UNIVERSITÀ DEGLI STUDI DI PADOVA

FACOLTÀ DI SCIENZE STATISTICHE

Corso di laurea in Statistica e Tecnologie Informatiche

TESI DI LAUREA

Ambiente per l'analisi della qualità dei software Open Source

Open Source software quality analysis environment

Creazione di un ambiente basato sulla piattaforma Spago4Q che utilizza i modelli di qualità QualiPSo

Laureando :
Patrick Masiero

Relatore :
Prof. Nicola Zingirian
Tutor Aziendale :
Ing. Davide dalle Carbonare

aa 2009/2010

ai miei genitori...

Indice generale

1 Sommario

L'adozione ed il rilascio di software Open Source[1] in ambito professionale sono recentemente divenuti una chiave strategica per la penetrazione dei mercati da parte delle Software House. Tuttavia sussistono ancora forti preoccupazioni sulla effettiva affidabilità di questa tipologia di software, e soprattutto forti dubbi sulla possibilità o meno di valutare tale affidabilità[2]. Tali dubbi hanno portato ad un tasso di adozione del software Open Source molto inferiore a quello previsto, soprattutto nell'ambito aziendale nel quale la qualità è un fattore critico.

Le attività riportate in questa tesi, sono state svolte presso Engineering S.p.a, azienda fortemente interessata all'argomento, ed hanno contribuito alla creazione di un ambiente di integrazione che, grazie all'utilizzo di particolari strumenti selezionati, sia in grado di rilevare in maniera automatizzata le informazioni necessarie per valutare la qualità dei progetti Open Source. Da tali strumenti sono stati estratti i dati da analizzare e veicolati nel data warehouse della piattaforma Open Source Spago4Q. Questo software sviluppato da Engineering S.p.a è pensato per valutare l'adeguata maturità ed efficacia dei processi di sviluppo del software e dei prodotti realizzati.

Successivamente sono state eseguite delle analisi su alcuni software Open Source, in riferimento ai Key Performance Indicators (KPI) correlati ai singoli elementi del modello di qualità predefinito per valutarne la bontà, per questo scopo si è utilizzato un sottoinsieme di indicatori ricavati dal modello MOSST[2] di QualiPSo, che si occupa di valutare la qualità dello sviluppo dei progetti Open Source.

Infine, dai dati raccolti sono stati creati appositi documenti analitici che mostrano i risultati ottenuti in una forma semplice e di rapida comprensione, utili per la valutazione dello stato di avanzamento del progetto e come supporto per l'individuazione delle azioni di miglioramento dei processi. Questo sistema avrà lo scopo di fornire degli indicatori, che verranno utilizzati dagli analisti che avranno il compito di certificare l'effettiva qualità di un software Open Source e di conseguenza favorirne l'adozione.

Durante l'attività di stage presso la divisione ricerca ed innovazione, è stata realizzata la documentazione dove vengono descritte le procedure per la creazione e utilizzo di un ambiente completo per l'analisi della qualità. La documentazione verrà utilizzata da Engineering e dai suoi partner per facilitare la creazione di FlossItaly, centro di competenza italiano per l'Open Source.

1 Open Source Software [OSS] indica un software i cui autori (più precisamente i detentori dei diritti) ne permettono, anzi ne favoriscono il libero studio e l'apporto di modifiche da parte di altri programmatori indipendenti. Questo è realizzato mediante l'applicazione di apposite licenze d'uso [1]

2 Model for Open Source Software Trustwothiness

2 Introduzione

2.1 Engineering

Engineering Ingegneria Informatica viene fondata a Padova il 6 giugno 1980. Ad oggi il gruppo Engineering con 11 società, 37 sedi operative in Italia e all'estero, oltre 6.200 professionisti IT ed uno dei maggiori Data Center europei per l'outsourcing. È la prima azienda nazionale di consulenza e servizi, capace di offrire servizi completi lungo l'intera catena del valore del software: progettazione e sviluppo, servizi di outsourcing, prodotti e soluzioni verticali proprietarie, consulenza IT e strategie di mercato su misura per i modelli di business dei clienti su tutti i mercati.

Engineering si estende su sette aree di business :

• Finanza

• Pubblica Amministrazione Centrale

• Pubblica Amministrazione Locale

• Sanità

• Energy & Utilities

• Industria & Servizi

• Telecom

affiancate da sei centri di competenza trasversali ad elevata specializzazione, e dalla Direzione Ricerca e Innovazione, che ha il doppio ruolo di promuovere la ricerca sul software a livello internazionale e trasferire l'innovazione al ciclo produttivo delle strutture di business.

Illustrazione 1: Struttura Engineering Ingegneria Informatica s.p.a.

2.1.1 Certificazioni di Qualità

Engineering è sempre stata all'avanguardia per quanto riguarda la qualità , infatti è stata tra le prime società italiane ad adottare la certificazione alla norma di qualità dei processi di produzione ISO 9000. Da allora la società ha continuato ad ottenere varie certificazioni per garantire ai propri clienti una qualità sempre maggiore nei propri servizi.

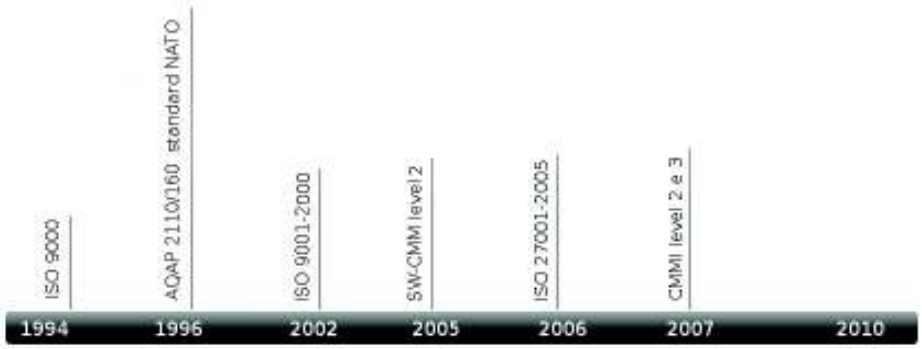

Illustrazione 2: Le certificazioni di Engineering

2.1.2 Divisione di mercato

Il Gruppo ha delle diverse divisioni di mercato specializzate, e società operative autonome secondo gli obiettivi aziendali diversificate per linee di competenza :

- SAP
- Pont St. Martin, centro europeo di Operations Management
- Enterprise Content Management
- Automazione e Controlli
- Sicurezza
- Open Source e Business Intelligence

Gli argomenti descritti in questa tesi sono stati svolti con la collaborazione del centro Open Source e Business Intelligence. Le principali competenze offerte dal centro ruotano intorno all'iniziativa SpagoWorld[3] nonostante collabori con molte altre iniziative[3].

3 *Competenze specifiche in ambito SOA/BPM, Enterprise Portals e Enterprise Content Management.*
Competenze derivanti dalla gestione di comunità globali quali Consorzio OW2, QualiPSo, NESSI OS Working Group, SpagoWorld.
Contribuzione a progetti open source: Eclipse STP/Intermediate Model, Apache ServiceMix, eXo platform,Mondrian, Harvard JHOVE.
Sviluppo di soluzioni infrastrutturali che applicative basate su Java EE, RedHat Enterprise Linux, Suse Novell Linux, JBoss AS, JOnAS, Ingres, Liferay, eXo, JetSpeed, PHPNuke, Alfresco, Struts, Spring, JSF, ServiceMix, PEtALS, Lucene, Carrot2, CAS, OFBiz, OLAT, Talend, JasperSoft.

La divisione di mercato che si occupa di Ricerca e Innovazione ha un doppio ruolo all'interno del ciclo produttivo del gruppo Engineering. Le risorse sviluppano innovazione e grazie alle sinergie con le business unit di mercato, trasferiscono i risultati della ricerca nella fase progettuale verso i clienti, generando valore e nuove opportunità per le strategie ed i modelli di business richiesti dal mercato.

Illustrazione 3: Ciclo dell'innovazione

La Direzione Ricerca e Innovazione ha utilizzato una strategia di approccio fondata sull'utilizzo, la realizzazione e l'integrazione di soluzioni Open Source come opportunità di business nell'offerta ai clienti. Per questo è strettamente correlata al centro di competenza Open Source e Business Intelligence anche attraverso la promozione e il supporto alle soluzioni gestionali raccolte sotto il brand SpagoWorld, utilizzate in Italia e all'estero come piattaforma di riferimento per lo sviluppo di applicazioni in progetti per Pubblica Amministrazione, Finanza e Industria.

