STIMA DEI COSTI DI SVILUPPO DEL SOFTWARE

Di Michele de Nittis

1. Generalità

Uno dei passi fondamentali dello sviluppo di un sistema software è la valutazione del suo impatto sociale (macroeconomico) ed aziendale (microeconomico). Tale valutazione, insieme alla stima dei costi e dei benefici, rientra in una delle prime fasi del *ciclo di sviluppo del software*: *lo studio di fattibilità*.

Nel seguito ci si concentrerà esclusivamente sugli aspetti microecnomici del problema.

2. Costi e Benefici del progetto SW

Nella valutazione dei costi e dei benefici di realizzazione di un sistema SW si devono tener in conto tre fattori:

1. Valutazione dell'Hardware, del Software di Base e dei Pacchetti Specifici (c.d. piattaforma);
2. Valutazione dello sforzo di sviluppo del SW applicativo;
3. Valutazione delle risorse umane da impiegare nelle fasi di formazione, avviamento e sviluppo.

2.1. Valutazione dell'Hardware, del Software di Base e dei Pacchetti Specifici

Non esiste una metodologia scientifica generale e consolidata per questo tipo di valutazione, infatti:

1. per ogni specifica di un progetto SW sono possibili più architetture HW;
2. i sistemi hardware ed i sistemi software di base sono soggetti ad una continua dinamica innovativa e ad una evoluzione della dinamica di scala (ad esempio un sistema originariamente costituito da elaboratori connessi ad una rete paritetica si evolve in architetture più grandi e sofisticate);
3. una medesima piattaforma/ambiente operativo può essere impiegato per diversi progetti.

La scelta dell'HW e del SW di base, tuttavia, viene facilitata nei casi in cui:

1. esista un bagaglio di consolidata esperienza più che decennale nel dimensionamento e scelta dei sistemi HW e SW di base;

2. esista un consolidamento nel mercato di un determinato set di architetture di base e sviluppo di processi di **standardizzazione** (vedi ISO ODMG, ANSI, W3C, etc.);
3. vi sia presenza di misuratori di capacità quali i MIPS (*Mega Instructions per Second*) che costituiscono una base comune di misura prestazionale e di raffronto tra diversi sistemi;
4. si faccia ricorso a test empirici quali i *Benchmark*;
5. vi sia la tendenza dei prodotti HW e SW di base a migliorare l'affidabilità, diminuendo i costi.

2.2. Valutazione dello sforzo di sviluppo del SW applicativo

E' un'operazione molto complessa, poiché dipende da diverse variabili tra loro correlate che rendono la soluzione del problema di tipo non lineare. Le più importanti variabili sono:

1. sviluppo separato dei moduli SW e loro integrazione;
2. coordinamento dei diversi gruppi di lavoro;
3. integrazione delle varie fasi del ciclo di sviluppo del SW, con particolare riguardo al *Project Management*, alla documentazione ed allo sviluppo di SW di supporto (ad esempio un pacchetto di amministrazione del pacchetto applicativo sviluppato);
4. vincoli specifici della stima;

Un'errata valutazione può portare ad effetti disastrosi che vanno dall'aumento sostanziale sei costi di sviluppo (fino anche a più del doppio) al fallimento del progetto.

Fortunatamente esistono modelli matematici e metodologie empiriche che, insieme all'esperienza personale, aiutano il progettista a compiere questo tipo di valutazione.

2.3. Valutazione delle risorse umane da impiegare nelle fasi di formazione, avviamento e sviluppo.

Le risorse umane costituiscono una quota parte rilevante nello sviluppo di un progetto, onde l'importanza di valutarne i costi. Tale valutazione ha significato sia nel caso di acquisizione delle risorse dall'esterno (*outsourcing*) sia nel caso di impiego di risorse interne ad una realtà aziendale (*insourcing*). Al riguardo si considerino i costi di formazione per la preparazione di adeguati *skill* professionali delle risorse interne.

Il problema da risolvere, in pratica, è quello della quantificazione dei tempi, misurati in mesi-uomo (oppure ore-uomo), necessari per lo svolgimento del progetto. Una volta stimata tale quantità, si determina il numero delle unità lavorative, compatibilmente con il loro costo. Il costo mensile (orario) di un'unità lavorativa di un determinato *skill* è una grandezza facilmente ricavabile o nota (contratti pregressi o stime di mercato sul costo dell'offerta di lavoro)

Per quanto concerne il personale da impiegare per l'avviamento di un pacchetto applicativo, si devono considerare anche i fattori 'culturali' e 'motivazionali' dell'utente finale.

