Università degli Studi di Salerno

Dipartimento di Informatica

Corso di Ingegneria Gestione ed Evoluzione del Software

Aniello Florido & Alexander Minichino

July 22, 2019

Metric 3.0 Web Edition

Documentazione di manutenzione:
Analisi

Coordinatore del progetto:

Nome
Prof.Andrea De Lucia

Partecipanti:

Nome	**Matricola**
Aniello Florido	0522500625
Alexander Minichino	0522500644

Revision History

Data	**Versione**	**Descrizione**	**Autori**
01/02/2019	1.0	Prima stesura	Aniello Florido Alexander Minichino

Contents

1 Scopo del documento

In questo documento saranno descritti gli obiettivi del processo di migrazione sul web del progetto Metric 3.0, in riferimento al documento di "Identificazione e classificazione delle modifiche richieste". Si andrà ad analizzare come tale

modifica impetterà sugli artefatti del sistema esistente, osservando il rapporto costi/benefici e i possibili rischi derivati dallea fase di progettazione.
Questo documento includerà uno studio di fattibilità e inoltre, all'interno di esso verranno pianificate le fasi successive di progettazione, implementazione e testing.

2 Panoramica del sistema attuale

Il tool Metric 3.0 è stato sviluppato presso l'Università degli studi di Salerno.
Metric 3.0 esegue un'analisi dettagliata del codice sorgente Java, ottenendo delle informazioni che vanno a descrivere, attraverso una serie di metriche, la qualità del codice analizzato, così da offrire agli sviluppatori un valido aiuto nella comprensione di progetti con scarsa documentazione.
Il tool è stato implementato utilizzando la libreia JDT Core, la quale permette l'accesso all'AST (Abstract Syntaxt Tree), fornendo gli strumenti e le strutture dati per analizzarlo.
I nodi dell'AST sono gli elementi presenti nel codice sorgente, come classi, metodi ,variabili,ecc.

2.1 Metriche

Le metriche utilizzate all'interno del tool Metric 3.0 per l'analisi del codice sorgente sono:

- **DIT** (Depth of Inheritance Tree)
- **WMC** (Weighted Method Class)
- **NOC** (Number of Childern)
- **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)

2.2 Architettura del sistema

Il sistema è organizzato in due blocchi principali: il blocco di analisi e il blocco di calcolo.
Questi due blocchi comunicano tra di loro mediante la condivisione di tre strutture dati che vengono prodotte dal blocco di analisi ed utilizzate dal blocco di calcolo.

2.3 Testing

Per quanto riguarda il testing, sono stati utilizzati come input per l'analisi i principali moduli di Apache Tomcat v.8. Così da verificare il corretto funzionamento del tool su progetti software di considerevoli dimensioni.

3 Analisi della modifica richiesta

Il processo di migrazione del Metric 3.0 sul web prevede che i requisiti funzionali della versione precedente rimangano invariati.
L'obiettivo principale consiste nel creare un interfaccia web che permetta un utilizzo più semplificato e intuitivo del tool da parte dell'utente.

4 Individuazione della soluzione progettuale

4.1 Problematiche Affrontate

Issue 1: Quali funzionalità verrano migrate nel nuovo sistema?

Resolution
Verrà effettuata una migrazione del core del sistema Metric 3.0, tralasciando i moduli inerenti la GUI. Verranno, quindi, migrate le funzionalità legate a : • Parsing del codice sorgente Java e analaisi dell'AST – Parsing e analisi di variabili e attributi – Parsing e analisi delle catene di invocazione dei metodi – Parsing e analisi dei commenti • calcolo delle metriche – Metriche code size – Metriche Chidamber e Kemerer (Object Oriented) – Metriche di pacchetto – Metriche relative ai commenti

Issue 2: Quali funzionalità del sistema hanno priorità maggiore?

Resolution
Le funzionalità del sistema che hanno priorità maggiore sono quelle legate al parsing del codice sorgente e quelle legate all'analisi dell' AST attraverso JDTCore.

Issue 3: Sono necessarie nuove funzionalità?

Resolution
Verrà implementato un nuovo modulo che rappresenterà un'interfaccia che permetterà la comunicazione tra il sistema esistente e il nuovo sistema. Nello specifico saranno implementate delle classi (serverlet Java) che avranno il ruolo di dispatcher delle richieste effettuate dalle nuove pagine JSP (Java Servlet Page). Vi sarà anche la possibilità di storicizzare i risultati delle analisi effettuate, le quali verranno associate agli utenti che le effettuano.