2.2 Spago4Q

Spago4Q (SpagoBI for Quality)[4] è una piattaforma di software libero[4] che fa parte del gruppo SpagoWorld per la misurazione, l'analisi ed il monitoraggio della Qualità di prodotti, processi e servizi. La piattaforma è facilmente adattabile a contesti organizzativi complessi, in modo indipendente dai processi di sviluppo e manutenzione dei software utilizzati, dai tool infrastrutturali e dai modelli di valutazione[5] e di misurazione[6] adottati, inoltre Spago4Q è utilizzabile nei processi di valutazione della maturità e dell'efficacia dei processi di sviluppo software, di erogazione di servizi IT e di verifica del software e dei prodotti realizzati. Permette inoltre di raccogliere dati e misure da vari progetti, repository di codice ed altri strumenti,

4 Free Libre Open Source Software – FLOSS
5 CMMI, ISO 9001:2008, ITIL
6 GQM

attraverso tecniche non invasive. Spago4Q è una verticalizzazione di SpagoBI, cui aggiunge uno specifico meta-modello per l'analisi delle informazioni sulla qualità del software, oltre a diversi estrattori per la raccolta dei dati.

Spago4Q è di supporto ad aziende ed organizzazioni impegnate a perseguire obiettivi di miglioramento continuo, sia nell'ambito di un processo di certificazione della qualità sia nel monitoraggio di un processo di sviluppo formalizzato o dell'erogazione di servizi IT. Inoltre, è a supporto dei diversi attori del processo di sviluppo nel fornire loro informazioni riguardanti l'avanzamento delle attività. Un numero configurabile di cruscotti, report e documenti OLAP permettono di monitorare i vari processi, quali la gestione dei requisiti, delle anomalie, dei rischi e dei test, fornendo una visione ad alto livello della qualità percepita di un determinato software, ma configurato in modo che l'utilizzatore possa scendere sempre più nel dettaglio in base alle proprie esigenze. L'evoluzione di Spago4Q è stata favorita dalla partecipazione di Engineering al progetto QualiPSo, progetto finanziato dalla Commissione Europea nell'ambito del sesto programma quadro[5].

2.3 QualiPSo

Il consorzio QualiPSo[6] è stato fondato per aiutare industrie e governi ad adottare un sistema di innovazione e di competitività, fornendo le modalità per un utilizzo della flessibilità del software Open Source per avere "affidabilità a basso costo", e che permetta di sviluppare sistemi informatici innovativi ed velidi. Per raggiungere questo obiettivo QualiPSo intende definire e attuare le tecnologie, processi e politiche, per facilitare lo sviluppo e l'utilizzo di componenti Open Source con lo stesso livello di affidabilità tradizionalmente offerto dal software proprietario. L'iniziativa è un'unica alleanza europea brasiliana e cinese tra operatori del settore delle TIC, le PMI, i governi e gli accademici per aiutare industrie e governi a migliorare l'innovazione e la competitività tramite l'utilizzo di software libero Open Source. Inoltre è una delle principali iniziative Free Open Source finanziate dalla Commissione Europea, ed è finanziato nell'ambito del sesto programma quadro dell'UE (6PQ)[5], nel settore delle iniziative tecnologiche della società dell'informazione (TSI). QualiPSo è lanciato in sinergia con le iniziative per l'Europa e tecnologie come NESSI[7] e Artemide[8].

2.3.1 QualiPSo e l'Open Source

L'obiettivo della ricerca di QualiPSo è l'identificazione, quantificazione e valutazione dei fattori di qualità che influenzano la fiducia nei prodotti Open Source Software (OSS). Questa ricerca vuole identificare i prerequisiti e i requisiti che permettono di stabilire le prove comuni per testare la qualità, gli approcci di riferimento e definire una metodologia per la valutazione della affidabilità dei prodotti OSS.

Questa metodologia comprenderà definizioni di metriche, pratiche di misurazione, analisi dei dati, definizione di una suite di test per l'analisi comparativa delle prestazioni, e calcolo degli indicatori. In questa tesi andremo ad utilizzare i modelli di qualità che QualiPSo sta creando a questo scopo, useremo quindi il modello MOSST (*Model for Open Source Software Trustwothiness)*.

Un vasto insieme di strumenti Open Source di supporto alla valutazione di affidabilità sarà realizzato e integrato in una piattaforma affidabile per sostenere lo sviluppo di questi tipi di programmi. Questi strumenti copriranno tutti gli aspetti della qualità del software Open Source e saranno basati su un'architettura aperta. Per raggiungere questo obiettivo il progetto QualiPSo promuoverà la nascita di competence center in Europa, Brasile e Cina.

2.4 FlossItaly

Il Centro di Competenza Italiano per l'Open Source è una iniziativa promossa da Engineering Ingegneria Informatica , la Libera Università di Bolzano, l'Università dell'Insubria e l'Università del Sannio per la realizzazione di servizi volti a facilitare l’adozione di software Open Source nella Pubblica Amministrazione[9].

L’operatività del Centro nasce sotto l'egida del Ministero per la Pubblica Amministrazione e l'Innovazione che, in data 30 Marzo 2010, ha firmato uno specifico Protocollo d'Intesa con Engineering.

Il Centro promuove i risultati tecnico-scientifici del progetto QualiPSo, co-finanziato dalla Commissione Europea. Ha la forma giuridica di un Consorzio “no-profit no-loss” ed opera su scala nazionale per assistere le Scuole, le Università, i piccoli Comuni, il settore sanitario/ospedaliero e le piccole e medie imprese, nella individuazione e nell’adozione di soluzioni Software Open Source, capaci di soddisfare pienamente le diverse esigenze di natura applicativa.

L'Obiettivo

Obiettivo del Centro è quello di consentire a fruitori di servizi e prodotti informatici per la produttività individuale e piccoli/medi business, che risentono in modo particolarmente intenso delle continue contrazioni della loro capacità di spesa, di continuare ad avere software di buona qualità, aggiornato e sicuro per gestire le loro attività quotidiane senza essere penalizzati dai costi legati alle licenze. Essi, inoltre, devono avere la possibilità di personalizzare i software di cui hanno necessità in base alle loro esigenze indipendentemente dai vincoli imposti dal software proprietario a pacchetto.

Per tali utenti, la possibilità di avvalersi delle competenze offerte dal CC italiano per supporto, consulenze ed addestramento rappresenta una grande risorsa.

Al fine dunque di promuovere efficacemente un ricorso sempre più diffuso all'adozione di software OS, il Centro di Competenza mette a disposizione i seguenti servizi:

- Consulenza e supporto tecnico sulle tecniche e la gestione di soluzioni Open Source con particolare enfasi ai processi di governance e gestione dei processi di misurazione della Qualità
- Servizi per la misurazione della qualità di prodotto e processo sottostante lo sviluppo di progetti Open Source
- Formazione sui temi più significativi legati all'Open Source con particolare enfasi ai processi di governance e gestione dei processi di misurazione della Qualità
- Sensibilizzazione e promozione nell'adozione delle soluzioni Open Source attraverso giornate di divulgazione gratuite svolte principalmente presso scuole ed università al fine di sensibilizzare la base dei futuri utenti circa l''importanza dell'adozione dell'Open Source e sui vantaggi che questo offre.[10]

3 Ambiente di integrazione

L'obiettivo è realizzare un ambiente funzionante che permetta ad un utente di eseguire le analisi necessarie allo scopo di verificare la bontà di un determinato software obiettivo. Per questo scopo è stato necessario inizialmente creare un ambiente adeguato per l'installazione e la convivenza degli strumenti utilizzati per le analisi. Questi strumenti hanno prodotto dei risultati che sono stati inseriti nel data warehouse di Spago4Q, e che sono stati aggregati secondo il modello predefinito per la visualizzazione e interpretazione dei risultati.

Al termine delle analisi gli strumenti hanno fornito i risultati attraverso diversi formati supportati da Spago4Q. Ad esempio sono stati forniti dati sotto forma di file XML e tabelle di database, che tramite apposite interfacce è stato possibile importare nel data warehouse. Un ulteriore modo di ricavare ed inserire dati nel data warehouse è la creazione di appositi estrattori personalizzati, che permettono di estrapolare informazioni da sorgenti che non sono solitamente accessibili utilizzando gli estrattori standard, questo dimostra una grande flessibilità di Spago4Q.