Il sistema informatico, infatti, ha maggiori probabilità di sopravvivenza successivamente al periodo di avviamento se il personale impiegato è in grado di stimolare e di formare l'utente finale in modo da fargli acquisire il necessario grado di autonomia.

3. Problemi legati al processo di stima dei costi

Il processo di stima dei costi può essere negativamente influenzato da un certo numero di fattori che ne determinano l'inesattezza. Secondo alcuni autori (De Marco, Boehm e Jones) questi fattori possono essere raggruppati in cinque categorie:

1. Sforzi per:
   - Ristrettezza dei tempi;
   - Non completa definizione e chiarezza dei requisiti del sistema informatico;
   - Non partecipazione degli estimatori, cioè coloro (organizzazioni o personi) a cui si chiede la stima, alla fase di stesura dei requisiti del progetto.
2. L'estimatore spesso non ha a disposizione dati storici per effettuare una stima accurata;
3. Pregiudizi verso la stima '*plausibile*' che portano a *sottostimare* il problema;
4. La tendenza del *Management* di portare in primo piano il raggiungimento degli obiettivi (massimizzazione della produzione) ed in secondo piano i costi da sostenere per conseguirli (minimizzazione del problema o sottostima dei costi);
5. Fattori soggettivi e legati alla natura umana dell'*estimatore* (umori, convinzioni, idee preconcette, etc.) ;

La scarsa accuratezza può portare a:

- *Sottostime dei costi*: si sviluppa un progetto con risorse (finanziarie, umane, tecnologiche e temporali) inadeguate, con alta probabilità di fallimento e scarsa qualità del prodotto finale (scarsa efficacia).
- *Sovrastima dei Costi:* si dedicano al progetto più risorse del necessario, sottraendole ad altri progetti e sottoimpiegandole (scarsa efficienza).

Le ripercussioni di un errore di stima possono incidere:

- Economicamente: Se il progetto che fallisce è sviluppato in proprio e non è possibile completarlo nei tempi programmati e/o nei costi preventivati, si ha lo spreco delle risorse già impiegate.

Diversamente, se il progetto è sviluppato per conto terzi, il *project manager* è costretto o a chiedere ulteriori risorse al committente o, se il budget preventivato è fisso, a rimetterci personalmente;

- Tecnicamente: i ritardi si ripercuotono sulla qualità del prodotto, specialmente sulle componenti accessorie (documentazione, test, avviamento, etc.);
- Organizzativamente: per far fronte ad un sovraccarico imprevisto (ad esempio un ritardo), il *prj. manager* cerca di far ricorso a risorse umane supplementari. Questo comporta un improvvisato riassetto dell'organico ed una rassegnazione degli incarichi e la demotivazione generale del personale.

La stima dei costi di sviluppo del SW non è rilevante soltanto nel momento iniziale del ciclo, ovvero nel raffronto tra i costi ed i benefici previsti o nelle decisioni sul *make-or-buy*, ma durante tutto lo svolgimento del progetto. La stima dei costi ha una importanza a livello **gestionale** del progetto, in ogni sua fase. Per tale motivo vi deve essere piena e continua sintonia tra l'*estimatore* ed il *project manager*. Quest'ultimo si deve avvalere delle stime in modo *adattivo*, controllando lo stato di avanzamento del progetto, misurandone i risultati parziali, e correggendo gli eventuali scostamenti dalle stime con opportune azioni correttive.

4. Metriche del Software (MS)

Le metriche sono quegli strumenti che consentono di giungere ad una misurazione del software dal punto di vista sia delle attività legate alla progettazione ed allo sviluppo del codice, sia delle attività correlate all'avviamento e alla manutenzione.

Gli studi del settore si sviluppano su due direttrici:

1. Metriche per la quantificazione del prodotto sviluppato (linee di codice, complessità strutturale dell'applicativo), facilmente applicabili anche in modo automatizzato;
2. Metriche per la quantificazione delle altre fasi del ciclo di sviluppo del SW (consistenza dei requisiti, produttività funzionale, strutturazione del design e della documentazione progettuale, livello di espressività del linguaggio, completezza dei test).

