **4** |
| **loginGui** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **cadocs_utils** | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **intent_manager** | 3 | 0 | 0 | **1** | 0 | **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **3** | 0 | 0 | 0 | 0 |
| **cadocs_intent** | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **custmException** | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **intent_resolver** | 6 | 0 | 0 | 0 | 0 | **5** | 0 | 0 | **2** | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **tool_selector** | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **tool_strategy** | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **tools** | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **CADOCS** | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **model_selector** | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **4** | 0 | 0 | 0 | 0 | 0 | 0 |
| **prediction_service** | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **2** | **3** | 0 | 0 | 0 | 0 | 0 |
| **utils** | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **oauth** | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **language_handler** | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **cadocs_messages** | 16 | 0 | 0 | 0 | 0 | **4** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Tabella 2.1:** Matrice delle dipendenze

| 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 |
| RF4: Report | Tale requisito fa riferimento alla capacità del sistema di fornire un report degli smells in una repository precedentemente analizzata. |

**Tabella 2.2:** Tabella dei requisiti funzionali del sistema Cadocs II

### 2.1.3 Use Cases

#### RF1: Get Smells

| | | |
|---|---|---|
| **Identificativo** | | UC_GS |
| **Nome** | | GetSmells |
| **Descrizione** | | Lo use case mostra la funzionalità get smells che permette di analizzare gli smells presenti in una repository GitHub. |
| **Attore Principale** | | **Utente** |
| **Attore Secondari** | | **CADOCS NLU, csDetector** |
| **Entry Condition** | | L'utente vuole trovare gli smells di una repository. |
| **Exit Condition (on success)** | | L'utente visualizza i community smells che aveva richiesto. |
| **Exit Condition (on failure)** | | L'utente non visualizza i community smells |
| **Rilevanza/User Priority** | | Alta |
| **Frequenza stimata** | | 20/settimana |
| **FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO** | | |
| **1** | **Utente** | Richiede di poter visualizzare gli smells fornendo un messaggio contenente il link alla repository che si vuole analizzare. |
| **2** | **Sistema** | Il sistema rileva il linguaggio della richiesta dell'utente |
| **3** | **Sistema** | Il sistema inoltra la richiesta al modello di NLU corretto. |
| **4** | **Sistema** | Il sistema riceve la risposta dal modello di NLU sull'intent della richiesta. |
| **5** | **Sistema** | Il sistema estrae ed inoltra il link della repository al tool che analizza i community smells (csDetector). |
| **6** | **Sistema** | Il sistema mostra tramite un messaggio gli smells presenti all'interno della repository analizzata. |
| **Flusso di eventi alternativo: il modello non riesce ad interpretare la richiesta.** | | |
| **5.a1** | **Sistema** | Il sistema non mostra nulla. |

**Figura 2.3:** Use Case Get Smells

### RF2: Get Smells By Date

| | | |
|---|---|---|
| **Identificativo** | | UC_GSD |
| **Nome** | | GetSmellsDate |
| **Descrizione** | | Lo use case mostra la funzionalità get smells date che permette di analizzare gli smells presenti in una repository GitHub a partire da una certa data in poi. |
| **Attore Principale** | | **Utente** |
| **Attore Secondario** | | **MODULO NLU, csDetector** |
| **Entry Condition** | | L'utente vuole trovare gli smells di una repository da una certa data in poi.. |
| **Exit Condition (on success)** | | L'utente visualizza i community smells che aveva richiesto. |
| **Exit Condition (on failure)** | | L'utente non visualizza i community smells |
| **Rilevanza/User Priority** | | Alta |
| **Frequenza stimata** | | 20/settimana |
| **FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO** | | |
| **1** | **Utente** | Richiede di poter visualizzare gli smells fornendo un messaggio contenente il link alla repository che si vuole analizzare e la data da cui trovare gli smells. |
| **2** | **Sistema** | Il sistema rileva il linguaggio della richiesta dell'utente |
| **3** | **Sistema** | Il sistema inoltra la richiesta al modello di NLU capace di interpretare la lingua corretta. |
| **3** | **Sistema** | Il sistema riceve la risposta dal modello di NLU sull'intent della richiesta. |
| **4** | **Sistema** | Il sistema estrae ed inoltra il link della repository al tool che analizza i community smells (csDetector). |
| **Flusso di eventi alternativo: il modello non riesce ad interpretare la richiesta.** | | |
| **5.a1** | **Sistema** | Il sistema non mostra nulla. |

