Integrating Amber and ju2jmh

Change Requests

Progetto ISTA

Nome e Cognome	Matricola
Benito Pigna	NF22500110
Michele Cillo	NF22500124

1. Introduzione e Obiettivi

1.1 Contesto

Ju2jmh è un framework e tool java che permette la conversione automatica di classi di test JUnit 4 in classi di benchmark JMH.

Il suo obiettivo è la facilitazione del processo di creazione di test suite per le prestazioni (Microbenchmark) partendo da test unitari esistenti.

FUNZIONAMENTO

Il funzionamento di ju2jmh è articolato in 2 passaggi:

- **IDENTIFICAZIONE**
  Il tool individua i singoli metodi JUnit e rileva le caratteristiche necessarie del test per la conversione.
- **GENERAZIONE ED ESECUZIONE**
  Viene generata una superclasse che istanzia ed accede all'istanza della classe JUnit, gestisce le eccezione ed esegue i benchmark.

LIMITAZIONI ATTUALI

Attualmente il tool presenta alcune limitazioni tecniche:

1. Mancato supporto a JUnit 5
2. Il tool converte interamente una classe JUnit in una classe JMH non permettendo la conversione dei singoli metodi
3. Una volta iniziata la conversione delle nuove classi, il tool svuota/sovrascrive le classi JMH precedentemente generate.

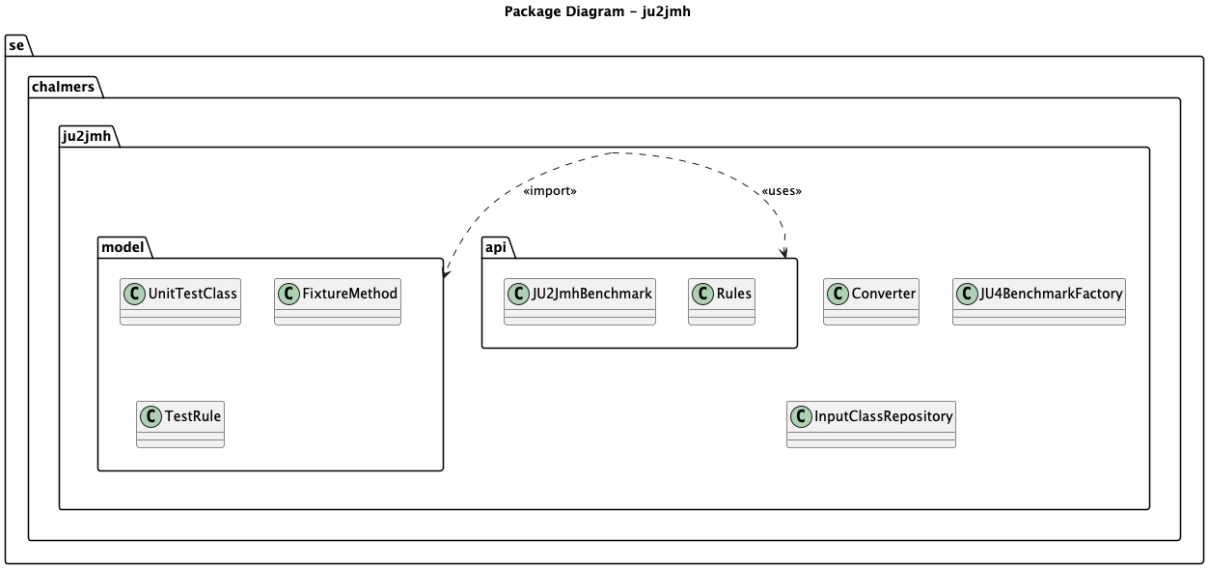
1.2 Scopo del Documento

Questo documento (Change Request) descrive una strategia per l'evoluzione e manutenzione del software al fine di superare le limitazioni precedentemente discusse.

