Metriche e modelli di stima dei costi del software

Indice

11. Metriche e modelli di stima dei costi del software

You can't control what you can't measure. 1 (DeMarco 1986)

Scopo della lezione

- **inquadramento del processo di stima dei costi nel processo produttivo del software**
- **analisi delle principali metriche di interesse alla stima dei costi del software (SCS)**
- **tassonomia e inquadramento storico dei principali modelli di SCS**
- **introduzione ai modelli composti di SCS più attualmente in uso (COCOMO + Punti Funzione)**

11.1 Il processo di stima dei costi del software

Stima dei costi del software (SCS)

comprensione e controllo

- comprensione dei costi ⇨ miglior controllo
- *controllo*
  - ⇨ *maturità* del processo
  - ⇨ *qualità* del processo e del prodotto
- **comprensione dei costi del software:**
  - **approccio a scatola nera**
    individuazione dei pesi relativi dei *fattori di costo*
  - **approccio a scatola trasparente**
    distribuzione dei costi fra le risorse, attività, fasi, etc.
- **entrambi gli approcci sono utili: complementari**

11.1.1 Stima dei costi e gestione del software

importanza di una buona SCS: imprescindibile per decidere, pianificare, difendere, controllare il processo di sviluppo

Decidere a ragion veduta: analisi costi-benefici, *make-or-buy, break-even,* etc.;
Pianificare l'implementazione delle decisioni, cioè il *processo produttivo;*
Difendere le decisioni e la pianificazione a fronte di pressioni esterne, quali: richieste di tagli di bilancio, dei tempi di sviluppo, etc.;
Controllare il processo di sviluppo e rilevarne eventuali anomalie quali scostamenti dalle previsioni di piano.

- **Sinergia di SCS e gestione del processo di sviluppo del software: processi dinamici**

Dinamica dell'interazione fra SCS e gestione SW

La *calibrazione* dei modelli di SCS ne migliora l'*affidabilità,* che a sua volta migliora la gestione del processo di sviluppo del software.

- **Produttività di sviluppo o del ciclo di vita?**

La produttività di sviluppo tiene conto solo dei costi di sviluppo. Migliorarla a scapito dei costi di manutenzione è una tentazione frequente, ma non è generalmente una buona idea. Occorre infatti considerare l'economia dell'intero ciclo di vita del prodotto. Una fenomenologia tipica è quella della propagazione di aumenti di costi e di ritardi nei tempi, con accumulo sulle ultime fasi dello sviluppo. Per "recuperare" la produttività di sviluppo persa si *tagliano* le ultime attività: collaudi, documentazione, formazione di utenti...
La produttività del ciclo di vita tiene conto di tutti i costi: di sviluppo e di *manutenzione.* Migliorarla a scapito dei costi di sviluppo può essere una buona idea, di fatto un *investimento* nella qualità del prodotto. Infatti, come tali vanno visti i maggiori costi di collaudo, documentazione, formazione: investimenti con ritorno a prodotto finito. La produttività di sviluppo sacrificata si recupera in termini di *quote di mercato* e *durata del ciclo di vita* . Inoltre, il vantaggio si *reitera* ad ogni iterazione di sviluppo e manutenzione!

11.1.2 Incertezza della stima dei costi del software

Quanto può essere incerta la stima dei costi del software?

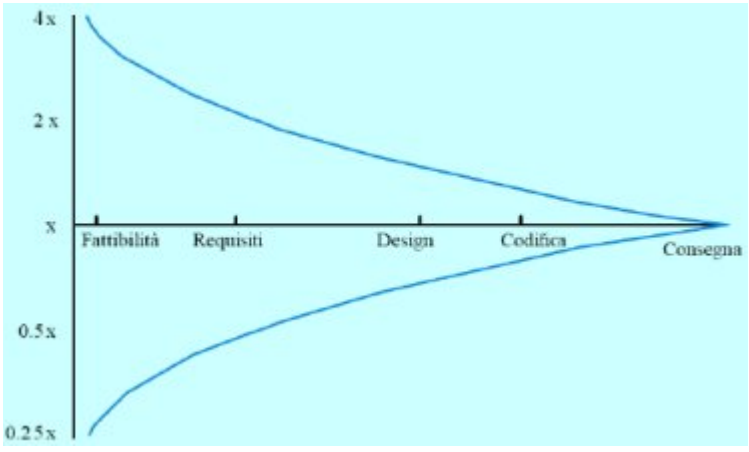


Figura 11.1: Riduzione d'incertezza della stima (Boehm et al. 1995)

Difficoltà inerenti alla stima dei costi del software
Costo della stima: la stima andrebbe organizzata come un mini-progetto, con i propri obiettivi, piano operativo, etc.
Frequente mancanza di esperienza degli addetti, nonché **mancanza di dati storici** per migliorarla col tempo.
Naturale tendenza alla sottostima: ad es. perché basata sulle capacità di persone più esperte della media, o perché si trascurano attività quali il coordinamento, il collaudo, lo sviluppo di codice di supporto, la documentazione, etc.
Opportunismo: concorrenza, ambizioni di carriera, etc.
Soggettività ineliminabile della stima.
Conseguenze di errori di stima

- **per eccesso (di cautela: sovrastima dei costi)**
  occasioni mancate, perdita di *quote di mercato*
- **per difetto (di cautela = eccesso di "ottimismo")**
  (*sottostima* di costi e tempi: fenomeno molto più diffuso)
  - **conseguenze economiche**
    danno economico (penalità contrattuali), fallimento del progetto
  - **conseguenze tecniche**
    qualità del prodotto, scarico di costi sul ciclo di vita
  - **conseguenze manageriali: legge di Brooks**
    "... poiché il processo di sviluppo del software è complesso, aggiungere personale allo staff oltre un certo numero fa aumentare, anziché diminuire, la durata del progetto".