Un chiaro esempio dell'ambiente che andremo a creare è rappresentato nella immagine seguente :

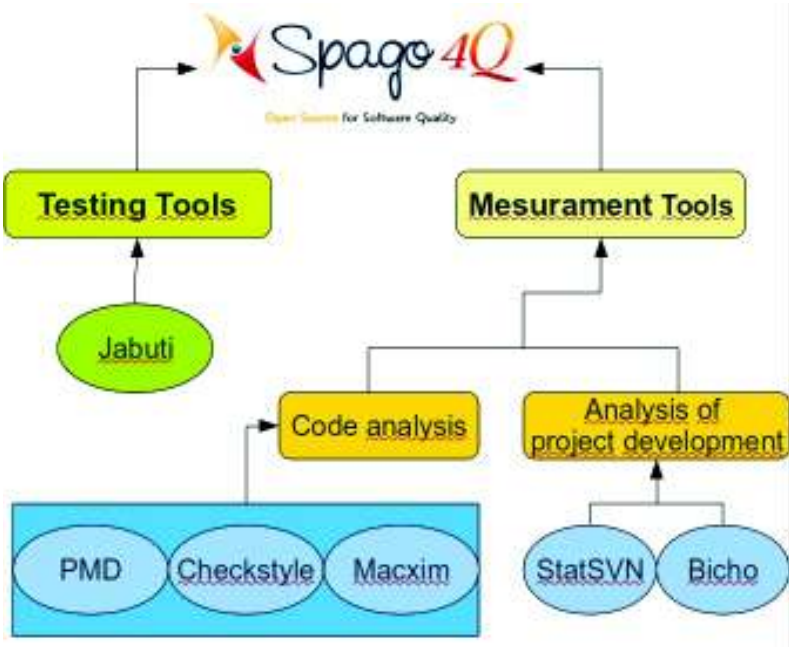

Illustrazione 4: Struttura dell'ambiente di integrazione

3.1 Spago4Q

Spago4q è una soluzione che dà una visione globale ed immediata degli indicatori di qualità di un prodotto software o di un processo di sviluppo. Questi indicatori vengono visualizzati in forma strutturata ed in riferimento a degli specifici modelli di qualità. Nel dettaglio una volta identificate le sorgenti dati e i calcoli per ottenere gli indicatori di qualità Spago4Q può essere applicato per estrarre, calcolare e visualizzare questi indicatori in modo indipendente dal contesto analizzato. I dati da analizzare provengono direttamente dagli strumenti utilizzati nella attività quotidiana di sviluppo di prodotto o di gestione del processo o di un servizio.

Una volta estratti, questi dati vengono caricati all'interno del data warehouse di Spago4q secondo delle opportune interfacce dati che consentono di mantenere separata la fase di estrazione dalla fase di analisi in modo che in un secondo momento possono essere sostituite le sorgenti dati mantenendo inalterate le fasi di analisi.

Spago4Q viene anche utilizzato per valutare la qualità di prodotti software e di processi utilizzati per il loro sviluppo. Per fare questo raccoglie le informazioni e i dati analizzati da tool esterni e li aggrega secondo dei modelli definiti, nel nostro caso utilizzeremo una parte del modello per la qualità dei progetti Open Source di QualiPSo.

Una componente essenziale di Spago4Q è il modulo di ETL (Extract Transform Load) che estrae i dati dai tools di infrastruttura e popola il data warehouse di Spago4Q, implementato sulla base di un sofisticato meta-modello[11]. Il modulo analitico è una verticalizzazione di SpagoBI che soddisfa tutti i requisiti di una soluzione di BI (Business intelligence) come analisi e gestione dei dati, amministrazione e sicurezza.

L'uso di SpagoBI semplifica la rappresentazione di KPI, metriche e relative soglie con differenti tipologie di documenti analitici (report, OLAP, dashboard, data mining, Query By Example).

Spago4Q comprende un set configurabile di dashboards, reports e OLAP per monitorare e valutare nell'ambito di un processo di sviluppo sotto processi quali: gestione requisiti, tracking di problemi e anomalie, test, risk management, controllo delle versioni.

Tutti i moduli sono configurabili mediante funzionalità di Configurazione ed Amministrazione quali: configurazione degli estrattori specializzati per la raccolta dati dai tools di infrastruttura predefiniti, controllo degli accessi, gestione dei modelli gerarchici di KPI, gestione algoritmi e soglie di riferimento per KPI.

L'architettura modulare di Spago4Q e la progettazione del meta-modello garantiscono estensibilità al sistema permettendo future integrazioni di nuovi tool di infrastruttura e differenti set di modelli, KPI e misure.

Illustrazione 5: Architettura Spago4Q

3.2 Meccanismi di Importazione

Illustrazione 6: Importazione dei dati in Spago4Q

Nell'illustrazione 5 è rappresentata l'architettura della piattaforma Spago4Q, i cui componenti principali che ne costituiscono l'infrastruttura sono:

Estrattori:

Procedure per l'estrazione dei dati di interesse dagli strumenti utilizzati in fase di sviluppo. Gli estrattori, che come possiamo vedere dall'illustrazione 6 possono essere sviluppati con diverse tecnologie, sono strumenti specializzati per l'estrazione da diverse sorgenti, in grado di caricare contemporaneamente più interfacce. Inoltre possono essere creati più estrattori per il medesimo strumento e ciascuno di questi può applicare determinate regole dinamiche per il filtraggio o la trasformazione dei dati raccolti in modo da renderli compatibili con le interfacce. Per questo scopo siamo ricorsi all'uso di un estrattore per recuperare le metriche di statSVN.

Interfacce:

Definiscono il formato e la tipologia dei dati utilizzati dai componenti analitici e nel calcolo dei KPI. Permettono così di creare una separazione tra gli estrattori e la logica analitica. Inoltre per ogni area di misura viene definita soltanto un'interfaccia per unificare il formato dei dati raccolti dai diversi strumenti, o progetti, e tale formato può essere definito utilizzando le funzionalità amministrative. Interessante è il fatto che sia possibile sviluppare in maniera semplice degli estrattori per integrare qualsiasi formato di dato proveniente da strumenti non ancora integrati con Spago4Q.

Data Warehouse:

Deposito per tutti i dati ricavati dai progetti sviluppati da un'organizzazione. Questa parte del database è identificata con le tabelle il cui nome inizia per #FT_# , che rappresentano le interfacce ossia quelle in cui vengono memorizzati i dati raccolti, oppure #DT_# per le tabelle utilizzate per descrivere l'ambiente o come fonti da cui raccogliere dati.

Strumenti analitici:

Analizzano i dati e forniscono le rappresentazioni dei KPI. Attraverso il componente analitico, infatti, i dati inseriti nel data warehouse possono essere analizzati e successivamente utilizzati per la costruzione di documenti analitici. Questo modulo è stato sviluppato all'interno di SpagoBI, in modo da sfruttare una serie di servizi già presenti all'interno di quella soluzione. Utilizzare la piattaforma SpagoBI per implementare il componente analitico di Spago4Q semplifica la rappresentazione dei KPI, delle metriche e delle relative soglie come istanze dei documenti analitici già presenti all'interno di SpagoBI (report, dashboard, data mining e interrogazioni libere).

Moduli di configurazione e amministrazione:

Indispensabili per definire le impostazioni del sistema. Attraverso questi moduli infatti possono essere configurati tutti gli altri componenti in quanto permettono:

- gestione dei modelli, degli indicatori KPI e dei valori delle soglie di riferimento
- controllo dei processi di estrazione che raccolgono i dati utilizzando gli estrattori e li adattano attraverso le interfacce
- configurazione delle interfacce, che consentono la definizione delle strutture dati che devono immagazzinare le informazioni estratte dagli strumenti di sviluppo [12]

3.3 Modelli di qualità

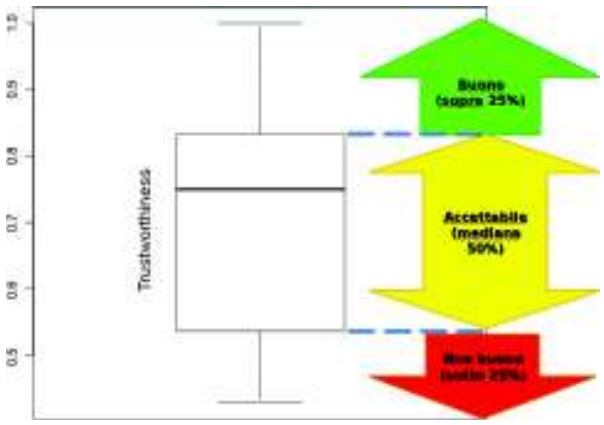

Illustrazione 7: Livelli di accettazione

3.3.1 Model for Open Source Software Trustwothiness (MOSST)

Questo modello nasce per rispondere all'esigenza di un metodo per valutare la qualità dei progetti Open Source. È un Modello composto da un gruppo di stimatori statistici di significatività (MOSST: Model of Open Source Software Trustworthiness) e serve per fornire un'indicazione ad alto livello della qualità percepita del software. Il modello MOSST è composto da un sottoinsieme di metriche, in particolare tutte quelle che possano essere rilevate in maniera autonoma da specifici tool per valutare la bontà di prodotto.

È un modello ancora in via di definizione presso QualiPSo quindi non abbiamo ancora informazioni definitive su come sarà composto in futuro. I livelli di accettazione al momento sono forniti dagli stessi sviluppatori dei tool per verificarne il corretto funzionamento, quindi al momento non possiamo ancora interpretare i risultati ottenuti dalle analisi come veritieri, poiché non è stato ancora definito come questi livelli debbano essere calcolati per valutare la qualità di ogni singola metrica.

