# Un Sistema Basato sulla Conoscenza per il Calcolo dei Function Point

Fausto Gramantieri[1], Evelina Lamma[1], Paola Mello[2], Fabrizio Riguzzi[1]

[1]DEIS - Universita` di Bologna
Viale Risorgimento, 2
40136 Bologna
{elamma,friguzzi}@deis.unibo.it

[2]Dipartimento di Ingegneria, Universita` di Ferrara
Via Saragat, 1
44100 Ferrara
pmello@ing.unife.it

## Extended Abstract

Il processo di sviluppo di software, come ogni altra disciplina ingegneristica, necessita di una valutazione o *misura*. La misura puo` essere applicata a tutte le parti coinvolte nel processo (risorse umane, programmi software, etc.) con l'obiettivo di prendere decisioni migliori e migliorare il processo stesso. Lo sviluppo del software richiede un'alta percentuale di costi fissi od inelastici che non sono associati con la codifica e non possono essere misurati da tecniche quali metriche del software basate sul conteggio delle linee di codice. La disponibilita` di una metrica applicabile nelle prime fasi di sviluppo di un prodotto software e` quindi un fattore di primaria importanza.

Albrecht [Albrecht 79], [Albrecht and Gaffney 83] ha proposto un metodo di misura software (noto come Metodo dei Function Point ) indipendente dal tipo di linguaggio usato per la codifica e applicabile gia` nelle fasi iniziali di analisi e specifica del processo di sviluppo del software. Il metodo si basa su alcune componenti (misurabili) del software classificate in cinque tipi diversi:

1) input all' applicazione;
2) output dall' applicazione;
3) richieste dell' utente;
4) file di dati propri dell'applicazione;
5) interfacce con altre applicazioni;

Queste cinque componenti vengono combinate attraverso una serie di fattori di peso empirici e una serie di regole per valutare la complessita` e ottenere una misura del software che si andra` a sviluppare.

L'International Function Point Users Group pubblica il manuale sulle regole del conteggio, che rappresenta lo standard per l'applicazione di questa metrica [IFPUG 94].

I Function Point (f.p., per brevita` nel seguito) forniscono, entro un certo ambito, un modo di predire in maniera sufficientemente precisa il numero di linee di codice sorgente che devono essere scritte per un progetto in funzione del linguaggio usato. Per ogni linguaggio di programmazione variera` il numero di istruzioni richieste per realizzare un function point.

Questa forma di misura e` di notevole interesse in quanto puo` fornire un'indicazione dello sforzo richiesto, in linee di codice, per convertire un'applicazione da un linguaggio a un altro e della produttivita` di progetti scritti in diversi linguaggi.

Il calcolo dei f.p. varia in funzione del:

- Tipo di conteggio (a seconda si tratti di un progetto di sviluppo, manutenzione evolutiva o applicazione);
- Confine dell'applicazione (ovvero della linea di separazione fra l' applicazione, od il progetto che si sta misurando e le applicazioni esterne, od il dominio utente)
- Calcolo delle funzioni di tipo dati (classificati come Internal Logical File o ILF ed External Interface File o EIF). Il numero di file logici e la loro relativa complessita` funzionale (basata sul numero di Data Element Type o DET e sul numero di Record Element Type o RET) determinano il contributo delle funzioni di tipo dati al numero dei f.p. detti non pesati;
- Calcolo delle funzioni di tipo transazione, che rappresentano le funzionalita` fornite da un'applicazione per elaborare i dati (definite come External Input o EI, External Output o EO ed External Query o EQ, a seconda della provenienza e della destinazione dei dati oggetto della computazione);
- Fattore di aggiustamento che tiene conto delle caratteristiche generali del sistema in cui l'applicazione sara` eseguita, per calcolare infine il numero finale dei f.p. pesati.

Gli strumenti esistenti per il calcolo dei f.p. sono ancora ad un primo stadio di sviluppo, con funzionalita` abbastanza limitate e sono molto spesso semplici fogli elettronici. La complessita` del metodo dei f.p. non e` semplicemente dovuta al calcolo, ma piuttosto a una parte che possiamo chiamare decisionale, completamente ignorata dai mezzi automatici esistenti e non facilmente traducibile in linguaggi di programmazione tradizionali a causa della sua informale espressione in linguaggio naturale che identifica gli elementi di ingresso al conteggio.

E` invece altamente desiderabile automatizzare il piu` possibile l'identificazione dei valori di ingresso al metodo di conteggio (relativi rispettivamente a dati - ILF, ELF- e transazioni - EI, EO, EQ -), almeno nel caso in cui il documento prodotto al termine della fase di specifica sia gia` scritto in un linguaggio formale o semi-formale.

