1. Introduzione

CodeSmile è uno strumento di analisi statica progettato per rilevare code smells specifici per il Machine Learning (ML-CSs) e riportarli tramite file .csv con informazioni come file name, function name, line e nome dello smell individuato.
Questo documento svolge la funzione di report generale del lavoro fatto per introdurre 2 delle 3 change requests proposte.
Il documento si divide in reverse engineering e forward engineering. Nella prima fase, a partire dal codice, si è effettuato a più passi un'astrazione per derivare i sottosistemi, requisiti funzionali e use cases dell'applicativo prima delle modifiche. Nel forward engineering sono presentate le change requests e come esse abbiano impattato le diverse astrazioni del software, modificando e aggiungendo artefatti. L'impact analysis è stata svolta in due momenti diversi: prima di implementare le change requests e dopo.
Per quanto riguarda il testing, si faccia riferimento agli altri documenti: Test Plan, Test Incident Report e Test Summary Report

2. Reverse Engineering

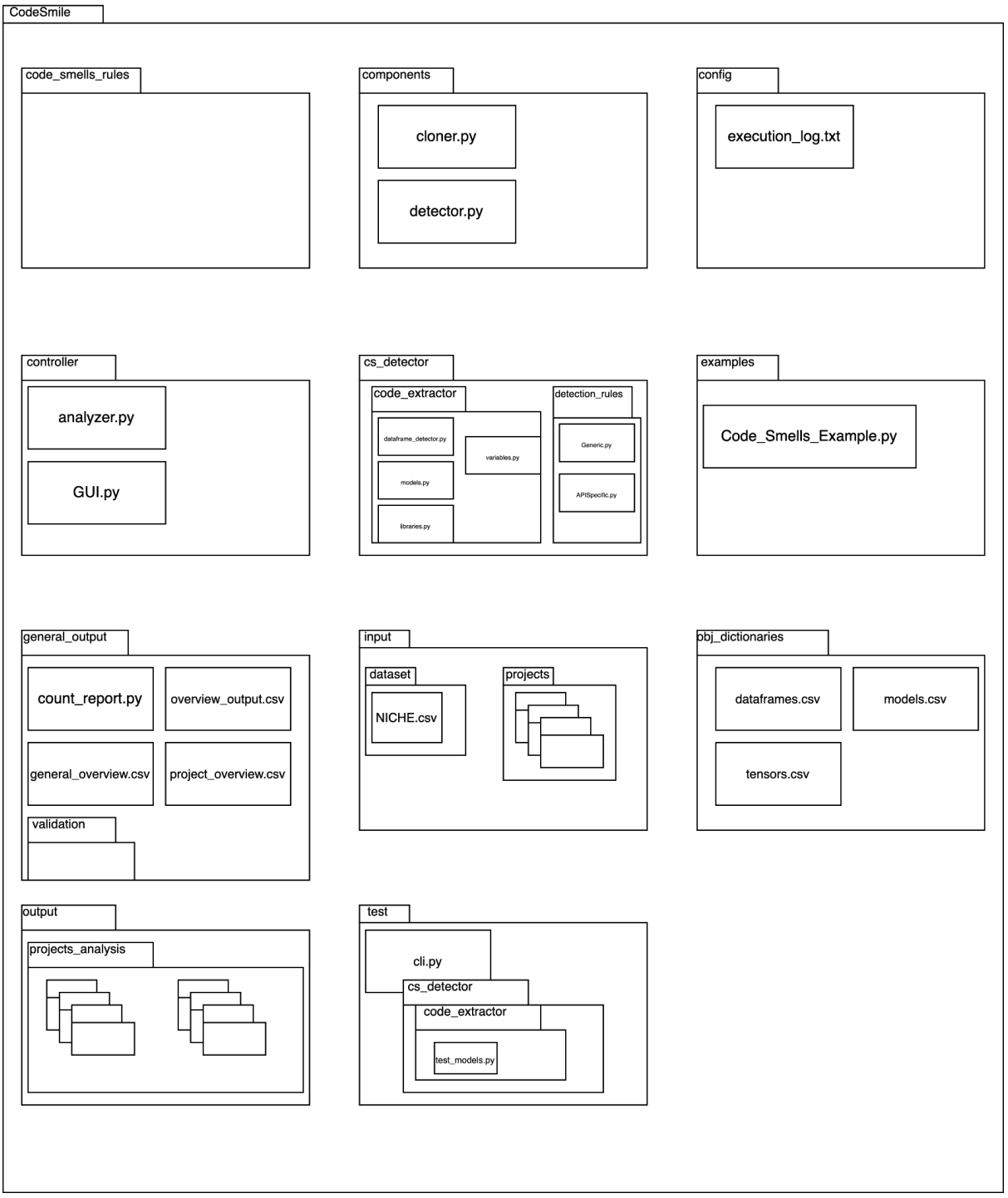
2.1 Code design

2.1.1 Packages

Il codice sorgente è suddiviso nei seguenti packages:

- ***code_smell_rules***, al momento vuoto
- ***components***, in cui è presente *detector.py* che si occupa di individuare i code smells all'interno di un singolo file e *cloner.py* che si occupa di copiare all'interno del progetto i progetti software presenti nel dataset *NICHE.csv*
- ***config***, in cui è presente *execution_log.txt*, usato per riprendere l'analisi dall'ultimo progetto analizzato
- ***controller***, in cui è presente il main del progetto, sottoforma di linea di comando in *analyzer.py* o di interfaccia grafica in *GUI.py*. Entrambi permettono l'analisi di un singolo o multipli progetti
- ***cs_detector***, suddiviso in ulteriori due packages
    - ***code_extractor***, in cui è presente *dataframe_detector.py* che si occupa di identificare e tracciare i dataframes di pandas all'interno di una funzione, *libraries.py* che contiene funzioni di utilità su librerie come estrarle da uno AST e ottenere la libreria utilizzata da un nodo,