Issue 4: che tipologia di interfaccia grafica adottare?

Resolution
Si andrà ad implementare una GUI semplice ed intuitiva, che permetterà di sfruttare a pieno le funzionalità del tool Metric 3.0. Inoltre, l'interfaccia sarà resa responsive così da garantire una resa ottimale su tutti i dispositivi.

Issue 5: che modello architetturale verrà utilizzato?

Resolution
Come modello architetturale si è sclto di utilizzare il Pattern Facade. L'implementazione di di tale pattern permette, attraverso un'interfaccia più semplice, l'accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro, nonchè a blocchi di codice complessi.

Issue 6: che web server o web container utilizzare?

Resolution
Come Web Server si è scelto Apache Tomcat v.9 in quanto si tratta del web server più utilizzato e diffuso per l'implementazione di sistemi client-server basati su Java.

Issue 7: che tipo di DBMS verrà utilizato?

Resolution
Per la persistenza dei dati si è scelto di non utilizzare un DBMS, ma di salvare i dati direttamente su un file. Tale scelta è stata presa poichè non sussiste la necessità di creare relazioni tra i dati prodotti dal tool (risultati analisi), tranne quella che vi è tra l'analisi effettuata e l'utente che l'ha richiesta.

Issue 8: che tipologia i linguaggio di programmazione utilizzare per lo sviluppo del nuovo sistema?

Resolution
Il linguaggio scelto per l'implementazione è JAVA così da garantire una piena compatibilità con il sistema esistente e il Web Server Apache Tomcat. Inoltre verranno utilizzati linguaggi di markup (come HTML e XML), di formattazione (CSS), di scripting (Javascript) e di rappresentazione dei dati (Json) per l'implementazione delle JSP.

4.2 Soluzione individuata

La soluzione individuata consiste nell'unione di tutte le resolutions presenti nel paragrafo precedente, di conseguenza la soluzione individuata avrà le seguenti caratteristiche:

- Migrazione del core del sistema tralasciando i moduli inerenti la GUI.
- Verrà implementata un'interfaccia che permetterà la comunicazione tra il sistema esistente e il nuovo sistema.
- Sarà permessa la storicizzazione dei dati, associandoli agli utenti che svolgono le analisi.
- Verrà implementrata una GUI che sia il più possibile intuitiva e semplice.
- Il modello architetturale scelto è il Pattern Facade.
- Il Web Server scelto è Apache Tomcat v.9

- I dati verranno salvati all'interno di un file.
- I linguaggi utilizzati sono Java, per garantire una totale compatibilità con il vecchio sistema. Inoltre, verrano utilizzati linguaggi di markup (HTML e XML), di formattazione (CSS), di scripting (Javascript) e di rappresentazione dei dati (JSON) per l'implementazione delle pagine JSP.

5 Identificazione dell'Impact Set

La soluzione individuata non è andata a modificare molti artefagtti del sistema precedente.
La tabella che segue descrive il Candidate Impact Set individuato, ovvero l'insieme di tutti gli artefatti che verranno modificati durante la fase di manutenzione. Inoltre, accanto ad ogni artefatto sarà associato un livello di impatto, ovvero tale livello farà riferimento al modo in cui la modifica va ad impattare sul sistema software attuale.

Per individuare le componenti del sistema impattate dalla modifica si utilizzerà un approccio topdown, ovvero a partire dai documenti di alto livello si andrà ad individuare gli artefatti di più basso livello che andranno ad essere modificati durante la fase di manutenzione.
La nuova interfaccia Web deve consentire all'utente di effettuare tutte le operazioni che vengono messe a disposizione dal tool Metric 3.0. Per tale motivo non vi sono sostanziali modifiche ai requisiti funzionali e ai casi d'uso presenti nel tool.
Le uniche differenze apportate rispetto al vecchio sistema stanno nella creazione di un'interfaccia GUI e della permanenza dei dati correlata dall'aggiunta di un profilo utente.
Per quanto riguarda l'architettura del sistema, essa verrà modificata in base alle nuove specifiche elencate precedentemente.
Nella tabella che segue sono indicati gli artefatti del documento che saranno impattati. L'impatto della modifica verrà valutato utilizzando tre categorie:

- FORTE: se saranno necessarie pesanti modifiche nell'artefatto o se l'artefatto dovrà essere completamente sostituito;
- MEDIO: se saranno necessarie sostanziali modifiche all'artefatto, non facendo cambiare però la sua struttura in maniera eccessiva;
- DEBOLE: se saranno necessarie solo modifiche marginali;

Artefatto	Impatto	Descrizione
Decomposizione in sottosistemi	Medio	
Mapping Hardware/Software	Forte	Sarà possibile accedere al sistema direttamente via Web, da qualsiasi browser. Tutto ciò sarà reso possibile dalla creazione di un'interfaccia
Flusso di controllo globale	Forte	Il sistema verrà implementato in modo da far comunicare il tool Metric 3.0 con l'interfaccia creata.