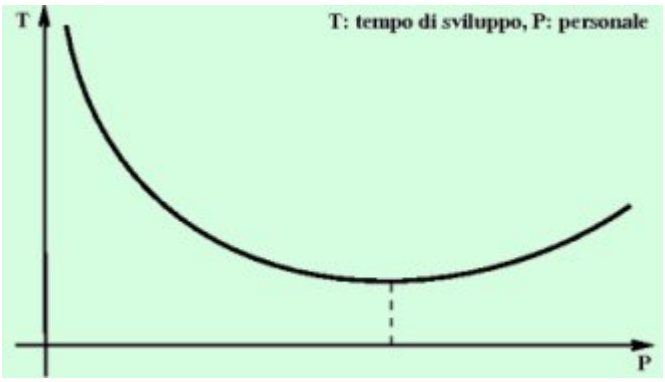


Figura 11.2: Legge di Brooks (Brooks 1975)

11.1.3 Approcci alla stima dei costi del software

- **ricorso al giudizio di esperti**
  per una migliore oggettività: *metodo Delphi*
- **stima per analogia**
  punti deboli:
  - scarsa sensibilità agli aspetti *nuovi* del progetto
  - presuppone l'esistenza di database di progetti *analoghi*
- **uso di modelli algoritmici**
  stima in funzione di *cost driver:* dipende dall'accuratezza della loro stima, è poco sensibile ad aspetti nuovi, *tuttavia* è più oggettiva, ripetibile, e meglio suscettibile di *calibrazione*

11.1.4 Sistema di misure per la stima dei costi del software

Parte integrante di un sistema di misure per il controllo di qualità, del processo e del prodotto, v. il modello ISO 9126

Cosa si misura? Sono rilevanti alla stima dei costi del software sia metriche dei *prodotti* software in senso lato (ad es. dimensione, complessità (McCabe), strutturazione, prestazioni, etc.), sia metriche dei *processi produttivi* del software (ad es.: durata, costi, produttività, maturità, etc.). Inoltre, la misura di metriche distinte *per fasi* del ciclo di vita risulta utile sia alla stima dei costi che al *controllo di qualità* . L'uso delle metriche a questo scopo può riassumersi nel seguente schema:
· definizione di *obiettivi* di qualità, quantificati da valori *target* di fattori misurabili di qualità (metriche);
· *misura* dei valori *actual* effettivi;
· *interventi correttivi* a fronte di eventuali deviazioni degli *actual* dai *target.*
Problemi tipici in questo processo:
· *identificazione* delle cause delle deviazioni;
· *attendibilità* della stima dei *target;*
· *soggettività* dell'interpretazione delle metriche.
Aspetti salienti della costruzione di un sistema di gestione della misura per la SCS

- **Valore del sistema di gestione della misura**
  I *benefici* (di non facile stima) includono:
  · acquisizione di quote di mercato,
  · aumento del livello di soddisfazione degli utenti,
  · aumento del profitto.
  Fra le metriche rilevanti alla stima del valore di tali benefici si annoverano:
  · *numero di difetti* riscontrati nei prodotti,
  · rilevazioni del grado di *soddisfazione degli utenti,*
  · *produttività.*
- **Costo del sistema di gestione della misura**
  Un articolato sistema di gestione delle misure presenta un costo significativo, dell'ordine di quello della gestione della contabilità globale. Tipicamente il suo ammontare è prossimo al 5% del costo totale dello sviluppo del software, ripartito in circa il 2% per misure di produttività e circa il 3% per misure di qualità dei prodotti e di soddisfazione degli utenti.
  L'impiego tipico di *risorse umane* per la stima dei costi del software, in grandi aziende, può essere dimensionato come segue:
  · circa 12 persone al livello centrale,
  · circa 5-6 persone per direzione periferica,
  · circa 100 manager interessati.
- **Profilo del personale per la gestione della misura, competenze richieste**
  · ingegneria del software, progettazione;
  · metodi di stima e pianificazione;
  · metodologie di controllo di qualità, ispezione, collaudo;
  · analisi statistiche e multivarianti;
  · principi di misurazione del software;
  · principi di contabilità.