models.py che contiene funzioni di utilità per verificare che il nome di un metodo sia la creazione di un modello di machine learning, *variables.py* che si occupa di identificare le definizioni e le assegnazioni di variabili tensors
    - ***detection_rules***, in cui è presente *APISpecific.py* con funzioni che individuano code smells API specific e *Generic.py* che individua code smells generici
- ***examples***, in cui è presente *Code_Smell_Example.py* contenente una funzione che esegue semplici operazioni su un dataframe pandas. Usato come file di input nel testing.
- ***general_output***, in cui è presente *count_report.py* che attraverso *overview_output.csv* genera altre due overview: *general_overview.csv* e *project_overview.csv*. Il primo mostra il numero di smells totali per tipo per tutti i progetti analizzati, il secondo mostra il numero di smells per progetto.
- ***input***, suddiviso in ulteriori due packages:
    - *dataset*, in cui è presente il dataset *NICHE.csv* contenente progetti software di GitHub
    - *projects*, in cui vengono salvati i progetti recuperati tramite il dataset
- ***obj_dictionaries***, in cui sono presenti diversi file .csv con all'interno salvati nomi di librerie e metodi
- ***output***, in cui è presente *projects_analysis* dove ogni cartella contiene l'analisi di un differente progetto analizzato durante l'esecuzione del programma. Ogni cartella contiene:
    - *overview_output.csv,* una panoramica generale di tutti i smells individuati nei progetti analizzati in sequenza/parallelo
    - un file per ogni tipo di smell individuato, in cui vengono esplicitate le funzioni in cui il tipo di smell appare e la linea di codice
    - *to_save.csv*, una panoramica generale di tutti i smells individuati in un singolo progetto
- ***test***, in cui è presente *cli.py* che utilizza comuni funzioni pandas e calcola i numeri di fibonacci tramite array tensors, leggendo inoltre da un file *test.csv* che però non è presente nella repository.
    - ***cs_detector***
        - ***code_extractor****,* in cui è presente *test_models.py che* testa la corretta identificazione dei modelli e librerie da parte di *models.py* e *libraries.py*



2.1.2 Componenti off-the-shelf

- ***pandas***, libreria usata per il reporting dei code smells sotto forma di diversi files .csv
- ***ast,*** libreria usata per analizzare il codice dei progetti, ottenendo per ogni file sorgente il suo AST (abstract syntax tree) in cui ogni nodo viene visitato per individuare i code smells

- ***NICHE***, dataset di progetti software da cui vengono estratti solo quelli machine learning-related e copiati all'interno della repository del progetto per essere analizzati e individuare i loro code smells.

2.1.3 Design patterns

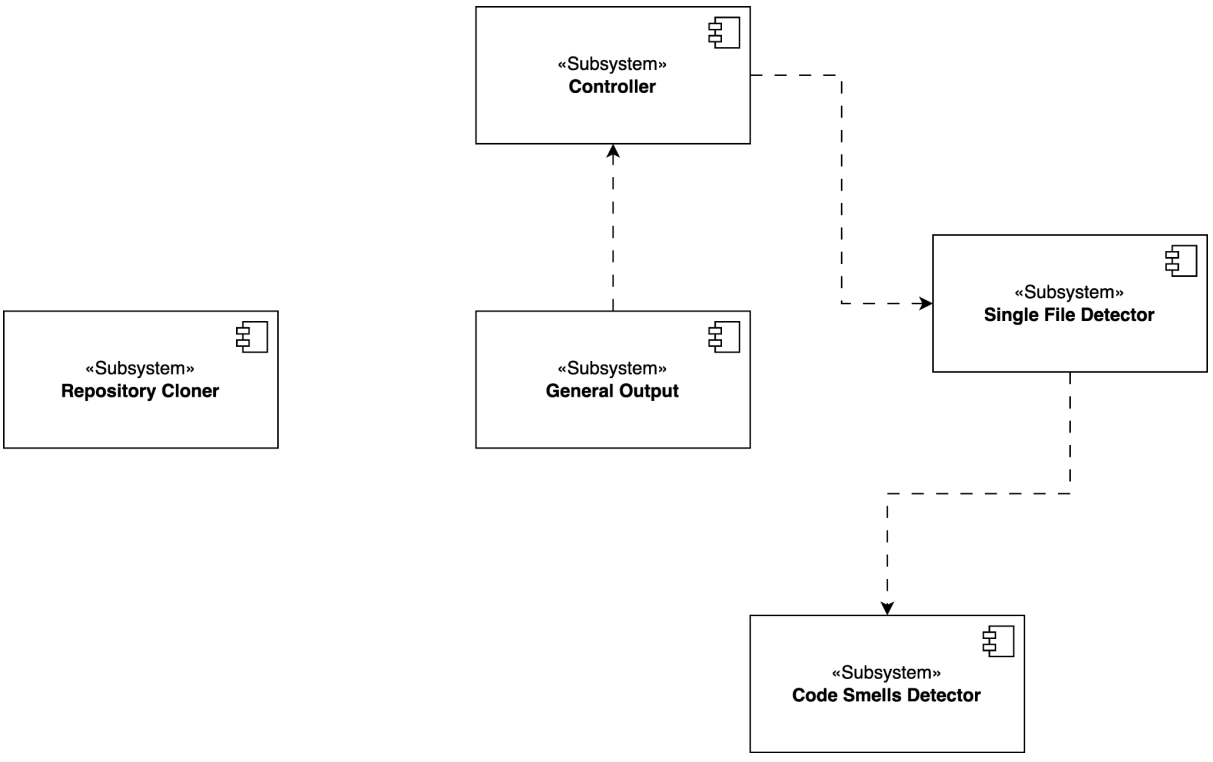
Non sono state individuate applicazioni particolari di design patterns.