Le metriche possono essere impiegate per:

1. Migliorare la qualità del **controllo** di gestione del SW (*SW Management*), sia in fase progettuale che operativa, misurando grandezze quali:
   - Risorse (uomini e mezzi) dedicate alla produzione;
   - Caratteristiche del SW (affidabilità, usabilità, dimensione, complessità, *performance*);
   - Caratteristiche dell'ambiente di sviluppo (skill del personale impiegato, tecnologia adottata);
2. Creare dei **modelli di stima** dello sforzo e del tempo necessari allo sviluppo di un progetto SW, nonché alcuni parametri caratteristici del progetto stesso, quali *performance*, affidabilità ed efficienza.

4.1. Il sistema di misurazione del software

Il problema che si pone all'interno di una realtà aziendale è l'individuazione del soggetto deputato alla stima e alla misurazione dei costi del SW, nonché alla interpretazione dei risultati di un progetto. Nasce l'esigenza di creare un vero e proprio **sistema di misurazione** del SW in termini quantitativi (numero di funzionalità), temporali e di impiego delle risorse disponibili (uomini, mezzi, capitale, etc.).

Nella creazione di tale sistema si devono tenere ben presenti alcuni fattori:

1. *il suo valore aggiunto, ovvero i benefici che può apportare all'organizzazione*: tali benefici incidono sull'immagine dell'organizzazione e sul grado di soddisfazione del cliente e sono legati alla capacità di misurare la qualità del prodotto e di stimare l'entità di eventuali interventi correttivi e migliorativi.

2. *i costi da sostenere per sua realizzazione:* totalizzano circa il 5% del costo di sviluppo del SW e comprendono i costi di misurazione della produttività, della qualità, del grado di soddisfazione del cliente e della formazione del personale appartenente al sistema di misurazione.
3. *l'impatto che esso ha sull'organizzazione dell'azienda*: l'organizzazione dovrebbe dedicare a tale struttura del personale, costituito in un gruppo di lavoro. Tale gruppo di lavoro non dovrebbe avere un peso marginale nella struttura organizzativa bensì centrale e direzionale, in quanto di supporto all'organo decisore.
4. *l'adozione di un piano operativo per la sua gestione*: le stime e le misure di produttività e di qualità non devono procedere a caso, ma secondo un programma o piano operativo. Tale piano può essere articolato in attività o *task*, tra le quali le più significative sono:
   - Misure Operazionali: tempi di funzionamento e tempi di risposta del sistema;
   - Misure Progettuali: misura degli scostamenti dei costi effettivi delle varie fasi del ciclo di vita del SW da quelli stimati;
   - Misure consuntive dei progetti;
   - Misure del grado di soddisfazione dell'utente;
   - Misure dei difetti rilevati;
   - Misure accessorie (o dei fattori *soft*): riferite ai costi per l'adeguato impiego del personale in quantità e qualità (*skills*), per l'impiego di idonei strumenti (*tools*) e metodologie di sviluppo e di misura, per l'adeguata organizzazione della struttura stessa.

4.2. Metriche per la stima dei costi del SW

Per compiere, dunque, una misura è necessario definire una metrica. Esistono diversi tipi di metriche:

- m. di input: in generale si basano sulle informazioni disponibili sin dalle prime fasi del ciclo di sviluppo del SW, quali analisi dei requisiti, schemi concettuali e logici, disegno dell'architettura e conteggio delle linee di codice - vedi oltre;
- m. di output: si basano sulla misura dei risultati;

Le prime si suddividono, a loro volta, in:

- Basiche: semplici conteggi di quantità osservabili empiricamente;
- Calcolate: derivate dal computo di una formula matematica;

L'approccio più immediato per la stima del costo di sviluppo del SW è quello del conteggio delle linee di codice del sorgente (*SLOC*), che, tuttavia,

conduce a risultati validi solo in casi di programmazione con linguaggi a basso livello. Applicando tale metodo a linguaggi ad alto livello, infatti, le stime perdono accuratezza e consistenza. I limiti del metodo SLOC si incontrano nei casi di programmi contenenti dichiarazioni di variabili, costanti e commenti e nel riuso di software.