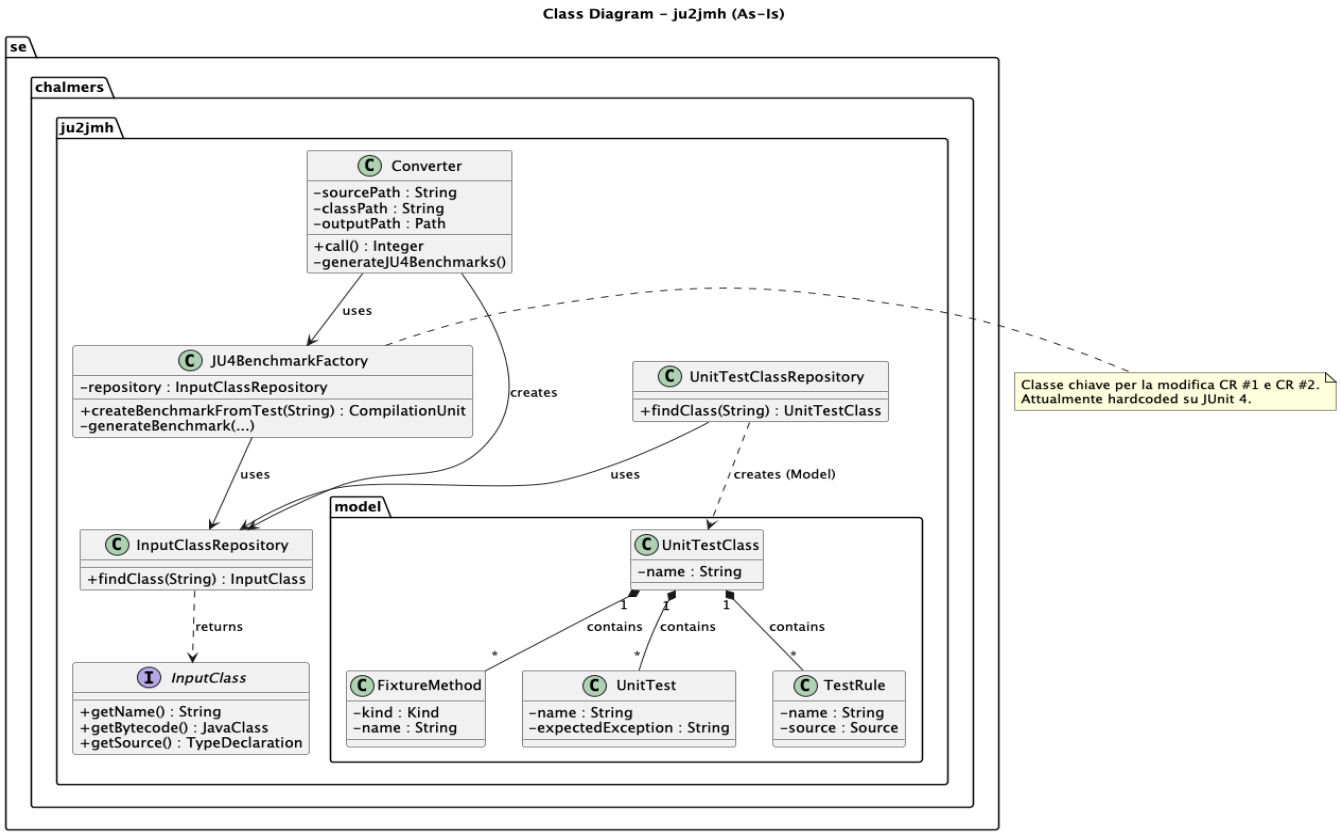
2. Analisi Architetturale ju2jmh

Attraverso un'attvità di Reverse Engineering sono stati prodotti diagrammi UML con lo scopo di comprendere la struttura interna del sistema ju2jmh e valutare l'impatto delle future modifiche.

