Università degli studi di Salerno
Dipartimento di informatica

Corso di Ingegneria Gestione ed Evoluzione del Software

Aniello Florido & Alexander Minichino

Metric 3.0

Reverse System Design Document

Ottobre 2020

Introduzione

Questo documento è stato stilato sulla base delle osservazioni che sono state fatte nella precedente fase di Object Design.
Questo documento, infatti, è parte di un processo più ampio di reverse engineering che segue un ordine inverso nello svolgersi dei processi del modello waterfall.
Pertanto, questo documento, è da considerarsi successivo al documento di Object Design.

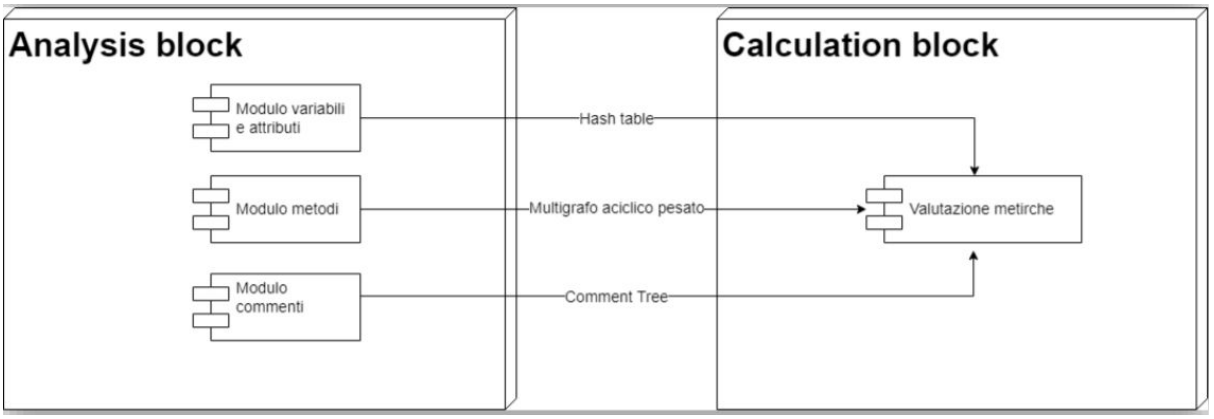
Software Architecture

Metric 3.0 è un tool che analizza un sistema e calcola una serie di metriche atte a descrivere la bontà del software.

Overview

Il tool si compone di due blocchi principali: un blocco di analisi ed un blocco di calcolo delle metriche.
Il blocco di analisi si occupa di effettuare il parsing del codice, analizzare le informazioni e popolare le strutture dati su cui il secondo blocco farà le sue operazioni di calcolo.
Il suo funzionamento si basa sulla libreria JDT Core tramite la quale è possibile costruire l'AST per visitare gli elementi del codice analizzato.
Le visite dell'AST vengono fatte in tre moduli diversi:

- uno che visita variabili ed attributi,
- uno che visita le dichiarazioni dei metodi e le invocazioni dei metodi
- uno che visita i commenti.



Inoltre,in Metric 3.0 è stato implementato successivamente un modulo sperimentale con l'obiettivo di studiare la relazione tra differenti tipi di commento presenti all'interno del codice e presenza di difetti. Tale modulo è risultato utile allo studio in questione e si presta bene per ulteriori studi ed approfondimenti futuri, tuttavia è da ritenersi supplementare e non strettamente collegato alle funzionalità di metric.