**Figura 2.4:** Use Case Get Smells Date

### RF3: Info

| | | |
|---|---|---|
| **Identificativo** | | UC_IN |
| **Nome** | | Info |
| **Descrizione** | | Lo use case mostra la funzionalità Info che consiste nell'inviare delle informazioni relative ai community smells in generale |
| **Attore Principale** | | **Utente** |
| **Attore Secondario** | | **Modello NLU** |
| **Entry Condition** | | L'utente deve aver installato il tool. |
| **Exit Condition (on success)** | | L'utente visualizza informazioni sui community smells |
| **Exit Condition (on failure)** | | L'utente non visualizza informazioni sui community smells |
| **Rilevanza/User Priority** | | Media |
| **Frequenza stimata** | | 20/settimana |
| **FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO** | | |
| **1** | **Utente** | Richiede informazioni sui community smells. |
| **2** | **Sistema** | Il sistema inoltra la richiesta al modello di NLU capace di interpretare la giusta lingua. |
| **3** | **Sistema** | Il sistema riceve la risposta dal modello di NLU sull'intent della richiesta. |
| **4** | **Sistema** | Il sistema mostra all'utente le informazioni sui community smells. |
| **Flusso di eventi alternativo: il modello non riesce ad interpretare la richiesta.** | | |
| **5.a1** | **Sistema** | Il sistema non mostra nulla. |

**Figura 2.5:** Use Case Info

**RF4: Report**

Attualmente tale requisito funzionale risulta non essere implementato dal sistema Cadocs II, pertanto è usabile solo tramite Slack e Cadocs.

