UNIVERSITÀ DEGLI STUDI DI SALERNO

Facoltà di Scienze Matematiche Fisiche e Naturali

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

INGEGNERIA GESTIONE ED EVOLUZIONE DEL SOFTWARE
ANNO ACCADEMICO 2016/2017

MINING MAILING LISTS

DOCUMENTO DI MANUTENZIONE:
ANALISI

VERSIONE 1.1
30 NOVEMBRE 2016

Coordinatore del progetto:

Nome
Prof. Andrea De Lucia

Candidato:

Nome	Matricola
ADP – Antonio De Piano	0522500397

Revision History

Data	Versione	Descrizione	Autore
06/11/2016	1.0	Prima stesura	Antonio De Piano
30/11/2016	1.1	Revisione del documento, aggiunta della soluzione individuata e dell'Impact Set	Antonio De Piano

Indice

1. Scopo del documento

In questo documento saranno descritti gli obiettivi del processo di integrazione del modulo “Mining Mailing Lists” all’interno del sistema esistente GitDM, in riferimento alle specifiche richieste.
Verrà studiato come l’integrazione potrà impattare sugli artefatti del sistema esistente, analizzando il rapporto costi/benefici e i possibili rischi derivanti dalla fase di progettazione.
Questo documento includerà uno studio di fattibilità e verranno pianificate le fasi successive di progettazione, implementazione e testing.

2. Panoramica del sistema esistente

Il sistema attuale, GitDM incapsula un insieme di tecniche e metodologie che hanno per oggetti l’estrazione di un sapere o di una conoscenza da grandi quantità di dati (attraverso metodi automatici o semi-automatici). In particolare GitDM permette di estrarre informazioni (commit, status log, checkout i merge ecc …) da un dato progetto disponibile sul noto repository online GitHub.

2.1 Attori e funzionalità

Il sistema permette di estrapolare informazioni circa i commit, i checkout, gli status log, i merge, l’aggiunta di file sull’ index e tutte le altre operazioni permesse su un progetto da GIT e mostrate online attraverso GitHub.

2.2 Design, implementazione e testing del sistema attuale

2.2.1 Design

Non risulta una documentazione relativa all’object design.
Pertanto da una prima analisi possiamo dire che il sistema GitDM è organizzato in sottosistemi:

- Core
- initRepo
- newcomers
- scattering

A loro volta, questi sottosistemi sono organizzati in package che opportunamente comunicano tra di loro.

2.2.2 Implementazione

Non risulta una documentazione relativa all’implementazione.
Analizzando il codice, possiamo dire che il sistema GitDM è stato sviluppato nel linguaggio di programmazione Java.

2.2.3 Testing

Non risulta una documentazione relativa al testing, così come l’implementazione delle classi di test.
Pertanto si prevede di implementare le classi di test delle nuove componenti sviluppate utilizzando il framework JUnit.

3. Analisi della modifica richiesta

Il processo di integrazione del modulo “Mailing Mailing Lists, MML” prevede che le funzionalità del sistema esistente rimangano invariate. L’obiettivo principale consiste nell’integrare un modulo di estrazione delle email dall’archivio email di Apache Foundation (http://mail-archives.apache.org/mod_mbox/) al sistema esistente GitDM.

4. Individualizzazione della soluzione progettuale

4.1 Problematiche affrontate

Issue 1: Quali funzionalità verranno integrate nel nuovo sistema?

Proposal 1.1
Un “minatore” per estrapolare le email scambiate dagli sviluppatori in un dato periodo, presenti nell’archivio email di Apache Foundation (http://mail-archives.apache.org/mod_mbox/). Ricercare, analizzare specifiche informazioni nelle email estrapolate (parte del messaggio, parole chiavi, indirizzo email etc..)
Proposal 1.2
Un parser che permetta di analizzare, ricercare le informazioni contenute online nell’archivio email di Apache Foundation (http://mail-archives.apache.org/mod_mbox/).

Criterion 1.1
Integrare funzionalità che permettano il corretto funzionamento del sistema e da cui acquisire reali vantaggi.
Criterion 1.2
Integrare solo la possibilità di analizzare e ricercare informazioni, interrogando l’archivio online.

