Misurare la stabilità di un processo di sviluppo software

di
Luca Biasin

Introduzione

Nel seguente articolo si illustrerà, con dati ricavati da un caso di utilizzo reale, come sia possibile servirsi di uno strumento software per il tracciamento delle segnalazioni dei malfunzionamenti al fine di misurare e tenere sotto controllo la stabilità di un processo di sviluppo software in ambiente industriale.

Contesto aziendale

Eles Semiconductor Equipment S.p.A., nata nel 1988 a Todi (PG), opera nel settore dei sistemi di collaudo ed affidabilità di dispositivi a semiconduttore. I suoi prodotti principali sono sistemi complessi, formati da un elevato numero di componenti meccanici ed elettronici, controllati da diversi strati di applicazioni software e firmware. L'area di applicazione di tali programmi si estende dal controllo a basso livello dell'elettronica fino alla raccolta e all'analisi statistica dei dati prodotti.

Dal 1988 ad oggi, Eles si è trovata a dover proporre ai propri clienti, presenti su svariate differenti locazioni geografiche, distribuite su quattro diversi continenti, soluzioni sempre più complesse ma allo stesso tempo flessibili ed affidabili. Questa tendenza ha causato un costante e progressivo aumento dell'importanza del processo di sviluppo software all'interno dell'azienda.

Evoluzione del processo di sviluppo software

L'evoluzione del processo di sviluppo software si è articolata attraverso l'attuazione di alcuni passi fondamentali, tutti orientati ad una maggiore definizione e controllo: è stato introdotto un sistema di controllo di versione (database centralizzato per la memorizzazione ed il controllo di accesso ai file sorgente del software), si è realizzato un sistema di gestione di configurazione, si è definita una politica di pianificazione delle emissioni delle *release*[1]. In sostanza si è curato l'aspetto fondamentale della definizione formale del processo di sviluppo e di quegli strumenti necessari a sostenerlo.

Il processo attuale prevede la manutenzione e lo sviluppo di circa quaranta applicativi di medie dimensioni (150/200 classi l'uno), suddivisi in tredici differenti pacchetti di distribuzione, per un totale quindi di 53 progetti. Ogni anno sono rilasciate in media tre *release* per ogni pacchetto di distribuzione, e un certo numero di aggiornamenti critici. Nel periodo che intercorre tra una *release* e l'altra, il gruppo di sviluppo – di sette persone – lavora sulla correzione degli errori residui segnalati dagli utilizzatori interni ed esterni e allo sviluppo delle nuove caratteristiche, secondo quanto pianificato dalla gestione del percorso aziendale di evoluzione dei sistemi. Il carico di lavoro dei nuovi sviluppi è stimato in base alla complessità delle nuove caratteristiche richieste, quindi è pianificato e tenuto sotto controllo per mezzo di un sistema informatico aziendale.

Obiettivi di misura del processo

Il processo sopra descritto può mantenersi stabile solo se è trascurabile il carico di lavoro dovuto alla correzione degli errori o alla richiesta di realizzazione di nuove caratteristiche non previste al momento della prima pianificazione. In pratica la complessità e la natura stessa dei prodotti realizzati fanno sì che questo non avvenga. Inoltre, spesso può essere conveniente attuare un processo che supporti una certa flessibilità nella gestione delle specifiche (processo di sviluppo *agile*), a patto di essere in grado di controllare il costo dei cambiamenti richiesti.

In questo scenario serviva dunque trovare un modo per indagare quale fosse l'effettivo impatto della gestione della correzione degli errori e dei piccoli cambiamenti imprevisti sulla normale polarizzazione degli sviluppi pianificati. In aggiunta si voleva indagare anche se fosse possibile mantenere costante e stabile la quantità del lavoro di manutenzione imprevista, tanto da poterlo pianificare in anticipo e, d'altra parte, indagare se un tasso troppo elevato di realizzazione di nuovi sviluppi potesse essere correlato con un aumento del tasso di errori.

Allo scopo di collezionare i dati necessari alla nostra analisi, si è cominciato a cercare uno strumento di tracciamento degli errori. Alla fine di un periodo di sperimentazione durato circa quattro mesi, la nostra scelta è caduta su Mantis: un software *open source*[2], distribuito gratuitamente. Tale software presenta un giusto grado di complessità, è facilmente adattabile alle esigenze dell'utente e fornisce un supporto all'analisi dei dati che colleziona.