Nella tabella sottostante verranno riportati gli artefatti che saranno impattati dalla modifica.

Artefatto	Impatto	Descrizione
Packages	Debole	Verrà aggiunto un Package contenente le nuove classi che compongono l'interfaccia
Comunicazione tra Package	Medio	Verranno specificati i package con i quali andranno a comunicare i nuovi package e il modo con cui essi comunicano.
Classi e Intefacce	Debole	Verrano specificate le classi che andranno a comporre i nuovi package

Per quanto concerne il codice, non vi sono modifiche che vanno ad impattare sul codice del tool Metric 3.0.

6 Studio di fattibilità

6.1 Identificazione, descrizione e valutazione dei costi

Identificazione	Valutazione	Motivazioni
Migrazione dei requisiti funzionali	Debole	Verrà effettuata una migrazione graduale del core del tool, così come specificato nella Issue 1.
Aggiunta di nuove funzionalità	Media	Le modifiche richieste prevedono l'aggiunta della persistenza dei dati con relativa gestione dei profili utenti a cui vengono abbinate le analisi. La creazione di un'interfaccia Web per interagire con il tool.
Alta somiglianza con l'interfaccia esistente	Forte	L'obiettivo principale del processo di migrazione prevede lo sviluppo dei moduli che si occupano dell'intefaccia grafica.
L'architettura usata è il Pattern Facade	Medio	Tale pattern prevede che il lavoro venga concentrato sulla presentazione dei dati con la creazione di un nuova interfaccia, mentre le classi che compongono il tool vengono mantenute così come sono.
Utilizzo del Web Server Apache e di Tomcat	Debole	Il Web Server Apache come Tomcat sono gratuiti pertanto non risulta disponibile a costo zero.

Implementazione con linguaggi di programmazione client-side HTML, CSS, JavaScript e serverside Java.	Debole	Il team ha un'ottima conoscenza di tali tecnologie.
Livello di complessità dell'interfaccia utente.	Media	L'utilizzo combinato di HTML, CSS e Javascript permette la creazione di un'interfaccia utente che possa essere quanto più avanzata. Inoltre verranno implementati alcuni controlli presenti nelle tradizionali applicazioni standalone.
Compatibilità con diversi tipi e versioni di browser.	Forte	Il sistema, funzionando sul web, verrà implementato con framework e linguaggi che possano essere facilemente interpretati da qualsiasi browser, così da avere una piena compatibilità.
Velocità di accesso alle risorse.	Debole	Le tecnologie attuali permettono di sorvolare gli aspetti di ottimizzazione dei tempi di risposta da parte del server.
Mantenimento dello stato dell'applicazione.	Media	Il modello di interazione di base del Web è privo del concetto di connessione, per cui l'esecuzione di un'applicazione web consiste di una serie di interazioni disconnesse. Pertanto si prevede l'utilizzo di informazioni codificate nella richiesta HTTP.

Organizzazione del lavoro nel team di sviluppo.	Forte	I componenti del team di sviluppo condividono poche ore settimanali per lo sviluppo del nuovo sistema. Pertanto verranno effettuate riunioni extra il sabato e/o la domenica.
Testing funzionale	Forte	I requisiti funzionali testati nel sistema attuale sono pochi e coprono pochi casi di test. Pertanto, il testing verrà affrontato ex-novo.

6.2 Identificazione, descrizione e classificazione dei benefici

Facilità di distribuzione e manutenzione.	Forte	L'applicazione web si trova interamente sul server, per cui la pubblicazione o l'aggiornamento di risorse sul nodo server è automaticamente reso disponibile a tutti i nodi client.
Accesso multipiattaforma.	Forte	L'accesso all'applicazione web è indipendente dall'hardware e dal sistema operativo utilizzato dagli utenti.
Riduzione del costo di gestione.	Forte	L'uso di Internet come infrastruttura per un'applicazione web riduce notevolmente sia i costi di connettività che i costi digestione dei client.