Il riuso del software come potenziale fattore di inquinamento dei data base per il benchmarking in Function Point

Roberto Meli
(CFPS) - D.P.O. Srl

Abstract

Le attività di benchmarking, a cui questo lavoro fa riferimento, sono basate sulla raccolta di dati tecnici e gestionali misurati attraverso l'uso di opportuni sistemi metrici tra cui primeggiano oggi i Function Point. I data base di benchmarking contengono dati consuntivi relativi alle dimensioni ma anche ai fattori produttivi relativi a progetti software. Essi possono essere utilizzati per molti usi importanti anche e soprattutto ai fini gestionali e contrattuali come ad esempio l'assessment dei processi produttivi, la elaborazione di modelli di produttività o la valutazione di offerte software. E' dunque essenziale avere la certezza che i dati in esso contenuti, e quindi le decisioni che da essi ne derivano, non siano inquinati da errori di ordini di grandezza inaccettabili. I Function Point sono una delle migliori scelte possibili per il dimensionamento (sizing) delle applicazioni software ma - così come sono definiti attualmente – non sono ancora una metrica puramente funzionale ed il loro uso acritico in attività di benchmarking può portare a conclusioni errate se applicate a singoli e specifici progetti. Oltre a suggerire strade possibili per il miglioramento della metrica Function Point in senso funzionale, il presente lavoro evidenzia come il riuso del software possa essere un fattore potenzialmente inquinante della validità di confronto dei dati di un data base di benchmarking. Sono infine introdotti alcuni semplici accorgimenti che permettono di risolvere i problemi evidenziati.

1. Premessa

La produzione del software è un processo industriale che appare piuttosto refrattario ad inquadrarsi nell'ambito delle pratiche scientifiche o almeno ingegneristiche tradizionali. Questo è sicuramente dovuto all'essenza stessa del prodotto che è in primo luogo immateriale e quindi sottoposto ad un certo grado ineliminabile di soggettività. Quest'ultima caratteristica si rispecchia anche nella difficoltà di mettere a punto delle metriche ben fondate che possano consentire di misurare con affidabilità i vari aspetti del software. L'uso di metriche siffatte è però essenziale se si vuole governare il processo produttivo anziché esserne da questo governati. I Function Point sono stati introdotti per ovviare ad alcuni importanti inconvenienti che le metriche precedentemente usate, ed in particolare le Linee di Codice, manifestavano. Non tutti i problemi però sono risolti. A nostro avviso i Function Point devono subire una nuova evoluzione per rispondere ai mutamenti di scenario tecnologico e produttivo che sono avvenuti negli ultimi anni ed anche per risolvere, invero, alcune incongruenze che si portano dietro fin dalla nascita.

2. Il benchmarking

Il benchmarking (almeno quel tipo di cui ci interessiamo) può essere pensato come *l'attività del confrontare quantitativamente un certo fenomeno produttivo rilevato all'interno di un'organizzazione con una serie di fenomeni analoghi o con fenomeni medi in senso statistico riportati dall'ambiente esterno*.

Perché si ricorre al benchmarking ? I principali motivi per cui le organizzazioni utilizzano tale pratica sono :

- valutare lo stato dei propri processi produttivi in un certo momento (assessment);
- identificare il proprio posizionamento competitivo rispetto al mercato;
- stabilire punti di partenza e di arrivo per il Process Improvement;
- ed in generale ricavare indicazioni utili per le attività di previsione, di monitoraggio e di governo della realtà oggetto di confronto.

Il benchmarking del software può essere attuato efficacemente attraverso l'utilizzo di un sistema di archiviazione - possibilmente automatizzato - che permetta di raccogliere, validare ed analizzare i dati attraverso diversi strumenti di indagine sia qualitativa / analogica sia quantitativa / statistica.

Una volta disponibili, tali dati potranno aiutare a raggiungere obiettivi di secondo livello, ma talvolta non meno importanti di quelli canonici, come ad esempio:

- creare modelli matematici di produttività, di costo e di durata;
- prevedere impegni, tempi e costi per un nuovo specifico progetto software;
- valutare la congruità di un'offerta software da parte di un fornitore.

