UNIVERSITA' DEGLI STUDI DI ROMA
TorVergata
Università

Modeling the Cost of a Software Product

Estimating Methods Classification

Algorithmic models

Expert judgement

Analogy

Design-to-cost (or Capacity problem or Parkinson)

Price-to-win

Top-down

Bottom-up

Algorithmic models /
Parametric models /
Which attributes should we include in a cost model?

Cost Model Attribute Classes

Product Factors

Personnel Factors

Platform Factors

Project Factors

COCOMO

SINTESI.
1. Constructive Cost Model. Versioni

ALCUNI DETTAGLI
1. Storia.
2. Basi.
3. Calcolo.
4. Classi di complessità.
5. Livelli.
6. Modelli base, intermedio e avanzato.
7. Coefficienti correttivi.
8. Stima a livello di Componenti.
9. Altri modelli.
10. Vedere a parte Esempio di calcolo.
11. Uso a fini di gestione

The Constructive Cost Model in Synthesis

Constructive Cost Model

COCOMO, abbreviazione di COnstructive COst MOdel, è utilizzato per:

- *stimare* parametri fondamentali quali **tempo di consegna** e **mesi-persona** necessari per lo sviluppo/manutenzione di un prodotto software
- *gestire* tale sviluppo ("che fare se …")

Constructive Cost Model

Il *Constructive Cost Model* è, dunque, un modello predittivo di misura.

Esso è fondato su esponenziali derivati da studi empirici con analisi basata su regressione statistica.

$$y = a * x^b$$

Constructive Cost Model's **Attributes**

Product Factors. Required Software Reliability. Database Size. Software Product Complexity. Required Reusability. Documentation Match to Life-Cycle Needs. Product Reliability and Complexity.

Personnel Factors. Analyst Capability. Applications Experience. Programmer Capability. Programming Language Experience. Virtual Machine Experience. Personnel Capability. Personnel Experience. Personnel Continuity. Platform Experience. Language and Tool Experience.

Platform Factors. Execution Time Constraint. Main Storage Constraint. Computer Turnaround Time. Virtual Machine Volatility. Virtual Machine Volatility: Host. Virtual Machine Volatility: Target. Platform Volatility. Platform Difficulty. Platform.

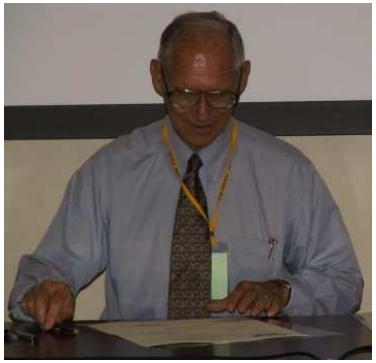
Project Factors. Use of Software Tools. Modern Programming Practices. Required Development Schedule. Classified Security Application. Multisite Development. Facilities. Requirements Volatility**.**

Constructive Cost Model

Il *Constructive Cost Model* fu creato, a partire dagli anni ’70, da Barry Boehm

(poi con la USC Andrea → Andrew Viterbi)

Barry W. Boehm @ UoTorVergata



29 September 2003

at Roma Tor Vergata 's Villa Mondragone Sala degli Svizzeri,
while signing certificates of attendance for the

1st Intl. Advanced School of Experimental Software Engineering (IASESE)

4 October 2003

Roma Tor Vergata Villa Mondragone Vasanzio Porticos,
in the final evaluation session of the

1st Experimental Software Engineering Int. Week (ESEIW)

Barry W. Boehm
giving talk at
Roma Tor Vergata DICII, July the 6th, 2012

Versioni

COCOMO 81 viene pubblicato nel 1981 (tale versione e sue varianti nel prosieguo potranno anche essere dette semplicemente **COCOMO** o **COCOMO I**).

COCOMO II viene rilasciato a partire da poco prima del 2000.

COCOMO 81

COCOMO 81 è basato sullo studio di sessanta progetti presso la compagnia Californiana TRW.

Sono stati esaminato progetti con dimensioni che vanno dalle 2000 alle 100.000 linee di codice, per linguaggi di programmazione che spaziano dagli assemblativi al PL/I.

COCOMO

COCOMO 81 è da intendersi un modello statico e analitico:

- statico, per quanto concerne le variabili di ingresso e di uscita,
- analitico, in quanto può essere anche applicato a parti di un progetto.