Funzionamento del sistema di tracciamento di segnalazioni d'errore

In generale, un software di tracciamento permette di memorizzare tutte le segnalazioni di malfunzionamento di un certo numero di programmi e di tracciare l'evoluzione del loro stato fino alla dichiarazione della loro correzione. Gli utenti del sistema sono suddivisi a seconda del ruolo che svolgono. Ruoli tipici sono: *Reporter* (segnalatore d'errore), *Developer* (sviluppatore), *Tester* (verificatore). In Figura 1 è illustrata, evidenziata in neretto, la tipica evoluzione dello stato di una segnalazione d'errore: un *Reporter* si collega al sistema e segnala un errore in un particolare programma; il sistema assegna l'analisi della segnalazione al *Developer* responsabile di quel programma; il *Developer* analizza la segnalazione e conferma la presenza del problema; successivamente, il *Developer* corregge il problema e lo dichiara risolto; durante il collaudo il *Tester* controlla l'effettiva risoluzione del problema e dichiara la segnalazione chiusa.

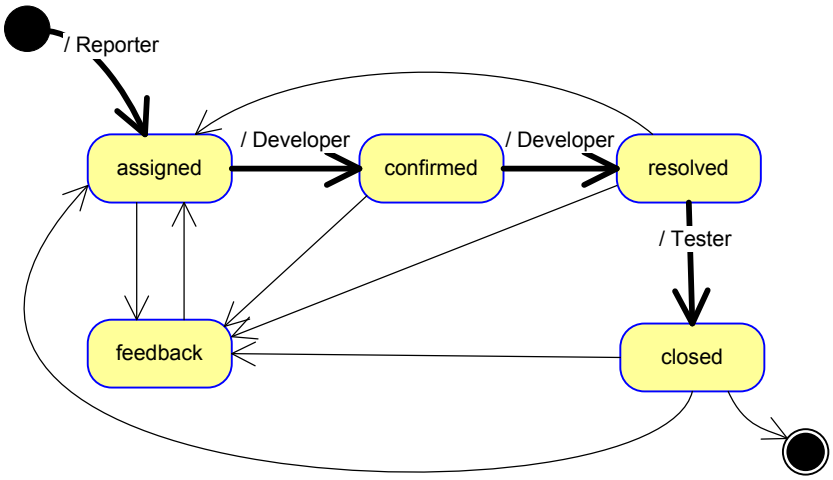

Figura 1: diagramma degli stati di una segnalazione errore.

Ogni transizione dello stato di una segnalazione d'errore viene memorizzata, assieme ai suoi dati temporali e ad altri attributi la cui trattazione esula dallo scopo di questo articolo.

Esempi di dati e misure ottenibili

La base dei dati che si viene a formare durante l'utilizzo del sistema di tracciamento delle segnalazioni rappresenta una fonte di informazione utile all'analisi dello stato del processo di sviluppo. Nel seguito commenteremo i dati collezionati dal nostro sistema nel corso di circa nove mesi di utilizzo. Al momento della stesura dell'articolo, la base dati comprendeva esattamente 200 segnalazioni d'errore su 53 progetti tracciati.

Una prima diagnosi della efficienza del processo di sviluppo è stata effettuata analizzando la distribuzione degli stati delle segnalazioni (Tabella 1: riassunto per stato). Il numero degli errori *Open* (ancora aperti) rappresenta in sostanza il lavoro accumulato non previsto ancora da svolgere da parte degli sviluppatori, il numero delle segnalazioni *Resolved* (risolte) ma non ancora *Closed* (dichiarate chiuse) rappresenta il lavoro pendente per i verificatori.

Tabella 1: riassunto per stato.

By Status	Open	Resolved	Closed	Total
new	0	0	0	0
feedback	6	0	0	6
acknowledged	0	0	0	0
confirmed	8	0	0	8
assigned	25	0	0	25
resolved	0	19	0	19
closed	0	0	142	142
Total	39	19	142	**200**

Altre analisi più dettagliate sono state effettuate esaminando la distribuzione della severità degli errori, così come viene dichiarata dagli utilizzatori al momento della segnalazione. In Figura 2, ad esempio, si può notare come la gran parte degli errori segnalati sia costituita da problemi generici (*major*, *minor*) che tuttavia non causano un blocco o un malfunzionamento grave del programma (*crash*, *block*). D'altra parte è anche messo in evidenza che l'incidenza delle segnalazioni corrispondenti a richieste di caratteristiche non pianificate (*feature*) non è trascurabile.

