

METODI AGILI E SISTEMI DI GESTIONE DELLE CONFIGURAZIONI

Bruno Rossi, Alberto Sillitti, Giancarlo Succi
Libera Università di Bolzano
[Bruno.Rossi, Alberto.Sillitti, Giancarlo.Succi]@unibz.it

I sistemi di gestione delle configurazioni sono una possibile fonte di informazioni riguardo lo stato di un progetto software. Questo articolo presenta un tool per l'estrazione e l'analisi di informazioni da repository CVS con lo scopo di fornire dati ai progetti agili. Il tool si propone di analizzare il processo di produzione del software senza interferire con lo sviluppo e raccogliere dati empirici riguardo l'efficacia dei Metodi Agili.

1. Introduzione

I sistemi di controllo delle versioni (es. CVS, Visual Source Safe, Clear Case, ecc.) sono strumenti indispensabili per lavorare in team e sviluppare prodotti software complessi [1]. Questi sistemi permettono ad un team di lavorare contemporaneamente sugli stessi file sorgente gestendo automaticamente operazioni come: tracciamento delle versioni di un file, unione di modifiche fatte da diversi sviluppatori, ripristino di una versione precedente, ecc.

Questi sistemi sono stati progettati per memorizzare e gestire codice sorgente ma, oltre a queste informazioni, raccolgono anche una moltitudine di altri dati riguardanti il processo di sviluppo [3]. Questi dati comprendono: dimensione del codice, numero di modifiche, nome dello sviluppatore che ha inserito le modifiche, ecc. I dati memorizzati possono essere usati per ricavare informazioni sul processo di sviluppo in modo non invasivo, cioè senza influenzare in alcun modo il processo di sviluppo [4]. Inoltre, è possibile analizzare non solo i progetti attuali ma anche tutti i progetti passati dei quali si dispone ancora dei repository e, quindi, confrontarli tra loro.

Ad esempio, l'analisi dei dati raccolti in questo modo e i dati sui difetti potrebbero essere un metodo efficace per identificare buone abitudini di programmazione e migliorare l'intero processo di sviluppo.

Tracciare le attività compiute dai programmatori non è un compito facile per diversi motivi tra cui:

1. richiede tempo per essere fatta con precisione [6]
2. sia gli sviluppatori sia i manager non considerano importante questa attività perché non produce benefici immediati [5].

Per questi motivi, i dati raccolti manualmente sono spesso errati e/o incompleti, in particolare quando sarebbe più importante disporre di informazioni corrette: durante periodi critici come l'avvicinamento di una scadenza importante.

Molte aziende portano avanti progetti software per molti anni e le informazioni raccolte involontariamente dai loro sistemi di gestione delle configurazioni potrebbero essere analizzate e produrre dati utili per l'azienda. In particolare, i project manager potrebbero usare queste informazioni storiche per verificare

quali cambiamenti al processo di sviluppo sono avvenuti nel tempo e quali hanno prodotto un effetto positivo sul software prodotto (es: meno difetti, minore effort, ecc.). Tutto questo lo si potrebbe ottenere senza aver pianificato di monitorare il processo di sviluppo e senza dover aspettare di aver raccolto una quantità sufficiente di dati, una grande quantità di dati è già disponibile in azienda.

Questo articolo presenta una prima applicazione di CodeMart, uno strumento per la raccolta ed analisi di informazioni di processo contenute nei sistemi di gestione delle configurazioni [11]. L'articolo è organizzato come segue: la sezione 2 propone un metodo di analisi dei dati; la sezione 3 presenta un primo esperimento; la sezione 4 trae le conclusioni.

2. Analisi di repository di codice

I sistemi di gestione delle configurazioni sono una fonte di dati per lo studio della produzione del software e sono allo studio innumerevoli metodi per estrarne informazioni utili [3].