- **Posizionamento della gestione della misura nell'organizzazione produttiva**
  Per garantire l'indipendenza delle attività di misura, il loro coordinamento e la loro continuità, è opportuna la creazione di un gruppo di lavoro *ad-hoc,* sotto diretta responsabilità della *direzione.* La struttura del gruppo di lavoro, nelle organizzazioni più grandi, dovrà realizzare un buon *decentramento* di responsabilità fra gruppo centrale e gruppi periferici.
- **Un piano operativo per la gestione di misure**
  è necessario al controllo del processo di misura. Questo può essere efficacemente articolato in base a varie categorie di misure, fra le quali si annoverano misure:
  · operazionali,
  · progettuali,
  · dei costi arretrati,
  · di soddisfazione degli utenti,
  · consuntive dei progetti,
  · dei fattori "soft",
  · dei difetti,
  · demografiche aziendali,
  · dell'opinione.

11.2 Metriche della stima dei costi del software

mattoni nella costruzione dei modelli di stima

importanza delle metriche per la stima dei costi:

affidabilità predittiva dei modelli di stima

- **ipotesi:**
  - **esistenza di relazioni fra metriche di fasi diverse del ciclo di vita**
  - **dette relazioni sono formalizzabili in modelli teorico-empirici**
- **scostamento stima - misura ⇨ correzione e calibrazione dei modelli di stima**

11.2.1 Tassonomia delle metriche di stima dei costi del software

- **metriche di input:**
  - **metriche di base: I (SLOC, KDSI, ...)**
  - **metriche calcolate: FP, SSM**
- **metriche di output: T, S**
- **metriche di elaborazione:**
  complessità ciclomatica (McCabe), ...

11.2.2 Dimensione del prodotto

I (SLOC, KDSI): modalità di conteggio

tre gruppi ortogonali di opzioni:

- **a livello di linea**
  linee *fisiche* , linee *logiche*
- **a livello di programma**
  linee *eseguibili* , + *def. dati* , + *commenti* , + *JCL*
- **a livello di progetto**
  linee *nuove* , + *modificate* , + *riusate* ,
  + *codice temporaneo* , + *codice di supporto*

11.2.3 Produttività

P, "equazione dello sforzo": S = I/P

- **produttività di che cosa?**
  - **quantità di SLOC:**
    metrica concreta, concetto *erroneo* di produttività
  - **quantità di "funzionalità", Function Points (FP):**
    metrica astratta, concetto *corretto* di produttività (*Allen J. Albrecht, 1979*)
- **concetto economico di produttività:**
  quantità di **bene o servizio** prodotto per unità di risorsa (tempo · persona)

11.2.4 Sforzo e tempo di sviluppo

- **sforzo S (mesi · persona) e tempo di sviluppo T (mesi) correlati alla quantità media di personale FSP (Full-time Software Personnel)**
- "*curva di lavoro* " FSP(t) ≈ *curva di Rayleigh* :

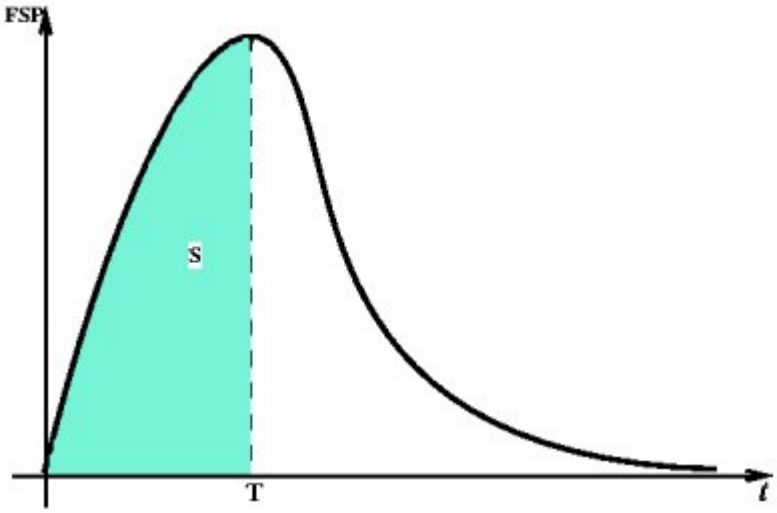


Figura 11.3: Curva di Rayleigh

11.3 Tassonomia di modelli algoritmici di stima dei costi del software

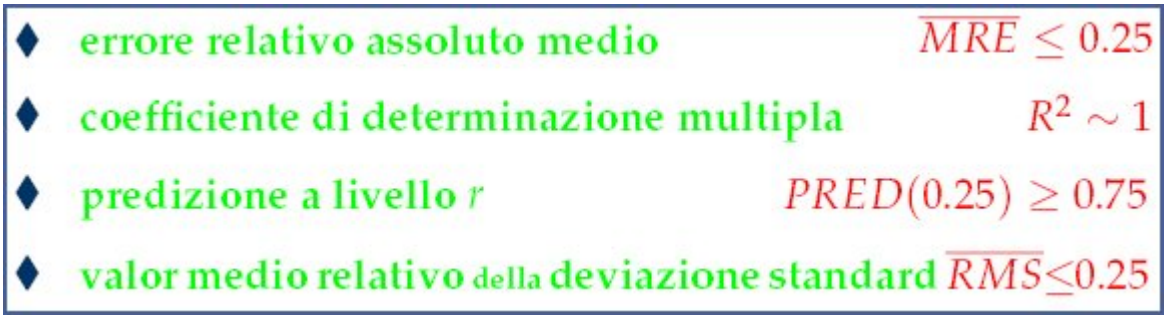
- **Criteri di valutazione dei modelli algoritmici di stima: soggettivi, oggettivi**
- **Modelli statistici**
- **Modelli storico-empirici**
- **Modelli teorici**
- **Modelli composti**