3.3.2 Open Maturity Model (OMM)

L'Open Source Maturity Model (d'ora in poi abbreviato in OMM) di QualiPSo è un modello formale di valutazione del livello di maturità che si concentra sulla qualità dei processi e sul miglioramento, e solo indirettamente sulla qualità del prodotto che comunque è strettamente legato alla qualità del processo. Il modello è attualmente in via di definizione presso QualiPSo. Attualmente siamo nella fase iniziale del processo che avrà come obiettivo raggiungere una piena maturità e riconoscibilità del modello, universalmente accettata. Le metriche che il modello utilizza non possono essere rilevate tutte automaticamente tramite meccanismi di analisi, molte di loro vanno quindi raccolte tramite questionari. I questionari serviranno per raccogliere le opinioni degli "addetti ai lavori" sulla importanza dei diversi aspetti, percepiti come utili dagli sviluppatori per valutare la qualità di un determinato progetto.

3.4 Strumenti utilizzati

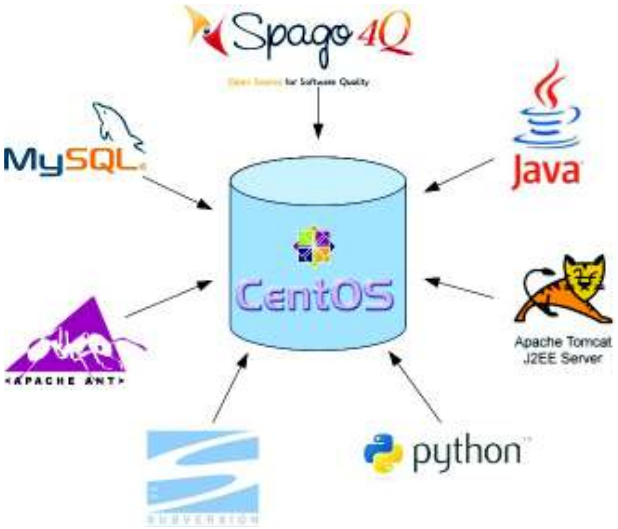

Illustrazione 8: Strumenti utilizzati nell'ambiente di produzione

Gli strumenti utilizzati sono tutti Open Source, a partire dal sistema operativo fino ad arrivare ai tool. Come sistema operativo si è usato una distribuzione Debian per la fase di test, ed una distribuzione CentOS per la macchina virtuale definitiva che l'azienda andrà ad utilizzare. Come Browser di riferimento è stato usato Mozilla Firefox, utilizzato per interagire con l'interfaccia web di Spago4Q. Quest'ultimo è scritto principalmente nel linguaggio Java e fa largo uso di database Mysql. L'accesso ai repository software è stato possibile grazie a Subversion (svn).

3.5 Preparazione dell'ambiente

La preparazione dell'ambiente consiste nel creare un ecosistema di applicazioni non in contrasto fra di loro, cercando di ottimizzare le risorse a nostra disposizione. Visto che l'ambiente d'uso dei tool è basato su GNU/Linux, andremo come prima cosa a installare tutte le dipendenze necessarie per l'esecuzione dei tool di analisi, a questo scopo è importante accertarsi di avere un accesso alla rete e i diritti necessari per l'esecuzione delle operazioni richieste. Per le analisi che andremo a fare siamo ricorsi all'uso di diversi strumenti Open Source come mostrato nell'illustrazione 8.

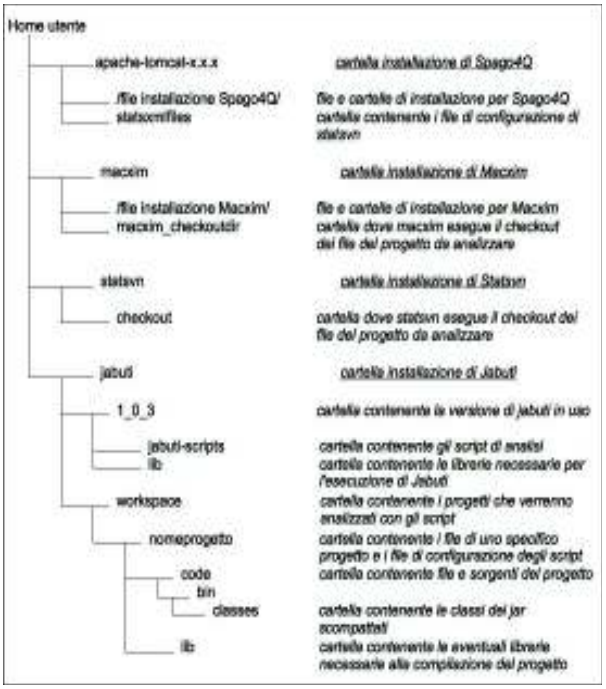

Illustrazione 9: Struttura delle cartelle scelta

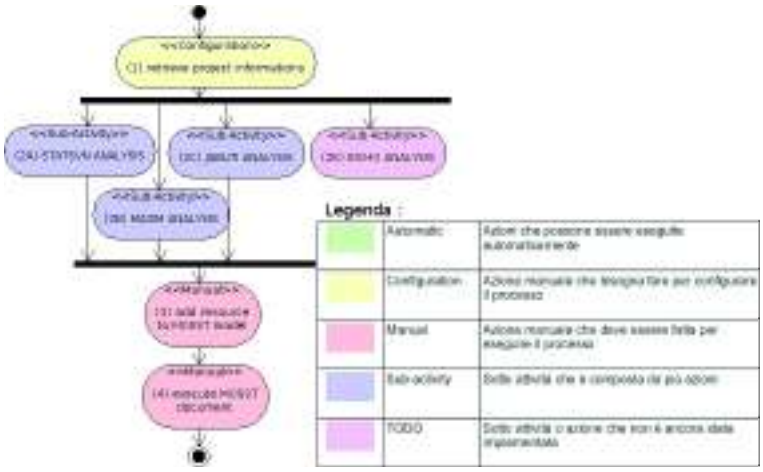

Illustrazione 10: Procedure di analisi

Creazione risorsa in Spago4Q

Prima di iniziare le analisi è stata creata la risorsa in Spago4Q, la quale riceverà in ingresso tutti i risultati dei diversi tool di analisi che utilizzeremo. Per creare la risorsa in Spago4Q si deve andare in Gestine KPI → Risorse → Inserire una nuova risorsa facendo attenzione a rispettare lo standard del nome in questo modo: nomeprogetto-versione-revisione, ad esempio: Spago4Q-2.3-776 . Una volta creata la risorsa dobbiamo renderla attiva, quindi spostiamoci in Gestione KPI → Istanza di Modello → MOSST (resources) e attivare la risorsa .

3.6 MacXim

MacXim[13] è uno strumento di analisi statica del codice che mira a scansionare il codice sorgente Java ed estrarre un Abstract Syntax Tree semplificato (AST) rappresentante il codice sorgente (in formato XML), insieme ad alcune misure a bassi livelli relativi al codice sorgente analizzato. L'AST dei file di origine vengono memorizzati in un database XML. Il database può essere interrogato tramite Xpath query, o con query complesse attuate direttamente in Java. I risultati della query sono in formato XML e possono essere archiviate nel database o scartati immediatamente. Le query possono avere scopi diversi: possono essere eseguiti nei confronti di un singolo progetto, diversi progetti differenti nello stesso momento, o diverse versioni dello stesso progetto. Il vantaggio principale di questo approccio, rispetto ad approcci simili, consiste nella separazione delle due attività che si trovano di solito incollati insieme:

- L'analisi del codice sorgente
- Calcolo delle misure di software.

Questo può essere un bel vantaggio, dal momento che mentre l'analisi del codice sorgente è più o meno lo stesso in molti strumenti diversi, le misure di software possono variare ampiamente.

Illustrazione 11: Struttura Macxim

3.6.1 Features

L'idea di MacXim è quella di supportare diversi tipi di misure, permettendo così di calcolare sia metriche native sia metriche di strumenti esterni. Attualmente si possono gestire solo due strumenti esterni: PMD e Checkstyle. Il primo è in grado di misurare alcune caratteristiche riguardo la complessità del software obiettivo, il secondo è in grado di misurare alcune caratteristiche relative allo stile di scrittura del codice sorgente del software analizzato.

3.6.2 Metriche

MacXim è un software modulare che svolge il suo lavoro in un processo di pipeline. A partire dal codice sorgente utilizzando il compilatore Eclipse, si ottiene una rappresentazione dell'Abstract Syntax Tree (AST), che è una rappresentazione logica del primo codice sorgente. In questo primo processo, MacXim esegue una sorta di filtro che rifiuta alcune informazioni presenti nel codice sorgente iniziale. Ad esempio le linee dei commenti non sono mantenute. Le informazioni dell'AST sono strutturate in un file XML e conservati in un database locale XML (eXist). Nel database eXist c'è una struttura gerarchica delle informazioni: una raccolta per le informazioni AST e una per gli script XQuery. Quest'ultimi sono alcuni script scritti nel linguaggio XQuery (un linguaggio standard per lavorare con dati XML) per calcolare le metriche utilizzando le informazioni AST.

