Ingegneria del Software

Misurare il Software
Metriche

A cosa ci interessiamo

- Valutazione di costi, tempi (*effort*)
- Dare una misura della *qualità*

- A cosa non ci interessiamo
  - Valutazioni delle prestazioni (a meno di quanto esse influenzino la qualità)

Misurare il software

"*Non si può controllare ciò che non si può misurare*"
(De Marco)

- Esempi:
  - Quanto è usabile un dato software?
  - Che tempo si prevede per lo sviluppo?
  - Quanto costa un dato processo?
  - Qual è la produttività dei programmatori?
  - Quant'è "buono" il codice prodotto?
  - Quanti programmatori si richiedono?
  - Quanto è portabile il software prodotto?

Misurazione

- *Misurazione:* il processo di assegnazione di simboli, normalmente numeri, per rappresentare un attributo dell'entità di interesse, secondo regole definite
- Quattro elementi:
  - **Entità**: l'oggetto o l'evento su cui si indaga (un tavolo, un viaggio..)
  - **Attributo**: caratteristica dell'oggetto (altezza, durata, costo, ..)
  - **Forma** (della rappresentazione): l'altezza si misura in centimetri; gli indumenti con S (small) , M (medium), L (large), XL (extra large); la benzina con "normale", "super".
  - **Regole**: per arrivare a determinare il valore dell'attributo (in modo da rendere il processo ripetibile, non soggettivo)

Metriche

- Si usa il termine "metrica" per indicare una misura *diretta* o *indiretta* di un qualche attributo di una entità di interesse.
- Si misurano gli attributi NON le entità
  - Si misura l'altezza di Franco **NON** Franco
  - Anche se nel parlare si dice "Giorgio è più alto di Franco" (per dire "l'altezza di Giorgio è maggiore di quella di Franco")
- Misura **diretta**: quando esiste uno strumento per effettuare direttamente la misura (lunghezza, peso, densità,…)
- Il metro è stato definito nel 1889 (a Parigi c'è il metro campione)

Misura indiretta

"Rendere misurabile ciò che non è direttamente misurabile"

- Si fanno misure dirette e si combinino per dare una quantificazione di un attributo non direttamente misurabile.
  - Esempio: nel decathlon si misurano tempi e lunghezze; le misure vengono poi combinate e pesate in modo da dare luogo ad uno "score" che identifica il "miglior atleta"
  - Quoziente di intelligenza (IQ): misurato attraverso una serie di test standardizzati

Metriche software

- Si riferiscono a misurazioni riguardanti:
  - Costi, tempi ed *effort* richiesti per lo sviluppo di un prodotto
  - Affidabilità, funzionalità, usabilità, portabilità, ecc.

- Metriche formali o informali (soggettive):
  - Affidabilità
    - Informale: quanto si "conta" sul prodotto
    - Formale: probabilità di assenza di malfunzionamenti su un periodo di tempo
  - Portabilità
    - Informale: quanto è facile (si ritiene facile) portare il prodotto
    - Formale: 1 - NStatementNuovi/NStatementTotale

Esempi di "metriche"

- Con il termine **Qualità** ci si riferisce all'insieme degli attributi che caratterizzano la "bonta" di un prodotto

Attributi e relative metriche

Attributo Interno
(visibile a che sviluppa)

	Misura
PRODOTTO	
Dimensione	SLOC, Mbyte
Complessità	N. ciclomatico
Funzionalità/ /Dimensione	Function Points

Attributo Esterno
(visibile all'utente)

	Misura
Usabilità	??
Manutenibilità	??
Portabilità	??
Affidabilità	MTTF
Prestazioni	tempo risp

- PROCESSO di sviluppo

Tempo sv.	Mesi
Effort	Mesi/uomo

Qualità

Stima dei costi software

(Costo dello sviluppo del software)

- Il software è una attività ad alta intensità di lavoro umano
  - Tendenzialmente il costo del software coincide con quello della forza lavoro. Eventualmente costi accessori (spese fisse, ….)