11.3.1 Criteri di valutazione di modelli di stima

Criteri soggettivi:

- **validità**
  lievi scostamenti dai dati di messa a punto;
- **oggettività**
  grado di indipendenza da fattori soggettivi;
- **facilità d'uso**
  · reperibilità dei dati, bassa numerosità,
  · disponibilità nelle fasi precoci del ciclo di vita;
- **robustezza**
  insensibilità a piccole variazioni dell'input;
- **trasportabilità**
  grado di indipendenza da fattori locali.

Criteri oggettivi:
estimatori statistici di validità del modello a fronte di un campione di misura: i criteri più in uso sono i seguenti



Criteri statistici di validità di un modello

Le definizioni degli estimatori statistici sono raccolte nella seguente tabella, dove *n* è la dimensione del campione, y_i è il valore misurato nell'elemento *i* del campione, $\overline{y}$ è il valor medio degli y_i, mentre $\hat{y}_i$ è il valore stimato dal modello nell'elemento *i* del campione.

- **errore relativo medio** $\overline{RE}$

$$\overline{RE} = (1/n)\sum_{i=1}^{n} RE_i = (1/n)\sum_{i=1}^{n}(\hat{Y}_i - Y_i)/Y_i$$

- **errore relativo assoluto medio**

$$MRE = |RE| \qquad \overline{MRE} = (1/n)\sum_{i=1}^{n} MRE_i$$

- **coefficiente di determinazione multipla**

$$R^2 = 1 - \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 / \sum_{i=1}^{n}(Y_i - \overline{Y})^2$$

- **predizione a livello** r

$$PRED(r) = k/n \qquad \text{dove: } k = |\{i | MRE_i \leq r, 1 \leq i \leq n\}|$$

- **valor medio relativo della deviazione standard**

$$RMS = \overline{SE}^{1/2} = \sqrt{(1/n)\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \qquad \overline{RMS} = RMS/\overline{Y}$$

Definizione di estimatori statistici di validità di un modello

11.3.2 Modelli statistici

Identificati mediante analisi di regressione

siano:

- S la variabile di **output** del modello (ad es.: lo sforzo)
- $x_i, i = 1, \ldots n$, le variabili di **input** del modello (ad es.: *cost driver*)
- $a_i, i = 1, \ldots m$, i **parametri** (da identificare) del modello,

ciascun modello esprime l'output come funzione F dell'input, per i dati parametri:

$$S = F(x_1, \ldots, x_n; a_1, \ldots, a_m)$$

l'**identificazione** del modello, cioè **dei parametri**, si ottiene minimizzando l'errore (quadratico), su un campione che fornisce i valori misurati S_i dell'output e $x_{i_1}, \ldots, x_{i_n}$ dell'input, per ciascun elemento i del campione, $i = 1, \ldots, N$:

$$Er(a_1, \ldots, a_m) = \sum_{i=1}^{N}(S_i - F(x_{i_1}, \ldots, x_{i_n}; a_1, \ldots, a_m))^2$$

Identificazione dei parametri di un modello statistico

Modelli statistici lineari

$$S = a_0 + \sum_{i=1}^{n} a_i x_i$$

Forma di un modello statistico lineare

- **i primi risalgono agli anni '60**
  ottenuti con l'analisi di regressione e selezionando i *cost driver* più significativi dai circa 400 potenzialmente rilevanti, ad es. *SDC* (14 *cost driver*), *Farr & Zagorski* (6 *cost driver*).
- **hanno prestazioni scadenti**
  verosimilmente perché la dipendenza dello sforzo (e del tempo di sviluppo) dai (molteplici) *cost driver* è *fondamentalmente non lineare*

Modelli statistici non lineari

$$S = S_{\text{nom}} \cdot m(\mathbf{X})$$

con:

equazione nominale di stima (da correggere poi con $m(\mathbf{X})$):

$$S_{\text{nom}} = a + b \cdot I^c$$

- I : KSLOC
- a, b, c : costanti del modello (regressione, ma talvolta dipendono dai *cost driver*)
- $m(\mathbf{X})$: fattore di correzione, con $\mathbf{X}$ il vettore dei *cost driver*

Forma frequente di un modello statistico non lineare dello sforzo

Occorre notare che modelli nominali "simili" (Walston-Felix, Bailey-Basili, Boehm, Doty, ...) differiscono per le modalità di conteggio di *I*

11.3.3 Modelli storico-empirici

Basati sulla raccolta di dati storici e sull'analogia

classico il modello di Wolverton (1975):

