

Lattix LDM for .NET

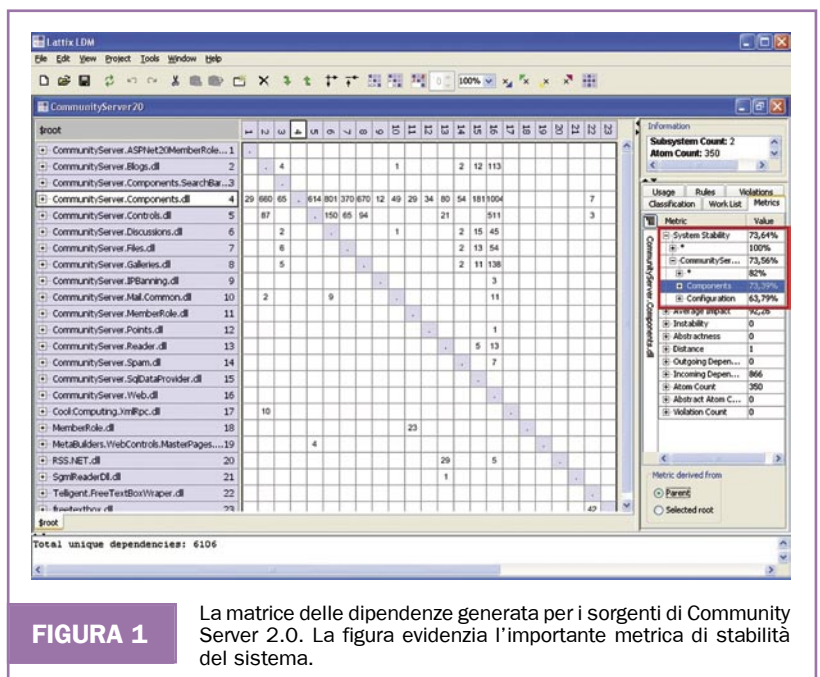
L'analisi matriciale delle dipendenze permette a questo tool per architetti software (disponibile per Java, C/C++, Oracle ed ora anche per .NET) di effettuare un'analisi oggettiva e ripetibile sulla qualità della modularizzazione e delle interdipendenze di un'architettura.

Gli ultimi Jolt Awards (un po' il "premio Nobel" dell'informatica) hanno visto, nella categoria Design Tools And Modeling, il trionfo a sorpresa di questo prodotto per architetti sul blasonato Borland Together. Lattix SDM usa una semplice ma efficace metodologia, la DSM (Dependency Structure Matrix or Design Structure Matrix, <http://www.lattix.com/technology/whatisdsm.htm>), elaborata al solito MIT, che riduce la valutazione di un'architettura software ad una semplice ispezione visiva su una matrice generata automaticamente. Naturalmente, il prodotto è in grado anche di generare una serie di metriche sul software che, insieme all'analisi architeturale, possono essere d'aiuto per alcuni processi del CMMI (<http://www.lattix.com/solutions/cmmi.htm>). Il prodotto nasce in ambito Java, ma vista la natura indipendente dal linguaggio dell'analisi matriciale, né è stato rilasciato, fra gli altri, un plug-in per l'analisi del codice .NET, oggetto di questa prova. Sono disponibili inoltre plug-in per Eclipse e NetBeans, per C/C++, e per Oracle.

Caratteristiche del prodotto.

A partire dal codice sorgente, Lattix LDM genera, una matrice sulle cui righe e colonne sono elencati tutti i moduli software del progetto; ove un modulo interagisca con un altro, nella cella situata all'incrocio dei due (la convenzione è che il modulo nella colonna usa quello nella riga) viene posto un numero che indica la "forza" della dipendenza: di default, rappresenta il numero di dipendenze che una classe ha verso le altre, ma è naturalmente modificabile dall'architetto. La generazione avviene in automatico, e la velocità di esecuzione dell'analisi è davvero notevole. La matrice generata ha alcune proprietà notevoli: per cominciare, un modulo cui corrisponde una riga (risp. colonna) vuota non è usato (risp. non usa) alcun altro modulo, per cui corrisponde alla radice (risp. all'ultimo strato) della gerarchia. Ugualmente, il prodotto evidenzia quei gruppi di moduli che hanno solo dipendenze *interne*,