C. Jones, uno dei massimi esponenti nella quantificazione del software, ritiene ancora applicabile il metodo con sufficiente accuratezza a buona parte dei linguaggi di programmazione ad alto livello, **purché si rispettino delle precise ed univoche convenzioni di conteggio**.

Un altro limite del metodo lo si incontra nel calcolo/stima dello sforzo necessario allo sviluppo. Detti I il numero delle istruzioni contate con il metodo SLOC e P la produttività espressa in numero di linee di codice prodotte da una persona in un intervallo di tempo di riferimento (mese, giorno), il rapporto **S=I/P** rappresenta lo **sforzo** sostenuto per lo sviluppo. Paradossalmente si osserva che la produttività **P** diminuisce all'aumentare del livello del linguaggio di programmazione e quindi, per assurdo, computando lo sforzo solo mediante il conteggio dei SLOC, si arriva alla conclusione che esso è maggiore quando il livello di astrazione del linguaggio è più alto. Il paradosso si spiega col fatto che il vero sforzo di sviluppo non risiede tanto nell'applicazione delle regole formali e sintattiche di un dato linguaggio, quanto nel concepimento della soluzione informatica di un problema e nell'applicazione delle astrazioni logiche offerte dal linguaggio adottato. All'aumentare del livello di astrazione, quindi, predomina lo sforzo per le attività indipendenti dal linguaggio (*non-coding tasks*) su quelle legate alla mera scrittura del codice (*language-dependent*). Per tali ragioni si ricorre sempre più spesso a metriche *language-independent* come quella dei *Function Point* introdotta per la prima volta da A.J. Albrecht.

4.3. Approcci alla Stima del Costo del SW

Come affrontare il problema della stima del costo del software?
Esistono tre approcci generali:

1. Ricorrere al giudizio di esperti: soffre dell'inconveniente dell'eccessiva soggettività ed è legato al grado di comprensione che l'esperto ha del problema.
2. Effettuare una stima per analogia con uno o più progetti con caratteristiche simili già completati: si deve fare attenzione al fatto che i progetti precedentemente eseguiti sono rappresentativi di condizioni ambientali, funzionali ed operative diverse da quelle in cui si deve collocare il nuovo progetto. Affinché questo approccio sia applicabile è necessario che l'estimatore abbia a disposizione molti dati storici di riferimento.
3. Impiegare dei modelli algoritmici, matematici e statistici: producono una stima del costo in funzione di un certo numero di

variabili dette *cost-driver*. Le valutazioni conseguibili con questi modelli sono oggettive, in quanto non inquinate da fattori umani, ripetibili e automatizzabili. Questo approccio soffre dell'inconveniente di non fornire stime attendibili in quei casi particolari che non possono essere ricondotti ai predetti modelli.

Concentrandoci solo sui modelli algoritmici, matematici e statistici (brevemente modelli algoritmici), si pone il problema di valutarne le caratteristiche e l'attendibilità. In altre parole la questione è quella di avere dei criteri di scelta del modello migliore per la stima dei costi in un determinato contesto.

Vi sono due classi di criteri: soggettivi ed oggettivi. I primi attengono a qualità del modello rilevabili soggettivamente (validità, facilità d'uso, oggettività, robustezza, trasportabilità, etc.), gli altri fanno ricorso dei coefficienti matematici, quali:

- Errore Medio Relativo;
- Errore Medio Relativo Assoluto;
- Coefficiente di Multipla Determinazione;
- Valor Medio Relativo della Deviazione Standard;
- Predizione a livello r.

Non scenderemo nel dettaglio di questi coefficienti.

4.4. Modelli Algoritmici di Stima del costo del SW

Molte sono le classificazioni dei modelli di stima del costo del SW ma quella più comune li raggruppa in *"relazione al metodo* ":

1. Modelli statistici;
2. Modelli storico empirici;
3. Modelli teorici;
4. Modelli composti.

4.4.1. Modelli statistici

I modelli statistici sono modelli matematici che, sulla base di dati statisticamente noti, consentono di ottenere la stima ottima della grandezza "*sforzo*" (Š - espressa in mesi-uomo), cioè quella grandezza stimata Š che rende minima la quantità (S - Š) detta "*Errore*".