Argument 1.1
Con l’integrazione del “minatore” possiamo memorizzare in locale l’intero archivio email degli sviluppatori di Apache Foundation ed effettuare l’analisi dei dati, oppure la ricerca di specifici contenuti. Con l’integrazione di questa funzionalità, la fase di analisi e ricerche dei contenuti è senz’altro più semplice.
Argument 1.2
Con l’integrazione del parser sarà possibile effettuare analisi e ricerche dei contenuti presenti nell’archivio email di Apache Foundation, senza dover scaricare i dati, disponibili online.

Resolution 1.1
L’Argument 1.1 è stato risolto applicando la Proposal 1.1

Issue 2: Quali funzionalità del sistema hanno priorità maggiore?

Proposal 2.1
Il “minatore”
Proposal 2.2
Motore di ricerca full-text

Criterion 2.1 - 2.2
Sviluppare le funzionalità fondamentali del sistema.

Argument 2.1

Il “minatore” è la funzionalità core del sistema. Le altre funzionalità potrebbero essere inizialmente emulate attraverso driver e stub.
Argument 2.2
Sviluppare per prima il motore di ricerca full-text consente di avere immediatamente a disposizione un analizzatore full-text su generici contenuti da ricercare.

Resolution 2.1
Analizzando le argomentazioni Argument 2.1 e Argument 2.2, l’Issue 2 è stata risolta applicando la Proposal 2.1.

Issue 3: Sono necessarie nuove funzionalità?

Proposal 3.1
Nessuna nuova funzionalità aggiunta.
Proposal 3.2
Aggiunta di una feature per visualizzare l’email (date, from, to, subject e content) in cui la query di ricerca ha prodotto risultato.

Criterion 3.1
Sviluppare funzionalità solo se strettamente necessario per ridurre i tempi di sviluppo.
Criterion 3.2
Fornire nuove funzionalità per la visualizzazione dei risultati.

Argument 3.1
Le funzionalità attuali consentono l’utilizzo del sistema. Nuove funzionalità non sono vitali.
Argument 3.2
Il sistema visualizza solo il nome del file in cui il motore di ricerca ha prodotto risultato. Fornire una visualizzazione dell’email (date, content, from, to, subject) consentirebbe di avere una visione più chiara del contesto d’uso della chiave ricercata.

Resolution 3.1
Analizzando le argomentazioni Argument 3.1, Argument 3.2 e Argument 3.3, l’Issue 3 è stata risolta applicando la Proposal 3.2.

Issue 4: Che tipo di interfaccia utente adottare?

Proposal 4.1
L’interfaccia deve ricordare quella presente nel sistema attuale.
Proposal 4.2
Interfaccia grafica user-friendly (GUI).

Criterion 4.1
L’interfaccia utente deve essere a caratteri (CLI).

Argument 4.1
Un’interfaccia simile a quella utilizzata nel sistema attuale consente agli utenti maggiore familiarità con la nuova funzionalità integrata.
Argument 4.2
Un’interfaccia grafica user-friendly consente all’utente di interagire con la macchina controllando oggetti grafici convenzionali.

Resolution 4.1
Analizzando le argomentazioni Argument 4.1 e Argument 4.2, l’Issue 4 è stata risolta applicando la Proposal 4.1.

Issue 5: Che tipo di file utilizzare per memorizzare le informazioni?

Proposal 5.1
Utilizzo del file di testo.
Proposal 5.2
Serializzazione di oggetti su file “file di oggetti java”

Criterion 5.1
Permette l’utilizzo della ricerca full-text sull’intero archivio email mensile.
Criterion 5.2
Permette una rapida consultazione dell’intero oggetto email.

Argument 5.1
Il file di testo è il modo più semplice per memorizzare informazioni non strutturate.
Argument 5.2
Serializzando oggetti su file, è garantito un supporto completo in fase di lettura e scrittura di oggetti java.

Resolution 5.1
Analizzando le argomentazioni Argument 5.1 e Argument 5.2, l’Issue è stata risolta applicando la Proposal 5.1.

Issue 6: Come ricercare le informazioni?