Un data base di benchmarking è, dunque, una collezione di dati tecnici e gestionali afferenti alle risorse, ai mezzi ed ai processi produttivi per il software. Essi includono dati preventivi e consuntivi di progetti conclusi da parte di varie organizzazioni e sono raccolti secondo precise e verificate modalità di sicurezza e di purificazione. I dati normalmente presenti vanno dalla nazionalità al tipo di organizzazione, dal tipo di progetto (sviluppo, manutenzione, porting etc) al suo dominio applicativo, dalla dimensione tecnica del software rilasciato (FP o KLOC) alla sua qualità, dall'impegno delle varie fasi di lavoro al tempo solare di rilascio e così via. Una volta inseriti i dati è possibile interrogare il data base filtrando i casi analoghi a quello su cui si desidera un confronto ed analizzando singolarmente i risultati oppure ottenendo dei dati medi con i rispettivi indicatori di dispersione e di correlazione statistici. E' spesso possibile applicare tecniche di indagine evoluta come l'analisi multifattoriale o le tecniche di correlazione.

Come in tutte le indagini empiriche, però, i dati non parlano da soli; occorrono, invece, delle ipotesi di relazione e legame tra le variabili registrate al fine di accettare o rigettare i modelli ipotizzati. Tale lavoro è generalmente materia da esperti di statistica perché le trappole che l'uso disinvolto degli strumenti di analisi possono disseminare nel terreno sono innumerevoli. Ciononostante l'utilità di un database di benchmarking è molto elevata e vale la pena di impegnarsi sia nella raccolta che nello studio di tali dati.

Come è già stato osservato, proprio per la delicatezza dei processi decisionali che tali dati supportano, è indispensabile avere la certezza di non includere nel data base una serie di osservazioni viziate da errori o disomogeneità di fondo che siano dello stesso ordine di grandezza delle relazioni numeriche ricercate.

Prima di addentrarci nel vivo dell'analisi chiediamoci: che relazioni esistono - nei progetti software - tra dimensione tecnica (size), impegno, durata lavorativa, staff e costo di produzione ?

Impegno e size

Come molti autori hanno già rilevato, tra impegno di sviluppo e size di un'applicazione software esiste sicuramente una relazione di dipendenza funzionale che vede coinvolte, però, innumerevoli altre variabili produttive in qualità di comparse ma talvolta di co-protagoniste. Alcune di queste variabili sono:

- metodologia di lavoro
- case tool
- linguaggi di programmazione
- piattaforme tecnologiche
- esperienza del team
- criticità ed affidabilità finali
- complessità e innovazione del problema da risolvere e del software
- qualità attesa
- contesto economico/organizzativo/competitivo
- nazionalità

E' comunemente accettato che il size sia il cosiddetto "*driver primario*" della relazione funzionale. Questo significa che le variazioni del size sono quelle che determinano in misura maggiore le variazioni dell'impegno collegato. E' per questo che si usa tenere conto delle variabili di produttività ausiliarie attraverso un Fattore di Aggiustamento Moltiplicativo che viene applicato a posteriori del calcolo dell'impegno a partire dal size. I modelli matematici che legano il size all'impegno sono dei più vari e vanno dalle equazioni lineari - le rette - a quelle esponenziali passando dalle polinomiali. La scelta di una o l'altra delle ipotesi è un fatto di convinzioni metodologiche, intuizioni, esperienze e talvolta (purtroppo) di puro gusto estetico. Saranno poi i dati empirici raccolti indipendentemente a confermare o rigettare i modelli produttivi proposti. Da qui discende l'importanza di avere a disposizione un serio data base di benchmarking.

Durata lavorativa, impegno e staff

Queste tre variabili sono, invece, legate tra loro in modo molto più diretto e vincolato della precedente coppia di variabili in quanto, per definizione di lavoro, questo è dato dal prodotto tra persone e tempo. Se c'è un lavoro di un mese persona da svolgere e abbiamo una sola persona a disposizione esso durerà (c'è qualcuno disposto a scommetterci ?) un mese lavorativo. Da molto tempo sappiamo che l'intercambiabilità tra mesi e persone non è verificata in tutti i punti dell'iperbole (tale è la curva matematica che rappresenta la relazione di interscambio dei due fattori) ma solo nella zona centrale, cioè nei punti non estremi. Un detto popolare che riassume molto bene quest'ultimo aspetto è il seguente: se una donna fa normalmente un bambino in nove mesi, nove donne non faranno mai un bambino in un mese!

Benché vi siano motivi di ritenere che un vincolo troppo stretto sullo staff - la sua qualità, la sua quantità - o sulla durata accettabile di un progetto influenzino l'impegno di sviluppo a parità di altre condizioni, si può affermare, in prima approssimazione, che è l'impegno ad essere usato come variabile primaria per determinare la durata e lo staff necessari al progetto.