La stima dello sforzo Š viene prodotta da una funzione, detta di *regressione*, che ha la seguente forma:

$$Š=F(x_1, x_2, \ldots. x_n; a_1, a_2, \ldots. a_m)$$

Le variabili indipendenti x_i sono quelle grandezze che si ritiene influenzino lo sforzo di produzione del SW ed i parametri a_j sono coefficienti (costanti) scelti per minimizzare la funzione errore, esprimibile come:

$$Er(a_1, a_2, ...a_m) = \sum_{i=1}^{N} [S_i - \check{S}]^2$$

In altre parole, i coefficienti $\{a_j\}$ sono quelli che minimizzano la funzione errore intesa come media delle *distanze* tra la stima Š e tutti gli sforzi S_i di un campionario (*database di campioni*) di *N* progetti noti.

I diversi modelli, che non approfondiremo in questa sede, differiscono tra loro per il tipo di funzione F(), per la scelta delle variabili di costo, o *cost driver* $\{x_i\}$, per il numero di tali variabili e per la determinazione dei valori dei coefficienti costanti $\{a_j\}$.

In base al tipo di funzione F(), i modelli statistici si distinguono in:

- Modelli a Regressione Lineare: $\check{S} = a_0 + \sum_{i=1}^{n} a_i * x_i$

Questi modelli non hanno dimostrato piena attinenza con la realtà poiché questa si è rivelata non modellabile come un sistema lineare;

- Modelli a Regressione Non Lineare: $\check{S} = (a + b * I^c) * m(\underline{X})$

dove:

a, b, c: sono parametri costanti, calcolati mediante analisi statistiche;

I: è il numero di linee di codice del progetto, espresso in KSLOC (migliaia di linee di codice sorgente). E' necessario che il computo degli sforzi S_i venga eseguito con le stesse modalità per tutti i progetti del campionario e ciò implica che il conteggio delle linee di codice sorgente (I) sia lo stesso per tutti i suddetti progetti.

$\underline{X}$: è il vettore dei *cost driver*;

$m(\underline{X})$: è una funzione di correzione di tipo non lineare.

La funzione di stima Š, così come sopra espressa, risulta complessa e difficilmente sviluppabile, pertanto si opera su una funzione di "stima nominale" $\check{S}_{nom} = (a + b * I^c)$ e si correggono, poi, i risultati mediante il fattore di correzione $m(\underline{X})$.

I vari modelli di stima non lineari differiscono tra loro per i coefficienti a, b, e c della stima nominale.

4.4.2. Modelli Storico-Empirici

Questi modelli si basano sull'esperienza di alcuni "*estimatori*" che hanno cercato di razionalizzare i propri metodi di stima producendo dei modelli empirici, in parte razionali ed obiettivi, in parte soggettivi.

Il più noto di questi modelli è quello di **Wolverton**, secondo il quale il costo totale di un sistema SW è dato dalla somma dei costi dei moduli funzionali in cui è possibile decomporlo:

$$C_{tot} = \sum_{k=1}^{n} C(k)$$

I costi C(k) dei moduli SW componenti il sistema sono dati da:

$$C(k) = \hat{I}(k) * C_{i(k),j(k)}$$

I(k) rappresenta la stima della dimensione del k-esimo modulo espressa il SLOC, mentre $C_{i(k),j(k)}$ è un coefficiente che tiene conto del tipo (i(k)) e della complessità (j(k)) del modulo SW. Il valore di tale coefficiente non è calcolato ma viene attribuito empiricamente dall'estimatore. L'estimatore, infatti, costruisce una matrice [C], detta 'matrice dei costi SW', i cui elementi rappresentano i costi in dollari (o altra valuta) di un modulo SW in relazione al suo tipo, o categoria funzionale, e alla sua complessità o difficoltà di realizzazione.

In pratica il metodo generalizza le possibili varie tipologie funzionali di moduli SW riconducendole a sole sei classi:

1. Controllo;
2. Input/Output;
3. Pre/post processamento;
4. Algoritmo;
5. Gestione Dati;
6. Criticità delle prestazioni.

Parimenti, il modello contempla solo sei classi o livelli di complessità di sviluppo:

1. Vecchio e semplice (OE);
2. Vecchio e medio (OM);
3. Vecchio e difficile (OH);
4. Nuovo e semplice (NE);
5. Nuovo e medio (NM);
6. Nuovo e difficile (NH);

Un esempio di matrice dei costi è il seguente, dove i valori rappresentano i costi dei moduli in funzione del livello di complessità.