### 2.1.4 Sequence Diagrams

#### Get Smells e Get Smells By Date

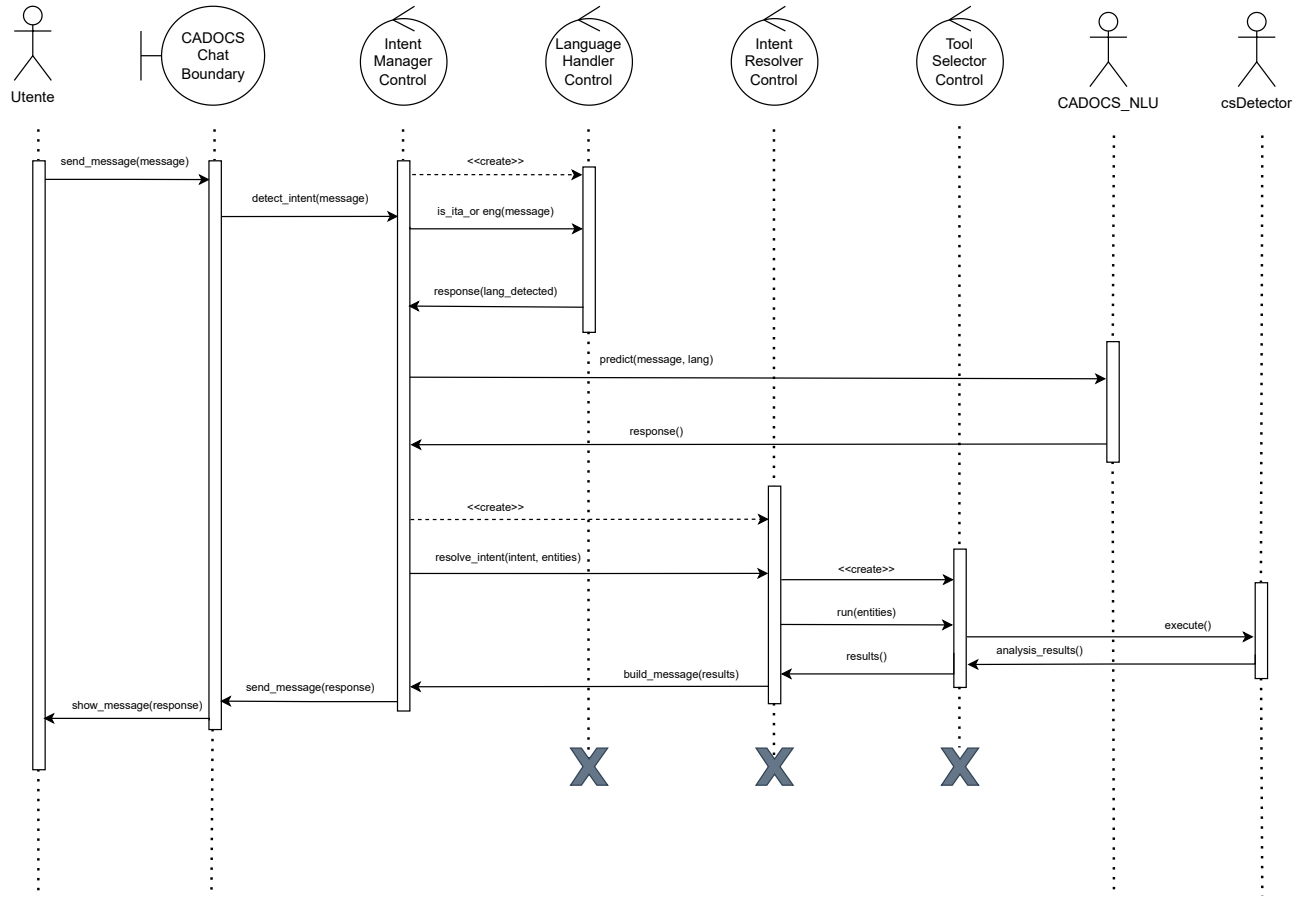


**Figura 2.6:** Sequence Diagram requisito Get Smells e Get Smells By Date

### Info

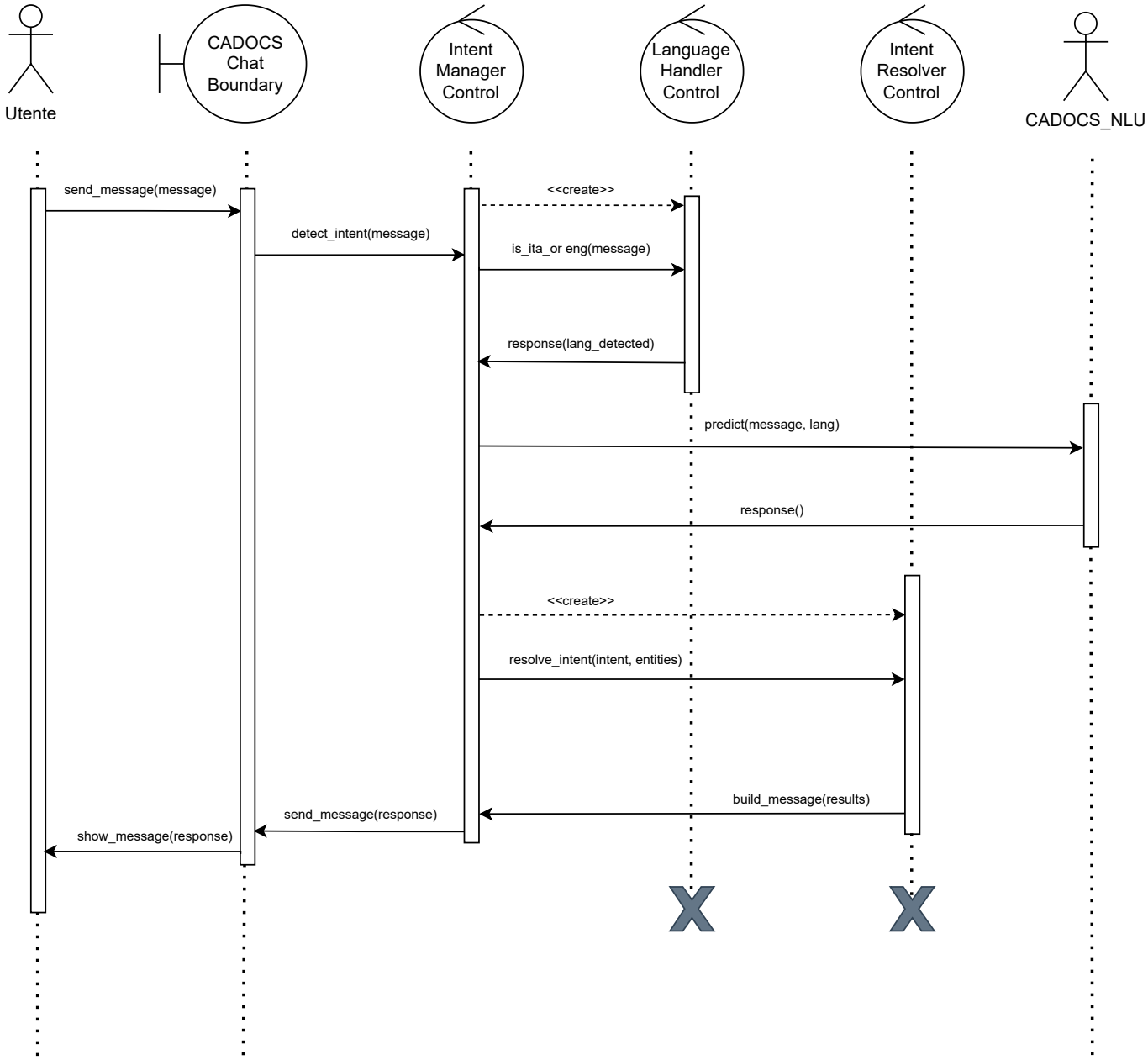


**Figura 2.7:** Sequence Diagram requisito Info

# CAPITOLO 3

# Change Requests

Il nucleo centrale del progetto verte sulla trasformazione di CADOCS II in un recommendation system informato dalla letteratura.

Nel libro *Recommendation Systems in Software Engineering* [1], un recommendation system viene definito come *un'applicazione software che fornisce informazioni ritenute utili per un compito di ingegneria del software in un dato contesto*. Per trasformare CADOCS II in un Recommendation System, sono state identificate quattro change requests.

Le change requests sono descritte secondo tre criteri principali: priorità, impatto ed effort. La priorità di una change request è correlata all'urgenza della sua implementazione. L'analisi dell'impatto rivela le conseguenze che la modifica avrà sull'intero sistema, sia a breve che a lungo termine. Infine, l'effort necessario per realizzare la modifica dipende dalla sua complessità e dalla quantità di risorse richieste. L'effort è stato stimato non solo consultando i precedenti sviluppatori, ma anche grazie a una prima analisi della matrice delle dipendenze 2.1.

## 3.1 CR_1: Separazione della logica applicativa

| **Descrizione** |
| --- |
| In questa change request, l'obiettivo è separare il componente Resolver dalle dipendenze precedenti. |
| **Motivazione** |
| Separazione del modulo di Machine Learning dall'interfaccia grafica, in modo da rendere possibile avere più frontend. |
| **Priorità** |
| **High[X]** Medium[] Low[] |
| **Effort** |
| **High[X]** Medium[] Low[] |
| **Conseguenze se non accettato** |
| Se tale change request non venisse accettata, CADOCS II, con l'avanzare del tempo diventerà sempre più accoppiato e dipendente dalla GUI attuale. |
| **Va fatto dopo** |
| NA |