2.2 System design

2.2.1 Suddivisione in sottosistemi

I sottosistemi individuati sono i seguenti:

- **Repository cloner**: ottiene e filtra le repositories contenute nel dataset *NICHE*.
- **Single file detector**: analizza un singolo file di un progetto e individua i code smells seguendo le rules in Code Smells Detector
- **Code Smells Detector**: individua code smells all'interno di un nodo dell'AST
- **Controller**: coordina la logica principale del sistema, permettendo l'analisi di un singolo progetto o più progetti, con CLI o GUI
- **General Output:** permette di creare file di report .csv in cui è esplicitata la frequenza di code smells per tipo e la frequenza di code smells per progetto



2.2.2 Servizi dei sottosistemi

Repository Cloner

Servizio	Descrizione
get_repo	Attraverso l'url di una repository, la clona sulla macchina su cui viene invocato il servizio
get_projects	Clona le repositories di progetti ML dal dataset NICHE

Single File Detector

Servizio	Descrizione
rule_check	Controlla se un singolo nodo dell'AST abbia un code smell, effettuando un check per ogni regola presente
save_single_file	Salva ogni code smell individuato in un file .csv, tenendo traccia della funzione in cui è stato trovato e la linea di codice
inspect	Ispeziona un intero file .py per trovare code smells al suo interno

Code Smells detector

Servizio	Descrizione
rules	Non un unico servizio, ma bensì un insieme di servizi, ognuno identifica un tipo di code smell specifico
extract_libraries	Dato un AST di un file .py, estrae una lista di librerie utilizzate

Controller

Servizio	Descrizione
analyze_project	Analizza un singolo progetto per individuare code smells, generando un

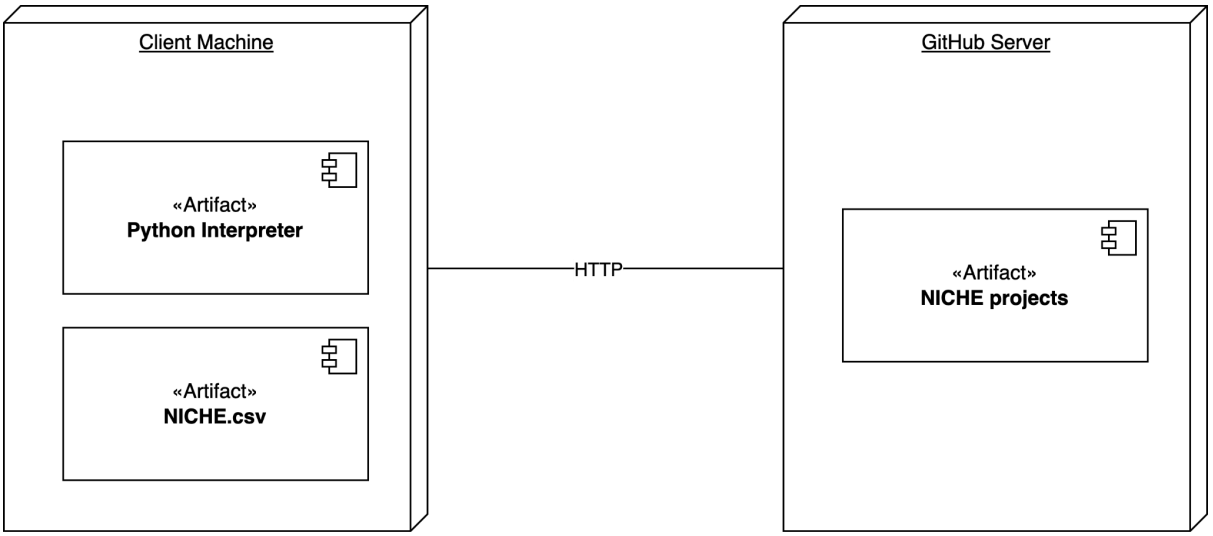
	file .csv per ogni tipo di code smell individuato e uno di overview
projects_analysis	Come analyze_project ma analizza più progetti in sequenza
parallel_projects_analysis	Come projects_analysis ma analizza i progetti in parallelo
merge_results	Fonde i risultati dell'analisi di più progetti in un unico file .csv

General output

Servizio	Descrizione
smell_report	Crea un report generale dell'analisi in cui è esplicitata la frequenza per ogni tipo di code smell
project_report	Crea un report generale dell'analisi in cui è esplicitata la frequenza di code smells per ogni progetto analizzato

2.2.3 Hardware software-mapping

Il sistema viene eseguito interamente in locale, ad eccezione delle richieste di cloning delle repositories contenute nel dataset NICHE



2.3 Requirements and analysis

2.3.1 Requisiti funzionali

Generic		
ID	**Nome**	**Descrizione**
RF_1	**Analisi progetto**	Il sistema permette di analizzare un progetto software per individuare ML code smells
RF_2	**Analisi progetti sequenziale**	Il sistema permette di analizzare più progetti in maniera sequenziale per individuare ML code smells
RF_3	**Analisi progetti parallela**	Il sistema permette di analizzare più progetti in maniera parallela per individuare ML code smells
RF_4	**Reporting generale**	Il sistema permette la creazione di un report generale dei data smells individuati, con informazioni su: file name, function name, numero di smells nella funzione, smell name, fix message. Il file di report è in formato .csv
RF_5	**Reporting specifico per tipo di code smell**	Il sistema permette la creazione di reports specifici per ogni tipo di code smell individuato, con informazioni su: file name, function name, smell name, line. I files di report sono in formato .csv

RF_6	**Reporting aggregato multi-progetto**	Il sistema permette la creazione di un report di tutti i progetti analizzati, con informazioni su: file name, function name, numero di smells nella funzione, smell name, fix message. Il file è in formato .csv
RF_7	**Reporting numero di code smells totali**	Il sistema permette la creazione di un report di tutti i progetti analizzati, con informazioni su: smell name e numero di volte che compare in tutta l'analisi. Il file è in formato .csv
RF_8	**Reporting numero di code smells per progetto**	Il sistema permette la creazione di un report di tutti i progetti analizzati, con informazioni su: nome progetto e numero di code smells individuati. Il file è in formato .csv
	Command line options	

Code Smells		
ID	**Nome**	**Descrizione**
RFCS_1	**Deterministic Algorithm Option Not Used**	Il sistema permette di individuare se viene usato un algoritmo deterministico
RFCS_2	**Merge API Parameter Not Explicitly Set**	Il sistema permette di individuare se viene invocata la funzione “merge” di pandas senza parametri
RFCS_3	**Columns and DataType Not Explicitly Set**	Il sistema permette di individuare se non vengono specificate colonne e datatypes in fase di import
RFCS_4	**Empty Column Misinitialization**	Il sistema permette di individuare se una colonna è stata inizializzata con valori che non siano null, ad esempio “0” o stringhe vuote
RFCS_5	**NaN Equivalence Comparison Misused**	Il sistema permette di individuare se ci sono comparazioni di equivalenza tra valori NaN
RFCS_6	**In-Place APIs Misused**	Il sistema permette di individuare se un’operazione non è stata salvata in una variabile
RFCS_7	**Memory Not Freed**	Il sistema permette di individuare se non vengono usati metodi per liberare la memoria durante la fase di training di un modello