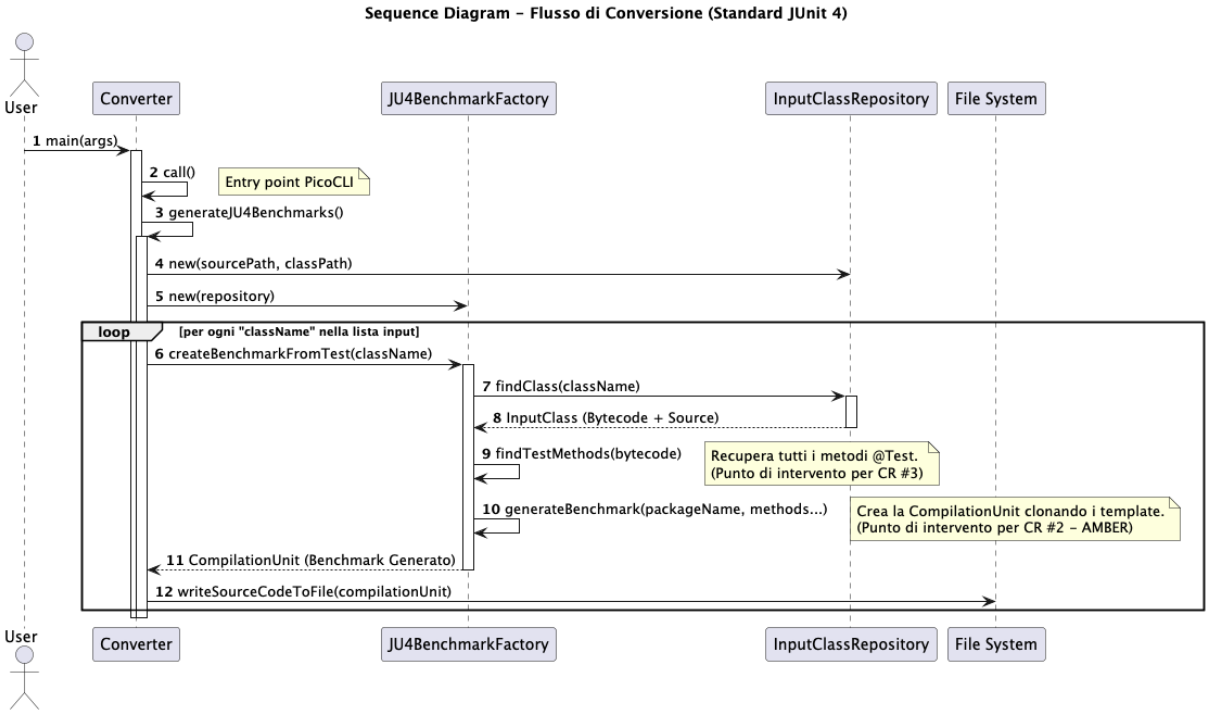
2.1 Package Diagram



2.2 Class Diagram



2.3 Sequence Diagram



3. Test Plan: Strategie e Tools impiegati

3.1 Strategia di Testing

Per il testing del sistema ju2jmh verrà impiegato un approccio basato su due strategie di testing:

White Box: Verrà mantenuta (aggiornata) la suite dei test unitari esistenti per la verifica dei singoli componenti interni

Black Box: Per la validazione del sistema, verrà impiegato quest'approccio che ignorerà la struttura interna del codice per focalizzarsi sulla correttezza dell'input/output.

E' prevista la creazione di un Golden set che verrà utilizzato come oracolo del test che avrà successo se e solo l'output generato sarà uguale sintatticamente e semanticamente all'oracolo.

3.2 Strategia di Regressione

Per il progetto ju2jmh verrà adottato l'approccio Retest-all al fine di garantire che le modifiche software non introducano regressioni.

Data la natura architetturale delle modifiche previste, la tecnica Retest-All garantisce la safety assoluta, in quanto è necessario rieseguire l'intera suite per rilevare eventuali anomalie ed imprevisti.

In particolare è previsto che per ogni modifica (commit) verrà lanciato il comando Gradle ":test" che lancerà tutti i test unitari e di sistema.

3.3 Tools Impiegati.

NOME	**DESCRIZIONE**
JUnit	Runner dei test
JaCoCo	Analisi della Code Coverage
PITest	Mutation Testing
Mockito	Isolamento delle dipendenze

4. Analisi e Valutazione della Test Suite esistente

E’ stata condotta un’analisi sulla Test Suit esistente per stabilire una Baseline per il progetto ju2jmh.

Quest’ analisi è stata realizzata mediante l’impiego di strumenti di verifica dinamica per misurare due indicatori chiave:

1. **COMPLETEZZA**
   Quanto codice viene eseguito (Code coverage).
2. **ROBUSTEZZA**
   Quanto sono efficaci i test per rilevare “difetti” (Mutation).

4.1 Analisi della Code Coverage

E’ stato configurato ed utilizzato il plugin JaCoCo per misurare la Code coverage (77%).

converter

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
se.chalmers.ju2jmh		80%		72%	82	355	187	931	32	219	2	27
se.chalmers.ju2jmh.model		68%		46%	52	108	44	192	13	66	0	7
Total	1.046 of 4.717	77%	116 of 343	66%	134	463	231	1.123	45	285	2	34

Punti di Forza: Il package se.chalmers.ju2jmh (Core) ha una copertura dell'80%.

Punti di Debolezza: Il package model ha una copertura inferiore (48%).

4.2 Mutation Testing

Per valutare la robustezza, è stato eseguito il PITest.

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
13	74%	810/1100	70%	315/447	91%	315/346

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
se.chalmers.ju2jmh	9	73%	664/910	73%	246/337	94%	246/263
se.chalmers.ju2jmh.model	4	77%	146/190	63%	69/110	83%	69/83

5. Change Requests (CR)

Sulla base dell'analisi, sono state pianificate le seguenti modifiche.

CR-01: Supporto JUnit 5

-Tipologia: aggiunta di una nuova funzionalità. (Manutenzione evolutiva)

-Descrizione: Il sistema deve essere aggiornato per supportare riconoscere e convertire le annotazioni di JUnit 5.

-Impatto: Componenti (NestedBenchmarkSuitebuilder, input class repository).

CR-02: Integrazione AMBER

-Tipologia: Manutenzione adattiva.

-Descrizione: Attualmente il parser non riesce ad elaborare i file in input che contengono i costrutti introdotti da AMBER, pertanto è necessario aggiornare le dipendenze e le configurazione del parser per supportare il livello linguistico Java 17/21.

-Impatto: Componenti (BuildGradle.kts, input class repository).

CR-03: Controllo Granularità

-Tipologia: Manutenzione perfettiva.

-Descrizione: E' necessario introdurre un meccanismo per permettere all'utente di specificare i metodi di test da convertire.

-Impatto: Componenti(Converter.java).