Livelli COCOMO

Esistono tre diversi modelli o **livelli** di **COCOMO 81**, i quali si differenziano per la precisione con cui vengono effettuate le stime:

1. Basic COCOMO
2. Intermediate COCOMO
3. Advanced COCOMO , detto anche Detailed COCOMO.

Basic COCOMO Level

È, dei tre, il livello COCOMO più facile da utilizzare ma anche il meno preciso, la stima di *effort* M viene fatta partendo dalla **dimensione** del software da sviluppare.

Questa viene calcolata - con varie differenze di conteggio - in KSLOC “Thousands of Source Lines of Code” o KNCSS “Thousands Non Commenting Source Statement”, (quest’ultima metrica, a volte, anche detta KDSI [Migliaia di], “Delivered Source Instruction”).

Intermediate COCOMO Level

Questo modello calcola lo sforzo di sviluppo del software, oltre che come funzione della grandezza del programma, espressa sempre in KNCSS, anche di un insieme di "indici di costi", detti ***Cost-driver***, che includono la valutazione di attributi relativi a prodotti, hardware, progetto e personale.

Intermediate COCOMO Level

In pratica, esistono dei fattori che vanno valutati dall'esperto in scala ordinale e poi, utilizzando un'apposita tabella, una per ogni complessità, vanno trasposti in scala reale, ciascuno con valore intorno a 1.00.

Alla fine, moltiplicando fra loro tali fattori, si computa un *fattore correttivo complessivo* che va moltiplicato per il coefficiente impiegato per la valutazione dello sforzo M in caso di metodo base.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Advanced COCOMO Level

Questo modello (detto anche Detailed) COCOMO) incorpora tutte le caratteristiche del COCOMO intermedio con l'aggiunta della valutazione dell'impatto dei vari costi per ogni "disciplina"/fase del processo software (analisi, progettazione, ecc.).

COCOMO DEVELOPMENT MODES

Per ciascun livello di COCOMO (I) esistono tre diverse possibili *development modes* o **complessità**: maggiore è quest'ultima maggiore è l'effort necessario per lo sviluppo.

Le complessità previste sono dette:

- Organic
- Semi-detached
- Embedded

COCOMO II

In COCOMO II, alcuni dei più importanti fattori che contribuiscono alla durata e al costo di un progetto sono gli "**Scale Drivers**". Essi rimpiazzano i *development modes* di COCOMO.

Some Details about the Constructive Cost Model in Synthesis

Storia

Oltre a quanto già detto, si può aggiungere:

- Il database dei progetti è stato periodicamente aggiornato e arricchito con dati relativi a ulteriori progetti
- Il modello è stato aggiornato anche attraverso la definizione di nuovi coefficienti di costo

Assunzioni (1/2)

1. Il modello di processo assunto a riferimento da COCOMO è quello a cascata.
    1. In tale ciclo di vita, il COCOMO identifica quattro fasi:
        1. *Pianificazione e analisi dei requisiti.*
        2. Progettazione dell'applicazione.
        3. Sviluppo (codifica e test di unità).
        4. Integrazione e test.
2. I requisiti vengono considerati essenzialmente stabili (primo rilascio).
3. Il progetto dell'architettura dell'applicazione è svolto da un piccolo numero di persone molto capaci e con una profonda conoscenza del problema applicativo.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Assunzioni (2/2)

4. Il progetto di dettaglio dell'applicazione, il suo sviluppo e il test a livello di modulo sono compiuti, per quanto possibile, in parallelo da diversi team di programmatori.
5. La documentazione viene scritta in modo incrementale durante l'intero corso del progetto.
6. Il progetto è gestito (vi è una struttura di gestione del progetto).

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Uscite del modello

Il COCOMO è costituito da formule per calcolare la **quantità di lavoro** richiesta **(sforzo, "Effort")**, M, in mesi persona, e il **tempo solare** di sviluppo, T.

La stima si basa principalmente sulla **dimensione del programma** da sviluppare espressa in KDSI.

Noti T ed M, è possibile calcolare la quantità di risorse umane necessarie.

Uscite del modello

Nei livelli più avanzati di COCOMO 81, vengono introdotti coefficienti correttivi che tengono conto di:

- caratteristiche del progetto
- ambiente complessivo di sviluppo.

Manutenzione

Il COCOMO permette di stimare separatamente e con altre formule anche lavoro e tempi di manutenzione.