## 3.2 CR_2: Aggiunta di un tool di raccomandazione

| **Descrizione** |
| --- |
| In tale change request, l'obiettivo è aggiungere un tool che offra raccomandazioni sugli "smell" basate su dati come dispersione culturale e geografica della community di sviluppatori |
| **Motivazione** |
| L'implementazione e aggiunta di questo tool semplificherebbe notevolmente il processo di monitoraggio dello stato di salute di una comunità di sviluppo. In aggiunta, aggiungerebbe agli utenti la possibilità di ottenere informazioni sulla comunità tramite una nuova dimensione, quella culturale. |
| **Priorità** |
| **High[X]** Medium[] Low[] |
| **Effort** |
| **High[X]** Medium[] Low[] |
| **Conseguenze se non accettato** |
| Se tale change request non venisse accettata, CADOCS II, con l'avanzare del tempo diventerà meno soddisfacente per i practitioners fino a diventare inutilizzato. |
| **Va fatto dopo** |
| CR1 |

## 3.3 CR_3: Sviluppo di una Recommendation System GUI

| **Descrizione** |
| --- |
| Reingegnerizzazione dell'interfaccia grafica che consenta agli utenti di interagire sia con il nuovo tool sia con le funzionalità preesistenti in CADOCS II, quali il ChatBot e il modello NLU. |
| **Motivazione** |
| La Reingegnerizzazione dell'interfaccia grafica risulta indispensabile per ottimizzare l'integrazione e consentire agli utenti di sfruttare appieno le nuove funzionalità introdotte nella CR2. Questa rinnovata interfaccia dovrà migliorare l'esperienza dell'utente, rendendo l'utilizzo del software più intuitivo e riducendo il tempo richiesto per apprenderne utilizzo. |
| **Priorità** |
| **High[X]** Medium[] Low[] |
| **Effort** |
| **High[X]** Medium[] Low[] |
| **Conseguenze se non accettato** |
| Se tale change request non venisse accettata, non ci sarebbe alcun modo per utilizzare il nuovo tool. |
| **Va fatto dopo** |
| CR2 |

## 3.4 CR_4: Dockerizzazione del Backend

| **Descrizione** |
| --- |
| La Dockerizzazione del Backend è un processo attraverso il quale si implementa l'uso di Docker per confezionare e distribuire il backend dell'applicazione in un ambiente containerizzato. Questo implica la creazione di un'immagine Docker contenente il backend dell'applicazione e tutte le sue dipendenze, configurazioni e risorse necessarie per l'esecuzione. Una volta containerizzato, il backend diventa portabile e può essere eseguito in modo consistente su diversi ambienti. |
| **Motivazione** |
| È necessario dockerizzare il backend dell'applicazione per garantire una distribuzione più efficiente e scalabile. Dockerizzare il backend consentirà di isolare l'applicazione e le sue dipendenze in container leggeri e portabili, facilitando la gestione delle risorse e la distribuzione su diversi ambienti. |
| **Priorità** |
| **High[X]** Medium[] Low[] |
| **Effort** |
| **High[X]** Medium[] Low[] |
| **Conseguenze se non accettato** |
| Se tale change request non venisse accettata, ci sarebbero delle difficoltà nell'installazione dell'applicazione |
| **Va fatto dopo** |
| CR2 |

# Bibliografia

[1] I. Avazpour, T. Pitakrat, L. Grunske, J. Grundy, M. Robillard, W. Maalej, R. Walker, and T. Zimmermann, "Recommendation systems in software engineering," *Dimensions and metrics for evaluating recommendation systems*, pp. 245–273, 2014. (Citato alle pagine 2 e 14)

[2] S. Lambiase, G. Catolino, D. A. Tamburri, A. Serebrenik, F. Palomba, and F. Ferrucci, "Good fences make good neighbours? on the impact of cultural and geographical dispersion on community smells," in *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society*, ser. ICSE-SEIS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 67–78. [Online]. Available: https://doi.org/10.1145/3510458.3513015 (Citato a pagina 2)