	OE	**OM**	**OH**	**NE**	**NM**	**NH**
Controllo	21	23	24	26	28	30
I/O	15	17	18	21	22	23
Pre/Post	16	18	19	25	28	29
Algoritmo	24	24	25	31	33	38
Dati	20	25	29	40	42	44
Criticità	67	67	67	67	67	67

Questo modello offre il vantaggio della semplicità di applicazione ma soffre di completezza ed esaustività in quanto non tiene in considerazione l'influenza di altri fattori di costo quali l'ambiente applicativo di sviluppo (linguaggio, sistema operativo, tecnologia), l'esperienza e la capacità del personale tecnico addetto allo sviluppo.

4.4.3. Modelli Teorici

Questi modelli si basano su precise teorie matematiche, riconducendo il formalismo di un linguaggio di programmazione (statement, variabili e costanti) al formalismo algebrico (operatori e operandi), oppure sulle ipotesi di come la mente umana operi durante il processo di sviluppo.

Non approfondiremo nel seguito della trattazione questi modelli.

4.4.4. Modelli Composti

In questa categoria rientrano tutti quei modelli che hanno un fondamento teorico, che impiegano indagini statistiche e che fanno uso dell'esperienza storica (*expert judgment*).

Spesso si classificano come composti anche quei modelli che, per natura e metodologia, non rientrano in alcuna delle predette classi.

I modelli composti più consolidati sono:

- PRICE-S (non trattato);
- ESSE o Expert System for Software cost Estimation (non trattato);
- COCOMO;
- FUNCTION POINTS;
- ESTIMACS (non trattato);
- FEATURES POINTS.

5. Modelli Consolidati di Stima del Costo del SW

5.1. Modello COnstructive COst MOdel (COCOMO)

Il modello COCOMO, introdotto da Barry Bohem, consente di stimare lo sforzo S di sviluppo del prodotto, espresso in mesi-persona, in funzione del numero di istruzioni **consegnate** (*I*), della modalità di sviluppo, del grado di conoscenza e di dettaglio delle informazioni attinenti il progetto da realizzare e di alcuni fattori correttivi, denominati *cost driver*.

Il modello consente, altresì, di stimare il tempo solare, espresso in mesi, necessario per lo sviluppo.

L'equazione generale del modello è la seguente:

$$S = a * I^b * \prod_i C_i$$

Dove C_i sono i fattori correttivi o *cost driver*, *a* e *b* sono delle costanti che dipendono dal "modello" e dalla "modalità" di sviluppo adottati, '*I*' rappresenta la quantità di codice consegnato, espresso migliaia di istruzioni (KDSI), escludendo tra queste i commenti e le istruzioni appositamente introdotte per facilitare la fase di *testing* e di *debugging* o per essere supporto allo sviluppo.

Come già detto, una volta noto lo sforzo di sviluppo, il modello COCOMO consente anche di calcolare il tempo solare di sviluppo (*schedule*), espresso in mesi, attraverso la seguente formula:

$$T = c * S^d$$

dove *c* e *d* sono costanti dipendenti dal "modello" e dalla "modalità" di sviluppo adottati (vedi oltre).

Nel modello COCOMO il mese-persona corrisponde a 152 ore lavorative (19 giorni lavorativi da otto ore ciascuno). Un anno-persona corrisponde a 12 mesi-persona a meno di 35 giornate lavorative per ferie e malattie.

Una stima effettuata col modello COCOMO mantiene la propria validità solo se i requisiti iniziali del SW non vengono modificati sostanzialmente durante lo sviluppo.

Nonostante in letteratura siano noti un gran numero di fattori correttivi *C* dello sforzo *S* di sviluppo del SW, molto spesso non generali e correlati tra loro, il modello COCOMO ne contempla solo 15, tutti generali ed indipendenti. L'influenza di tali fattori nel computo della stima dipende dal "modello di sviluppo" adottato.

