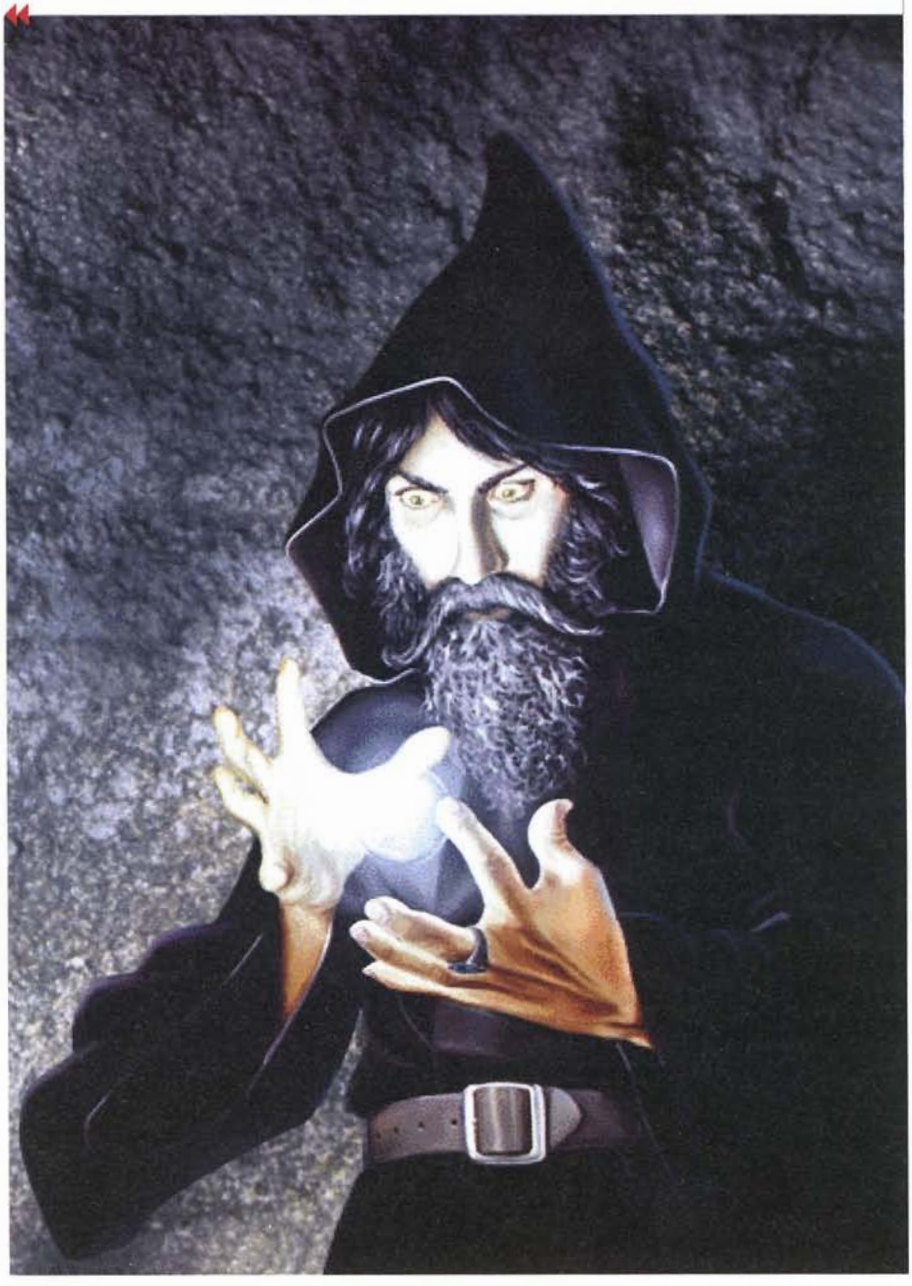
Nel corso della storia informatica si è tentato più volte di definire dei metodi certi per quantificare e valorizzare lo sviluppo e la produzione del software. Nell'articolo le tappe compiute dalla comunità scientifica e lo stato dell'arte.