ovvero si usano l'un l'altro; poiché costituiscono dei "gruppi funzionali indipendenti", tali gruppi vengono aggregati, sommando i rispettivi pesi, e conservando naturalmente sempre la possibilità di fare un *drill down* sui componenti di un medesimo gruppo funzionale. L'intervento, banale, dell'architetto è richiesto solamente per avere, riordinando i moduli ottenuti, una matrice il più possibile simile ad una **triangolare bassa** (anche solo a blocchi): in tal caso, evidentemente, avremo una situazione in cui i moduli *in alto* usano quelli immediatamente "in basso", ovvero una completa stratificazione dell'architettura. Inoltre, nel caso in cui non riuscissimo ad avere una matrice triangolare bassa, i numeri posti al di sopra della diagonale principale consentono di identificare immediatamente le dipendenze cicliche. Ancora, l'esame dei pesi permette di identificare moduli critici per il progetto, assieme ad altre considerazioni: ad esempio, se ad un modulo corrispondono sia una riga che una colonna "piene", questo indica che il modulo usa (ed è usato da) tutti gli altri moduli, per cui è sicuramente di centrale importanza. Sottolineiamo ancora una volta che l'esperienza dell'architetto ha sempre il suo peso, ma la generazione della matrice e molte analisi su di essa si prestano ad essere automatizzate in larga parte. Ne consegue che un gran numero di considerazioni sulla stratificazione dell'architettura e sull'indipendenza/dipendenza funzionale dei singoli moduli può essere espresso sulla base dei "fattori di forma" della matrice, che costituiscono un dato **oggettivo**, piuttosto che osservando i pesi attribuiti alle interazioni, che possono essere modificati in maniera **sogettiva**. Immaginiamo ora di dover effettuare la stessa analisi sul diagramma UML del software (un significativo esempio è riportato nella documentazione del prodotto): un architetto riuscirebbe ad evidenziare le dipendenze fra i vari moduli solo navigando, certamente



non in maniera automatica, il diagramma, utilizzando dunque i colori per evidenziare quelle classi che rappresentano punti di estensione, moment-interval. Chiunque abbia familiarità con il linguaggio di modellazione in questione sa che al crescere della complessità del progetto diventa difficile scovare le dipendenze e risolverle, utilizzando unicamente i diagrammi. La matrice DSM invece “scala” magnificamente man mano che le linee di codice si moltiplicano: la dimensione della matrice, ovvero il numero di righe e colonne, aumenta, ma rimane sempre la possibilità di analizzarla “a colpo d’occhio” guardandone semplicemente le macrocaratteristiche. Inoltre, indipendentemente dalle dimensioni, la matrice generata da Lattix permette con grande facilità di aggregare macrofunzionalità, piuttosto che, al contrario, di esaminare nel dettaglio i moduli che ci interessano: dal punto di vista pratico, si compattano dei moduli aggregando le rispettive righe e colonne, o viceversa si esplodono i moduli che compongono le macrofunzionalità, con lo stesso meccanismo con cui espandiamo/comprimiamo un albero di cartelle in “Esplora Risorse”. Una caratteristica importante del prodotto è quella di permettere la definizione di regole architetturali; è possibile ad esempio impedire dipendenze esterne o fra particolari moduli. Tali regole vengono editate, in maniera visuale ed assistita da Lattix, che ha già a questo punto *consito* tutte le componenti del software; i moduli che le violano vengono visibilmente evidenziati con un flag rosso. È altresì particolarmente efficace la rianalisi di una diversa versione di un programma per cui sia già stata generata la DSM, che evidenzia, ove presenti, nuove violazioni delle regole architetturali.