Il modello CO.CO.MO. comprende, in realtà, tre modelli di sviluppo per adattarsi a diversi tipi di esigenze e modalità di sviluppo:

1. Modello COCOMO base: adatto se l'esigenza è quella di pervenire ad una stima immediata del costo in presenza di poche informazioni sul profilo delle risorse umane e sull'ambiente tecnologico ed operativo del progetto. **Non tiene conto dei fattori correttivi**.
2. Modello COCOMO intermedio: è il modello più usato poiché tiene conto di tutti i fattori correttivi ed è impiegabile qualora siano note le caratteristiche di tipo generale del progetto, cioè non dipendenti dalla particolare fase del ciclo di sviluppo. I valori dei 15 fattori correttivi *C* sono assunti costanti per tutto il ciclo di sviluppo e dipendono dal livello di incidenza nel progetto. I possibili livelli d'incidenza contemplati dal modello sono: Molto Basso, Basso, Nominale, Alto, Molto Alto, Altissimo. B. Bohem, partendo da un'indagine statistica, ha proposto una matrice attraverso cui, noti i livelli d'incidenza dei suddetti 15 fattori correttivi, si ricavano i valori da inserire nelle formule di calcolo dello sforzo.
3. Modello COCOMO dettagliato: è il modello che più si addice alla stima di progetti molto complessi poiché tiene conto delle diverse fasi del ciclo di sviluppo del SW e, pertanto, i valori dei 15 fattori correttivi variano in funzione della fase del ciclo di sviluppo considerata. Questo modello, quindi, impiega diverse matrici dei livelli di incidenza dei fattori correttivi, una per ogni fase del ciclo di sviluppo.

Il modello COCOMO è uno dei più completi strumenti di stima anche perché si applica a diverse modalità di sviluppo. Il modello distingue tre categorie generali di modalità di sviluppo in funzione di alcuni parametri qualitativi quali: dimensioni del progetto, dimensioni del gruppo di lavoro ed esperienza dei suoi membri, tipo di approccio allo sviluppo (artigianale, industriale) e tecnica di sviluppo (sequenziale, parallela). In funzione della

modalità e del modello di sviluppo variano i coefficienti *a,b,c,d* delle formule dello sforzo *S* e del tempo *T*, come si evince nella seguente tabella

	COCOMO base				**COCOMO Intermedio**				**COCOMO dettagliato**			
Modalità	*a*	*b*	*C*	*d*	*a*	*B*	*c*	*d*	*a*	*b*	*c*	*d*
Poco strutturata	2,4	1,5	2,5	0,38	3,0	1,12	2,5	0,35	3,6	1,20	2,5	0,32
Mediamente strutturata	3,2	1,05	2,5	0,38	3,0	1,12	2,5	0,35	**2,8**	1,20	2,5	0,32
Fortemente strutturata	3,2	1,05	2,5	0,38	3,0	1,12	2,5	0,35	2,8	1,20	2,5	0,32

Ricapitolando, per effettuare la stima secondo il modello COCOMO si devono eseguire i seguenti passi:

1. Stimare la dimensione del progetto in KDSI;
2. Individuare la modalità di sviluppo del progetto;
3. Scegliere il modello di stima più adeguato;
4. Calcolare lo sforzo *S;*
5. Calcolare il tempo solare *T*.

5.2. Metodo di conteggio dei Function Points (FP)

Il modello dei FP introdotto da Allen Albrecht si propone come obiettivo la quantificazione del software in termini di funzioni consegnate all'utente e può essere applicato per effettuare la stima dei costi e dello sforzo di produzione (fase preventiva), per valutare e quantificare i risultati intermedi e finali dello sviluppo (fase consuntiva) e per valutare la qualità dello sviluppo.

Lo scopo del metodo è quello di quantificare solo le funzionalità che il programma offre all'utente. Il punto di vista, quindi, non è quello dell'imprenditore che tende a remunerare tutti gli sforzi di sviluppo ma quello del committente che vuole pagare solo le funzionalità richieste ed effettivamente consegnate, indipendentemente dal come queste ultime siano state realizzate.

Per tale motivo questa tecnica è fondata su due requisiti fondamentali:

1. *I FP devono essere indipendenti dalla tecnologia;*
2. *I FP devono misurare tutte e sole le funzioni consegnate all'utente;*

Questa "astrazione" dai dettagli realizzativi si ottiene considerando cinque entità funzionali generali:

1. Internal Logical File (ILF): *gruppo logicamente correlato di dati o di informazioni di controllo identificabili dall'utente, generati, mantenuti e utilizzati all'interno dell'applicazione.* Esempio: le tabelle di un database, le tabelle di codifica, le tabelle dei messaggi e degli errori, i file di configurazione dei profili degli utenti.
2. Interface File: *Gruppo logicamente correlato di dati o di informazioni di controllo identificabile dall'utente, utilizzato all'interno dell'applicazione, ma creato e mantenuto da altre applicazioni.*

Esempio: Tabelle di altre applicazioni, tabelle degli errori del sistema operativo, database degli utenti del sistema operativo, etc.