La fase finale è l'esecuzione delle metriche che utilizza Xquery, la MacXim Runtime Engine calcola tutti i parametri disponibili e inserisce i risultati in un database MySQL server. Il calcolo è effettuato per ciascun livello di granularità supportato[7].

Grazie ad un file .ini di configurazione, l'utente può scegliere quale database server vuole utilizzare. In primo luogo, MacXim calcola le metriche native ed in seguito calcola le metriche dei tool esterni ed inserisce i risultati nel database MySQL.

7 Al momento Macxim supporta solo livelli di granularità "Application" e "Package"

3.6.3 Integrazione con Spago4Q

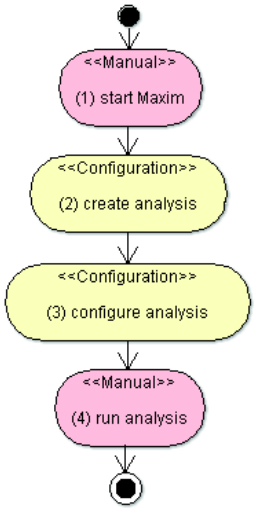

Illustrazione 12: Processo di analisi di Macxim

Processo nuova analisi

Come possiamo vedere dall'illustrazione 12, una nuova analisi tramite MacXim comporta ancora diverse fasi manuali di configurazione ed esecuzione da parte dell'utente di alcuni processi. In particolare non è al momento possibile pianificare un'analisi automatica e le fasi di creazione e configurazione (punti 2,3,4) vanno eseguite ripetutamente ad ogni nuova analisi. Al punto 4 è giusto specificare che le metriche in questo caso vengono calcolate prendendo i dati direttamente dal database di MacXim, senza l'utilizzo di un particolare estrattore.

Lista metriche Fornite

M4.1.1.1	Comment Lines up to eLOC	[CLeLOC-0]

M4.3.1.2	CBO per class	[CBO-0]
M4.3.1.3	LCOM per class	[LCOM-0]
M4.3.4	McCabeIndex	[McCabe-0]

M4.4.1	eLOC per class	[eloc-0]
M4.4.2	numParametersPerMethod	[NPM-0]
M4.4.3	numPackages	[NP-0]
M4.4.4	numMethodsPerInterface	[NMPI-0]
M4.4.5	numMethodsPerClass	[NMPC-0]
M4.4.6	commentLinesPerClass	[CLPC-0]
M4.4.7	Number of Attrubutes per Class	[NAPC-0]
M4.4.8	Number of Classes	[NC-0]
M4.4.9	Number of Interfaces	[NI-0]

M4.9.1	num Interfaces per Class	[NIPC-0]
M4.9.2	num Methods per Interface	[NMPI-0]
M4.9.3	num classes with defined attr...	[NCDA-0]
M4.9.10	num classes with defined meth...	[NCDM-0]

3.7 JaBUTi

JaBUTi (Java Bytecode Understanding and Testing) è progettato per lavorare con il bytecode Java in modo che non sia necessario il codice sorgente per svolgere le sue attività. E' composto da uno strumento di analisi della copertura, da uno strumento di taglio e da un complesso strumento di misura delle metriche. Lo strumento di copertura può essere utilizzato per valutare la qualità di un determinato insieme di test o per generare insiemi di test basati su differenti criteri di controllo di flusso (control-flow) e flusso di dati (data-flow). Lo strumento di taglio (slicing tool) può essere utilizzato per identificare le aree del codice soggette ad errori, rendendolo utile per il debugging e per la comprensione del programma. Il complesso strumento di misura delle metriche può essere utilizzato per identificare la complessità e la dimensione di ogni classe testata sulla base delle informazioni statiche.

3.7.1 Features

JaBUTi implementa un sottoinsieme delle funzionalità fornite da xSuds, una suite completa di strumenti per C e C + +. L'interfaccia grafica permette di esplorare il controllo di flusso (control-flow) e le prove sul flusso di dati (data-flow). Inoltre, fornisce un modo per visualizzare quali parti delle classi testate sono già coperte e quali no. Nell'illustrazione 13 possiamo vedere una vista generale dell'interfaccia grafica di JaBUTi, compreso il suo menù.

Illustrazione 13: Jabuti GUI

3.7.2 Metriche

LK's Metrics Applied to Classes

NPIM	Number of public instance methods in a class
NIV	Number of instance variables in a class
NCM	Number of class methods in a class
NCV	Number of class variables in a class
ANPM	Average number of parameters per method
AMZ	Average method size
UMI	Use of multiple inheritance
NMOS	Number of methods overridden by a subclass
NMIS	Number of methods inherited by a subclass
NMAS	Number of methods added by a subclass
SI	Specialization index

CK's Metrics Applied to Classes

NOC	Number of Children
DIT	Depth of Inheritance Tree
WMC	Weighted Methods per Class
LCOM	Lack of Cohesion in Methods
RFC	Response for a Class
CBO	Coupling Between Object

Another Metrics Applied to Methods

CC	Cyclomatic Complexity Metric

3.7.3 Integrazione con Spago4Q

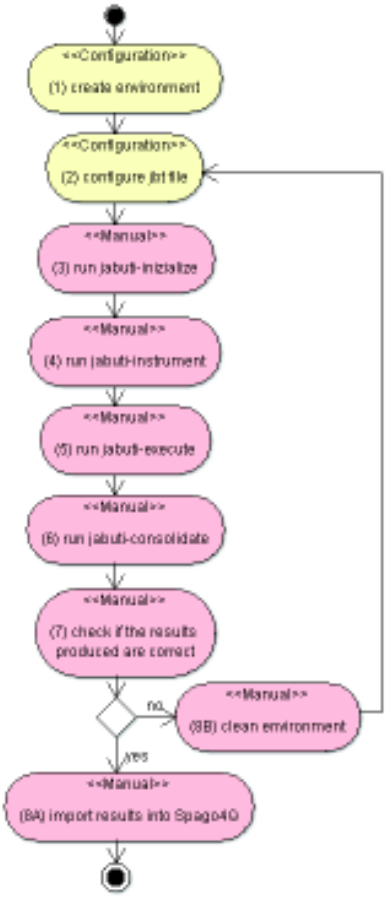

Illustrazione 14: Processo di analisi di Jabuti

Processo nuova analisi

Jabuti è sicuramente il tool che ha richiesto la maggior attenzione da parte dell'utente. Per prima cosa è stato necessario creare un ambiente favorevole all'esecuzione degli script di analisi, poiché abbiamo registrato ben poca elasticità da parte del tool nella personalizzazione delle cartelle di lavoro (punto 1). Successivamente si è passati alla configurazione del file contenente le indicazioni per effettuare l'analisi (punto 2). Questa parte è molto delicata e l'utente deve conoscere molto bene le caratteristiche del software obiettivo per compilare adeguatamente i campi necessari a Jabuti. Sempre manualmente sono stati eseguiti i quattro script (punti 3 4 5 6) che hanno fornito i risultati come un file XML da caricare manualmente tramite l'interfaccia Direct Metric (punto 8A) in Spago4Q. Per procedere ad una nuova analisi è stato necessario scrivere uno script che agevoli la rimozione di tutti i file superflui creati da Jabuti nella fase di analisi (punto 8B).

Lista Metriche fornite

M3.1.2.1	All-Edges-ei
M3.1.2.2	All-Edges-ed
M3.1.3.1	All-Nodes-ei
M3.1.3.2	All-Nodes-ed
M3.1.4.1	All-Uses-ei
M3.1.4.2	All-Uses-ed
M3.1.4.3	All-Pot-Uses-ei
M3.1.4.4	All-Pot-Uses-ed

3.8 StatSVN

StatSVN è una applicazione Java rilasciata con licenza LGPL che recupera informazioni da un repository Subversion e genera varie tabelle e grafici che descrivono l'evoluzione del progetto, ad esempio:

- quando è stato generato il report
- ultima revisione
- su che periodo si basa
- numero totale di file
- numero totale di linee di codice
- numero di sviluppatori
- un collegamento a Twitter
- un menu che mostra maggiori dettagli
- il grafico sommario delle linee di codice
- la tag cloud delle parole usate nei messaggi di log
- la struttura gerarchica delle directory, dove per ognuna è mostrato il numero di file e le linee di codice contenuti.