- Diversi aspetti
  - Stima dei costi
  - Stima della forza lavoro da allocare a un progetto
  - Stima del tempo di sviluppo di un prodotto software

..Stime dei costi software

- E' ragionevole assumere che il costo (di sviluppo) di un prodotto software sia funzione della dimensione (ovvero del volume, v) del prodotto stesso

$$C = f(v)$$

- Problema:
  - come si misura v ?
  - che forma ha f ?

- Per il momento soffermiamoci su v. Le metriche più usate sono:
  - Il numero di linee di codice (SLOC)
  - I punti funzione (function points)

Si dovrebbe dire "le metriche più usate sono riferite a questi attributi"

SLOC - *Source Lines Of Code*

- E' la misura software più antica (anni settanta)
  - Facile da effettuare
  - Per quanto primitiva è forse ancora la misura che riscuote più consenso
  - Spesso è la base per derivare altre misure (per esempio misure di produttività, effort, ecc.) o per modelli più rifiniti di previsione dei costi (COCOMO)

Uso per valutazione o previsione

Produttività: SLOC/GU (ovvero SLOC/MU)

- Valutazione ex-post:
  - Misura SLOC
  - Misura dei GU spesi (non solo in programmazione, ma anche in analisi, progetto, sviluppo, testing, …)
  - La produttività è una misura di come si è svolto il progetto
- Previsione
  - Previsione SLOC
  - Previsione della produttività
    - Ambedue sulla base della tipologia di sistema e di dati di esperienze precedenti
  - Calcolo dei GU necessari allo sviluppo *(effort)*

Come si contano?

- A seconda di come si contano si possono avere variazioni fino al 500% !!! [Jones].
  - Questa valutazione è un po' vecchia, quando non si era raggiunto il grado di parziale standardizzazione del tempo presente

Problemi:

- Si contano i commenti ?
- Come si contano le linee multi-statement?
- Come si contano i contributi delle librerie?
- Come si contano le eventuali linee generate dai tools o dai framework.
- I linguaggi di programmazione hanno differente espressività: uno statement C equivale a 5-10 o più linee Assembler; come se ne tiene conto?

…Come si contano?

Tendenzialmente

- Escludere dal conteggio linee bianche e commenti
- Escludere le linee di libreria e quelle generate automaticamente da tools o altro
- Per quanto riguarda gli statement multi riga:
  - Contare uno statement come
  - **Contare uno statement in base al numero di righe occupate**

  Gran parte dei tool in circolazione adotta la seconda regola. Essa implica il conteggio dei "return", escludendo quelli relativi a righe bianche e commenti

- Si parla di SLOC fisiche (**PSLOC**), anche si usa normalmente il simbolo SLOC

SLOC (qualche dato)

- NASA space shuttle flight control
  430 K (shuttle) 1,4 M (ground)
- Microsoft Windows 3.1 (1992) 3 M
- Microsoft NT (1992) 4 M
- Sun Solaris (1998) 7 M
- Microsoft Windows 95 14 M
- Microsoft Windows 98 18 M
- Microsoft NT5.0 (1998) 20 M
- RedHatLinux 6.2 (2000) 17 M
- RedHatLinux 7.1 (2002) 32 M
- Microsoft Windows XP 40 M
- Microsoft Vista 50 M

TOOLS

- Ci sono svariati strumenti per misurarle:
  - **http://sunset.usc.edu/research/CODECOUNT/download.html**
  - **http://www.kclee.de/clemens/java/javancss/**
  - Metrics (Plugin di Eclipse) (E' quello che useremo noi)
    - Conta i "return" escludendo linee bianche e commenti
- Ovviamente quando si usa un tool la prima cosa da fare è capire cosa misura

Un esempio: Uso SLOC presso JPL

Uso delle SLOC per stimare il costo di progetto

- Il primo passo è fare una previsione delle PSLOC in base a
  - Analogia con altri progetti di cui si hanno i dati
  - Giudizio di personale esperto ("guru")
  - Valutazioni ad hoc

Handbook for Software Cost Estimation – Jet Propulsion Laboratory – May 2003 (scaricabile da rete)

Coefficienti di abbattimento (da [JPL])