- **matrice di costo** del software $C_{i,j}$
  - i: **tipo** di funzionalità: controllo, I/O, pre/post-processing, algoritmo, gestione dati, criticità delle prestazioni
  - j: **difficoltà**: OE, OM, OH, NE, NM, NH
- $C_{i,j}$: costo unitario (*i.e.* per linea di codice)
- $i(k), j(k)$: stima di **tipo** e **difficoltà** del modulo k $(1, \ldots, n)$
- $\hat{I}(k)$: stima della **dimensione** del modulo k
- $C(k) = \hat{I}(k) \cdot C_{i(k),j(k)}$: costo del modulo k

Costo totale $= \sum_{k=1}^{n} C(k)$

Modello di Wolverton

11.3.4 Modelli teorici

Derivano da ipotesi sull'operare della mente umana nel processo di sviluppo del software, o su leggi matematiche che si presume lo governino

- **modello di Halstead (1972):** *Software Science Effort Model*
  - **ipotesi**:
    n. operatori (verbi) e n. operandi (nomi) in un programma sono correlati al n. errori
  - **analogia** con un testo letterario
  - entità elementari di un programma: **operatori, operandi**
  - **4 metriche di base** di input: $n1$: n. operatori unici
    $n2$: n. operandi unici
    $N1$: n. occorrenze di operatori
    $N2$: n. occorrenze di operandi
  - **metriche calcolate**:
    - **vocabolario**: $n = n1 + n2$
    - **lunghezza**: $N = N1 + N2$
    - **volume**: $V = N \cdot \log_2(n)$
  - **metriche di output** (sulla base di una "metafisica intuitiva"):
    - **livello di astrazione**: $L = \frac{2}{n1} \cdot \frac{n2}{N2}$ (inverso della difficoltà di codifica)
    - **contenuto di intelligenza**: $I = L \cdot V$
    - **livello del linguaggio**: $l = L^2 V$
    - **sforzo mentale**: $S = V/L$
    - **tempo di sviluppo**: $T = S/\beta$ ($5 \leq \beta \leq 20$, *Stround number*)
    - **errori consegnati**: $E = V/3000$
  - **modello di stima** derivato, basato sulla stima della dimensione del software:
    - dalle metriche suddette di deduce per lo sforzo: $S = \frac{n1}{2}\frac{N2}{n2} \cdot N\log_2(n)$
    - ipotesi: $n1 = n2 = n/2$, e $N1 = N2 = N/2$
    - da cui: $S = \frac{1}{4} \cdot N^2 \cdot \log_2(n)$
    - relazione lineare fra N e I con k dipendente solo dal linguaggio: $N = k \cdot I$
    - con un procedimento iterativo si ottiene infine:

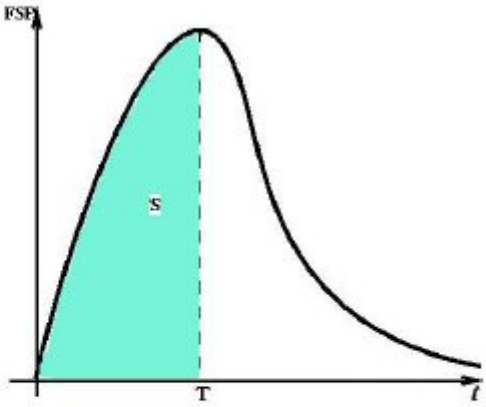
$$S = \frac{1}{4} \cdot k^{2.28} \cdot I^{2.28}$$

Modello di Halstead

- **modello di Putnam (1978):** *Resource Allocation Model*

  - **ipotesi**: l'allocazione di risorse durante il **ciclo di vita** segue l'andamento della **curva di Rayleigh**, definita dall'equazione $\dot{y} = 2kate^{-at^2}$
  - dove: $\dot{y}$ è il grado di utilizzo del personale
    t è il tempo dall'inizio del progetto
    a è un parametro
    k è l'area sottesa dalla curva in $[0, \infty]$
  - integrando in $[0, t]$ si ottiene lo sforzo $y(t)$ dall'inizio del progetto al tempo t:

    $$y(t) = k(1 - e^{-at^2})$$



  - ponendo $a = 1/2T^2$ si trova che $\dot{y}$ assume il valore massimo in T che dunque corrisponde $\sim$ al tempo di sviluppo
  - da quanto detto si deduce una **stima dello sforzo**: $S = y(T) = 0.3945k$
  - inoltre, Putnam propone una **metrica per la difficoltà**: $D = k/T^2$
  - da osservazioni empiriche (~ 50 progetti) ipotizza una relazione fra difficoltà e **produttività** di forma: $P = cD^b$
  - la produttività è per definizione: $P = I/S$
  - l'analisi di regressione sul suddetto campione di progetti ha dato: $P = cD^{-2/3}$
  - da cui si ottiene la **Software equation di Putnam**:
    $I = PS = cD^{-2/3}0.3945k = C \cdot (\frac{k}{T^2})^{-2/3} \cdot k$
  - dopo semplici calcoli si ottiene l'**equazione dello sforzo**:

    $$S = KI^3/T^4$$

    in cui K rappresenta una **costante tecnologica**

  - **limiti** del modello di Putnam:
    - non segue la legge di Brook
    - sovrastima progetti medio/piccoli
    - influenza eccessiva di T su S
    - stima di S eccessivamente sensibile alle variazioni di K
    - richiede taratura, che presuppone notevole conoscenza teorica del modello