Nel menu dei dettagli si può vedere ad esempio, quante linee di codice ha scritto un particolare autore, il numero di commit per ora e giorno della settimana, i log dei commit, che tipi di file ci sono ordinati per loro numero (1o file php, 4 xml ecc.) e quante linee di codice in media ci sono per ogni tipo, lista dei file più grandi e più revisionati, tabella e grafico a torta delle dimensioni delle cartelle e tanto altro.

3.8.1 Features

StatSVN è una applicazione usabile via riga di comando in grado di generare report HTML o XML. Questi rapporti coprono una serie di qualità, che possono essere suddivisi nei seguenti gruppi :

- metriche correlate a file e directory
- metriche correlate alle attività degli sviluppatori

3.8.2 Metriche

Approfittando dei report e delle statistiche generate, StatSVN può calcolare le seguenti metriche :

Numero medio di linee modificate per anno
Numero di major releases per anno
Numero di minor releases per anno

Nel modello MOSST, queste misure sono utilizzate per valutare la sezione definita come "New Release Rate". Tuttavia, i parametri di cui sopra non possono essere ottenuti direttamente dai report, ma esiste la possibilità di calcolarli manualmente uilizzando le seguenti statistiche come base :

linee di codice aggiunto
linee di codice rimosso
linee di codice modificate
velocità di rilascio delle Major e Minor release basato sulla lista di tag

3.8.3 Integrazione con Spago4Q

L'estrazione dei dati e la raccolta manuale è un compito noioso. Pertanto StatSVN integra questi strumenti e li combina con Spago4Q attraverso un estrattore. Calcola anche i valori delle metriche, fornendo i risultati finali come parte dei report stilati da Spago4Q. StatSVN è in grado di fornire le seguenti metriche:

REG-Mjr-REL-PER-Y
REG-MINOR-REL-PER-Y
REG-LOC-CH-PER-Y

Tra gli altri strumenti (ad esempio, Maxcim, Jabuti), StatSVN è integrato con la piattaforma Spago4Q attraverso la propria interfaccia di programmazione. L'integrazione si avvale di un'interfaccia di Spago4Q detta estrattore che permette di gestire il processo di esecuzione degli strumenti integrati.

L'architettura complessiva dell'integrazione si presenta come nell'illustrazione seguente :

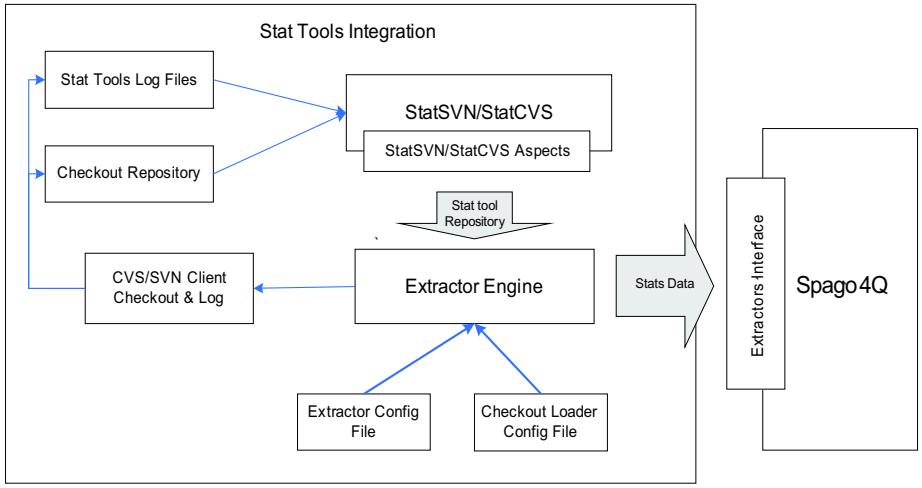

Illustrazione 15: Schema dell'integrazione di StatSVN

Gli strumenti di misura sono integrati con la piattaforma Spago4Q. Esse forniscono dati attraverso l'interfaccia dell'estrattore per coprire i parametri inclusi nel modello GQM. Il calcolo delle metriche è gestito da meccanismi della piattaforma Spago4Q così come la visualizzazione del modello GQM.

Procedura nuova analisi

StatSVN viene fornito come estrattore da integrare in Spago4Q. Come vediamo dall'illustrazione 16, una volta installato e configurato per una corretta integrazione con la piattaforma, le uniche attività che l'utente dovrà compiere sono configurare il file per istruire il tool su cosa dovrà compiere l'analisi e successivamente lanciare l'analisi direttamente da Spago4Q. I risultati verranno integrati direttamente nel data warehouse e saranno disponibili fin da subito nel report del modello associato alle metriche rilevate. Essendo un estrattore di Spago4Q può avvalersi di alcune caratteristiche utili al nostro scopo, come la schedulazione del processo di analisi.

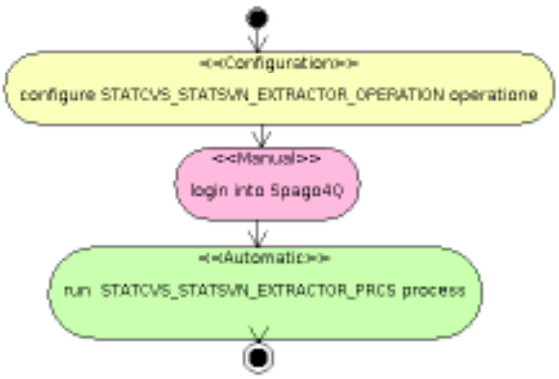

Illustrazione 16: Processo di analisi di StatSVN/StatCVS

3.9 Bicho

3.9.1 Introduzione a Bicho

Bicho è uno strumento da riga di comando sviluppato in Python utilizzato per analizzare i sistemi di bug tracking. L'obiettivo è quello di ricavare dati da svariati sistemi di bug tracking in maniera automatica e memorizzarli in un database per poter essere poi rielaborati. Bicho popola principalmente tre tabelle con tutti i dettagli relativi ai bugs, i commenti collegati ad essi ed informazioni su eventuali allegati trovati. Attualmente il software è ancora in via di sviluppo e presenta diversi bugs. Al momento è possibile ricavare informazioni da bug tracker come SourceForge[8], ed i Bugzilla di Gnome, Kde e Apache.

3.9.2 Features

Bicho estrae le informazioni dei bugs come ad esempio: id, priority, resolution, status, comments, descriptions, changes, attachments ed altro e inserisce il tutto in un database che ci consentirà di usare quelle informazioni per studi e analisi. I processi di analisi sono in fase di sviluppo, in particolare per quanto concerne Spago4Q è previsto l'implementazione di un servizio REST[9] che permetta di ricavare le informazioni dal database in un formato comprensibile a Spago4Q (in questo caso xml) per poi lasciare a lui il compito di elaborare le informazioni. Un esempio del servizio REST che verrà implementato per spago4q lo si può trovare online nel sito[14] del progetto.

3.9.3 Metriche

L'attuale stato di sviluppo di Bicho non ci ha permesso di importare i risultati nella nostra piattaforma poiché gli sviluppatori non hanno ancora definito delle procedure per l'estrazione di metriche dai dati raccolti e quindi l'integrazione con Spago4Q.

3.9.4 Integrazione con Spago4Q

L'integrazione di Bicho con Spago4Q è in via di sviluppo e sarà fornita tramite un servizio REST che restituisca un file xml contenente le metriche richieste dal modello OMM. Come si può vedere dall'illustrazione 17, durante la fase di test si è riusciti ad eseguire l'analisi e importare i dati raccolti nel database ma non è ancora possibile ricavare le metriche da essi (punto 3).

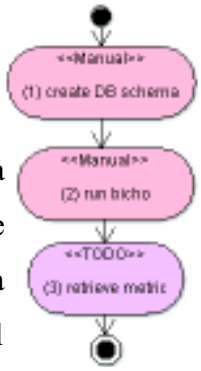

Illustrazione 17: Processo di analisi di Bicho

8 Utilizza parsing HTML

9 *Representational State Transfer*. È un servizio che si basa su un protocollo di comunicazione stateless, client-server, chacheable e si appoggia sul protocollo HTTP.

4 Casi d'uso

4.1 Log4J

Log4j è un framework che permette di utilizzare il meccanismo dei log all'interno di un'applicazione in modo semplice e non invasivo. Il log è un'informazione che può essere utilizzata per effettuare un'analisi in caso di errore, per il ripristino di situazioni precedenti, per verificare se alcune operazioni sono state eseguite da chi o cosa, o più in generale, per riassumere quanto accaduto in un determinato arco temporale. Log4j è divenuto molto popolare tra gli sviluppatori Java ed è distribuito sotto licenza della Apache Fundation, quindi completamente Open Source. Nel nostro caso analizzeremo una determinata versione del software prelevata da svn : Log4J. v1.2.15 revisione 569610 poiché ci è stata utile per calibrare il nostro sistema di analisi verificando di ottenere gli stessi risultati riscontrati dai partner e dagli stessi sviluppatori nelle loro fasi di test.