Installazione ed utilizzo

Sia una versione di valutazione della versione attuale, la 2.61, che il piccolo JAR contenente il plug-in per .NET (meno di 250 KB) possono essere scaricati all’URL <http://www.lattix.com/beta-dotnet/betadl.htm>. L’installazione, quanto mai semplice, richiede però la presenza del JRE 1.4.2 o superiore; è richiesto l’inserimento di una chiave di licenza; una evaluation deve essere richiesta direttamente alla Lattix. Dopo la registrazione, si può semplicemente copiare il plug-in JAR per .NET nella cartella dei plug-in (di default C:\Programmi\Lattix2.6.1\bin\plugins). Se ora lanciamo un nuovo progetto di Lattix LDM da “File/New Project”, avremo disponibile la nuova voce “.NET” nell’area “Project Type”; per le prime prove abbiamo scelto il codice sorgente della versione 2.0 di Community Server (<http://communityserver.org>), un software per la creazione di portali community sempre più popolare nel mondo .NET. Poiché è necessario specificare degli assembly .dll o .exe già compilati abbiamo selezionato, in particolare, i moduli presenti nella sottocartella “bin” dello ZIP scaricabile dal sito. Nella creazione del progetto consigliamo anche di stratificare i sorgenti in sottosistemi e di far generare un file di log. Su un processore Pentium Mobile da 2 GHz con 1 GB di RAM l’analisi di 26 assembly di codice, con generazione dei sottosistemi e del file di log, ha richiesto poco più di un minuto: non proprio un cattivo risultato,

diremmo. La **Figura 1** mostra la matrice generata: abbiamo selezionato un modulo con un elevato livello di interazione (individuabile “ad occhio” dalla riga “piena”), evidenziandone nella finestra delle proprietà il tab relativo alle metriche di progetto, ed in particolar modo alla “stabilità di sistema”, una misura della sensibilità del sistema ai cambiamenti, ovvero, di quant’è probabile che un cambiamento localizzato ne provochi altri “a catena” in altre parti del sistema, calcolata come complemento al 100% del rapporto fra numero di classi potenzialmente affette e numero di nodi foglia nell’albero delle dipendenze, “esplosa” per i vari moduli per un’interessante panoramica sulle metriche

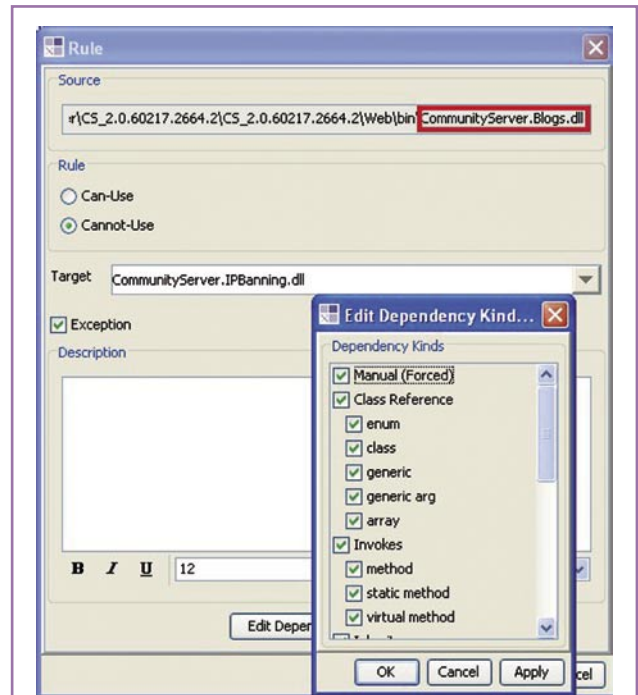


FIGURA 2

La definizione di una regola architetturale, comprese eventuali eccezioni e tipi di dipendenze da considerare.