Costo di produzione e impegno

Quando parliamo di costo di produzione ci riferiamo al costo del lavoro, delle attrezzature, dei fattori organizzativi e delle materie prime necessarie al progetto. Generalmente la parte più difficile da stimare è quella legata al lavoro da svolgere. Ed è qui che nasce l'esigenza di un modello che leghi, appunto, il costo del lavoro alle mansioni impiegate, al processo adottato, ai costi unitari delle risorse ed in primo luogo all'impegno da erogare. Tale modello, però, non pone particolari problemi concettuali per essere sviluppato ed è di tipo più amministrativo che tecnico.

Come si può vedere una volta fatto il primo passo – la produzione di un modello funzionale che leghi il size all'impegno di produzione – il resto è una strada in discesa. Ed è su questo primo passo, infatti, che si concentra in genere l'attenzione dei maggiori ricercatori aziendali.

Un errore che viene troppo spesso commesso in questo campo dalle imprese che producono e acquistano software, è quello di considerare la relazione tra size ed impegno come un rapporto costante espresso da un valore di produttività media. Non so se abbiano fatto più danni alle aziende le persone che girano armate del fatidico numero - un solo numero! - Function Point per mese persona o peggio Lira per Function Point con cui confrontare tutto ciò che capita a tiro o i consulenti che tali braccia hanno armate di valori la cui origine è spesso imperscrutabile ed insondabile.

Da quanto qui solo accennato appare chiaro, invece, che il legame tra size e impegno – e quindi costo di produzione – è mediato da una serie di fattori produttivi che possono mutare in modo estremamente significativo il rapporto di produttività. Ecco perché al semplice numero FP per mese persona dovremmo innanzitutto sostituire un'equazione che tenga in conto il fatto che la produttività non è affatto costante al variare delle dimensioni del progetto. Si cita spesso il fatto che progetti piccoli tendono ad essere più produttivi di progetti medi e questi a loro volta più produttivi di progetti grandi. In ogni caso è poco credibile che la produttività sia costante al variare delle dimensioni del software da sviluppare. Questo significa che al posto del fatidico numero dovremmo almeno usare una tabella che associ numeri diversi per ogni range di size in cui abbiamo diviso la scala di misura di quest'ultimo. Un problema non da poco di questa soluzione è, però, che la tabella è di fatto una funzione discontinua delle sue variabili. Ciò significa che progetti che differiscono tra loro solo di pochi FP potrebbero sviluppare previsioni di impegno distanti di decine di mesi persona. E' chiaro, dunque, che l'unica pratica corretta sia quella di usare una funzione continua e regolare come un'equazione matematica che leghi le due variabili primarie citate. Oltre a ciò, però, dovremmo dotarci anche di una serie di fattori di aggiustamento per tenere conto di tutte le condizioni produttive che condizionano la capacità di sviluppo come, ad esempio, quelle citate precedentemente.

Un data base di benchmarking dovrebbe considerare il più possibile tali fattori per poter consentire un'opportuna analisi multifattoriale. Purtroppo il principale, il più referenziato e quasi unico studio pubblico di questo tipo risale agli anni 80 ed è stato realizzato da Barry Boehm nel modello COCOMO.

Veniamo ora alla parte più delicata che è rappresentata dal legame tra costo di produzione e prezzo di mercato. Per esplorare questo rapporto è necessario introdurre nel discorso alcuni elementi di base della disciplina economica che ci permetteranno di scoprire una realtà alquanto bizzarra circa i Function Point.

3. Il software come merce di scambio

Da molto tempo ormai il software ha assunto la veste di merce di scambio ed è oggetto di transazioni economiche rilevanti al pari di altri e più tradizionali prodotti. Il tempo del software interamente sviluppato in casa è decisamente tramontato e tutte le organizzazioni ricorrono in misura più o meno elevata al mercato dei fornitori di software. Questo significa che il meccanismo di formazione del prezzo di un software è influenzato dalle leggi empiriche della domanda e offerta. Quello che ci interessa rilevare qui è la relazione tra Function Point e prezzo del software. Prima di ciò è opportuno ricordare alcuni elementi fondamentali della disciplina economica. [1]