RFCS_8	**Chain Indexing**	Il sistema permette di individuare se, in ambito pandas, viene usato il chain indexing
RFCS_9	**Dataframe Conversion API Misused**	Il sistema permette di individuare se, in ambito pandas, viene usato il metodo ".values" su un dataframe
RFCS_10	**Matrix Multiplication API Misused**	Il sistema permette di individuare se, in ambito numpy, viene usato il metodo ".dot"
RFCS_11	**Gradients Not Cleared before Backward Propagation**	Il sistema permette di individuare se, in ambito PyTorch, non è stato usato il metodo ".zero_grad" prima di ".backward"
RFCS_12	**TensorArray Not Used**	Il sistema permette di individuare se, in ambito TensorFlow, viene usato ".constant" per inizializzare un array che viene modificato in un loop
RFCS_13	**Pytorch Call Method Misused**	Il sistema permette di individuare se, in ambito PyTorch, viene usato il metodo ".forward"
RFCS_14	**Unnecessary Iteration**	Il sistema permette di individuare se vengono effettuate iterazioni su dataframes
RFCS_15	**Broadcasting Feature Not Used**	Il sistema permette di individuare se non viene usata la feature broadcasting

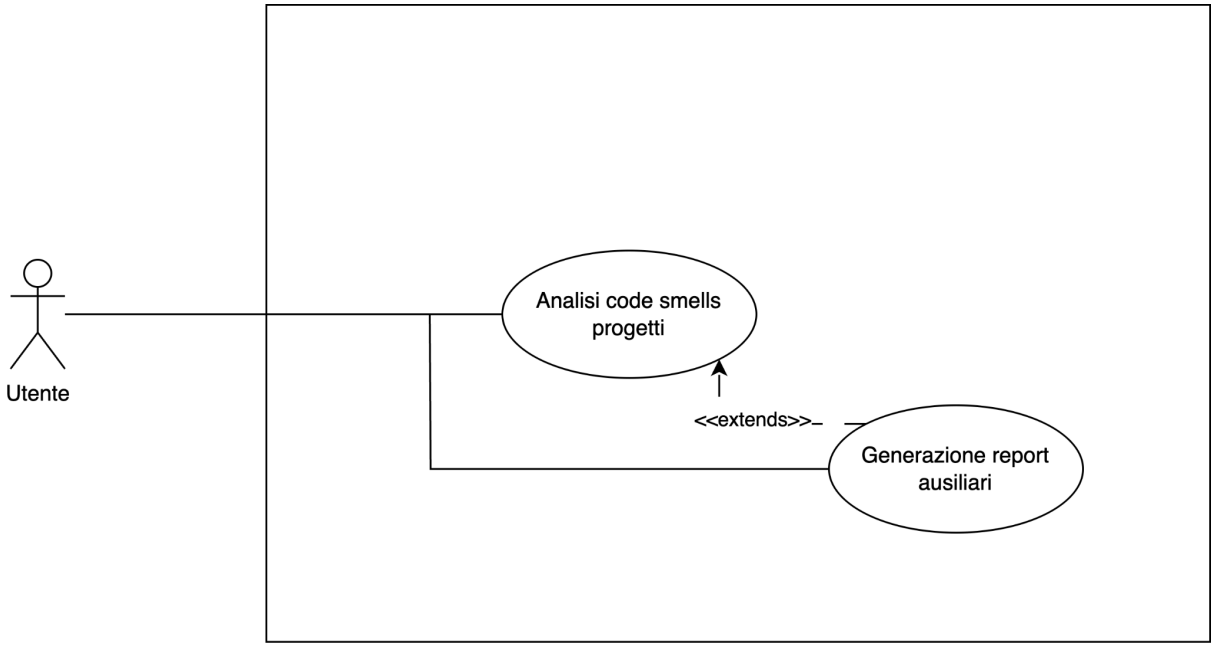
2.3.2 Requisiti non funzionali

Gli unici due requisiti non funzionali estraibili direttamente dal codice riguardano performance e affidabilità

RNF_1. Il sistema, al fine di velocizzare la procedura, deve permettere l'analisi in parallelo di più progetti - la funzione parallel_projects_analysis in analyzer.py

RNF_2. Il sistema, in caso di fail improvviso, deve permettere la ripresa dell'analisi dall'ultimo progetto analizzato. (L'analisi di più progetti non deve ripartire da capo) - la funziona projects_analysis in analyzer.py utilizza un execution log

2.3.3 Use cases



Identificativo: ***UC_1***	Analisi code smells progetti (RF_1…6, RFCS_1…15)	**Data**	27/08/2024
		Vers.	1.0
		Autore	Carmine Iemmino
Descrizione	Lo UC fornisce la funzionalità di analizzare il codice di uno o più progetti software e riportare i code smells individuati		
Attore principale	**Utente**		
Entry condition	L'utente ha fornito una directory in cui è presente uno o più progetti python		
Exit condition (on success)	Il progetto o più progetti sono stati analizzati e sono stati generati reports sui code smells individuati		
Exit condition (on failure)	I reports non vengono generati		

FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO		
1	Utente:	Esegue lo script, specificando i parametri obbligatori: • *input*, path al progetto o progetti da analizzare • *output*, path in cui salvare i file report generati I parametri opzionali sono: • *multiple*, flag che permette l'analisi di più progetti • *parallel*, flag che permette l'analisi di più progetti in parallelo • *max_workers*, numero di thread per l'analisi parallela • *resume*, flag che permette di riprendere un'analisi interrotta
2	Sistema:	Legge i parametri di input e inizia l'analisi di un singolo progetto o di più progetti se il flag multiple è true
3	Sistema:	Per ogni file presente per ogni progetto, esclusi i file di