3. External Input: *Un input esterno che processa dati e/o informazioni di controllo che fanno parte dell'applicazione e causa aggiunte, cancellazioni e modifiche negli ILF.* Esempio: transazioni INSERT, UPDATE e DELETE;
4. External Inquiry: *Unica combinazione di Input/Output dove un input provoca un output immediato e non si verifica alcuna modifica negli ILF.* Esempio: Query SELECT, messaggi di errore, elaborazioni di controllo degli accessi, informazioni di HELP.
5. External Output: *Flussi informativi e di controllo in uscita dall'applicazione e derivati da una elaborazione.* Esempio: report, risultati di una *stored procedure*.

Queste entità funzionali possono essere ricavate dalla documentazione di progetto (design), da quella di supporto alla fase di progettazione concettuale (specifiche, schema E-R), dai sorgenti dell'applicativo ed dagli script di definizione della base di dati.

Una volta individuate e contate le suddette entità, si procede a stimarne la complessità percepita dall'utente pesando i valori assunti da tali entità nell'ambito dell'applicazione da realizzare. I gradi di complessità contemplati sono solo tre, BASSA, MEDIA ed ALTA, a ciascuno dei quali è attribuito un peso, inteso come coefficiente moltiplicativo numerico. Il metodo di assegnazione del grado di complessità alle entità funzionali individuate e la scelta del relativo coefficiente di peso dipendono dalle diverse metodologie di conteggio (IFPUG ver. 4.0, IFPUG ver. 3.5., IBM etc.).

Sommando i valori delle suddette entità funzionali, ponderate secondo il rispettivo peso o grado di complessità, si ottiene il numero di *Unadjusted Funcion Points*, cioè il totale grezzo di FP contati.

$$UFP = a*ILF + b*IF + c*EI + d*EINQ + e*EO$$

Il numero di UFP deve essere corretto in considerazione delle peculiarità dell'applicazione e dell'ambiente operativo. Per trovare il valore del fattore di correzione (VFC) si deve valutare l'incidenza sul progetto di 14 voci denominate Caratteristiche Generali del Sistema (CGS):

1. Comunicazione Dati;
2. Elaborazione Dati Distribuita;
3. Prestazioni;
4. Carico sul sistema HW;
5. Volume di transazioni;
6. Immissione dati On-Line;
7. Facilità d'uso;
8. Facilità di modifiche on-line;
9. Complessità di elaborazione;

10. Riusabilità;
11. Semplicità di installazione;
12. Semplicità di gestione operativa;
13. Installazioni Plurime;
14. Semplicità di modifica.

A ciascuna di queste caratteristiche viene assegnato un punteggio da 0 (assente) a 5 (massima influenza).

Sommando tutti i 14 punteggi si ottiene un valore N che è legato al VFC dalla seguente formula empirica:

$$VFC=(0,01*N) + 0,65$$

Il numero totale di FP contati è, dunque:

$$FP=UFP * VFC$$

5.3. *Features Points di Capers Jones*

Capers Jones, grande studioso dei metodi di stima del costo del SW, ha rilevato un limite nel metodo di conteggio dei *function points*: esso si applica con successo solo a sistemi informativi transazionali ma non è proficuamente applicabile ai sistemi real-time (sistemi di controllo, sistemi d'arma militari), ai sistemi elaborativi per applicazioni scientifiche ed in tutti quei sistemi SW in cui è più elevata la complessità della componente algoritmica su quella transazionale.

Modificando il metodo di conteggio dei FP con l'introduzione di una nuova Caratteristica Generale del Sistema (la quindicesima), ovvero il "*numero di algoritmi*", e di un nuovo metodo di assegnazione della complessità alle cinque entità funzionali attraverso il calcolo di un Fattore Correttivo di Complessità (FCC), Jones è pervenuto ad un nuovo metodo di stima del costo dello sviluppo del SW: il metodo di conteggio dei *Features Points*. Tale metodo, attualmente, è ancora in fase di perfezionamento e raramente è applicato a sistemi commerciali.