Valore di scambio, valore lavoro e valore d'uso.
Il valore di un bene viene spesso definito come la sua qualità di poter venire scambiato con un altro bene. Il valore di un bene, in questa accezione, rappresenta quindi la proprietà del bene di acquistare altri beni o servizi attraverso lo scambio. In termini monetari si può sostenere che il valore di un'unità del bene considerato è rappresentato dal suo prezzo. Nella scienza economica esistono due teorie del valore: la teoria del valore-lavoro e quella dell'utilità. La teoria del valore-lavoro di Marx discende dalla teoria classica del valore. Stando ai classici (Smith, Ricardo), il valore di un bene dipende dal suo costo di produzione ed in particolare dal costo del lavoro. Così un bene che domandasse tre giornate di lavoro per essere prodotto dovrebbe valere di più di un bene che ne chiedesse solo due. Per Marx il valore di un bene è sempre ed unicamente rappresentato dalla quantità di lavoro socialmente necessario in esso incorporato, sia sotto forma di prestazioni dirette della forza lavoro nella sua produzione sia nella specie di prestazioni indirette incorporate nei beni capitali utilizzati nella sua produzione. Secondo questa teoria, dunque, il valore è una proprietà oggettiva del bene. I teorici dell'utilità avvicinarono, invece, il problema partendo dalla domanda dei beni. Per loro il valore non è più una proprietà oggettiva del bene stesso ma dipende dalla sua utilità, vale a dire dalla sua capacità di soddisfare uno o più bisogni del consumatore e quindi dalla relazione soggettiva tra bene e consumatore. Ne discende che il medesimo bene può avere valore diverso per consumatori diversi.

Il meccanismo di formazione del prezzo
In un'economia di mercato il prezzo viene determinato dall'incontro tra domanda ed offerta.

Nella teoria elementare della domanda di un bene la quantità domandata è determinata da una serie di fattori tra cui spiccano:

- il prezzo di mercato di tale bene
- il reddito del consumatore
- il prezzo di altri beni
- il gusto o le preferenze del consumatore
- le finalità del consumatore

Nella teoria elementare dell'offerta di un bene, invece, la quantità offerta viene determinata da una serie di fattori come:

- le finalità delle aziende produttrici
- il prezzo di mercato di tale bene
- il prezzo di altri beni
- il prezzo dei fattori di produzione
- il progresso tecnico

La teoria elementare del prezzo ci dice che esiste un unico prezzo che è in grado di eguagliare la quantità di bene offerta con quella domandata e quindi il valore di scambio o prezzo del bene non dipende in modo esclusivo ne' dal solo costo di produzione ne' dal solo valore d'uso.

Questa è dunque una conseguenza molto importante anche per il mercato del software: il prezzo monetario di una fornitura non è legato in modo esclusivo ne' al valore-lavoro (costo di produzione) ne' al valore d'uso (benefici per l'utente) ma solo al raggiungimento dell'equilibrio tra domanda e offerta che, come abbiamo visto sono influenzate da una serie molto più ampia di variabili.

Nel campo del software esistono due diversi scenari possibili, quello del package di consumo e quello dello sviluppo su commessa. Solo nel primo le leggi economiche appena citate sono applicabili in modo appropriato, coerentemente con le caratteristiche di mercato concorrenziale. Tuttavia nel secondo scenario i fattori elencati continueranno ad influenzare le singole decisioni di acquisto seppure in modo imperfetto. Per una singola fornitura di software da sviluppare su commessa più che di quantità domandata ed offerta di un bene si potrà parlare di disponibilità di acquistare o vendere quel particolare bene a determinate condizioni.

Dal momento che nel mercato software, sia di consumo che su commessa, il prezzo del bene è legato ad una molteplicità di fattori di pari importanza questo vuol dire che non esiste un driver primario a cui collegare il prezzo in modo semplice. Non possiamo aspettarci, dunque, una forte correlazione tra le variabili prezzo e size. Se questa si manifestasse, in un particolare insieme di dati raccolti, nasconderebbe quasi certamente una realtà media fatta di inapparenti sprechi o forzate perdite. Non sembra una buona idea, dunque, adottare un fattore fisso di conversione tra Function Point e prezzo di mercato in denaro per valutare uno specifico progetto. E' una buona idea, invece, utilizzare tale rapporto come un macroindicatore di posizionamento competitivo generale dell'organizzazione in quanto sui numeri più grandi gli errori tendono a compensarsi.

Che conseguenze ha quanto detto rispetto ai Function Point ed al benchmarking?

4. I Function Point tra valore-lavoro e valore d'uso

Si usa dichiarare che i Function Point misurano il size funzionale di un'applicazione software dal punto di vista dell'utente esperto e nello stesso tempo li si vuole legati all'utilità che il software riveste per l'utente medesimo cioè al suo valore d'uso. Sarebbe come dire che il chilogrammo con cui misuriamo il pane è una misura anche del grado con cui quella quantità di pane può soddisfare la fame di un consumatore. E' verosimile una tale proprietà ? Quanto più cresce il size tanto più cresce il valore d'uso ? E dal momento che la teoria utilitaristica prevede che per ogni bene non esista un solo valore d'uso ma tanti quanti sono i diversi consumatori (utenti in questo caso) del software dovremmo avere tante misure diverse in Function Point per quanti diversi punti di vista esistono ? Per rispondere a queste domande cerchiamo di capire chi sia l'utente e cosa sia il valore d'uso per esso.