Subsystem decomposition

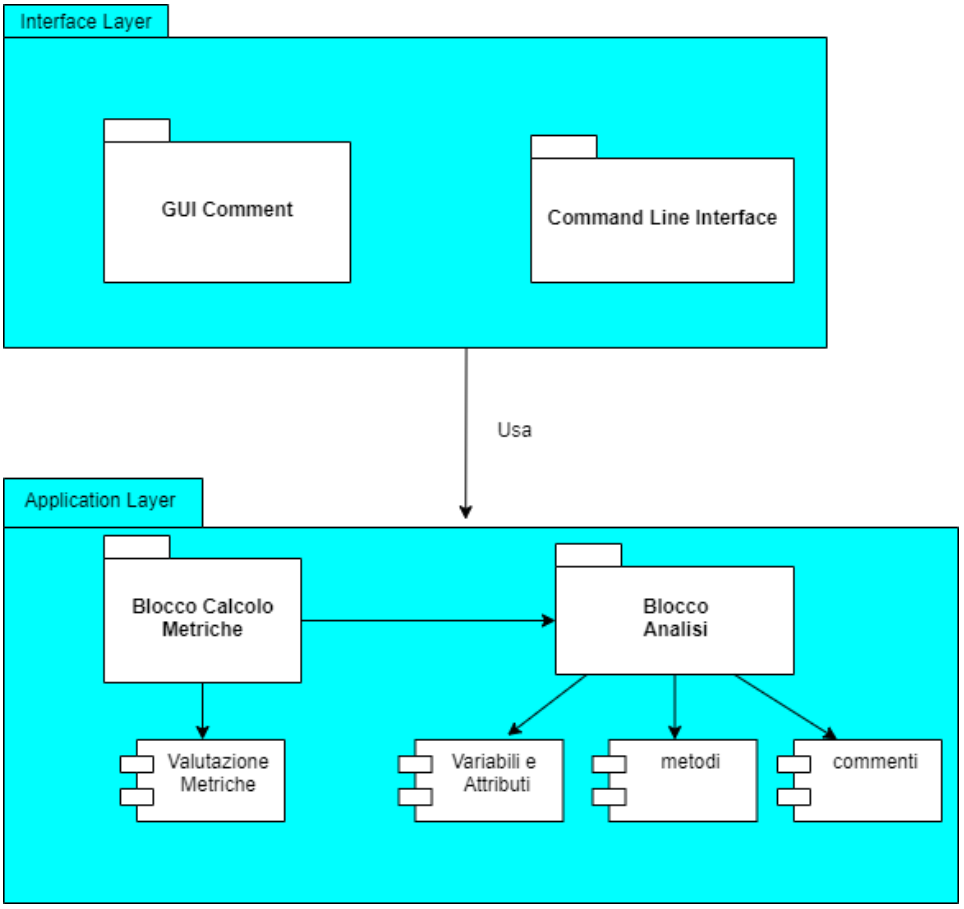
Il sistema può essere suddiviso in due livelli principali:

- Interface layer: racchiude i relativo alla rappresentazione dell'interfaccia utente e della rappresentazione dei risultati.
  A sua volta si può suddividere in:
  - GUI comment: relativo alla sola interfaccia grafica che permette la visualizzazione della struttura dati ad albero usata per rappresentare i commenti.
  - CLI: L'interfaccia testuale che permette l'avvio del tool specificando parametri e permette la visualizzazione dei risultati in forma testuale.
- Application Layer: livello relativo alle funzionalità offerte dal tool metric 3.0, come descritto in precedenza, questo livello si compone a sua volta di due moduli principali:
  - Blocco* di analisi: relativo al parsing del codice sorgente, all'analisi delle informazioni e al popolamento delle strutture dati.
  - Blocco di calcolo delle metriche: contiene la logica legata al calcolo delle metriche.

**Blocco: utilizzato come sinonimo di sottosistema*

Schema Generale

Di seguito è riportato uno schema riassuntivo dei sottosistemi principali dell'architettura del sistema.



Sottosistemi

Il blocco di analisi a sua volta si suddivide in tre moduli:

- **Analisi di variabili e attributi:** si occupa di visitare l'AST e raccogliere informazioni relative alle variabili ed ai field organizzandole in una tabella hash
- **Analisi della catena di invocazione dei metodi**: si occupa di visitare l'AST e raccogliere informazioni relative ai metodi e alle catene di invocazioni dei metodi organizzando il tutto in un multigrafo orientato aciclico pesato.
- **Analisi dei commenti:** si occupa di visitare l'AST e raccogliere informazioni relative ai commenti ed agli elementi Java a cui il commento veniva associato.