I sistemi di gestione delle configurazioni memorizzano il codice sorgente e le modifiche introdotte ma non raccolgono nessuna informazione relativa al tipo di modifica introdotta (es. aggiunta di una nuova funzionalità, bug fix, ecc.). Ai fini dell'analisi dei dati, sarebbe molto utile disporre di queste informazioni ma richiedere che gli sviluppatori inseriscano commenti significativi all'interno del codice e/o quando inseriscono il codice all'interno del sistema di gestione delle configurazioni è estremamente difficile. Nonostante questa difficoltà oggettiva, è possibile implementare un semplice sistema di classificazione automatica delle modifiche introdotte nel codice analizzando le differenze tra le diverse versioni dello stesso file memorizzate all'interno del sistema di gestione delle configurazioni [7].

Una classificazione semplice delle modifiche introdotte nel codice è la seguente [11]: **(1)** commenti, **(2)** modifiche strutturali, **(3)** modifiche non strutturali

Le modifiche di commento riguardano tutti i cambiamenti relativi ai commenti nel codice sorgente, quindi nessuna istruzione del linguaggio.

Le modifiche strutturali sono modifiche ad istruzioni del codice sorgente che influenzano i possibili percorsi di esecuzione del programma (es. i costrutti if-then-else, for, do-while, switch, ecc.)

Le modifiche non strutturali sono modifiche ad istruzioni del codice sorgente che non modificano i possibili percorsi di esecuzione del programma.

Un sistema di analisi statica del codice può facilmente individuare questi tipi di modifiche tra le diverse versioni di uno stesso file sorgente.

Normalmente, i sistemi di gestione delle configurazioni non forniscono alcuno strumento di supporto all'analisi dei dati, quindi è stato sviluppato CodeMart, uno strumento in grado di estrarre il codice, pre-processarlo, memorizzare i dati interessanti in un data warehouse e, infine, fare delle analisi [2] [9].

Una delle possibili analisi che si possono effettuare sui dati raccolti in questo modo si basa sull'analisi del comportamento degli sviluppatori. In particolare, l'analisi di come si evolve il comportamento nel tempo in progetti simili ed in progetti molto diversi. Per fare una analisi di questo tipo è possibile usare la Gamma analisi [10] che è utilizzata nelle scienze sociali.

In questo tipo di analisi si identifica un insieme di stati possibili del modello (fasi) e si analizza la loro evoluzione temporale. Nel caso delle modifiche al codice sorgente (strutturali, non-strutturali e di commento), uno sviluppatore può introdurre una qualunque combinazione di queste modifiche ad ogni invio del codice al sistema di gestione delle configurazioni (Tabella 1).

Tabella 1: Fasi

Phase	Structural modifications	Non-structural modifications	Comment modifications
A	0	0	0
B	≠ 0	0	0
C	0	≠ 0	0
D	0	0	≠ 0
E	≠ 0	≠ 0	0
F	≠ 0	0	≠ 0
G	0	≠ 0	≠ 0
H	≠ 0	≠ 0	≠ 0

La Gamma analisi descrive come le fasi si susseguono nel tempo e fornisce una misura delle sovrapposizioni.

3. Analisi dei dati

Il web server Apache è stato considerato spesso in studi riguardanti l'Open Source e lo sviluppo software [8], per questo motivo è stato utilizzato per fare un primo test di questo strumento analizzando i repository delle versioni 1.2, 1.3 e 2.0.

Questo primo esperimento vuole investigare l'applicabilità della Gamma analisi all'analisi del processo di sviluppo del software. Nel caso considerato, l'ipotesi sperimentale consiste nel ritenere che i precedence score e i separation score relativi ai progetti considerati siano scorrelati. Dato il numero esiguo di progetti considerati, non è possibile fare una analisi statistica vera e propria ma i valori ottenuti saranno valutati qualitativamente.