Modello di Putnam

- **Modello di Jensen (1984): variante del modello di Putnam**
  - modifica del modello di Jensen per attenuare l'eccessiva influenza di T e K
  - la Software Equation viene così modificata: $I = C_{te} k^{1/2} \cdot T$
  - la **costante tecnologica effettiva** C_{te} è specificata dall'equazione di stima

$$C_{te} = C_{tb} \prod_{i=1}^{15} f_i$$

dove:

- C_{tb} : **costante tecnologica di base**
- $f_i (i = 1, \ldots 15)$: **cost driver**

  - con semplici calcoli si ottiene l'**equazione dello sforzo** nel modello di Jensen:

$$S = cI^2 / T^2$$

dove la costante c incorpora fattori tecnologici e cambiamento di scala

  - valutazioni empiriche indicano che il modello di Jensen si comporta meglio di quello di Putnam, ma i risultati non sono eccellenti

Modello di Jensen

11.4 Modelli composti di stima dei costi del software

Combinazione di idee, metodi e risultati dalle tre precedenti categorie di modelli: statistici, storico-empirici e teorici.

11.4.1 Modelli COCOMO ('81)

COnstructive COst MOdel: risultante dall'analisi statistica di 63 progetti (Boehm 1981)

- **famiglia di nove modelli basata su due classificazioni ortogonali:**
  - **livello di dettaglio**
    dell'informazione espressa nelle metriche di input
    tre livelli: *di base, intermedio, dettagliato*
  - **modalità di sviluppo:**
    tre modalità: *organic, semi-detached, embedded*
- **forma generale dei modelli COCOMO:**

$$S = a \cdot I^b \cdot \prod_i C_i \qquad T = c \cdot S^d$$

Modelli COCOMO '81

- *problema:* la stima della dimensione del prodotto *I* (KDSI)
- *soluzione:* stima dei Punti Funzione (v. appresso) + livello del linguaggio

Modelli COCOMO di livello di base

Al livello di dettaglio di base, non si richiede stima di *cost drivers* , perché lo sforzo è stimato mediante l'equazione nominale di stima. Le equazioni per la stima di sforzo e tempo di sviluppo sono dunque quelle della forma generale, ma con valore nominale (unitario) dei *cost drivers,* e dove i valori dei parametri *a, b, c, d,* dipendono dalla modalità di sviluppo, come segue:

Development Mode	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Parametri dei modelli COCOMO'81 di livello di base

Si tratta quindi di modelli applicabili in sede di *studio di fattibilità* , giacché la stima richiede una sola metrica di input, la dimensione del prodotto *I,* oltre all'indicazione della modalità di sviluppo. La prima può essere stabilita per analogia o in base al giudizio di esperti, mentre la seconda dipende da caratteristiche molto generali del processo produttivo, riassunte nella seguente tabella.

Development Mode	Project Characteristics			
	Size	*Innovation*	*Deadline/constraints*	*Dev. Environment*
Organic	Small	Little	Not tight	Stable
Semi-detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex hardware customer interfaces

Modalità di sviluppo nei modelli COCOMO'81

Modelli COCOMO di livello intermedio

I parametri *b, c, d,* hanno gli stessi valori che nei modelli di livello di base, mentre quelli del parametro *a* differiscono, come indicato nella seguente tabella.

Development Mode	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Parametri dei modelli COCOMO'81 di livello intermedio

Nei modelli COCOMO di livello di dettaglio intermedio si richiede *stima globale* (cioè, unica per l'intero progetto) di *15 cost drivers,* tabulati come segue.

Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
ACAP Analyst Capability	1.46	1.19	1.00	0.86	0.71	-
AEXP Applications Experience	1.29	1.13	1.00	0.91	0.82	-
CPLX Product Complexity	0.70	0.85	1.00	1.15	1.30	1.65
DATA Database Size	-	0.94	1.00	1.08	1.16	-
LEXP Language Experience	1.14	1.07	1.00	0.95	-	-
MODP Modern Programming Practices	1.24	1.10	1.00	0.91	0.82	-
PCAP Programmer Capability	1.42	1.17	1.00	0.86	0.70	-
RELY Required Software Reliability	0.75	0.88	1.00	1.15	1.40	-
SCED Required Development Schedule	1.23	1.08	1.00	1.04	1.10	-
STOR Main Storage Constraint	-	-	1.00	1.06	1.21	1.56
TIME Execution Time Constraint	-	-	1.00	1.11	1.30	1.66
TOOL Use of Software Tools	1.24	1.10	1.00	0.91	0.83	-
TURN Computer Turnaround Time	-	0.87	1.00	1.07	1.15	-
VEXP Virtual Machine Experience	1.21	1.10	1.00	0.90	-	-
VIRT Virtual Machine Volatility	-	0.87	1.00	1.15	1.30	-

Cost drivers nei modelli COCOMO '81

Il livello intermedio è usualmente adeguato alla stima in sede di *prima stesura del piano operativo,* dove l'architettura del prodotto non è verosimilmente ancora nota, ma tuttavia sono disponibili, dalle decisioni di pianificazione e da una prima analisi dei requisiti, sufficienti informazioni per determinare con buona confidenza i valori appropriati dei *15 cost drivers.*
Inoltre, la disponibilità di una *specifica dei requisiti* può permettere una stima più affidabile della dimensione del prodotto mediante i Punti Funzione, come indicato più avanti.