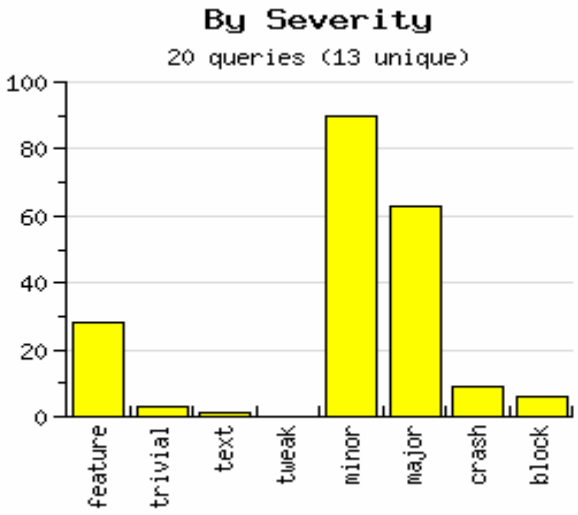

Figura 2: distribuzione delle segnalazioni per severità.

L'efficienza del tipo di risoluzione è stata indagata utilizzando la distribuzione delle risoluzioni per tipo di soluzione, così come viene dichiarata dagli sviluppatori. In Figura 3 è possibile notare come il grado di efficienza delle soluzioni sia elevato: l'incidenza di segnalazioni non replicabili, non

risolvibili, sospese, o che si è deciso di non risolvere (*unable to duplicate, not fixable, suspended, won't fix*) è praticamente nulla.

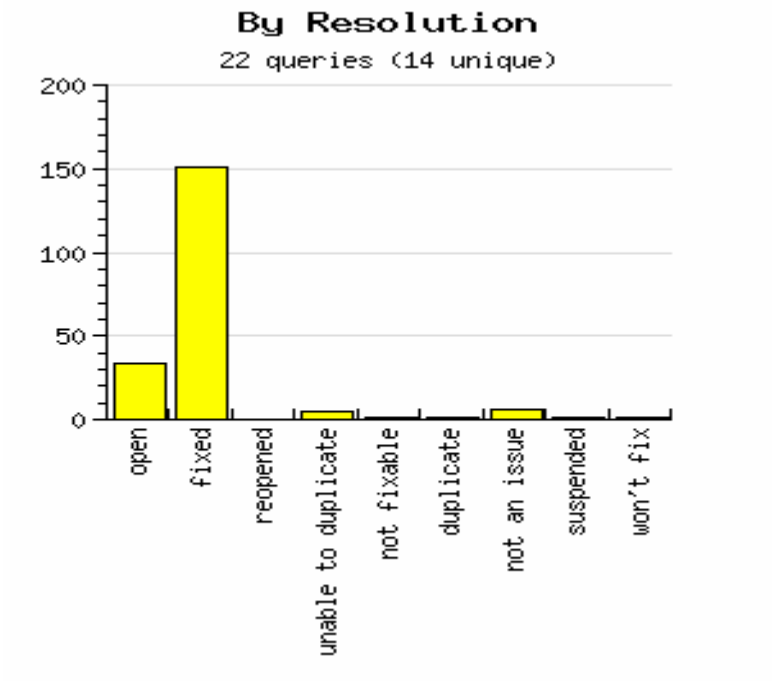

Figura 3: distribuzione delle risoluzioni per tipo di soluzione.

Un primo approccio all'estrapolazione di un indice riassuntivo dell'efficienza del processo di risoluzione degli errori è stato quello del calcolo del tempo medio di risoluzione. In Tabella 2, si vede che tale valore è pari a circa 12 giorni.

Tabella 2: statistiche sul tempo di risoluzione delle segnalazioni.

Time Stats For Resolved Issues (days)	
Longest open issue	0000036
Longest open	223.04
Average time	12.38
Total time	1,993.41

In realtà il calcolo della semplice media dei tempi di risoluzione, cioè la media del tempo che intercorre tra una segnalazione di un problema e la sua risoluzione, è solamente una sottostima dell'MTTR (Mean Time To Repair), in quanto non tiene conto dei problemi ancora aperti.