Negli standard IFPUG 4.0 [2] non è data una chiara ed inequivocabile definizione di Utente. In più punti è suggerito, però, un significato del termine allargato a comprendere l'utilizzatore diretto, quello indiretto, il responsabile gerarchico, l'utente tecnico operativo etc. Per Utente esperto si può intendere allora un'astrazione, una figura virtuale che in realtà è la composizione di diverse figure fisiche che sono legittimate ad esprimere requisiti sul progetto software guidate in questo da un esperto di tecniche di analisi. In questo modo si risolve il problema legato alla molteplicità degli utenti. Per ogni applicazione esiste uno ed un solo utente esperto che è la composizione di tutti i soggetti indicati precedentemente. Non a caso nella recente disciplina del Requirement Engineering, gli approcci basati sui Punti di Vista si stanno guadagnando un momento di gloria.

Cos'è, dunque, il valore d'uso del software per un soggetto fisicamente inesistente benché fin troppo esigente come quello delineato ?

Una prima considerazione riguarda il fatto che in ogni progetto software esistono funzioni più rilevanti di altre. Una funzionalità che permette di controllare il traffico aereo non sarà ugualmente importante di una che stabilisce su quale stampante dirigere il log-file delle chiamate del giorno ! Se si assume come ipotesi semplificatrice che tutte le funzioni sono democraticamente uguali e che il size funzionale è legato più al numero di esse che alla loro missione intrinseca anche il secondo elemento di soggettività viene a cadere. Questo è esattamente quello che si fa quando si conteggiano campi elementari ed archivi referenziati per determinare la complessità di un External Input o di un altro processo elementare. Se questa approssimazione fosse accettabile potremmo dire che i Function Point sono legati alla quantità di cose che è possibile fare con un certo pezzo di software e che questo corrisponde al suo valore d'uso.

Sono accettabili queste due ipotesi che abbiamo fatto ? La risposta è a nostro avviso positiva: non abbiamo motivo di pensare che tali assunzioni possano introdurre più problemi di quanti non ne risolvano, considerando che siamo nel campo di beni immateriali dove la soggettività è – in qualche misura - ineliminabile.

Il problema che riguarda i Function Point giunge, in realtà, da un'altra direzione. Per come sono definiti oggi i Function Point, infatti, essi sembrano più legati al valore-lavoro che non all'utilità per l'Utente. Se così fosse avremmo una misura intrinsecamente incoerente che vorrebbe misurare il valore d'uso del software ma in realtà è legata ai suoi costi di produzione cioè al valore-lavoro. Abbiamo visto, invece, come in un'economia di mercato il valore d'uso non è collegato se non fortuitamente ai suoi costi di produzione. Per usare una metafora è come se volessimo misurare il pane non con il chilogrammo (misura abbastanza proporzionale alla capacità del bene di sfamare un consumatore) quanto con le ore di lavoro necessarie a produrre quello stesso pezzo di pane ! Con la varietà offerta dalle moderne tecnologie potrebbe accadere che una certa quantità di pane possa essere prodotta in mezza giornata oppure in una ora pur mantenendo la medesima capacità di sfamare. Una misura del genere non sarebbe affatto legata all'attitudine del software di soddisfare il bisogno del consumatore ma solo alle condizioni produttive. Strano destino per una metrica nata per il mercato quello di essere perfettamente coerente con un approccio marxista !

Gli elementi che portano a sostenere una tesi come la precedente sono principalmente due:

1. l'esistenza nel metodo di conteggio IFPUG del Value Adjustment Factor (VAF; VAFA; VAFB) che non aggiunge in realtà una sola funzione elementare a quelle richieste dall'utente esperto ma che, caso mai, rappresenta più un modo di considerare le difficoltà realizzative attraverso l'aumento o diminuzione fino al 35% del valore funzionale puro in base al maggiore o minore costo di produzione;
2. la composizione dei pesi relativi dei processi elementari EI, EO, EQ, ILF ed EIF che sembrano essere stati assegnati più sulla base della difficoltà realizzativa legata alle tecnologie degli anni in cui i Function Point sono nati, che non della percezione di utilità per l'utente esperto. Perché, ad esempio, un External Input deve pesare meno di un External Output ? Forse perché con le tecnologie a cui facevamo riferimento era più facile progettare e realizzare una maschera di acquisizione dati che un report di stampa. Oggi le cose potrebbero essere addirittura invertite.