La misura del software: scienza o arte?

di Giancarlo Magnaghi



La misura del valore economico del software è un tema molto dibattuto e ancora aperto.
Non esiste e probabilmente non esisterà mai una metrica univoca per determinare il valore del software (Software Metrics) poiché esistono scenari tecnici e commerciali radicalmente diversi.
Una prima grande divisione è tra software prodotto in serie (software commerciale, shareware, open source) e software prodotto ad hoc (progetti software). Poi ci sono varie tipologie: software di base, middleware, pacchetti di produttività individuale, software per il web, software gestionale, quello per il controllo di processo, Cad, Cam, software embedded, firmware e via discorrendo, per non parlare dei molteplici ambienti di sviluppo e delle centinaia di linguaggi e dialetti di programmazione.
Cercare di mettere ordine in questa Babele può sembrare a prima vista un'impresa disperata, e in effetti lo è, ma esistono alcune metodologie di misura del software che, anche se ancora rozze e poco precise, consentono comunque almeno di impostare in modo sistematico la tematica della misura e della valorizzazione del software.

Perché si misura il software

Il controllo quantitativo e la misura dei progetti di sviluppo software ha assunto notevole rilevanza con la diffusione dei contratti di sviluppo a prezzo fisso, dell'applicazione delle penali nei confronti degli sviluppatori da parte dei clienti, dell'affidamento in outsourcing e degli accordi di partnership. I clienti sono sempre meno disposti ad accettare scarsa qualità, ritardi nelle consegne e aumenti imprevisti dei costi. È molto sentita anche la necessità di utilizzare misure quantitative per confrontare la propria organizzazione con le altre (benchmarking).

Poiché non esiste un'unica soluzione magica in grado di misurare in modo semplice e accurato tutto il processo di sviluppo del software, è consigliabile utilizzare più di una metodologia e poi confrontare i risultati, poiché le stime, ancorché guidate da metriche e da modelli matematici sono più un'arte che una scienza e dipendono dal giudizio personale di chi fa le valutazioni, infatti i modelli utilizzati comportano per definizione una semplificazione della realtà e quindi limitano la precisione delle stime ottenibili.

Nell'utilizzo pratico, il primo principio della misura del software è: "pragmatismo e compromesso" poiché bisogna affrontare il problema con un approccio non accademico, ma manageriale per ottimizzare il rapporto tra la bontà delle misure ottenute e gli sforzi necessari per realizzarle. Molti programmi di misura del software non vanno a buon fine perché l'impegno necessario viene sottostimato, non hanno un adeguato supporto da parte del management e la precisione dei risultati ottenuti è giudicata insufficiente.

Sono disponibili sul mercato molti pacchetti che forniscono stime relative al dimensionamento di costi e durata dei progetti software, con prezzi che variano da poco più di 100 Euro per i programmi shareware ai 2mila -5mila Euro per i prodotti commerciali come **Costar** di Softstar Systems e **Workbench** dell'australiana Charismatek.

Poiché le stime prodotte contengono tipicamente una percentuale di errore del 20-35%, i risultati di prodotti da questi programmi devono sempre essere valutati con cautela e pesati con coefficienti di sicurezza.

Origine e sviluppo delle metriche

La prima e più intuitiva misura del software è rappresentata dal numero

Iso 4143 - Functional Size Measurement

Parte	Titolo	Descrizione
1	Functional Size Measurement Concepts	Concetti generali di functional size measurement (FMS)
2	Compliance	Definisce il processo con cui un metodo può essere certificato formalmente come conforme
3	Verification	Definisce vari metodi con cui può essere determinata l'utilità di un metodo
4	Reference Model	Definisce un modello di riferimento per valutare se una metodologia produce i risultati attesi in determinate situazioni
5	Software Domains	Definisce il significato dei domini software
6	Guide to 14143 & related standards	Guida alle altre parti della norma 14143 e ai metodi che sono standard internazionali

Iso ha anche standardizzato quattro metodi di conteggio dei Function Point, che sono ovviamente conformi alla norma 14143