Proposal 6.1
Leggendo e confrontando la query con il contenuto del file (rigo per rigo) prodotto dal “minatore”.
Proposal 6.2
Per l’implementazione del motore di ricerca delle informazioni full-text si utilizza Apache Lucene

Criterion 6.1
Permette oltre che alla ricerca della query, anche un analisi non formale sul contenuto rigo per rigo del file.
Criterion 6.2

Lucene è un API gratuita ed Open Source per il reperimento di informazioni. Permette di realizzare applicazioni che necessitano di funzionalità di indicizzazione e ricerca full-text.

Argument 6.1
Utilizzando una lettura da file possiamo analizzare anche i sinonimi utilizzati nell'email oltre che ricercare la specifica query.
Argument 6.2
Utilizzando Lucene è possibile effettuare ricerche full-text su una collezione di file. L'output di Lucene saranno i nomi dei file in cui la query è contenuta.

Resolution 6.1
Analizzando le argomentazioni Argument 6.1 e Argument 6.2, l'Issue è stata risolta applicando la Proposal 6.2.

4.2 Soluzione individuata

La soluzione individuata consiste nell'unione di tutte le "resolution" ottenute nel paragrafo 4.1. In dettaglio, la soluzione individuata dovrà avere le seguenti caratteristiche:

- Integrare un "minatore" per estrapolare le email scambiate dagli sviluppatori in un dato periodo, presenti nell'archivio email pubblico di Apache Foundation (http://mail-archives.apache.org/mod_mbox/). Ricercare specifiche informazioni nelle email estrapolate (parte del messaggio, parole chiavi, indirizzo email etc).
- Priorità alta allo sviluppo del "minatore".
- Aggiunta di una feature per visualizzare l'email (date, from, to, subject e content) in cui la query di ricerca ha prodotto risultato.
- L'interfaccia deve ricordare quella presente nel sistema attuale.
- Utilizzo del file di testo per memorizzare le informazioni "minate" dall'archivio email di Apache Foundation
- Per l'implementazione del motore di ricerca full-text si utilizza Apache Lucene, un API gratuita ed Open Source. Permette di realizzare applicazioni che necessitano di funzionalità di indicizzazione e ricerca full-text.

4.3 Soluzioni alternative

Le soluzioni alternative sono rappresentate da tutte le proposte presentate e affrontate nel paragrafo 4.1, ma che non sono state selezionate per la soluzione adottata.

5. Identificazione dell'Impact Set

La soluzione individuata non ha coinvolto nessuno degli artefatti del sistema attuale. Pertanto il sistema attuale non ha subito modifiche hardware/software dovute all'implementazione del modulo qui sopra descritto.

In definitiva, nel documento di analisi e specifica dei requisiti di dovrà solo aggiungere i requisiti funzionali e i casi d'uso che riguardano le funzionalità che saranno aggiunte, ovvero la funzionalità di estrapolare, analizzare e ricercare informazioni dall'archivio email di Apache Foundation.

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	DEBOLE	Alla divisione in sottosistemi verrà aggiunta solo quella relativa alla nuova funzionalità da integrare.
Mapping Hardaware/Software	DEBOLE	Sarà possibile accedere al sistema MML attraverso un'interfaccia CLI, perfettamente integrata con il resto del sistema.
Flusso di controllo globale	MEDIO	Dovrà essere studiato un nuovo modo per eseguire il sottosistema MML all'interno del sistema.

Nell' Object Design Document (ODD), si può stilare una prima lista degli artefatti che verranno impattati dalla modifica:

Artefatto	Impatto	Descrizione
Packages	DEBOLE	Nell'artefatto andranno aggiunti i nuovi package del modulo sviluppato.
Comunicazione tra i packages	DEBOLE	I packages relativi al nuovo sottosistema sviluppato non comunicano con nessun altro packages del sistema esistente.
Classi ed interfacce	MEDIO	Nell'artefatto andranno esplicitate le classi che compongono i nuovi package aggiunti.

Dopo l'analisi dell'Object Design, è possibile stilare una lista delle classi java che saranno impattate dalla modifica.

Da un analisi accurata, risulta che il sottosistema implementato non impatta su nessuna classe java esistente, in quanto esso implementa una funzionalità esterna al focus del sistema esistente, arricchendo così il dominio applicativo.

Modifiche minime dovranno sicuramente essere fatte per la sola ragione di integrare il sottosistema sviluppato all'interno del sistema esistente.