Occorre, dunque, condurre i Function Point alla pura misura di size o – dopo quanto affermato – del valore d'uso del software, abbandonando ogni forma di commistione con i costi / impegni di produzione. Questo non deve spaventare tutti coloro che cercano nelle metriche del software degli strumenti per gestire previsioni, progetti e contratti e per i quali il legame tra size e impegno è di fondamentale importanza. In realtà anche loro hanno tutto da guadagnare da una depurazione degli aspetti non funzionali dalla metrica Function Point in quanto, se questo avverrà, potranno avere a disposizione modelli più robusti e consistenti anche per il calcolo delle variabili economiche.

Al riguardo della composizione dei pesi relativi dei processi elementari, appare estremamente promettente, per l'evoluzione dei Function Point, il lavoro sviluppato da Wittig, Morris, Finnie e Rudolph [3] per l'introduzione di un approccio metodologico formale alla definizione delle percezioni Utente circa i pesi relativi dei fattori alla base della Function Point Analysis. Lo studio citato va esattamente nella direzione di svincolare totalmente i pesi degli EI, EO, EQ, ILF ed EIF dalla difficoltà realizzativa legandoli, invece, al valore d'uso di cui abbiamo trattato finora. Come primo risultato possiamo concludere che i pesi originati dalla ricerca citata sono abbastanza differenti da quelli inclusi nelle tabelle standard del CPM IFPUG 4.0 (questa è una opinione diversa da quella degli autori).

Il secondo intervento indispensabile è quello di abbandonare l'uso del VAF (Value Adjustment Factor) perlomeno come modificatore della misura del size dell'applicazione recuperandolo eventualmente o come qualificatore (l'applicazione è tanto migliore quanto più il VAF è elevato) o come modificatore del valore della produttività che influenzerà i costi di produzione ma non il size dell'applicazione. [4]

5. Il riuso del software come fattore potenzialmente inquinante per il benchmarking

Abbiamo già visto che un data base per il benchmarking è uno strumento utile per collezionare informazioni storiche al fine di elaborare previsioni per il futuro e valutazioni per il presente che riguardano il software, in particolare per i processi produttivi ad esso legati. In altri termini esso può essere visto come un valido supporto per la creazione di modelli di produttività e come mezzo di validazione delle proposte di mercato. E' essenziale, però, che i dati raccolti nel data base siano il più possibile esenti da errori di tipo sistematico o casuale ma comunque di entità elevata. Al di là delle precauzioni gestionali che si possono prendere al fine di assicurarsi che il processo di raccolta dei dati sia sicuro, affidabile, riservato e fedele alla realtà, esistono dei potenziali pericoli che derivano dalla natura stessa delle metriche alla base della raccolta. In particolare vedremo come la pratica del riuso del software – in tutte le possibili forme in cui si manifesta – può inquinare in modo rilevante la validità dei confronti fatti con un data base che non ne tenga conto in alcun modo.

Come si è già avuto occasione di scrivere, la produttività è una funzione multifattoriale legata ad un numero molto ampio di variabili attribuibili alle seguenti classi principali:

- caratteristiche del problema da risolvere
- caratteristiche del prodotto da realizzare o esistente
- caratteristiche del processo produttivo
- caratteristiche delle tecnologie da utilizzare o utilizzate
- caratteristiche dell'ambiente lavorativo di sviluppo
- caratteristiche del gruppo di lavoro
- caratteristiche del contesto

L'uso di una relazione funzionale semplice tra impegno e size in Function Point è possibile solo se quest'ultimo è un driver primario cioè se la sua influenza è di almeno un ordine di grandezza superiore a quella di altri fattori che determineranno il "rumore" del modello o il range di errore probabile. Questa assunzione è probabilmente vera per tutti i fattori tranne che per il riuso in quanto esso può influenzare in modo significativo il size stesso cioè il candidato driver primario.