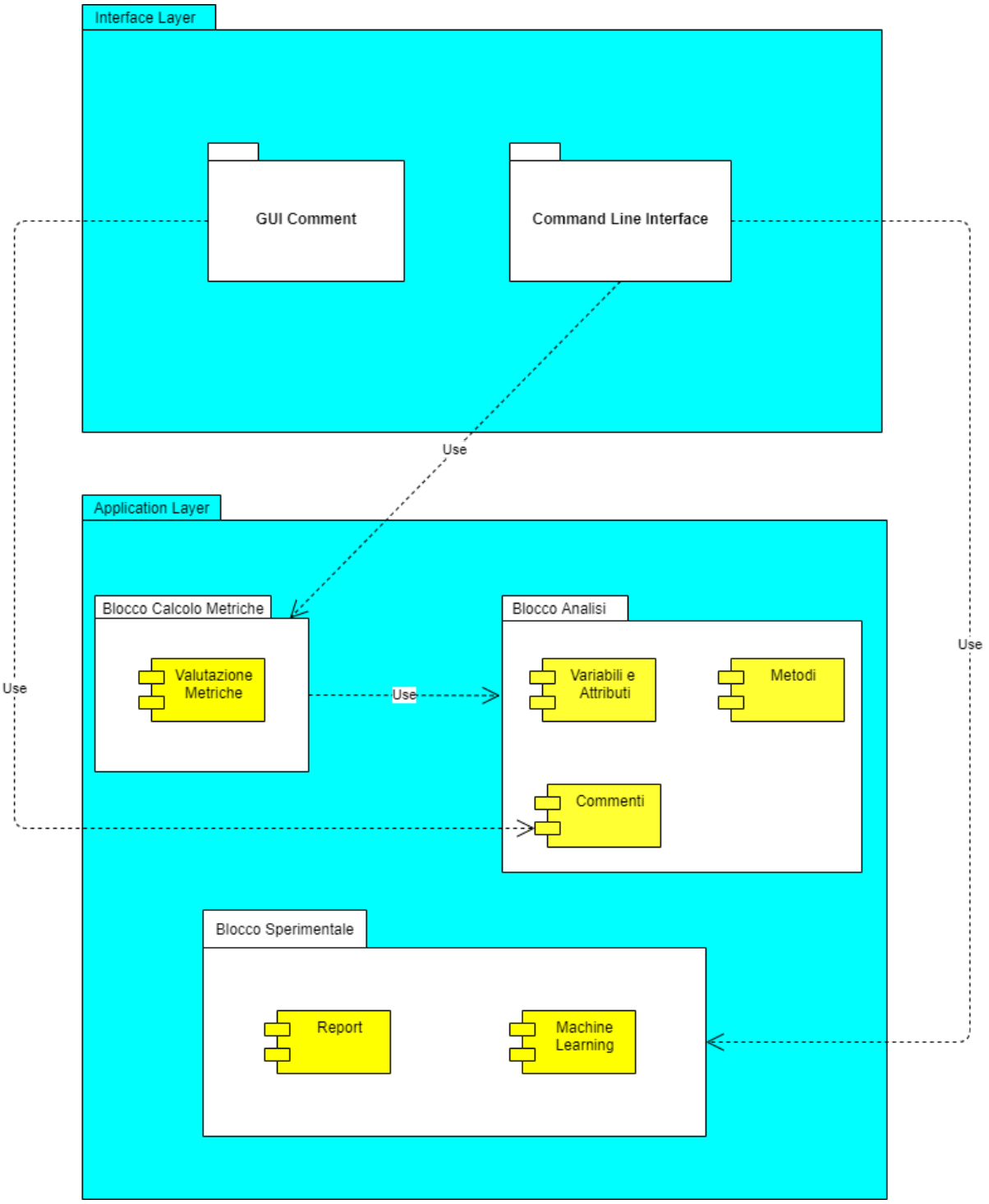
Il blocco di calcolo delle metriche ha un solo modulo:

- **Valutazione delle metriche**: questo modulo, si occupa di calcolare diverse metriche: tra cui quelle di Chidamber e Kemerer, basandosi solo su due delle tre strutture dati prodotte: il multigrafo orientato aciclico pesato e l'albero dei commenti.
  L'implementazione del calcolo delle metriche non prevede ulteriori visite dell'AST e si basa solo su informazioni già contenute nelle due strutture dati.