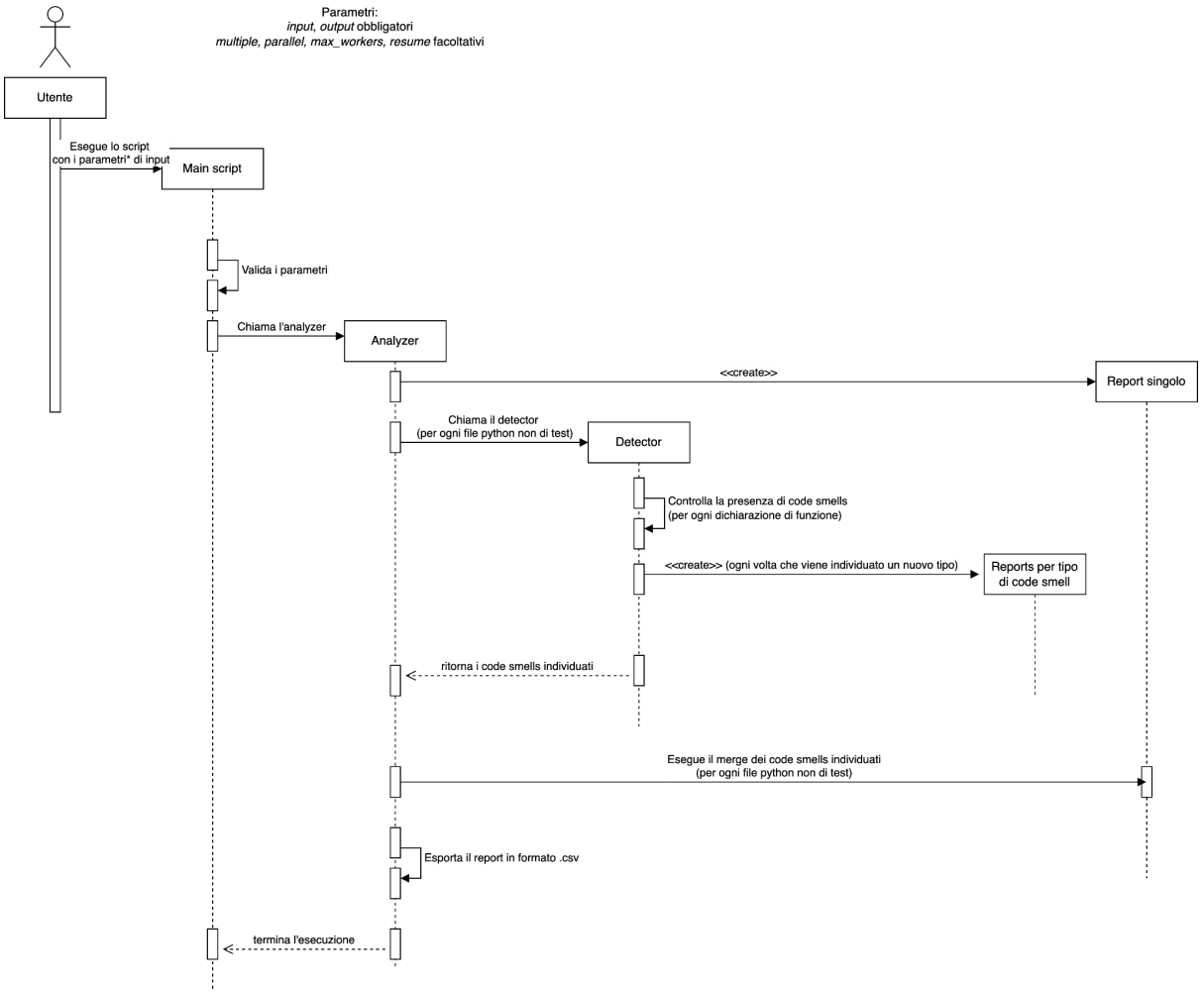
		test, esegue un'ispezione per individuare code smells. Ogni file ispezionato viene convertito in un albero. Ad ogni nodo dell'albero, se è una definizione di funzione, vengono controllate una ad una le regole definite per individuare i code smells. Se vengono individuati code smells sono riportati in un file specifico .csv per ogni tipo di code smell, con informazioni su file name, function name, smell name e linea di codice Ogni volta che finisce l'ispezione di un file, viene fatto un merge dei code smells individuati fino a quel momento in un unico file .csv, con informazioni su file name, function name, numero di smells nella funzione, smell name e fix message
4	Sistema:	Crea un ulteriore file .csv di overview dell'analisi, in cui vengono riportati tutti i code smells per tutti i progetti analizzati.
Scenario/Flusso di eventi di ERRORE		L'utente fornisce una cartella di input senza files python oppure una cartella di output non esistente
2err.1	Sistema:	Informa l'utente dell'errore.
2Err.2	Sistema:	Termina con un insuccesso.

Identificativo: ***UC_2***	Generazione reports ausiliari (RF_7, RF_8)	**Data**	27/08/2024
		Vers.	1.0
		Autore	Carmine Iemmino
Descrizione	Lo UC fornisce la funzionalità di generare reports: numero di code smells per progetto e numero di code smells per tipo		
Attore principale	**Utente**		
Entry condition	UC_1 è stato eseguito con successo e overview_output.csv è stato generato		
Exit condition (on success)	Vengono generati i due nuovi report project_overview.csv e general_overview.csv		
Exit condition (on failure)	I report non vengono generati		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO			
1	Utente:	Dopo aver eseguito UC_1, sposta nella cartella general_output il file overview_output.csv appena generato ed esegue lo script da count_report.py	
2	Sistema:	Legge overview_output.csv e tramite le informazioni contenenti in esso genera project_overview.csv con colonne “project_name” e “smell”	
3	Sistema:	Legge overview_output.csv e tramite le informazioni contenenti in esso genera general_overview.csv con colonne “name_smell” e “smell”	
Scenario/Flusso di eventi di ERRORE		L’utente non fornisce il file overview_output.csv nella cartella general_output	
2err.1	Sistema:	Informa l’utente dell’assenza del file.	
2Err.2	Sistema:	Termina con un insuccesso.	

Scenario/Flusso di eventi di ERRORE		L'utente fornisce il file overview_output.csv ma non presenta le colonne necessarie "name_smell", "smell", "filename",
2err.1	Sistema:	Informa l'utente del file non ben formato.
2Err.2	Sistema:	Termina con un insuccesso.

2.3.4 Sequence diagrams

Data la natura procedurale (e a non ad oggetti) dell'applicativo, le unità del sequence diagram sono moduli e files.