Un esempio dovrebbe chiarire quanto scritto finora. Supponiamo di avere un progetto di sviluppo di un'applicazione conteggiata, secondo le norme standard IFPUG 4.0, in 500 FP. Supponiamo anche che in tale occasione ci sia la possibilità di recuperare da un precedente progetto una serie di moduli software già realizzati e testati autonomamente per un totale di 100 FP e che vengano reperiti sul mercato dei componenti software altri 150 FP già pronti per essere inseriti nel software progettato ad hoc. In questo caso avremo che circa la metà delle funzionalità previste per l'applicazione da sviluppare saranno ottenute con un minimo o addirittura nessuno sforzo. Supponiamo infine che il nostro data base di benchmarking sia stato nutrito con dati esclusivamente forniti da organizzazioni che sviluppano interamente il software in casa senza alcun tipo di riuso. Se la media della produttività sui progetti del data base analoghi a quello che ci riguarda fosse 10 FP/mese persona ed applicassimo una semplice formula che, ad esempio, divide i 500 FP per 10 FP/mp otterremmo che per sviluppare il nuovo progetto occorrono 50 mesi persona quando, con tutta probabilità e tenendo conto del risparmio per riuso, ne bastano forse la metà. Lo spreco è assicurato dalla nota proprietà per cui un progetto software è come un gas che tende ad occupare tutto lo spazio che gli è concesso dilatandosi fino ad assorbire i 50 mesi persona senza che vi sia neanche il sospetto che se ne potrebbero evitare una buona metà.

Lo scenario esemplificativo appena descritto è un caso niente affatto ipotetico e che sarà sempre più norma negli anni a venire quando il mercato delle componenti sarà sviluppato come quello dei packages ed i sistemi informativi si potranno ottenere attraverso l'assemblaggio di parti, per così dire prefabbricate, già pronte perché sviluppate internamente od esternamente, poco importa.

Un data base di benchmarking che non differenzi i progetti in base alla percentuale di riuso del software utilizzata per ogni caso rischia di essere inquinato da dati imparagonabili tra loro. D'altra parte non è pratico confidare che la produttività media sia frutto della composizione bilanciata di livelli diversificati di riuso e che questo smussi gli spigoli del problema. Non è utile, infatti, fare una media di tutti i diversi livelli di riuso come per gli altri fattori perché sul singolo progetto questo parametro potrebbe assumere entità troppo rilevanti. Il valore delle medie di produttività a prescindere dal riuso, invece, è utile come indice macroeconomico di confronto tra realtà organizzative diverse perché, dal punto di vista competitivo, saper sfruttare il riuso è un fattore di vantaggio sulla concorrenza.

6. Come considerare adeguatamente il riuso del software nei modelli

Il Counting Practices Committe del Gruppo Utenti Function Point Italia (GUFPI) ha affrontato recentemente la questione [5] evidenziando le argomentazioni che seguono.

Il manuale CPM 4.0 riporta a pag. 2-2 la seguente affermazione: "i function point misurano il software quantificando le funzionalità in esso contenute e fornite all'utente basandosi principalmente sul disegno logico."

In effetti le funzionalità già esistenti ed incorporate - attraverso l'acquisizione esterna di package generalizzati o l'utilizzo casalingo di moduli sviluppati in altre occasioni - sono pur sempre funzionalità richieste ed ottenute dall'utente e pertanto vanno conteggiate così come quelle realizzate ex-novo al fine di ottenere la misura in FP del dimensionamento (size) dell'applicazione. Questo significa che il riuso non è un fattore che può influenzare il size almeno dal punto di vista esterno del valore d'uso. Esso, però, sicuramente influenza il lavoro da svolgere e quindi il conseguente costo di produzione eventualmente rendendo inapplicabile il legame funzionale tra size ed impegno che abbiamo faticosamente costruito attraverso il ricorso al data base di benchmarking. Come si può risolvere questo conflitto tra FP rilasciati all'utente (tutti) e quelli sviluppati dal progetto (solo una parte) utili ai fini previsionali ?

Un possibile approccio al problema precedente è il seguente: definire per ogni progetto due misure diverse in Function Point: una legata alla vista utente esterna del software che corrisponde ai Function Point così come sono definiti attualmente e l'altra legata alle necessità gestionali e produttive interne al costruttore di software che vuole conoscere quante funzionalità devono essere sviluppate più o meno ex-novo in modo da prevedere ed assegnare le sole risorse necessarie e sufficienti per la realizzazione dell'applicazione. Questa nuova misura potrà chiamarsi Developed Function Point (DFP). I Developed Function Point terranno conto delle sole funzioni che occorre sviluppare interamente o in parte ma non di quelle che risultano ereditate senza sforzo. In questo modo le previsioni di impegno basate su una produttività storica misurata in DFP per mese persona, non saranno inquinate dal fenomeno del riuso.