Modelli COCOMO di livello dettagliato

La forma generale e i valori dei parametri sono quelli del livello di dettaglio intermedio, ma sforzo e tempo di sviluppo vengono stimati separatamente per ciascuna delle *quattro fasi* in cui si suppone articolato lo sviluppo: Analisi, Disegno, Codifica, Integrazione e collaudo. Per ciascuna di esse si possono quindi adoperare valori diversi dei *cost drivers.*
Inoltre, nella fase di codifica la stima può essere ulteriormente disaggregata a livello di modulo. Per questa fase vengono adoperati solo quattro dei *cost drivers* visti in precedenza: CPLX, PCAP, VEXP, LEXP.
Questo livello di dettaglio è appropriato alla stima per la *stesura avanzata del piano operativo.*

11.4.2 La metodologia dei Punti Funzione

Presupposti

motivazione: misura economica della produttività P = F/S
dove *F* è una misura astratta della quantità di funzionalità prodotta
Requisiti per F (Allen Albrecht, 1979):

- **astrazione:** indipendenza dalla tecnologia
- **completezza:** misura di tutte le funzioni consegnate
- **proprietà:** misura solo delle funzioni consegnate

prerequisito per la stima dei PF: specifica dei requisiti

PF, una metrica funzionale:
· cinque tipi t di entità funzionali:

1. **file logici interni**
2. **file logici di interfaccia**
3. **input esterni**
4. **inquiry esterne**
5. **output esterni**

· tre livelli di complessità c: bassa, media, alta

Calcolo dei PF nominali, o *Unadjusted Function Points* (UFP):

$$\mathrm{UFP} = \sum_{t=1}^{5} \sum_{c=1}^{3} \pi_{t,c} \cdot n_{t,c}$$

Calcolo dei Punti Funzione nominali

dove $n_{t,c}$ è il numero di entità funzionali di tipo *t* e complessità *c,* mentre $\pi_{t,c}$ è il valore dei PF nominali per una entità funzionale di tale tipo e complessità, dato dalla tabella

Type of Component	Complexity of Components		
	Low	Average	High
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

PF nominali per tipo e complessità delle entità funzionali

Correzione dei PF: le CGS
Caratteristiche Generali del Sistema: 14 proprietà *non funzionali* del prodotto, della cui influenza si tiene conto per correggere la stima dei PF entro il ∓ 35%

· Comunicazione dati
· Elaborazione distribuita dei dati
· Prestazioni
· Carico sul sistema HW target
· Volume di transazioni
· Immissione dati on-line
· Facilità d'uso dell'applicazione
· Facilità di modifiche on-line
· Complessità di elaborazione
· Riusabilità
· Semplicità di installazione
· Semplicità di gestione operativa
· Installazioni plurime
· Semplicità di modifica

Calcolo dei PF corretti:
detta v_i l'influenza della CGS *i,* con $0 \leq v_i \leq 5$ (i = 1,...,14), si calcola:

$$N = \sum_{i=1}^{14} v_i, \qquad \mathbf{VFC} = 0.01 \cdot N + 0.65, \qquad \mathbf{PF} = \mathbf{UFP} \cdot \mathbf{VFC}$$

Correzione del calcolo dei PF

Uso dei PF per la stima della dimensione in linee di codice

Livello λ *di un linguaggio di programmazione* L:
reciproco (a meno di un fattore di scala) del numero medio n_L di istruzioni che realizzano 1 PF in L: $\lambda \cdot n_L = 320$.
Ad esempio:
· linguaggi assembler: $\lambda = 1$
· linguaggi ad oggetti: $12 \leq \lambda \leq 15$
· linguaggi di interrogazione di database: $\lambda = 25$

Calcolo di I_L *(KDSI nel linguaggio* L *di livello* λ *) dai PF* :

$I_L = n_L \cdot \text{PF} / 1000 = 0.32 \cdot \text{PF} / \lambda$

Limitazioni dei PF
Limitazioni *inerenti* i presupposti dei PF:
· i PF sono *poco sensibili agli aspetti interni* del prodotto SW, quali, ad es., la complessità strutturale e logica: nidificazioni, ramificazioni, chiamate ricorsive, etc.;
· la stima è affidabile *solo in alcuni domini applicativi,* quali, ad es., i sistemi informativi e gestionali, ma non per applicazioni scientifiche, sistemi real-time, etc.
Limitazioni *non inerenti* i presupposti dei PF:
· *tipi funzionali poco adatti* alle tecnologie attuali, ad es. quelle dei sistemi interattivi, alle quali sarebbero meglio adatti tipi quali: entità, relazioni, menu, schermate, etc.;
· eccessiva *semplificazione della complessità* in solo 3 livelli;
· tipi funzionali *non ortogonali;*
· alcuni tipi funzionali sarebbero *non correlati allo sforzo* (Kitchenham et al. 1993).