4.1.1 Visualizzazione risultati

Nell'illustrazione seguente vediamo la visione generale che Spago4Q ci fornisce con il suo report sul modello MOSST. Nella prima parte si può vedere come Jabuti ci spieghi che i test presenti all'interno del codice non vadano a coprire la maggior parte di esso. Possiamo interpretare quei valori come percentuali di area coperta dai test per ogni metrica. Questi livelli possono suggerire ad uno sviluppatore che c'è la necessità di inserire una maggior quantità di test per garantire più sicurezza. Passiamo poi alla sezione dedicata alle caratteristiche del codice, si noti come la quantità di commenti nelle linee di codice sorgente sia molto alta. Dalle metriche che descrivono le caratteristiche del codice notiamo che c'è una buona complessità e stile di scrittura generale, tutto ciò può indicare una buona qualità tecnica del software. L'analisi del repository utilizzato dagli sviluppatori ci mostra come siano davvero pochi i rilasci annuali del software, sembra infatti che non ci siano stati rilasci nell'ultimo anno.

MOSST

Model of Open Source Software Trustworthiness

RESOURCE: log4j-1.2.15R569610

MODEL	KPI	KPI CHART
A5-GQM - MOSST		
QF3 - Actual Reliability		
Q3.1 - Correctness		
M3.1.2 - TestConditionCoverage	18.88	
M3.1.2.1 - All-Edges-ei	18.3	
M3.1.2.2 - All-Edges-ed	0.58	
M3.1.3 - TestInstructionCoverage	24.41	
M3.1.3.1 - All-Nodes-ei	20.96	
M3.1.3.2 - All-Nodes-ed	3.47	
M3.1.4 - TestPathCoverage	42.73	
M3.1.4.1 - All-Uses-ei	18.21	
M3.1.4.2 - All-Uses-ed	3.68	
M3.1.4.3 - All-Pot-Uses-ei	16.81	
M3.1.4.4 - All-Pot-Uses-ed	4.04	
QF4 - Actual Exploit in dev Maintainability		
Q4.1 - Analyzability For Maintenance		
Q4.1.1 - Code Documentation		
M4.1.1.1 - Comment Lines up to eLOC	0.86	
Q4.3 - Code Quality		
M4.3.3 - ECARules		
M4.3.3.10 - Duplicated Code		
M4.3.3.11 - Missing Braces If Stmts		
M4.3.3.12 - Missing Braces While Loops		
M4.3.3.13 - Missing Braces If Else Stmts		
M4.3.3.14 - Missing Braces For Loops		
M4.3.3.15 - Field Never Initialized Properly		
M4.3.3.16 - Excessive Method Length		
M4.3.4 - McCabe Index		
M4.3.5 - Chidamber Kemerer		
M4.3.5.2 - CBO per class	34.0	
M4.3.5.3 - LCOM per class	66.0	
Q4.4 - Code Size		
M4.4.1 - eLOC per class	77.27	
M4.4.2 - numParametersPerMethod	0.76	
M4.4.3 - numPackages	40.0	
M4.4.4 - numMethodsPerInterface	2.48	
M4.4.5 - numMethodsPerClass	6.86	
M4.4.6 - commentLinesPerClass	66.82	
M4.4.7 - Number of Attributes per Class	3.52	
M4.4.8 - Number of Classes	303.0	
M4.4.9 - Number of Interfaces	24.0	
Q4.7 - New Release Rate		
M4.7.2 - PRJ-LOC-CH-PER-Y	238.0	
M4.7.3 - PRJ-MJR-REL-PER-Y	0.0	
M4.7.4 - PRJ-MINOR-REL-PER-Y	0.0	
Q4.9 - Code Modularity		
M4.9.01 - num Interfaces per Class	0.27	
M4.9.02 - num Methods per Interface	2.48	
M4.9.03 - num classes with defined attributes (perc)	88.12	
M4.9.10 - num classes with defined methods (perc)	118.49	
QFS - Actual Testability		
QFS.1 - Test Size		
MS.1.1 - Number of test cases		
QFS.2 - Test Result Availability		
MS.2.1 - Successful rate of test cases		

Illustrazione 18: Risultati Log4J

4.2 Spago4Q

Andremo ora ad analizzare lo stesso software che ci ha permesso di creare questo ambiente per l'analisi della qualità dei progetti Open Source. Si è scelto di analizzare l'ultima versione rilasciata per fornire delle informazioni utili agli sviluppatori sulla qualità del loro software e verificare che i risultati combacino con quelli ottenuti da altri strumenti di test che sono stati usati in precedenza per valutare la qualità del codice di Spago4Q.

4.2.1 Visualizzazione risultati

Dai risultati riportati nell'illustrazione seguente possiamo notare come Jabuti ci mostri che ci sono dei problemi con i test interni al codice, sembra infatti che per alcune categorie di test non siano stati previsti (indicato dai valori dei KPI nulli). A seguire troviamo tutte le metriche ricavate dai tool Macxim e StatSVN, anche se purtroppo non è ancora possibile fare delle valutazioni sui livelli ricavati dall'analisi poiché le soglie di valutazione dei KPI non sono ancora state definite.

Illustrazione 19: Dettaglio andamento di una metrica nel tempo

MOSST

Model of Open Source Software Trustworthiness

RESOURCE: Spago4Q-2.3R776

MODEL	KPI	KPI CHART
A5-GQM - MOSST		
QF3 - Actual Reliability		
Q3.1 - Correctness		
M3.1.2 - TestConditionCoverage	17.5	
M3.1.2.1 - All-Edges-ei	17.5	
M3.1.2.2 - All-Edges-ed	0.0	
M3.1.3 - TestInstructionCoverage	35.55	
M3.1.3.1 - All-Nodes-ei	35.55	
M3.1.3.2 - All-Nodes-ed	0.0	
M3.1.4 - TestPathCoverage	42.89	
M3.1.4.1 - All-Uses-ei	20.7	
M3.1.4.2 - All-Uses-ed	0.0	
M3.1.4.3 - All-Pot-Uses-ei	22.2	
M3.1.4.4 - All-Pot-Uses-ed	0.0	
QF4 - Actual Exploit in dev Maintainability		
Q4.1 - Analyzability For Maintenance		
Q4.1.1 - Code Documentation		
M4.1.1.1 - Comment Lines up to eLOC	0.39	
Q4.3 - Code Quality		
M4.3.3 - ECARules		
M4.3.3.10 - Duplicated Code		
M4.3.3.11 - Missing Braces If Stmts		
M4.3.3.12 - Missing Braces While Loops		
M4.3.3.13 - Missing Braces If Else Stmts		
M4.3.3.14 - Missing Braces For Loops		
M4.3.3.15 - Field Never Initialized Properly		
M4.3.3.16 - Excessive Method Length		
M4.3.4 - McCabe Index		
M4.3.5 - Chidamber Kemerer		
M4.3.5.2 - CBO per class	20.0	
M4.3.5.3 - LCOM per class	10.0	
Q4.4 - Code Size		
M4.4.1 - eLOC per class	58.09	
M4.4.2 - numParametersPerMethod	1.03	
M4.4.3 - numPackages	32.0	
M4.4.4 - numMethodsPerInterface	1.86	
M4.4.5 - numMethodsPerClass	5.75	
M4.4.6 - commentLinesPerClass	22.63	
M4.4.7 - Number of Attributes per Class	2.61	
M4.4.8 - Number of Classes	184.0	
M4.4.9 - Number of Interfaces	20.0	
Q4.7 - New Release Rate		
M4.7.2 - PRJ-LOC-CH-PER-Y	22808.0	
M4.7.3 - PRJ-MJR-REL-PER-Y	0.0	
M4.7.4 - PRJ-MINOR-REL-PER-Y	1.0	
Q4.9 - Code Modularity		
M4.9.01 - num Interfaces per Class	0.28	
M4.9.02 - num Methods per Interface	1.86	
M4.9.03 - num classes with defined attributes (perc)	79.9	
M4.9.10 - num classes with defined methods (perc)	98.92	
QFS - Actual Testability		
QFS.1 - Test Size		
M5.1.1 - Number of test cases	20.0	
QFS.2 - Test Result Availability		
M5.2.1 - Successful rate of test cases	1.0	