Standard ISO	Ente proponente	Descrizione
ISO/IEC 20926: 2003	IFPUG www.ifpug.org	Metodologia IFPUG (International Floating Point User Group) derivata dall'originale. Solo gli unadjusted FP (UFP) sono conformi a ISO 14143-1
ISO/IEC 20968: 2002	MKII www.uksma.co.uk	MKII Function Point Analysis di UKSMA (United Kingdom Software Metrics Association). Usata soprattutto in Inghilterra. Il metodo considera solo 3 componenti: Input, Output e Entity Reference.
ISO/IEC 19761: 2003	COSMIC www.cosmicon.com	Metodologia di COSMIC (Common Software International Consortium) che supporta anche i sistemi Real Time
ISO/IEC 24570: 2004	NESMA www.nesma.org	Metodologia olandese di NESMA (Nederlandse Software Metrieken Associatie)

Esempio di calcolo dei Fp

Elemento	Tipo	Bassa	P	Media	P	Alta	P	Totale
External Input	EI		3		4		6	0
External Output	EO		4	81	5		7	405
External Query	EQ		3	283	4		6	1.132
Internal Logical File	ILF		7	104	10	130	15	2.990
External Logical File	EIF		5		7		10	0
Totale FP								**4.527**

Produttività media in Fp/mese/persona per varie applicazioni

Tipo di sistema	FP/giorno	FP/mese
Client - Server	0,8	16
Gestionale su Main Frame	0,65	13
Web statico	1,15	23
Web e-Business	0,7	14
Data Warehouse	0,4	8
Valore Medio	**0,78**	**16**

Fonte: David Consulting Group

di linee di codice (Loc, Lines Of Code) che risale al 1974. Sebbene la misura delle Loc sia tuttora utilizzata, presenta alcuni problemi che hanno motivato la ricerca di metriche alternative.

Negli anni '70 nacquero metodi per misurare la dimensione tecnica e la complessità del software, come il *numero ciclomatico* di McCabe (1976), che misura la complessità logica degli schemi di flusso, e la *software science* di Halstead (1977) che misura il *programming effort* (cioè l'attività mentale richiesta per convertire un algoritmo esistente in una serie di istruzioni in un dato linguaggio di programmazione), nonché varie metriche funzionali, per ottenere una misura del software indipendente dal linguaggio di programmazione e da altri aspetti tecnici. Nel 1979 Allan Albrecht introdusse in Ibm il metodo della *Function Point Analysis*, che definisce le dimensioni dei prodotti software in termini di funzionalità fornite all'utilizzatore.

Rispetto alle Loc, le analisi prodotte dalle metriche funzionali mostrano una migliore correlazione con i costi di realizzazione, una maggiore applicabilità nelle fasi alte del progetto software, e segnano il passaggio dalle misure "fisiche" o "tecniche" o "dimensionali" a misure "logiche" o "funzionali" per la stima anticipata del software, la negoziazione contrattuale e la valutazione di un portafoglio di applicazioni. Nel corso di quasi un trentennio, la Function Point Analysis, standardizzata dall'Ifpug (International Function Point User Group), a partire dal 1986 si è evoluta e ha generato numerose varianti e proposte di miglioramento, come Feature Point (1986), Function Point Mark II (1988), Function Point Nesma (1990), Function Point 3D (1992), Micro Function Point (1995), Full Function Point (1997).

Nel 1998 un gruppo di esperti a livello internazionale hanno dato vita al Common Software Measurement Consortium Group (Cosmic), pubblicando il metodo dei Csomic Full Function Point, approvato come metodo standard dall'Iso/Iec nel 2003, insieme ai precedenti metodi proposti da Nesma, Mark II e Ifpug.

Parallelamente, si sono sviluppate le *misure di processo* e i relativi modelli di stima, come il modello Slim di Putnam (1978) e i modelli algoritmici Cocomo (1981) e Cocomo II (1997) di Barry Boehm.