Tabella 2: Precedence score

Apache Version	A	B	C	D	E	F	G	H
1.2	-0.10	-0.01	-0.07	0.10	0.02	0.06	0.00	0.01
1.3	-0.15	-0.04	-0.07	0.24	-0.01	0.02	-0.02	0.03
2.0	-0.13	0.06	0.00	0.15	0.11	0.02	-0.01	-0.19

Tabella 3: Separation score

Apache Version	A	B	C	D	E	F	G	H
1.2	0.33	0.24	0.35	0.38	0.35	0.08	0.29	0.30
1.3	0.28	0.27	0.30	0.38	0.30	0.08	0.33	0.29
2.0	0.32	0.29	0.32	0.35	0.31	0.16	0.33	0.33

La Tabella 2 riassume i precedence score dei progetti evidenziando il fatto che tutti questi valori non sono rilevanti, visto che quasi tutti sono compresi nell'intervallo [-0.2; 0.2]. Questo significa che non è possibile identificare un ordinamento all'interno delle fasi. Inoltre, in ogni categoria, i valori dei tre progetti considerati sono piuttosto "simili".

I separation score elencati nella Tabella 3 sono anch'essi molto "simili" all'interno delle diverse categorie ma presentano una situazione diversa. Eccetto i valori associati alla fase F, tutti i valori sono maggiori di 0.2, quindi esiste una separazione tra le fasi (più o meno marcata a seconda dei casi).

Secondo questi dati, gli sviluppatori tenderebbero a separare le fasi identificate nella Tabella 1. Non è possibile affermare quali tipi di modifiche vengano fatte prima ma è possibile affermare che le modifiche sono organizzate a blocchi che tendono a non mischiarsi tra loro.

4. Conclusioni

Il sistema presentato in questo articolo è la prima versione di un sistema progettato per l'identificazione di sequeze di dati rilevanti all'interno dei sistemi di gestione delle configurazioni ed avente lo scopo di estrarre informazioni riguardo al processo di sviluppo. Questo lavoro è solo una prima analisi esplorativa dell'applicabilità dell'analisi Gamma ai dati provenienti da sistemi di gestione delle configurazioni.

5. Bibliografia

- [1] P. Cederqvist, "Version Management with CVS" – website: <http://www.cvshome.org/manual>
- [2] R. Cooley, B. Mobasher, J. Srivastava, "Web Mining: Information and Pattern Discovery on the World Wide Web", Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97), Novembre 1997.
- [3] M. Fisher, J. Oberleitner, J. Ratzinger, H. Gall, "Mining Evolution Data of a Product Family", 2nd International Workshop on Mining Software Repositories (MSR 2005), St. Louis, MO, USA, 17 Maggio 2005.
- [4] P.M. Johnson, "You can't even ask them to push a button: Toward ubiquitous, developer-centric, empirical software engineering", The NSF Workshop for New Visions for Software Design and Productivity: Research and Applications, Nashville, TN, USA, Dicembre 2001.
- [5] P.M. Johnson, H. Kou, J. Agustin, C. Chan, C. Moore, J. Miglani, S. Zhen, W.E.J. Doane, "Beyond the Personal Software Process: Metrics collection and analysis for the differently disciplined", 25th International Conference on Software Engineering, Portland, OR, USA, 3 - 10 Maggio 2003.
- [6] P.M. Johnson, A.M. Disney A.M., "A critical analysis of PSP data quality: Results from a case study", Journal of Empirical Software Engineering, Dicembre 1999.
- [7] S.J. Metsker, "Building Parsers with Java", Addison-Wesley, 2001.
- [8] A. Mokus, R.T. Fielding, J. Herbsleb, "A Case Study of Open Source Development: The Apache Server", International Conference on Software Engineering, Limerick, Ireland, Maggio 2000.
- [9] J. Myllymaki, J. Jackson, "Web-based data mining, Automatically extract information with HTML, XML, and Java", IBM developerWorks, <http://www-106.ibm.com/developerworks/web/library/wa-wbdlm/?dwzone=web>
- [10] D.C. Pelz, "Innovation Complexity and Sequence of Innovating Strategies", Knowledge: Creation Diffusion, Utilization, Vol. 6, 1985, pp. 261-291.
- [11] Sillitti A., Succi G., "Source Code Repositories and Agile Methods", 6th International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP2005), Sheffield, UK, 18 - 23 Giugno 2005.