11.4.3 Altri modelli e metodologie

- *COCOMO II, COCOMO Suite* : (Boehm et al. 2000)
- *Feature Points* (Capers Jones, 1986)
- COSMIC *Full Function Points* : Standard (ISO/IEC 2003)
- *System Dynamics* : originata da (Forrester 1961)
- *Software Project Dynamics* (Abdel-Hamid et al. 1991)

11.5 Approfondimenti

Altre risorse per approfondimenti su metriche, modelli e metodologie di stima dei costi del software

11.5.1 Altre risorse per approfondimenti

- **Ulteriori letture consigliate:**
  (Sedehi 1997), (Buglione 2003), (Garmus et al. 2000)
- **Capitoli sulla stima dei costi in testi di Ingegneria del software:**
  Ch. 26 in (Sommerville 2005), *liberamente disponibile*
  Cap. 18 in (Pressman 2004)
  Ch. 7 in (Van Vliet 2000)
- **Altri siti per la ricerca di documentazione:**
  (GUFPI-ISMA), (IFPUG)
  (R.S. Pressman & Associates)
- **Analisi comparativa di metodologie proprietarie:** (ProjectExperts 2006)

11.6 Note

1. Non si può controllare ciò che non si può misurare.

Prentice Hall: http://vig.prenhall.com

11.7 Bibliografia

- **Abdel-Hamid, T. & Madnik, S.E.**, 1991. *Software Project Dynamics: An Integrated Approach*. Prentice Hall.
- **Albrecht, A.J.**, 1979. Measuring application development productivity. *In: Proc. IBM Applications Development Symp.*. CA, USA: Guide Int. and Share, Inc., IBM Corp., 83.
- **Boehm, B.W.**, 1981. *Software Engineering Economics*. Prentice Hall.

- **Boehm, B.W., Abts, C., Brown, A.W., Chulani, A., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D., Steefe, B.**, 2000. *Software Cost Estimation with COCOMO II*. Prentice Hall.
- **Boehm, B.W., Clark, B.K., Horowitz, E., Westland, C., Madachy, R., Selby, R.**, 1995. Cost models for future software life cyle processes: COCOMO 2.0. *Annals of Software Engineering*, 1:1, 57-94.
  Web: http://www.springerlink.com/content/y2386315010g7113
- **Brooks, F.P. Jr.**, 1975. *The Mythical Man-Month*. Addison-Wesley.
- **Buglione, L.**, 2003. *Misurare il software, Seconda Ed.*. Milano: FrancoAngeli.
  Web: http://www.geocities.com/lbu_measure/libri/mis.htm
- **DeMarco, T.**, 1986. *Controlling Software Projects: Management, Measurement, and Estimates*. Prentice Hall.
- **Forrester, J.W.**, 1961. *Industrial Dynamics*. New York, USA: MIT Press and John Wiley & Sons, Inc..
- **Garmus, D., Herron, D.**, 2000. *Function Point Analysis: Measurement Practices for Successful Software Projects*. Addison Wesley Professional, IT Series.
  Web: http://www.aw.com/cseng/series/it
- **GUFPI-ISMA**. *Gruppo Utenti Function Point Italia*. Italian Software Metrics Association. Web: http://www.gufpi-isma.org.
- **IFPUG**. *International Function Point Users Group*. Web: http://www.ifpug.org.
- **ISO/IEC**, 2003. *Software engineering -- COSMIC-FFP -- A functional size measurement method*. Geneva, CH: ISO/IEC, ISO/IEC 19761.
  Web: http://www.iso.org
- **Kitchenham, B.A. & Känsälä, K.**, 1993. Inter-item Correlations among Function Points. *In: Proc. 15th Int'l Conference on Software Engineering*. Baltimore, ML, USA: IEEE Computer Society / ACM Press, 477-480.
  Web: http://portal.acm.org/citation.cfm?id=257572.257677
- **Pressman, R.S.**, 2004. Cap. 18: Stime per il progetto software. *In: Principi di Ingegneria del software, 4/Ed.*. McGraw-Hill Italia.
  Web: http://www.mcgraw-hill.it
- **ProjectExperts** (2006). *Twenty Years of Better IT Estimating Software*. Web: http://www.projectexperts.com/articles/estimating_sw.htm.
- **R.S. Pressman & Associates, Inc.**. *Software Engineering Resources*. Web: http://www.rspa.com/spi.
- **Sedehi, H.**, 1997. *Ingegneria economica del software*. Milano; Roma: Apogeo; EUCOS, Libreria Italiana (2003).
  Web: http://www.libreriaitaliana.it
- **Sommerville, I.**, 2005. Chapter 26: Software Cost Estimation. *In: Software Engineering, 7th Ed.*. Addison-Wesley, Pearson Education.
  Web: http://www.comp.lancs.ac.uk/computing/resources/IanS/SE7/SampleChapters/ch26.pdf
- **Van Vliet, H.**, 2000. Chapter 7: Cost Estimation. *In: Software Engineering - Principles and Practice, 2nd Ed.*. John Wiley & Sons.
  Web: http://www.wiley.co.uk/vanvliet