Per determinare praticamente i Developed Function Point a partire dai Function Point basterà assegnare ad ogni elemento (EI, EO, EQ, ILF, EIF) classificato e valutato secondo le tabelle dei contributi standard, un fattore moltiplicativo che assuma valori da 0 a 1 in base al risparmio stimato per via del riuso per quel particolare elemento. Sommando, poi, nel modo usuale i contributi modificati con i nuovi coefficienti di riuso, otterremo il valore globale dei DFP che sarà inferiore o uguale a quello dei FP.

In un data base di benchmarking dovrebbero essere presenti entrambe le misure al fine di poter elaborare dei modelli di produttività interni ed esterni.

Un esempio tratto dal Counting Practices Manual IFPUG 4.0 ed opportunamente modificato dovrebbe chiarire l'utilizzo del metodo proposto:

.

Transactional Function Types	FTRs	DETs	Functional Complexity	UFP	Reuse	DUFP
External Inputs						
Assignment report definition	1	5	Low	3	Low/0.8	2.4
Add job information (screen input)	1	7	Low	3	None/1	3
Add job information (batch input)	2	6	Low	3	High/0.4	1.2
Correct suspended jobs	1	7	Low	3	Very H/0.2	0.6
Employee job assignment	3	7	High	6	All/0	0
External Outputs						
Jobs with employees report	4	5	Average	5	Low/0.8	4
New dependent transactions to Benefits	1	5	Low	4	Very H/0.2	0.8
Notification message	3	4	Low	4	Low/0.8	3.2
Employees by Assignment Duration Report	3	7	Average	5	Low/0.8	4
External Inquiries	**In/Out**	**In/Out**				
List of retrieved data	2/1	2/3	Low	3	High/0.4	1.2
Drop-down list box	1/1	2/1	Low	3	High/0.4	1.2
Field level help	1/1	2/4	Low	3	None/1	3

Per questo esempio avremmo 45 Unadjusted FP e soli 24.6 Developed Unadjusted FP.

Al fine di standardizzare le tipologie di riuso, le corrispondenti percentuali numeriche ed i valori di aggiustamento moltiplicativi è stata avviata un'attività di ricerca che vede coinvolti professionisti di varie realtà organizzative con l'obiettivo di ridurre il più possibile la soggettività nell'attribuzione dei pesi e dei coefficienti. Il risultato sarà presentato in un apposito rapporto tecnico al termine della sperimentazione.

7. Conclusioni

Il presente lavoro costituisce uno stimolo contemporaneo all'evoluzione degli standard IFPUG sui Function Point - per liberare questa preziosa metrica dalle impurità della sua componente non funzionale - ma anche alla strutturazione dei data base di benchmarking – affinché si tenga in giusto conto il riuso come fattore potenzialmente inquinante o migliorativo dei dati e delle decisioni che ne derivano.

8. Referenze

[1] Richard G. Lipsey, Introduzione all'economia, Etas Libri, 1981
[2] Function Point Counting Practices Manual, Release 4.0, *International Function Point Users Group*, Blendonview Office Park, 5008-28 Pine Creek Drive, Westerville, OH 43081-4899, USA, 1994
[3] Wittig, Morris, Finnie e Rudolph, Formal Methodology to Establish Function Point Coefficients, IFPUG - Fall Conference - Scottsdale, Arizona USA- September 15-19, 1997
[4] Roberto Meli, Early and Extended Function Point: a new method for Function Points estimation - IFPUG - Fall Conference - Scottsdale, Arizona USA- September 15-19, 1997
[5] Linee Guida Italiane per il conteggio dei Function Point, Counting Practices Committee Gruppo Utenti Function Point Italia, http://www.gufpi.com/cpc, 1998

9. L'autore

Roberto Meli
Si è laureato con lode in Scienze dell'Informazione presso l'Università degli Studi di Bari nel 1984. Dallo stesso anno ha iniziato a svolgere attività di consulenza e formazione per alcune tra le maggiori organizzazioni italiane. Esperto in Project Management e Metriche del Software, ha scritto lavori per congressi internazionali e per riviste del settore Information Technology. Ha seguito ed erogato corsi di formazione all'estero in ambiti qualificati ed ha ottenuto la nomina di Certified Function Point Specialist riconosciuta dall'IFPUG (International Function Point Users Group). E' coordinatore del Counting Practices Committee del GUFPI (Gruppo Utenti Function Point Italia). Dal 1990 è Direttore della D.P.O. Srl.
e-mail: roberto.meli@iol.it http://web.tin.it/dpo ; http://www.dpo.it