In questo lavoro abbiamo preso in considerazione il caso di specifiche espresse in notazione grafica, ed in particolare che utilizzano i diagrammi Entita`-Relazione (ER, [Chen 76]) e i Diagrammi di Flusso dei Dati (DFD, [DeMarco 78]) e abbiamo sviluppato un sistema basato sulla conoscenza che identifica tali parametri di ingresso a partire dai diagrammi ER e DFD. Il problema e` quindi assimilabile ad un problema di classificazione. La soluzione a problemi di questo tipo viene scelta da un insieme i cui elementi (limitati in numero) sono gia` stati enumerati. Nel nostro caso le soluzioni sono rappresentate dai due tipi di funzioni di tipo dati e dai tre tipi di funzioni di tipo transazioni che devono essere associate agli elementi in ingresso al sistema, quali entita` e processi, in modo da raggrupparli in classi e calcolarne il contributo.

Il sistema e` stato realizzato utilizzando l'ambiente Kappa su Personal Computer [Kappa 92], realizzando una gerarchia che rappresenta i diagrammi ER e DFD in ingresso (si veda Figura 1) e con un insieme di regole che identificano file logici di dati e transizioni (anch'essi rappresentati da classi nella gerarchia).

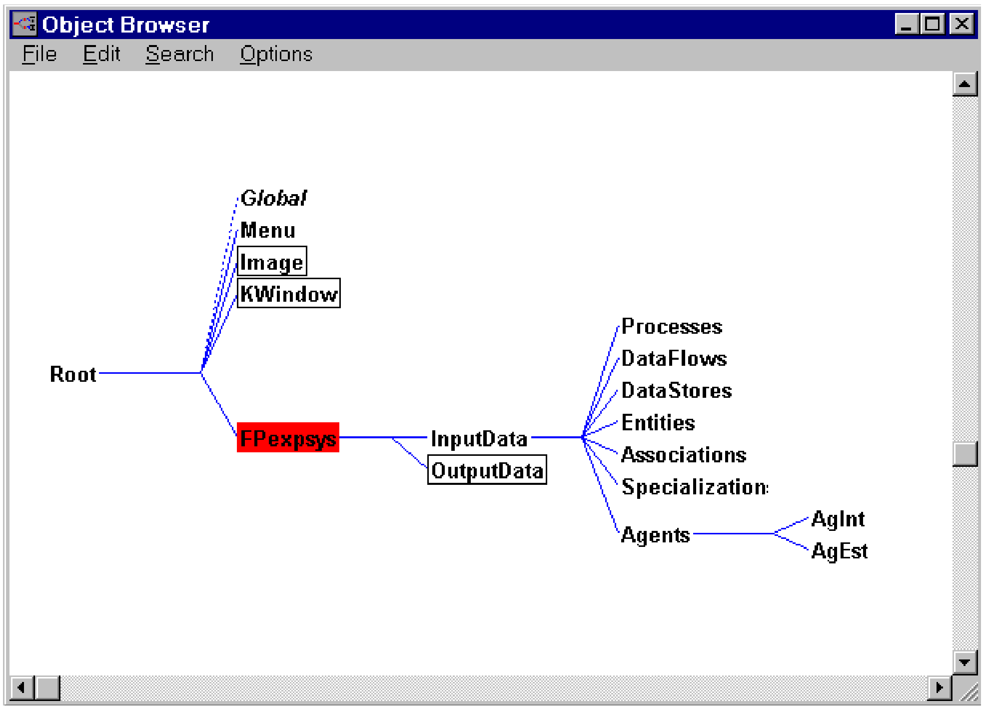


**Figura 1**

La radice della gerarchia (creata direttamente sotto alla classe di sistema Root) e\` FPexpsys (Function Point Expert System) e comprende tutte le sottoclassi necessarie al sistema. La sottoclasse InputData, il cui uso e\` dedotto dal nome, e\` divisa in Agents, Processes, DataFlows, DataStores, Entities, Associations, Specializations. La classe Agents e\` divisa in AgInt ed AgEst per poter includere sia dati riguardanti l'applicazione in esame, sia riguardanti le applicazioni esterne. In Figura 1 e\` riportata la struttura della gerarchia. Le istanze di (alcune delle) classi introdotte rappresenteranno gli elementi dei diagrammi ER e DFD in ingresso. Altre classi rappresentano gli elementi da identificare.
Le classi Agents, Processes, DataStores, Entities devono essere identificate dal loro nome, in quanto sono gli elementi principali, mentre le classi DataFlows Associations e Specializations vengono determinate univocamente dagli elementi principali (processi ed entita\`) che legano. Ad esempio, le istanze della classe Entities hanno un nome coincidente con quello della entita\` del diagramma ER che rappresentano, inoltre possono comprendere altri slot che indicano i vari attributi dell'entita\`. Lo slot KeySlotList identifica gli attributi che compongono la chiave primaria. Le istanze della classe Associations hanno gli slot Source e Destination che indicano le entita\` coinvolte dalla relazione e gli slot Card_min e Card_max che ne indicano la cardinalita\` minima e massima.