Calcolo per sviluppo ex-novo

Il calcolo viene sviluppato attraverso i seguenti due passi:

1. Vengono *stimati* i valori di M e T *per le ultime tre fasi:* Progettazione, Sviluppo (codifica e test), Integrazione e test.
2. Vengono *calcolati* separatamente i valori relativi alla pianificazione: a tale scopo, una serie di *valori correttivi* è messa a disposizione da COCOMO 81.

Classi di complessità

Come già accennato, IN COCOMO 81 incide sul calcolo il *development mode* (complessità) dell'applicazione:

- Applicazioni semplici (*Organic mode*)
- Applicazioni intermedie (*Semi-detached mode*)
- Applicazioni complesse (*Embedded mode*).

Livelli del modello

Come anche già detto COCOMO è strutturato in una gerarchia di modelli a finezza crescente:

- Modello Base
- Modello Intermedio
- Modello Avanzato.

Calcolo di M e T.
Modello Base

La stima viene effettuata considerando una sola variabile indipendente: la dimensione prevista (S, "Size)") per l'applicazione da sviluppare (in KDSI) e parametri dipendenti dalla complessità ("*development mode*") di sviluppo dell'applicazione stessa.

$$M = a_b * (S)^{b_b}$$

$$T = C_b * (M)^{d_b}$$

Complessità	a_b	b_b	c_b	d_b
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

Carico e tempi per fase

Quantità di lavoro (M)

Modo	Fasi	Dimensione (in KDSI) 2	8	32	128	512
Organic	Pian. e requis.	6	6	6	6	
	Progetto	16	16	16	16	
	Sviluppo	68	65	62	59	
	Integ. e test	16	19	22	25	
Semidetached	Pian. e requis.	7	7	7	7	7
	Progetto	17	17	17	17	17
	Sviluppo	64	61	58	55	52
	Integ. e test	19	22	25	28	31
Embedded	Pian. e requis.	8	8	8	8	8
	Progetto	18	18	18	18	18
	Sviluppo	60	57	54	51	48
	Integ. e test	22	25	28	31	34

Tempo (T)

Modo	Fasi	Dimensione (in KDSI) 2	8	32	128	512
Organic	Pian. e requis.	10	11	12	13	
	Progetto	19	19	19	19	
	Sviluppo	63	59	55	51	
	Integ. e test	18	22	26	30	
Semidetached	Pian. e requis.	16	18	20	22	24
	Progetto	24	25	26	27	28
	Sviluppo	56	52	48	44	40
	Integ. e test	20	23	26	29	32
Embedded	Pian. e requis.	24	28	32	36	40
	Progetto	30	32	34	36	38
	Sviluppo	48	44	40	36	32
	Integ. e test	22	24	26	28	30

Lavoro per fase
Organic Mode

Modo	Fasi	Dimensione (in KDSI) 2	8	32	128	512
Quantità di lavoro (M)						
Organic	Pian. e requis.	6	6	6	6	
	Progetto	16	16	16	16	
	Sviluppo	68	65	62	59	
	Integ. e test	16	19	22	25	

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Lavoro per fase
Semidetached Mode

Semidetached						
	Pian. e requis.	7	7	7	7	7
	Progetto	17	17	17	17	17
	Sviluppo	64	61	58	55	52
	Integ. e test	19	22	25	28	31

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Lavoro per fase
Embedded Mode

Embedded	Pian. e requis.	8	8	8	8	8
	Progetto	18	18	18	18	18
	Sviluppo	60	57	54	51	48
	Integ. e test	22	25	28	31	34

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Tempo per fase
Organic Mode

Modo	Fasi	Dimensione (in KDSI) 2	8	32	128	512
Tempo (T)						
Organic	Pian. e requis.	10	11	12	13	
	Progetto	19	19	19	19	
	Sviluppo	63	59	55	51	
	Integ. e test	18	22	26	30	

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Tempo per fase
Semi-detached Mode

Pian. e requis.	16	18	20	22	24
Progetto	24	25	26	27	28
Sviluppo	56	52	48	44	40
Integ. e test	20	23	26	29	32

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Tempo per fase
Embedded Mode

Pian. e requis.	24	28	32	36	40
Progetto	30	32	34	36	38
Sviluppo	48	44	40	36	32
	22	24	26	28	30

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Sintesi:
comportamenti % vs. dimensione

All'aumentare della **dimensione**:

- Per quanto riguarda tempo e lavoro:
  - In *Sviluppo*, decrescono entrambi
  - In *Integrazione e test*, crescono entrambi.
- Per quanto riguarda il tempo:
  - In *Pianificazione e requisiti*, cresce
  - In *Progettazione*, cresce.
- Per quanto riguarda il lavoro:
  - In *Pianificazione e requisiti* e in *Progettazione*, è indipendente.