Illustrazione 20: Risultati Spago4Q

4.3 Problemi riscontrati

Ci sono state diverse problematiche incontrate durante il percorso di creazione dell'ambiente per l'analisi della qualità, tralasciando i vari problemi puramente tecnici legati alle varie dipendenze e configurazioni del sistema operativo e dei software utilizzati ci siamo imbattuti contro curiosi bugs, ma anche con problemi che ci hanno fatto riflettere e ci hanno portato a modificare in parte il nostro sistema per adattarlo meglio alle esigenze delle analisi. Partiamo con il parlare del problema più curioso che ci è capitato, ossia un bug scovato nel codice sorgente di Jabuti che non permetteva di compiere le analisi su sistemi operativi diversi da quelli basati su distribuzione Debian. Il fatto è stato molto curioso poiché su macchina virtuale con sistema operativo Debian le analisi venivano eseguite correttamente, mentre replicando esattamente l'ambiente su sistema operativo CentOS sulla macchina aziendale venivano restituiti risultati decisamente improbabili, oppure errori con comportamenti anomali del processo. Grazie alla natura Open Source del software in questione siamo stati in grado di scovare il bug e segnalarlo agli sviluppatori che hanno prontamente corretto il problema. Con Jabuti abbiamo avuto anche altri problemi nelle configurazioni delle analisi dei software obiettivo, in particolare il programma non permetteva di scatenare i test interni al codice sorgente java di Spago4Q poiché quest'ultimo, essendo di natura modulare, era suddiviso in tre eseguibili java che venivano creati in fase di compilazione. I test dovrebbero essere stati eseguiti correttamente mantenendo la modularità ma non è stato possibile farlo. Questo ci ha spinti a dover modificare la fase di compilazione di Spago4Q per far sì che creasse un unico eseguibile da fornire a Jabuti per l'analisi. Il problema riscontrato ci ha fatto pensare che in futuro questa non elasticità di Jabuti potrà creare dei problemi poiché, se la persona incaricata non conosce bene il software da analizzare, avrà seri problemi nel configurare una corretta analisi tramite questo tool. I risultati forniti inizialmente da Statsvn ci hano fatto notare che vi era un problema con la struttura del repository di sviluppo di Spago4Q. Il problema essenzialmente era che, quando il tool scendeva in profondità nella struttura del repository, si imbatteva in sezioni che non riguardavano strettamente rilasci e versioni di sviluppo di Spago4Q, ma vi erano altre sezioni dedicate a progetti affini e sperimentazioni non inerenti al nostro obiettivo. Il problema potrebbe essere risolto semplicemente modificando la gerarchia delle cartelle del repository svn. StatSVN richiede molte librerie aggiuntive per la corretta esecuzione del suo estrattore in Spago4Q, questo ha portato a dei comportamenti anomali della libreria java dedicata al disegno dei grafici.

5 Conclusioni

L'obiettivo era di creare un ambiente di integrazione che potesse ricavare la maggior parte delle metriche necessarie al modello MOSST per fornire una visione della qualità di un progetto Open Source. Allo stato attuale, i tool utilizzati richiedono ancora diverse fasi manuali nella loro configurazione e utilizzo, questo vanifica in parte l'obiettivo di avere un sistema che nella maniera più autonoma possa portare a termine le analisi richieste. Attualmente il tool più vicino a questo obiettivo è StatSVN, che essendo fornito sotto forma di estrattore per Spago4Q permette di essere configurato con pochi passaggi, ed ha la possibilità di schedulare le proprie attività di analisi, rendendolo automatizzato.

Il report del modello MOSST ci ha fornito una visione generale delle metriche ricavate dai software analizzati, ma essendo ancora in piena fase di definizione non possiamo ancora interpretarle poiché le soglie saranno scelte nelle prossime fasi di definizione del modello da parte di QualiPSo. A questo scopo sarebbe opportuno definire un modello statistico che riesca in maniera dinamica a valutare la significatività di ogni metrica nel proprio contesto di utilizzo, e quindi fornire le soglie per interpretare al meglio i dati rilevati.

L'integrazione di software Open Source non è totalmente esente da problemi come si vuol far credere[15], ma ha la particolarità che essendo seguito da una grande comunità attiva di sviluppatori la soluzione giunge spesso in maniera rapida. Durante l'esperienza di stage è stato possibile constatare che la risoluzione di un bug nel tool è stata favorita dal fatto che si era in possesso del codice sorgente, questo ci ha consentito di poter analizzare nel dettaglio il problema, e con la collaborazione degli sviluppatori porne rimedio in breve tempo.

Avere una certificazione affidabile ed universalmente accettata è una grande opportunità per il software Open Source. Un indicatore di questo tipo è un ottimo modo per favorirne l'adozione di massa in tutti i settori che attualmente fanno affidamento su software proprietari, comportando grossi benefici in termini di costi e qualità. Neelie Kroes, EU Internet Commissioner, in un suo recente discorso[16] si è espressa in maniera molto favorevole sull'adozione di queste tipologie di software, affermando che i governi e le aziende dovrebbero avere una buona giustificazione se promuovono ed utilizzano software e standard proprietari che richiedono royalty o impongono restrizioni al posto delle valide alternative Open Source.

6 Ringraziamenti

I Ringraziamenti più sentiti vanno sicuramente ai miei genitori, che con una immensa pazienza e amore mi hanno sempre supportato, facendomi diventare quello che sono. Ringrazio mio fratello, perché nonostante le litigate quotidiane mi vuole sempre bene, ed anche io gliene voglio.

Ringrazio i miei amici, quelli veri, perché loro ci sono sempre, per tutti i sorrisi che mi regalano e per tutte le avventure che abbiamo passato insieme in passato, nel presente e per tutto quello che sicuramente faremo in futuro.

Un ringraziamento a tutti i miei compagni di università per i bellissimi momenti passati insieme in questi anni, in particolare a quei pochi che mi hanno sempre aiutato ad andare avanti anche quando perdevo le speranze, senza chiedere mai nulla in cambio.

Ringrazio le poche ragazze che ho avuto, per tutto quello che ho passato con loro nel bene e nel male. Queste esperienze mi hanno permesso di crescere e diventare un uomo migliore.

Ringrazio tutti i miei compagni di chat di una volta, con loro ho passato gran parte dell'adolescenza e insieme abbiamo scoperto il mondo della rete. È sopratutto grazie a quelle esperienze che oggi ho un'inesauribile passione per l'informatica e per tutto quello che può offrire.

Ringrazio Gabriele Ruffati, perché grazie a lui ho potuto fare questa esperienza di stage presso una delle aziende che ammiro di più. Un grazie sentito a Davide Dalle Carbonare e Nicola Bertazzo per la loro infinita disponibilità nel seguirmi in tutte le fasi di questa esperienza lavorativa.

Infine, ringrazio il prof. Nicola Zingirian per la disponibilità ed il supporto che mi ha fornito nella stesura di questa tesi.

Grazie a tutti.

Indice delle illustrazioni

Bibliografia

[1]: *Wikipedia, Definizione Open Source,* 2010, <http://it.wikipedia.org/wiki/Open_source>

[2]: *QualiPSo, Boosting innovation and growthby fostering Open Source Software,* 2010, <http://www.qualipso.org/sites/default/files/QualipsoProjectBrochure.pdf>

[3]: *Spagoworld, Website,* 2009, <http://www.spagoworld.org>

[4]: *Spago4Q team, SpagoBI for Quality,* 2010, <http://www.spagoworld.org/xwiki/bin/view/Spago4Q>

[5]: *UE, Quality platform for open source software,* 2006, <http://cordis.europa.eu/fetch?CALLER=FP6_PROJ&ACTION=D&DOC=1&CAT=PROJ&QUERY=012889126c6b:9d97:01c42654&RCN=80465>

[6]: *Qualipso, Qualipso Website,* 2010, <http://www.qualipso.org>

[7]: *EU, Website,* 2010, <http://www.nessi-europe.com>

[8]: *Artemide, Website,* 2010, <http://www.artemide.org>

[9]: *Gabriele Ruffati, Pubblica Amministrazione e Industria:criteri di scelta di una soluzione FOSS,* 2007, <http://www.math.unipd.it/~ruffatti/docs/FOSS_PA_Industria_criteri%20di%20scelta.pdf>

[10]: *Flossitaly, Descrizione degli obiettivi,* 2010, <http://www.flossitaly.it>

[11]: *Spago4Q team, Il Meta-Modello,* 2010, <http://www.spagoworld.org/xwiki/bin/view/Spago4Q/Meta-Model>

[12]: *Nicola Salier, Aree di misura ed Indicatori per l'applicazione open source Spago4Q,* 2009

[13]: *University Of Insubria, Macxim website,* 2010, <http://qualipso.dscpi.uninsubria.it/macxim>

[14]: *Bicho Team, Servizio REST per il progetto QualiPSo,* 2010, <http://qualipsoa6.libresoft.es>

[15]: *Hermann Härtig, Claude-Joachim Hamann, and Michael Roitzsch, The Mathematics of Obscurity:On the Trustworthiness of Open Source,* 2010, <http://weis2010.econinfosec.org/papers/session6/weis2010_haertig.pdf>

[16]: *Neelie Kroes, Openness at the heart of the EU Digital Agenda,* 2010, <http://europa.eu/rapid/pressReleasesAction.do?reference=SPEECH/10/300&format=PDF&aged=0&language=EN&guiLanguage=en>