3. Forward Engineering

3.1 Change requests

ID	Nome	Descrizione	Tipologia	Priorità
CR_1	**Estensione capacità di rilevamento di code smells**	Rilevare almeno due nuovi tipi di code smells, estendendo così il numero di tipi di code smells rilevati ad almeno 17.	Perfettiva/additiva	Media
CR_2	**Implementazione refactoring automatico di code smells**	Implementazione di refactoring automatico di almeno due tipi di code smells, permettendo di automatizzare il processo di rimozione dei code smells.	Additiva	Bassa
CR_3	**Aggiunta di grafici e diagrammi al reporting**	Aggiungere la possibilità di creare grafici a barre che raffigurino il numero di code smells per tipo, grafici a linee per visualizzare l'andamento temporale dei code smells rilevati e diagrammi a torta per rappresentare la percentuale dei diversi tipi di code smells individuati.	Additiva	Alta

3.2 Requirements and analysis

3.2.1 Requisiti funzionali

In base alle change requests, vengono aggiunti i seguenti requisiti funzionali

Generic			
ID	**Nome**	**Descrizione**	**Change request**
RF_9	**Grafico a barre**	Il sistema dovrebbe permettere di visualizzare il numero di code smells per tipo attraverso un grafico a barre	CR_3
RF_10	**Grafico a torta**	Il sistema dovrebbe permettere di visualizzare la percentuale di presenza di ogni tipo di code smell attraverso un grafico a torta	CR_3
RF_11	**Grafico andamento temporale**	Il sistema dovrebbe permettere di visualizzare l'andamento temporale del numero di code smells rilevati tra diverse esecuzioni dello script	CR_3

Code Smells

ID	Nome	Descrizione	Change request
RFCS_16	**Hyperparameters not explicitly set**	Il sistema dovrà permettere di individuare se gli hyperparameters di un modello ML non stati esplicitati	CR_1
RFCS_17	**Randomness Uncontrolled**	Il sistema dovrà permettere di individuare se non è stato esplicitamente impostato il random seed nella creazione di un modello ML	CR_1

3.2.2 Use cases

In base alle change requests, il seguente use case è modificato (modifiche in **grassetto**)

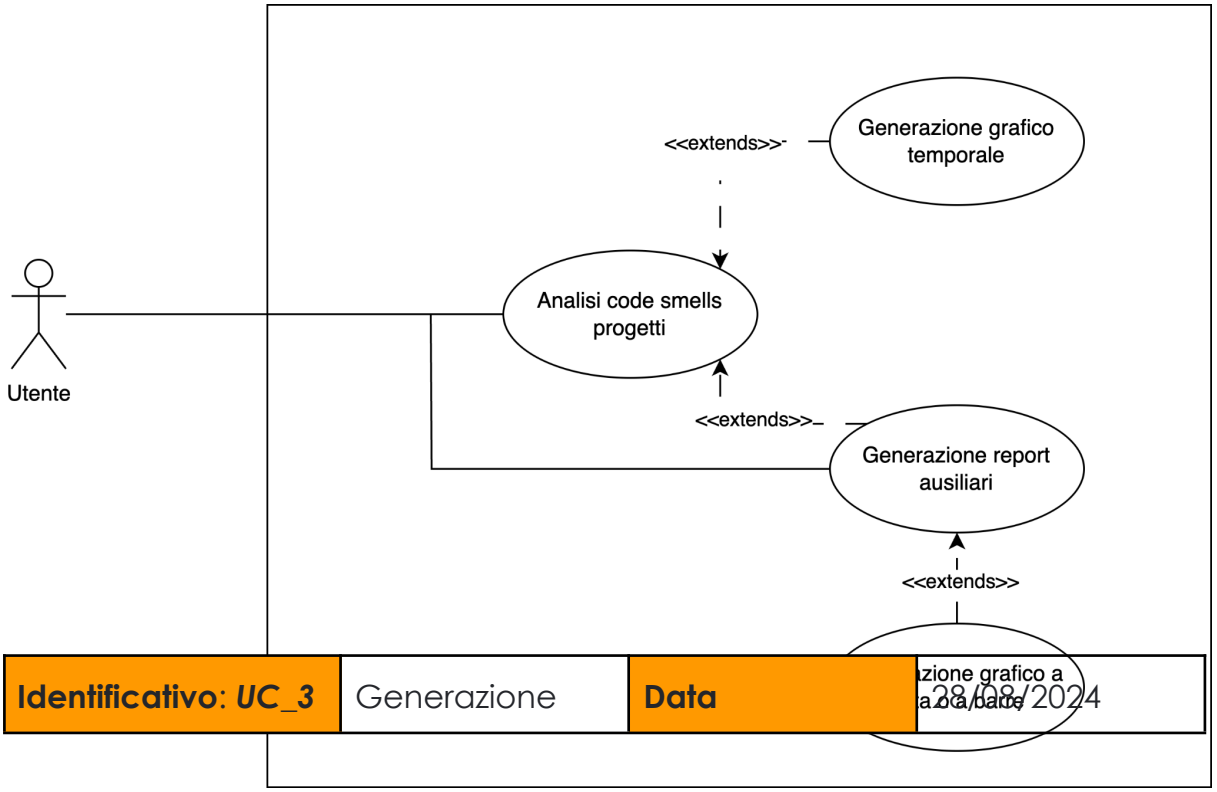
Identificativo: *UC_1*	Analisi code smells progetto (RF_1, RF_4, RF_5, RFCS_1…**16**, **17**)	**Data**	27/08/2024
		Vers.	1.0
		Autore	Carmine Iemmino
Descrizione	Lo UC fornisce la funzionalità di analizzare il codice di un progetto software e riportare i code smells individuati		
Attore principale	Utente		
Entry condition	L'utente ha fornito una directory in cui è presente un progetto python		
Exit condition (on	Il progetto è stato analizzato e sono stati generati		