Sintesi comportamenti % vs. complessità

All'aumentare della **complessità di sviluppo**:

- Per quanto riguarda tempo e lavoro:
  - In *Pianificazione e requisiti* e in *Progettazione*, crescono
  - In *Sviluppo*, decrescono.
- In *Integrazione e test*: il lavoro cresce, mentre il tempo cresce fino a 32K e decresce dopo.

Attributi correttivi

Sono previsti 15 attributi, fra cui, ad esempio:

- Required Software Reliability
- Virtual Machine Volatility
- Applications experience
- Use of Modern Programming Practice

Questi vengono prima misurati in una scala ordinale e poi trasformati in coefficienti reali, $C_{j=1..15}$.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Coefficienti correttivi

Coefficiente correttivo	Valore del coefficiente: Molto Basso	Basso	Nom.	Alto	Molto Alto	Extra Alto
Product attributes						
RELY – REquired reliabiLitY (Affidabilità richiesta)	0,75	0,88	1,0	1,15	1,40	
DATA – DATAbase size (Dimensione della base di dati)		0,94	1,0	1,08	1,16	
CPLX – product C om PLeXity (Complessità del prodotto)	0,70	0,85	1,0	1,15	1,30	1,65
Computer attributes						
TIME – execution TIMEconstraints (Requisiti di efficienza)			1,0	1,11	1,30	1,66
STOR – main STORage constraints (requisiti di memoria centrale)			1,0	1,06	1,21	1,56
VIRT –VIRTual machine volatilty (variabilità dell'ambiente di sviluppo)		0,87	1,0	1,15	1,30	
TURN – computer TURNaround time (tempi di risposta)		0,87	1,0	1,07	1,15	

Coefficiente correttivo	Valore del coefficiente: Molto Basso	Basso	Nom.	Alto	Molto Alto	Extra Alto
Personnel attributes						
ACAP – Analyst CAPability (esperienza degli analisiti)	1,40	1,19	1,0	0,86	0,71	
AEXP – Application EXPerience (esperienza nel settore applicativo)	1,29	1,13	1,0	0,91	0,82	
PCAP – Programmer CAPability (esperienza dei programmatori)	1,42	1,17	1,0	0,86	0,70	
VEXP – Virtual machine EXPerience (consocenza dell'ambiente di sviluppo)	1,21	1,10	1,0	0,90		
LEXP – programming Language EXPerience (esperienza nell'uso del linguaggio di programmazione)	1,14	1,07	1,0	0,95		
Project attribute						
MODP – use of MODern programming Practice (uso di tecniche di programmazione avanzate)	1,24	1,10	1,0	0,91	0,82	
TOOL – use of sofware TOOLs (uso di strumenti avanzati)	1,24	1,10	1,0	0,91	0,83	
SCED – required development SChEDule (vincoli sui tempi di sviluppo)	1,23	1,08	1,0	1,04	1,10	

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Required Software Reliability

Esprime il livello di affidabilità richiesto al software prodotto.

- **Very low**: l'effetto di un guasto è un semplice inconveniente che non procura danni reali
  - Esempio: errore in un programma di demo.
- **Low**: l'effetto di un errore è facilmente recuperabile
  - Esempio: errore in un programma di simulazione delle condizioni meteorologiche.
- **Nominal**: l'effetto è considerato in modo significativo dall'utente, che comunque è in grado di gestirlo senza eccessivi problemi
  - Esempio: gestione del magazzino.
- **High**: l'effetto di un errore può causare gravi perdite finanziarie
  - Esempio: sistemi bancari.
- **Very high**: rischi per la vita umana
  - Esempio: avionica.

Virtual Machine Volatility

Esprime il livello di stabilità dgli ambienti utilizzati per lo sviluppo e l'esecuzione dell'applicazione:

- Very high: cambi da giornalieri a quindicinali.
- High: cambi da mensili a bimestrali.
- Nominal: cambi da trimestrali a semestrali.
- Low: cambi ogni sette o più mesi.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Application experience

Esprime il livello di esperienza media del team di sviluppo nel settore applicativo considerato.

- **Very low**: ? 4 mesi.
- **Low**: 1-2 anno.
- **Nominal**: 3-5 anni.
- **High**: 6-11 anni.
- **Very high**: 12 anni.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Use of Modern Programming Techniques

Esprime il livello qualitativo delle tecniche di sviluppo utilizzate nel progetto.