Ad esempio, le regole che riguardano l'identificazione degli Internal Logical File considerano ogni entita\` dei depositi di dati interni all'applicazione, non gia\` compresa in un file logico interno. Un'entita\` di un diagramma ER puo\` essere un ILF se tutte le relazioni che ha con con le altre entita\` hanno cardinalita\` minima uguale a zero. Questo corrisponde ad un modello grafico del tipo:

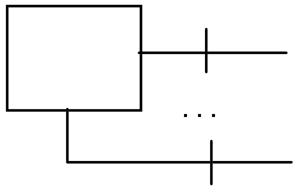

**Figura 2**

In tutti gli altri casi (nel caso di associazione sia 1:1 sia 1:n) le due entita` coinvolte e l'associazione che le lega identificano un ILF. Ogni elemento di un ILF deve possedere uno o piu` campi che non siano chiavi primarie per alcun altro elemento della classe ILF. Ciascuno di questi campi rappresenta un DET per l'ILF identificato. Nessun elemento della classe ILF deve essere destinatario di flussi di dati provenienti da agenti esterni o processi che appartengono ad agenti esterni.
La regola che identifica gli External Input considera ogni processo del Diagramma di Flusso dei Dati. Per ciascuno, si verifica che esista almeno un flusso di dati che parta dall'esterno dell'applicazione e sia diretto verso il processo e da questo verso un deposito dati. Questo corrisponde ad un modello grafico del tipo:

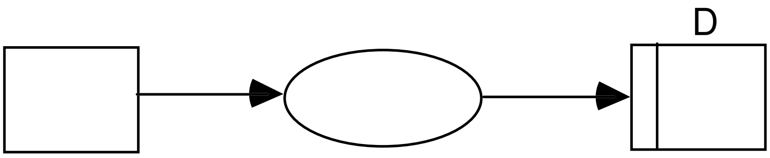


**Figura 3**

La regola che individua gli External Output e` esattamente duale, corrispondendo ad un modello grafico del tipo:

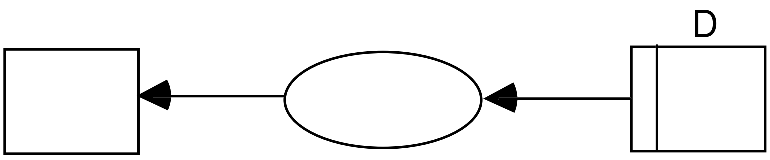


**Figura 4**

e quella per identificare le External Query ad un modello grafico del tipo:

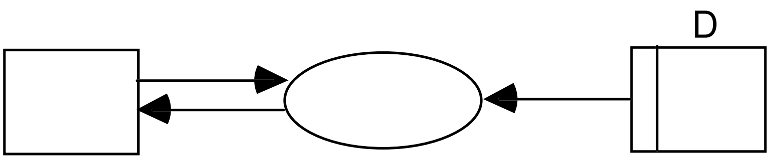


**Figura 5**

In tutti e tre i casi i flussi in uscita o in ingresso al processo relativi a dati devono essere indirizzati o provenire da un deposito dati D che contiene entita` raggruppate in almeno un elemento della classe ILF.

Il sistema e` stato realizzato utilizzando l'ambiente Kappa (ver. 2.0) per personal computer dotati di sistema operativo MS DOS ed ambiente Windows 3.0. Attualmente si sta verificando il sistema realizzato su alcuni esempi. Nel futuro si intende sviluppare un interfaccia utente che consenta di immettere la descrizione ER e DFD in modo piu` agevole all'utente, eventualmente anche per via grafica.

## Bibliografia


| | |
|---|---|
| [Albrecht 79] | A. Albrecht. *Measuring application development productivity.* in *Proc. Joint SHARE/GUIDE/IBM Applications Development Symposium*, Monterey, CA, 1979 |
| [Albrecht, Gaffney 83] | A. Albrecht, J. Gaffney: *Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation*; in IEEE Trans. Software Eng., 9(6), 1983, pp. 639-648 |
| [Chen 76] | P.P. Chen. *The Entity-Relationship model. Toward a unified view of data.* ACM Transactions On Database System, Vol. 1, No. 1, Marzo, 1976 |
| [DeMarco 78] | T. DeMarco. *Structured Analysis and System Specification.* Yourdon Press, New York, 1978. |
| [Kappa 92] | *Kappa-PC User Guide*, IntelliCorp Inc. Novembre 1992. |
| [IFPUG 94] | International Function Points User Group, *Function Point: Manuale sulle Regole del Conteggio*, *Versione 4.0*, 1994. |