Metodi di misura

Per la valutazione pratica del valore di un prodotto software si utilizzano metodi che si possono ricondurre a tre modelli logici: "lavoro impiegato/risorse impiegate", "valore di mercato"e "funzionalità offerte/quantità di codice prodotto". Ciascuno di questi metodi presenta valenze più o meno significative in funzione dell'ambito di applicazione e delle finalità della valutazione.

Il primo metodo si riduce alla valorizzazione del lavoro e delle risorse impiegate per la realizzazione di un prodotto/progetto software. Se si dispone della documentazione di progetti realizzati, completa di ore di lavoro, numero e tipo delle risorse impiegate in ciascuna delle fasi di sviluppo del progetto, è possibile stimare per analogia l'impegno richiesto dai progetti futuri.

Il valore di mercato di prodotti simili viene utilizzato principalmente per stabilire il prezzo di mercato delle licenze software o per decidere se acquistare un pacchetto esistente o svilupparlo in proprio.

Il modelli che misurano il valore intrinseco del software prodotto spaziano dal semplice conteggio delle linee di codice (Loc), a quelli più avanzati in cui la stima dei costi è basata su modelli matematici derivati da dati ottenuti dall'analisi dei progetti realizzati. Queste metodologie devono essere calibrate per essere applicate a organizzazioni e progetti differenti senza diminuire l'attendibilità.

Indubbiamente le metriche del software devono ancora raggiungere livelli di maturità e di standardizzazione adeguati, ma sono stati fatti passi importanti, ed è in corso l'integrazione di misure, metriche e indicatori, finora disgiunti o addirittura contrapposti.

Le metriche funzionali

Le metriche delle dimensioni funzionali o Fsm (*Functional Size Measurement Methods*) misurano le funzionalità che il software offre agli utenti, indipendentemente dagli aspetti tecnici e qualitativi.

I concetti relativi alla misura della dimensione funzionale del software sono stati definiti nel 1997 da Iso (*International Organization for Standardisation*) nello standard internazionale Iso 1413, composto da sei parti, che forniscono un metodo per normalizzare le misure di produttività e di funzionalità, stabilendo un criterio comune di misura dei prodotti realizzati. Lo standard definisce i concetti e le modalità delle misure quantitative della dimensione funzionale del software, ma non impone un particolare metodo per la misura delle dimensioni delle funzionalità, limitandosi a descrivere le caratteristiche che deve avere una metodologia per potere essere accettata. Attualmente la Function Point Analysis, basata sull'analisi di un'applicazione in base al numero e tipo delle informazioni in entrata, in uscita e memorizzazione, è la metrica funzionale più diffusa, applicabile a prescindere dal tipo di applicazione e dall'architettura di riferimento, ampiamente utilizzata in ambito internazionale e supportata da numerose associazioni, banche dati di progetti e società di consulenza, come Ifpug, Isbsg (International Software Benchmarking Standards Group), Spr (Software Productivity Research) e Gartner Group.

I Function Point sono di ausilio per stimare a priori l'impegno necessario per realizzare un progetto e a posteriori per calcolare il valore del prodotto e del patrimonio software.

Il metodo Fp è utilizzato dalla maggior parte delle pubbliche amministrazioni e delle grandi organizzazioni di tutto il mondo.

In Italia, dove l'interesse verso i Function Point è iniziato negli anni '90, è utilizzato dal Cnipa (Centro Nazionale per l'Informatica nella Pubblica Amministrazione) per la previsione dei costi di forniture software e per la determinazione della base per le relative gare, dalla Banca d'Italia, da grandi software house e da molte aziende, tra cui Fiat.

Il riferimento italiano degli utenti dei Function Point è Gufpi-Isma (*Gruppo Utenti Function Point Italia - Italian software metrics association*), nato nel 1990, che promuove lavori di ricerca e applicativi sulle regole di conteggio dei Fp, sul benchmarking e sulla misurazione in generale.