success)		report sui code smells individuati
Exit condition (on failure)		I report non vengono generati
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO		
1	Utente:	Esegue lo script, specificando i parametri obbligatori: • *input*, path al progetto da analizzare • *output*, path in cui salvare i file report generati I parametri opzionali sono: • *multiple*, flag che permette l'analisi di più progetti • *parallel*, flag che permette l'analisi di più progetti in parallelo • *max_workers*, numero di thread per l'analisi parallela • *resume*, flag che permette di riprendere un'analisi interrotta
2	Sistema:	Legge i parametri di input e inizia l'analisi di un singolo progetto
3	Sistema:	Per ogni file presente nel progetto, esclusi i file di test, esegue un'ispezione per individuare code smells. Ogni file ispezionato viene convertito in un albero. Ad ogni nodo dell'albero, se è una definizione di funzione, vengono controllate una ad una le regole definite per individuare i code smells. Se vengono individuati code smells sono riportati in un file specifico .csv per ogni tipo di code smell, con informazioni su file name, function name, smell name e linea di codice Ogni volta che finisce l'ispezione di un file, viene fatto un merge dei code smells individuati fino a quel momento in un unico file .csv, con informazioni su file name, function name, numero di smells nella

		funzione, smell name e fix message
4	Sistema:	Crea un ulteriore file .csv di overview dell'analisi, in cui vengono riportati tutti i code smells per tutti i progetti analizzati.
5	**Sistema:**	**Crea o aggiorna un ulteriore file .csv in cui è riportato il numero di code smells individuati e la data di esecuzione dell'analisi**
Scenario/Flusso di eventi di ERRORE		L'utente fornisce una cartella di input senza files python oppure una cartella di output non esistente
1err.1	Sistema:	Informa l'utente dell'errore.
1Err.2	Sistema:	Termina con un insuccesso.

In base alle change requests vengono aggiunti i seguenti use cases:

- Generazione grafico a torta o a barre
- Generazione grafico temporale



Identificativo: *UC_3*	Generazione	**Data**	28/08/2024

	grafico a torta o a barre(RF_7, RF_9, RF_10)	**Vers.**	1.0
		Autore	Carmine Iemmino
Descrizione	Lo UC fornisce la funzionalità di creare un grafico a torta che mostra la percentuale di presenza per ogni tipo di code smell o un grafico a barre che mostra il numero di code smells per tipo		
Attore principale	Utente		
Entry condition	UC_2 è stato eseguito con successo e general_overview.csv è stato generato		
Exit condition (on success)	Viene creato il grafico a torta o a barre		
Exit condition (on failure)	Il grafico a torta o a barre non viene creato		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO			
1	Utente:	Esegue lo script, eventualmente specificando il parametro opzionale: • *pie*, flag che se è impostato a vero fa restituire un grafico a torta, a barre altrimenti	
2	Sistema:	Legge il file general_overview.csv contenuto nella stessa cartella dello script	
3	Sistema:	Sulla base del file .csv, crea un grafico a torta o a barre	
4	Sistema:	Visualizza il grafico a video	
Scenario/Flusso di eventi di ERRORE	Il file general_overview.csv non è presente nella cartella dello script		
1err.1	Sistema:	Informa l'utente dell'errore	

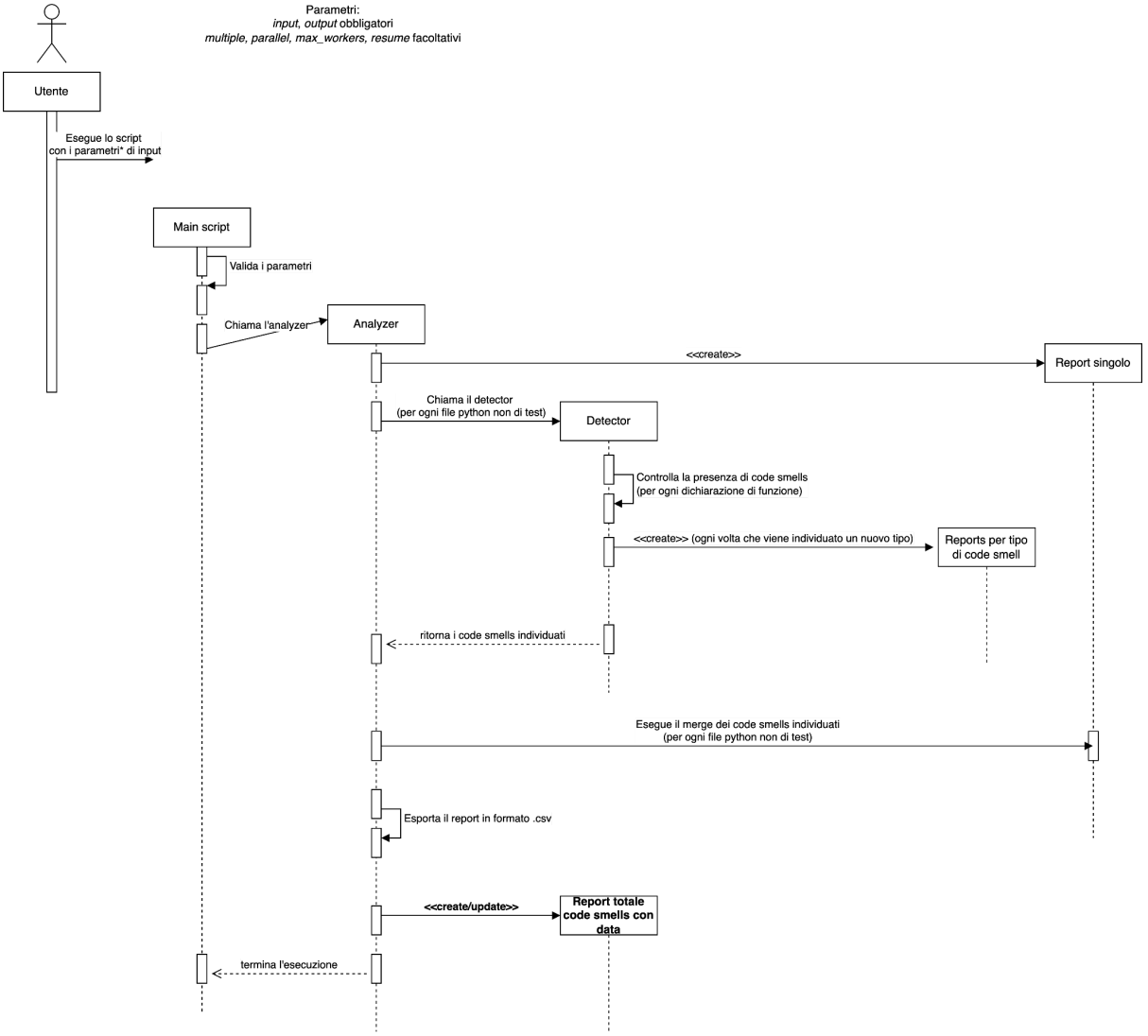
1Err.2	Sistema:	Termina con un insuccesso.
Scenario/Flusso di eventi di ERRORE		Il file general_overview.csv non contiene le colonne “code_smell” e “smell”
1err.1	Sistema:	Informa l’utente di fornire un file .csv con i suddetti campi
1Err.2	Sistema:	Termina con un insuccesso.