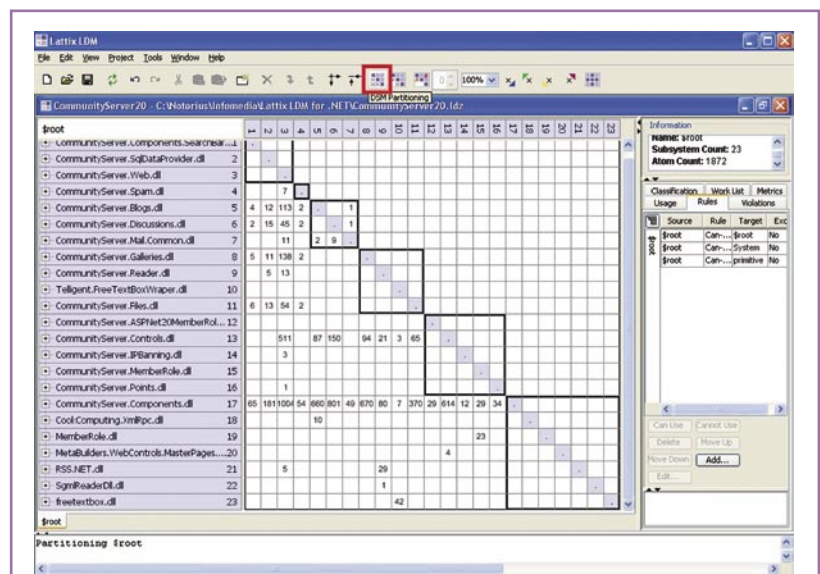


FIGURA 3

L’algoritmo di partizionamento DSM porta ad una struttura a blocchi che stratifica l’applicazione in layer, oltre a mostrare eventuali dipendenze cicliche (numeri al di sopra della diagonale principale).

disponibili, si veda il Capitolo 6 dell'Help del prodotto. Quando andiamo a salvare il progetto (in questa nuova versione, in formato compresso di default), otterremo un progetto .ldz di meno di 200 KB, a fronte di più di 5 MB di codice sorgente. Ciò che paradossalmente occupa spazio su disco è il file di log (quasi 43 MB!), che però contiene, assembly per assembly, l'elenco di tutti i costrutti .NET che hanno generato una particolare dipendenza (per ulteriori dettagli, in particolare sul significato delle righe contrassegnate con FILE, CREATEATOM e DEPENDENCY, si veda l'Help del prodotto). Selezioniamo ora uno dei moduli e dalle proprietà, scheda "Rules", andiamo a definire una nuova regola architetturale per il progetto; per esempio, possiamo chiedere che il modulo dei blog non usi il banning degli indirizzi IP (in **Figura 2** una scelta, naturalmente, che si presta a qualche obiezione, ma la facciamo a puro titolo di esempio), specificando anche eventuali eccezioni e, soprattutto, i tipi di dipendenze su cui vogliamo lavorare. La nuova regola comparirà nell'elenco; da menu contestuale, sulla cella all'incrocio fra i due moduli interessati dalla regola (si ricordi la convenzione: il modulo sulla riga è usato da quello nella colonna), andiamo a creare artificiosamente una dipendenza, e clicchiamo poi sul pulsante della barra degli strumenti evidenziato in figura: un visibile flag rosso ci mostrerà le violazioni delle regole architetturali definite. È giunto il momento del "piatto forte", ovvero dell'analisi DSM: selezioniamo il nodo "\$root" della matrice delle dipendenze, e clicchiamo sul pulsante "DSM Partitioning" nella barra degli strumenti; in men che lo si dica, ecco pronta la matrice DSM associata all'applicazione (**Figura 3**). Una perfetta matrice triangolare bassa a blocchi: con molti complimenti agli autori di Community Server 2.0. Una volta partizionata un'applicazione, è possibile applicarvi ricorsivamente l'algoritmo Provider Proximity (tramite il tasto posto immediatamente a destra di DSM Partitioning), che cerca di spostare il sottosistema selezionato il più vicino possibile a tutti i sottosistemi che dipendono da esso. Quando riteniamo di aver lavorato abbastanza sull'architettura, possiamo ottenerne un modello concettuale: clicchiamo col tasto destro su un sottosistema qualsiasi (noi abbiamo scelto, naturalmente, "\$root"), escegliamo la voce "Conceptual Architecture Diagram". Verrà generato un diagramma per l'architettura (**Figura 4**), che mostra la stratificazione ottenuta a seguito dell'applicazione del-