Un approccio più corretto al calcolo dell'MTTR, ipotizzando una distribuzione esponenziale per i tempi di segnalazione, si basa sulla formula illustrata nella Equazione 1, dove N_R è il numero delle segnalazioni risolte, N_A è il numero delle segnalazioni ancora aperte, i TS_j e i TR_j sono, rispettivamente, i tempi di segnalazione e di risoluzione dei singoli problemi e T0 è l'istante in cui si calcola il valore dell'MTTR. Applicando tale formula alla nostra base dati si è ottenuto un valore di circa 32 giorni.

$$MTTR = \frac{\sum_{j=1}^{N_R}(TR_j - TS_j) + \sum_{j=1}^{N_A}(T0 - TS_j)}{N_R}$$

Equazione 1: formula per il calcolo dell'MTTR.

Anche se tale MTTR può sembrare elevato è tuttavia importante notare che è perfettamente in accordo con l'ordine di grandezza del periodo di emissione delle *release*, di circa quattro mesi, e non è in realtà correlato con la percezione che ha l'utente dell'effettivo tempo di consegna di una nuova versione del software priva del problema segnalato. Uno studio più approfondito di tale tempo dovrebbe tener conto delle date di emissione delle nuove versioni.

Una analisi dell'andamento del processo del tempo è stata affrontata esaminando la Figura 4, nella quale sono disegnati gli andamenti temporali del numero dei problemi riportati (in blu), del numero dei problemi risolti (in nero) e di quelli ancora aperti (in rosso).

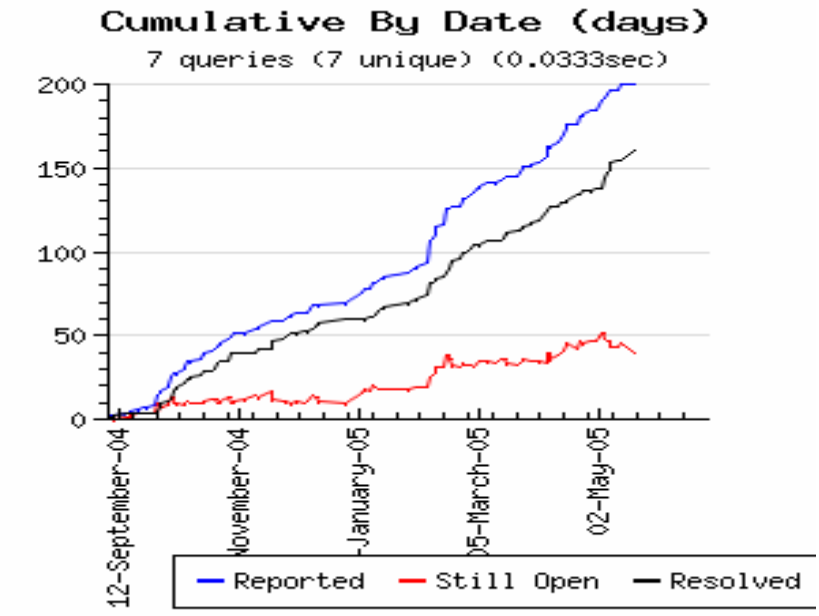

Figura 4: andamento temporale delle segnalazioni.

In particolare tale grafico rappresenta uno strumento molto efficace per tenere sotto controllo la stabilità del processo di sviluppo. La pendenza del grafico dei problemi segnalati rappresenta il tasso d'incidenza dei problemi nei nostri prodotti. Ovviamente l'ottimizzazione del processo di sviluppo e di testing dovrebbe tendere a rendere questa curva la più piatta possibile.

Un tasso di segnalazione degli errori costante, in presenza di un tasso di sviluppo costante, può essere interpretato come un indice di stabilità del processo. Al contrario, un aumento costante della percentuale di errori non chiusi è indice di un tasso di introduzione di nuove caratteristiche pianificate sproporzionato rispetto alla capacità di sviluppo e necessita di correzioni.