Il blocco sperimentale si compone di due moduli:

- **Report**: questo modulo è delegato alla gestione dei file di report, offre le facilities per acquisire e trattare informazioni presenti su file di report.
- **Machine Learning**:modulo sperimentale utilizzato per effettuare uno studio empirico sulle potenzialità tool metric nel predire difetti all'interno del codice.

Di seguito è riportato uno schema che mostra quanto elencato precedentemente:



Persistent data management

In Metric 3.0 non vi è nessun meccanismo di persistenza dei dati.

Obiettivi del sistema

Overview

L'obiettivo principale del sistema è quello di offrire all'utente un tool per l'analisi statica del codice sorgente Java.
Nello specifico, si vogliono fornire una serie di metriche che possano descrivere il software analizzato sia in termini qualitativi che quantitativi.

Metriche

Come accennato, l'obiettivo principale del tool è quello di produrre una serie di metriche atte a descrivere il codice sorgente analizzato, tali metriche sono:

- DIT (Depth of Inheritance Tree)
- WMC (Weighted Method Class)
- NOC (Number of Children)
- RFC (Response For a Class)
- CBO (Coupling Between Object Classes)
- LCOM (Lock of Choisen of Methods)
- LOC (Lines of Code)
- NC (Number of Comments)
- FI (Fan-in Fan-out)
- NMS (Number of Messages)
- NM (Number of Methods)
- NCp (Number of Classes)
- CC (Concrete Classes)
- AC (Abstract Classes)
- Ca (Coupling Afference)
- Ce (Coupling Efference)
- A (Abstractiveness)
- I (Instability)
- DFP (Distance From Parent)
- NOW (Number of Words)

Parametri

Essendo un sistema batch, l'interazione con l'utente è ridotta al solo avvio del sistema con la possibilità di indicare dei parametri.
Tra questi parametri vi sono, sia parametri essenziali al funzionamento che parametri opzionali.
I parametri possono alterare l'esecuzione del sistema, permettendo ad esempio di analizzare una parte del sistema anziché l'intero sistema, oppure di utilizzare una diversa versione di Java per l'analisi.
I parametri esprimibili sotto forma di argomenti passati al main sono:

- Parametri necessari:
    - **path [STRING]**: Una stringa contenente la path del progetto JAVA da analizzare;
    - **sourcePath [STRING]**: Permette di specificare la cartella dove risiede il codice sorgente nel progetto (tipicamente src)
- Parametri opzionali:
    - **java [INTEGER]**: Versione di java da utilizzare (di default 9);
    - **classfilter [BOOLEAN]**: Abilita il class filter, una funzionalità che permette di analizzare solo le classi selezionate
    - **class [STRING]**: Se abilitato "classfilter" permette di specificare i singoli path delle classi separati da virgola
    - **commentsGUI [BOOLEAN]**: Abilita una visualizzazione grafica dei commenti presenti nel progetto
- Parametri opzionali del modulo sperimentale:
    - **reportFilePath [STRING]**: Path report file(xlsx or xls)
    - **createDataset [BOOLEAN]**: Parametro che indica al sistema se creare un nuovo dataset di addestramento
    - **predictBugPresence [BOOLEAN]**: Abilita il modulo di defect prediction

Requisiti Funzionali

A valle delle analisi effettuate, sono stati individuati i seguenti requisiti funzionali e sono stati classificati secondo tre livelli di priorità (alta, media e bassa).
Tali requisiti sono:

- Alta priorità:
    - **RF1:** Creazione nuova analisi dell'intero progetto
    - **RF2:** Creazione nuova analisi di una parte del progetto
- Media priorità:
    - **RF3**: Creazione nuova analisi dell'intero progetto con visualizzazione finale dell'albero dei commenti (GUI)
    - **RF4**: Creazione nuova analisi di una parte del progetto con visualizzazione finale dell'albero dei commenti (GUI)
- Bassa priorità (modulo sperimentale):
    - **RF5**: Creazione nuova analisi dell'intero progetto con creazione dataset di addestramento in base a report processato
    - **RF6**: Creazione nuova analisi di una parte del progetto con creazione dataset di addestramento in base a report processato
    - **RF7**: Creazione nuova analisi dell'intero progetto con stima di difetti all'interno del codice
    - **RF8**: Creazione nuova analisi di una parte del progetto con stima di difetti all'interno del codice