l'algoritmo DSM, su cui possiamo fare un po' di editing grafico per poi esportarlo in JPEG: la figura è stata ottenuta proprio per esportazione da Lattix LDM. È sicuramente un "plus" del prodotto, infine, il poter effettuare l'esportazione dei dati di analisi in un file Excel, purtroppo in tal caso senza poter effettuare il raggruppamento/espansione dei sottosistemi.

Pro

Un ottimo prodotto per effettuare, in maniera oggettiva e ripetibile, analisi di qualità sull'architettura di un progetto software, basate sull'analisi delle dipendenze. Le caratteristiche di analisi per .NET sono allo stesso livello qualitativo di quelle per Java, linguaggio d'elezione per il prodotto. Ottima la parte di reportistica.

Contro

Lattix non è un tool *two-way*, ma è limitato all'analisi a posteriori: nessuna scelta architetturale permette di generare corrispondenti regole nel codice.

Bibliografia

- [1] N. Sangal, E. Jordan, V. Sinha, E. Jackson, "Using Dependency Models to Manage Complex Software Architecture", OOPSLA'05, October 16–20, 2005, San Diego, California, USA; disponibile all'URL <http://sdg.lcs.mit.edu/pubs/2005/oopsla05-dsm.pdf>



SCHEDA PRODOTTO

Nome e versione	Lattix LDM 2.61 for .NET
Categoria	Analisi matriciale delle dipendenze di software ad oggetti
Produttore	Lattix, Inc 8 Harper Circle, Andover, MA 01810 Tel: 978-474-5022 - Fax: 630-604-3261 E-mail vendite: sales@lattix.com E-mail supporto: support@lattix.com
Distributore	Lattix, Inc 8 Harper Circle, Andover, MA 01810 Tel: 978-474-5022 - Fax: 630-604-3261 E-mail vendite: sales@lattix.com E-mail supporto: support@lattix.com
Prelevabile da	http://www.lattix.com/beta-dotnet/betadl.htm
Prezzo	Enterprise Edition: \$2995 View Edition: \$395 Professional Edition: \$495 Professional Plus Edition: \$695 Moduli aggiuntivi e supporto: www.lattix.com/products/LicensingPricing.htm
Sistema Operativo	Windows XP Professional con Service Pack 2; Windows 2000 Professional, Server, Advanced Server con Service Pack 4; Windows Server 2003
Processore	Intel Pentium III 1 GHz (consigliato Pentium IV 3 GHz)
Memoria	512 Mb RAM (consigliato 1 Gb RAM)
Spazio Disco	100 Mb
Scheda Grafica	SVGA
Compatibilità e note	Necessario JRE 1.4.2 o superiore

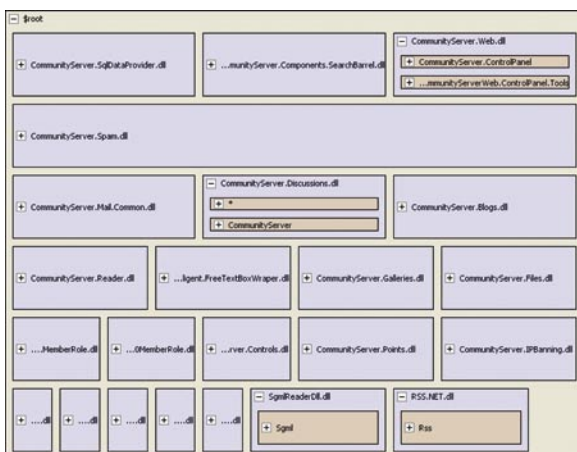


FIGURA 4 Il diagramma concettuale della nostra architettura, esportato direttamente da Lattix LDM.