Produttività ottenibile con vari livelli di linguaggio

Livello del linguaggio	FP/mese min	FP/mese max
1 - 3	5	10
4 - 8	10	20
9 - 15	16	23
16 - 23	15	30
24 - 55	30	50
Oltre 55	**40**	**100**

Fonte: Capers Jones

Caratteristiche e calcolo dei Function Point Ifpug

Per eseguire la Function Point Analysis (Fpa) si suddivide un'applicazione in componenti elementari di tipo predefinito, si classificano in base alla loro complessità (bassa, media o alta) e si contano attribuendo a ciascuno un numero di Fp in base al numero e tipo delle informazioni in entrata (Ei), in uscita (Eo), alle query possibili (Eq) e alle strutture per la memorizzazione dei dati interne (Ilf) ed esterne (Eif) all'applicazione.

Il numero dei Fp così calcolato e moltiplicato per il costo unitario di un Fp (attualmente attorno ai 400-500 Euro/Fp, che corrisponde a una produttività media di 15/20 Fp persona/mese) fornisce la valutazione del valore del sistema.

Il valore finale ottenuto dipende dal valore dei pesi attribuiti durante la valutazione. L'errore del metodo varia da un minimo del 10% a un massimo che può anche essere superiore all'80%, con un valore tipico del 30% ed è simile per tutti i metodi di valutazione certificati Iso 14143. Per diminuire queste discrepanze è consigliabile utilizzare valutatori certificati, che seguono un procedimento uniforme per eseguire le valutazioni.

Per verificare l'attendibilità dei risultati, si possono incrociare i risultati con quelli ottenuti dal calcolo diretto dei Fp e da altri metodi di valutazione basati sul lavoro impiegato, sul numero di istruzioni sviluppate (Loc) e sulla produttività media dei programmatori. Si può verificare ulteriormente la congruità delle stime confrontando i risultati con i dati di benchmarking di progetti software riportati nella banca dati di Isbsg (International Software Benchmarking Standards Group - www.isbg.org), l'ente australiano di riferimento mondiale per il benchmarking dei progetti software che mantiene un database con i dati di migliaia di progetti, fornisce le produttività medie in Fp/mese/uomo in varie condizioni, e permette di confrontare i propri progetti con progetti simili. Infatti la produttività dipende da molti parametri, come il tipo di piattaforma, il linguaggio di programmazione, il tipo di software, la dimensione del progetto e l'esperienza del team di sviluppo.

Esistono anche altre basi di documentazione mantenute da società di consulenza come Gartner Group o David Consulting Group.

Il metodo "Backfire"

Il metodo "**Backfire**", introdotto nel 1985 da Capers Jones di Spr (Software Productivity Research), si basa su relazioni empiriche tra linee di codice sorgente (Loc) e Function Point per i vari linguaggi di programmazione. Il documento "Programming Languages Table", pubblicato da Spr riporta i fattori di conversione Loc/Fp di alcune centinaia di linguaggi di programmazione e attribuisce a ogni linguaggio di programmazione un "livello" che esprime il numero medio di istruzioni Assembler a cui equivale una istruzione del linguaggio stesso. Questi dati si applicano solo alla fase di codifica del progetto, che tipicamente pesa dal 50% al 30% del totale. L'accuratezza del processo di backfiring comporta generalmente un errore pari a +/- 25% o superiore.

Le tabelle consentono una stima approssimativa dei Fp se si conosce il numero delle linee di codice (Loc) o viceversa.

Produttività ottenibile con alcuni linguaggi di programmazione

Linguaggio	Livello	LOC/FP
Assembly Language	1	320
Basic ANSI	5	64
C	2,5	128
C++	6	53
Clipper	17	19
Cobol ANSI 85	3,5	91
Cold Fusion	18	18
Delphi	11	29
Excel 5 (spreadsheet)	57	9
Flex	7	46
Fortran 95	4,5	71
Fourth Generation Languages	16	20
HTML 4.0	23	14
Java	9	36
Java J2EE 1.4	18	18
OO Language default	11	29
Perl	15	21
PHP 5	25	13
PI/1	4	80
PowerBuilder	20	16
Spreadsheet Default	50	6
SQL default	25	13
Visula Basic .Net	14,92	21
Visual C++	9,5	34
Visual Studio .Net	20	16

Fonte: Capers Jones