6. Studio di fattibilità

6.1 Identificazione, descrizione e valutazione dei costi

Identificazione	Valutazione	Motivazioni
Integrazione dei requisiti funzionali.	DEBOLE	Il processo di integrazione è effettuato in modo incrementale. Pertanto in primo momento verranno integrate le funzionalità relative all'estrapolazione dei contenuti da Apache Foundation.
Alta somiglianza con l'interfaccia esistente.	MEDIA	E' uno degli obiettivi principali del processo di sviluppo del sottosistema MML.
Utilizzo di Apache Lucene API	DEBOLE	Apache Lucene è un API gratuita ed open source. Pertanto sarà disponibile a costo zero.
Utilizzo dei file di testo per memorizzare le informazioni estrapolate	DEBOLE	L'archivio email di Apache Foundation permette di estrapolare le informazioni in formato testo.
Implementazione con il linguaggio JAVA	DEBOLE	Lo sviluppatore ha un ottima conoscenza del linguaggio.
Compatibilità con diversi S.O	FORTE	Il sottosistema MML è stato sviluppato in Java per garantire la conformità con l'intero sistema ed è stato progettato per essere portatile.
Velocità di accesso alle risorse	DEBOLE	La velocità di accesso agli archivi email consultati dipende dalla loro dimensione. I tempi di risposta legati alla ricerca di contenuti con Apache Lucene sono sorvolabili in quanto le API garantiscono ottime prestazioni.
Gestione della sicurezza	DEBOLE	La gestione della sicurezza dei file mantenuti sul client dipende dal client stesso.
Testing funzionale	FORTE	Il testing verrà affrontato ex-novo.
Testing di accettazione	FORTE	E' necessaria l'interazione con il cliente finale per analizzare i risultati del test.

6.2 Identificazione, descrizione e classificazione dei benefici

Identificazione	Classificazione	Motivazione
Facilità di distribuzione e manutenzione	FORTE	L'applicazione è stata progettata per essere distribuita su diversi S.O (Windows, Unix, Linux).
Facilità di ricerca e analisi dei dati	FORTE	L'utente non si deve più preoccupare di analizzare ogni singolo archivio mensile attraverso il sito web (http://mail-archives.apache.org/mod_mbox/) ma potrà interrogare il motore di ricerca full-text sui dati scaricati.
Scalabilità	FORTE	Il sottosistema MML può crescere insieme alle esigenza del cliente, senza particolare problemi.
Facilità nel consultare l'archivio dati.	FORTE	L'intero archivio o parte di essi (da mm/aa a mm/aa) sarà memorizzato sul HDD del utente. L'utente potrà consultarlo in qualsiasi momento offline.

6.3 Identificazione, descrizione e classificazione dei rischi

Identificazione	Probabilità	Motivazioni
Introduzioni di nuovi fault.	MEDIA	Il testing che si intende effettuare aumenta le probabilità di riscontrare nuovi errori.
Carico di lavoro eccessivo per il Client	MEDIA	I client che intendono estrapolare informazioni dall'archivio email di Apache Foundation, sono a conoscenza che il tempo di esecuzione dello script varia in base all'intervallo temporale fornito dall'utente (mm/aa - mm/aa) e dal contenuto presente nell'archivio consultato.
Il server rendere disponibile l'archivio email a un nuovo URL (Es. vecchio URL http://mail-archives.apache.org/mod_mbox/)	FORTE	Il client estrapola i dati a partire dal nome del progetto e dall'intervallo mm/aa a mm/aa fornito. Pertanto se l'archivio email viene spostato, il client non sarà in grado di recuperare le informazioni.
Il server modifica la struttura testuale degli archivi mensili.	FORTE	Il client organizza i messaggi presenti negli archivi in base alle parole chiavi: "Message-ID: ","In-Reply-To: ","Subject: ","Date: " e il Content. Qualora una di queste parole chiavi non sia presente

		negli archivi, il client non sarà in grado di recuperare tutte le informazioni relative al singolo messaggio.

6.4 Risultati previsti

La realizzazione del sistema software è definita secondo la soluzione individuata nel paragrafo 4.2, tenendo in considerazione costi, benefici e rischi di cui al paragrafo 6.