Un andamento costante del grafico dei problemi aperti, o addirittura decrescente, indica che il processo è stabile. Un andamento crescente che si protrae nel tempo indica un accumulo di problemi irrisolti a cui si dovrebbe far fronte aumentando il tempo dedicato dal gruppo di sviluppo alla risoluzione dei problemi rispetto al tempo dedicato agli sviluppi pianificati.

Nel nostro grafico, si può notare un tasso di segnalazione errori pressoché costante, accompagnato da un grafico di risoluzione con un tasso lievemente minore. In accordo con questo si riscontra un

leggero accumulo dei problemi aperti che si incrementa velocemente a partire dall'inizio di febbraio 2005 e che comincia a stabilizzarsi solo all'inizio di maggio 2005. Nel nostro caso è interessante notare che l'istante di aumento dell'accumulo coincide con la diminuzione temporanea del numero di sviluppatori di una unità e che l'istante di diminuzione sia immediatamente successivo ad una serie di azioni di risoluzione di problemi intraprese per stabilizzare il processo.

Conclusioni

L'utilizzo e l'interpretazione corretta dei dati restituiti da un sistema di tracciamento delle segnalazioni degli errori possono evidenziare i problemi, prima che questi si manifestino con evidente gravità, e permettono di effettuare le giuste correzioni.

D'altra parte la nostra esperienza ha evidenziato un aspetto non trascurabile, tipico dell'introduzione di sistemi informativi in ambienti lavorativi: quello della resistenza al cambiamento. Nel nostro caso è stato seguito un approccio graduale: la sperimentazione ha coinvolto all'inizio il solo gruppo di sviluppo software e un numero limitato di progetti. Quando il sistema è stato completamente sotto controllo, la sperimentazione è stata estesa a tutti i progetti in corso e ad un elevato numero di utenti (pari a circa la totalità dell'area Ricerca e Sviluppo). In sostanza, ogni utente che si è trovato ad utilizzare il sistema è stato incoraggiato dalla relativa facilità d'utilizzo, dal continuo supporto del gruppo di sviluppo e, soprattutto, dall'immediato riscontro che il sistema era in grado di fornire ad ogni segnalazione inviata.

Purtroppo, allo stato attuale, il nostro sistema di tracciamento non prevede campi in grado di memorizzare il tempo di lavorazione effettiva di ciascun problema e quindi non è stato ancora possibile accumulare dati utili a stimare in anticipo l'incidenza del lavoro di manutenzione rispetto a quello di sviluppo.

Sviluppi futuri

E' attualmente in esame, tramite l'utilizzo di campi personalizzabili, la possibilità di tracciare il numero di ore di lavoro effettivamente dedicate alla risoluzione di ogni problema segnalato.

E' sempre in fase di studio l'introduzione di ulteriori indici numerici riassuntivi, ricavabili dalla base dati del sistema di tracciamento, e si prevede il loro utilizzo per una misura formale e il controllo della qualità del processo.

Riferimenti bibliografici

- B1: Agile Modeling, Scott W. Ambler, John Wiley & Sons, Inc., New York, ISBN 0-471-20282-7
- B2: Misurare il software, Luigi Buglione, Franco Angeli, 2003, ISBN 88-464-4634-8
- B3: Applied Statistics for Software Managers, Katrina D. Maxwell, Prentice Hall, ISBN 0-13-041789-0
- B4: Commentary – Software: Metrics Mentality versus Statistical Mentality, John D. Healy, IEEE Transaction on Reliability, Vol. 40, No. 3, September 2000

Siti Web

- L1: sito web dove è possibile scaricare il sistema di tracciamento bachi Mantis: www.mantisbt.org
- L2: sito ufficiale dell'*Agile Modeling*: www.agilemodeling.com
- L3: sito ufficiale italiano delle metodologie di sviluppo agili: www.agilemovement.it

[1] Con il termine *release* di un applicazione si indica una particolare versione di quella applicazione. Il rilascio o la distribuzione di diverse *release* di un'applicazione, in un certo periodo di tempo, implica che questa venga distribuita in istanti differenti, in stati di evoluzione e maturità differenti, caratterizzati da diverse stringhe identificative di versione.

[2] Software *Open Source*: è un software distribuito assieme ai suoi sorgenti. A seconda della licenza di distribuzione è possibile per l'utente modificarli e addirittura utilizzarli come base per nuovi prodotti software.