Identificativo: ***UC_4***	Generazione grafico temporale (RF_11)	**Data**	28/08/2024
		Vers.	1.0
		Autore	Carmine Iemmino
Descrizione	Lo UC fornisce la funzionalità di creare un grafico temporale, in cui è mostrato il numero di smells individuati tra diverse esecuzioni in diverse date		
Attore principale	Utente		
Entry condition	UC_1 è stato eseguito con successo e smell_count_dates.csv è stato generato/aggiornato		
Exit condition (on success)	Viene creato il grafico temporale		
Exit condition (on failure)	Il grafico temporale non viene creato		
FLUSSO DI EVENTI PRINCIPALE/MAIN SCENARIO			
1	Utente:	Dopo aver eseguito UC_1, sposta nella cartella general_output il file smell_count_dates.csv appena generato/aggiornato ed esegue lo script	
2	Sistema:	Legge smell_count_dates.csv	

3	Sistema:	Sulla base del file .csv, crea un grafico temporale
4	Sistema:	Visualizza il grafico a video
Scenario/Flusso di eventi di ERRORE		L'utente non fornisce il file smell_count_dates.csv nella cartella general_output
2err.1	Sistema:	Informa l'utente dell'assenza del file.
2Err.2	Sistema:	Termina con un insuccesso.
Scenario/Flusso di eventi di ERRORE		Il file smell_count_dates.csv non contiene le colonne "smells" e "date"
1err.1	Sistema:	Informa l'utente di fornire un file .csv con i suddetti campi
1Err.2	Sistema:	Termina con un insuccesso.

3.2.3 Sequence diagrams

In base alle change request, il seguente sequence diagram è modificato (modifiche in **grassetto**)



3.3 System design

3.3.1 Suddivisione in sottosistemi

In base alle change requests, viene modificato il seguenti sottosistema (modifiche in **grassetto**)

- **General Output:** permette di creare file di report .csv in cui è esplicitata la frequenza di code smells per tipo e la frequenza di code smells per progetto. **Permette inoltre la creazione di grafici a torta o a barre per la frequenza dei diversi tipi di code smells; grafici temporali per visualizzare la frequenza dei code smells tra diverse esecuzioni.**

3.3.2 Servizi dei sottosistemi

In base alle change requests, vengono modificati i seguenti servizi dei sottosistemi (modifiche in **grassetto**)

Controller

Servizio	Descrizione
analyze_project	Analizza un singolo progetto per individuare code smells, generando un file .csv per ogni tipo di code smell individuato e uno di overview. **Inoltre crea/aggiorna un ulteriore file .csv in cui è riportato il numero di code smells individuati e la data di esecuzione dell'analisi**

In base alle change requests, vengono aggiunti i seguenti servizi dei sottosistemi

General output

Servizio	Descrizione
pie_or_bar_chart	Crea un grafico a torta o a barre in cui vengono mostrati i diversi tipi di code smells e la loro frequenza. Prende in input un file .csv con tipo di code smells e frequenza
temporal_chart	Crea un grafico temporale in cui viene mostrato il numero di code smells individuati per ogni esecuzione. Prende in input un file .csv con numero di code smells e data di esecuzione

3.4 Code design

3.4.1 Packages

In base alle change requests, vengono modificati i seguenti packages
(modifiche in **grassetto**)

- ***general_output***, in cui è presente *count.py* che attraverso *overview_output.csv* genera altre due overview: *general_overview.csv* e *project_overview.csv*. Il primo mostra il numero di smells totali per tipo per tutti i progetti analizzati, il secondo mostra il numero di smells per progetto. **Tramite *general_overview.csv* può essere generato un grafico a torta o a barre eseguendo graphic_report.py. Attraverso *smell_count_dates.csv* è possibile con temporal_report.py generare un grafico temporale del numero di code smells tra diverse analisi**
- ***output***, in cui è presente *projects_analysis* dove ogni cartella contiene l'analisi di un differente progetto analizzato durante l'esecuzione del programma. Ogni cartella contiene:
    - *overview_output.csv,* una panoramica generale di tutti i smells individuati nei progetti analizzati in sequenza/parallelo
    - un file per ogni tipo di smell individuato, in cui vengono esplicitate le funzioni in cui il tipo di smell appare e la linea di codice
    - *to_save.csv*, una panoramica generale di tutti i smells individuati in un singolo progetto
    - ***smell_count_dates.csv, numero di code smells individuati tra diverse esecuzioni***

3.5 Impact analysis

CR_1

SIS	CIS	AIS	DIS
• Generic.py	• Generic.py (SIS) • rule_check (detector.py)	• **Generic.py** • **rule_check (detector.py)** • **models.csv**	• models.csv

- In *Generic.py* è stata aggiunta la funzione/rule *hyperparameters_randomness_not_explictly_set*
- In *rule_check*, funzione contenuta in *detector.py*, è stata aggiunta la chiamata alla nuova rule e la creazione del singole file *hyperparameters_not_explictly_set.csv*
- *models.csv*, che nell'impact analysis inizialmente non è stato considerato, è stato modificato aggiungendo ulteriori modelli e aggiungendo la colonna *hyperparameters*

CR_2 (non implementata)

SIS	CIS	AIS	DIS
• Generic.py	• Generic.py (SIS) • analyzer.py	/	/

CR_3

SIS	CIS	AIS	DIS
• analyze_project (analyzer.py)	• analyze_project (analyzer.py) (SIS)	• **analyze_project (analyzer.py)**	/

- In *analyze_project*, funzione di *analyzer.py*, viene chiamata una nuova funzione *temporal_results* che genera/aggiorna un file .csv denominato *smell_count_dates.csv* in cui sono riportati numero di smells e data di esecuzione dell'analisi