- **Very low**: nessuna tecnica utilizzata.
- **Low**: Inizio di uso sperimentale.
- **Nominal**: Esperienza significativa nell'uso di qualche tecnica.
- **High**: Esperienza significativa nell'uso della maggior parte delle tecniche.
- **Very high**: Uso di routine delle diverse tecniche.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Valori dei Coefficienti correttivi (1/2)

Coefficiente correttivo	Valore del coefficiente: Molto Basso	Basso	Nom.	Alto	Molto Alto	Extra Alto
Product attributes						
RELY – REquired reliabiLitY (Affidabilità richiesta)	0,75	0,88	1,0	1,15	1,40	
DATA – DATAbase size (Dimensione della base di dati)		0,94	1,0	1,08	1,16	
CPLX – product C om PLeXity (Complessità del prodotto)	0,70	0,85	1,0	1,15	1,30	1,65
Computer attributes						
TIME – execution TIMEconstraints (Requisiti di efficienza)			1,0	1,11	1,30	1,66
STOR – main STORage constraints (requisiti di memoria centrale)			1,0	1,06	1,21	1,56
VIRT –VIRTual machine volatiltv (variabilità dell'ambiente di sviluppo)		0,87	1,0	1,15	1,30	
TURN – computer TURNaround time (tempi di risposta)		0,87	1,0	1,07	1,15	

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Valori dei Coefficienti correttivi (2/2)

Coefficiente correttivo	Valore del coefficiente: Molto Basso	Basso	Nom.	Alto	Molto Alto	Extra Alto
Personnel attributes						
ACAP – Analyst CAPability (esperienza degli analisiti)	1,40	1,19	1,0	0,86	0,71	
AEXP – Application EXPerience (esperienza nel settore applicativo)	1,29	1,13	1,0	0,91	0,82	
PCAP – Programmer CAPability (esperienza dei programmatori)	1,42	1,17	1,0	0,86	0,70	
VEXP – Virtual machine EXPerience (consocenza dell'ambiente di sviluppo)	1,21	1,10	1,0	0,90		
LEXP – programming Language EXPerience (esperienza nell'uso del linguaggio di programmazione)	1,14	1,07	1,0	0,95		
Project attribute						
MODP – use of MODern programming Practice (uso di tecniche di programmazione avanzate)	1,24	1,10	1,0	0,91	0,82	
TOOL – use of software TOOLs (uso di strumenti avanzati)	1,24	1,10	1,0	0,91	0,83	
SCED – required development SChEDule (vincoli sui tempi di sviluppo)	1,23	1,08	1,0	1,04	1,10	

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Formula aggiornata per M

Si calcola prima lo sforzo nominale.

$$M_{NOM} = a_i * (S)^{b_i}$$

Poi lo si moltiplica per ciascuno dei coefficienti correttivi.

$$M = M_{NOM} * \Pi_{j=1..15} C_j$$

Mode	a_i	b_i
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Component Level Estimation (CLE)

Component Level Estimation (CLE)

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Modello avanzato

Per rendere più agevole ed efficace il procedimento di stima, i dati vengono organizzati in una gerarchia *bottom-up* a tre livelli:

1. Al primo livello vengono considerati i moduli che costituiscono l'applicazione.
2. Al secondo livello vengono considerati i sottosistemi: un sottosistema è costituito da un sottoinsieme dei moduli descritti al primo livello.
3. Al terzo livello si considera il prodotto nel suo complesso come costituito dall'insieme dei sottosistemi identificati al punto precedente.

Ciascun coefficiente correttivo viene applicato solo a uno dei tre livelli.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Modello avanzato

Se nel modello intermedio di COCOMO la stima della distribuzione dello sforzo nelle varie fasi viene fatto dipendere solo dalle dimensioni del programma, nel modello avanzato i coefficienti correttivi variano da fase a fase.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

Precisione di COCOMO

Definendo una stima accurata quando il suo errore è minore del 20%, COCOMO produce stime accurate con le seguenti probabilità:

- nel 25% dei casi, per il modello base.
- nel 68% dei casi, per il modello intermedio.
- nel 70% dei casi, per il modello avanzato.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

COCOMO vs. altri modelli per il dimensionamento dello staff

COCOMO vs. altri modelli.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti

COCOMO II

COCOMO vs. altri modelli.

Basato su COCOMO, Luigi Lavazza, CEFRIEL e Politecnico di Milano e altre fonti