- Successivamente vengono applicati questi coefficienti di abbattimento in relazione al linguaggio usato. Si ottengono le cosiddette LSLOC (Logical SLOC)

Language	To Derive Logical SLOC
Assembly and Fortran	Assume Physical SLOC = Logical SLOC
Third-Generation Languages[3] (C, Cobol, Pascal, Ada 83)	Reduce Physical SLOC by 25%
Fourth-Generation Languages (e.g., SQL, Perl, Oracle)	Reduce Physical SLOC by 40%
Object-oriented Languages[3] (e.g., Ada 95, C++, Java, Python)	Reduce Physical SLOC by 30%

Il manuale del JPL prevede anche la possibilità di conteggiare il codice autogenerato. Le SLOC autogenerate vengono riportare a LSLOC attraverso un coefficiente di abbattimento che per i linguaggi OO è valutato circa 0,09

Introduciamo la produttività

- Usare dati storici: ogni azienda dovrebbe costruirsi un DB di progetti da cui derivare la produttività per tipologia di progetto
- In mancanza in JPL fanno riferimento a questa tabella

Table 4. Software Development Productivity for Industry Average Projects

Characteristic	Software Development Productivity (SLOC/WM)
Classical rates	130 – 195
Evolutionary approaches[4]	244 – 325
New embedded flight software	17 - 105

Sistemi più semplici di un sistema di un flight system

- Si calcola Effort_0 = SLOC preventivate/Produttività

(*) WM: Work-Month

…produttività

- NASA e JPL hanno anche questa tabella di riferimento

Table 3-7. Software Development Productivity for JPL and NASA Average Projects (Equivalent Logical SLOC)

Software Class	**Mean SW Development Productivity (SLOC/WM)**	**Range SW Development Productivity (SLOC/WM)**
Mission Critical Flight SW	125	13-467
Mission Support Flight SW	184	80-262
DSMS	197	148-347
Mission Critical Ground SW	239	116-519
Mission Support Ground SW	295	103-607
Development Support Ground SW	157	129-207

Tener conto del RIUSO

- JPL applica un coefficiente che tiene conto del riuso

Table 5. Effort Adjustment Multipliers for Software Heritage

Software Heritage Category	Effort Multiplier
New design and new code	1.2
Similar design and new code (nominal case)	1.0
Similar design and some code reuse	0.8
Similar design and extensive code reuse	0.6

- Si calcola Effort_1 = Effort_0 x Effort Multiplier

- Inoltre sono previsti altri fattori correttivi che tengono conto di alcuni aspetti del processo software (tabella prossima)

Altri fattori correttivi [JPL]

- Aumentare in base a questi fattori per arrivare a Effort finale

Table 6. Effort To Be Added to Software Development Effort Estimate for Additional Activities Based on Industry Data[6]

WBS Category	% of SW Development Effort
Software Management	Add 6-27%
System-level Test Support (includes SW Development Test-bed, SW System-level Test Support, ATLO Support)	Add 34 - 112%
Software Quality Assurance	Add 6 - 11 %
IV&V	Add 9 - 45 %
Supplemental Activities:	
Project Configuration Management	Add 3 – 6 %
Project management	Add 8 - 11 %
Acquisition management	Add 11 - 22 %
Rework	Add 17 - 22 %
Maintenance – First five years	Add 22% of SW Development Effort per year of Maintenance

Una precisazione

- L'*effort* calcolato con la procedura vista si riferisce all'intero processo di sviluppo ed è quindi comprensivo delle fasi in cui un progetto viene scomposto.
  - Analisi dei requisiti
  - Progetto
  - Progetto dettagliato
  - Codifica
  - Unit testing
  - Integrazione

Previsione schedule time [JPL]

- In precedenza abbiamo tacitamente assunto che i conti si riferissero al SW nel suo complesso.
- Nella pratica occorre definire la l'organizzazione del lavoro (*Work Breakdown Strucure* – WBS) e individuare gli elementi che la compongono e in che relazioni stanno. Per esempio:
  - Il sistema è diviso in più sottosistemi o packages (*WBS elements*)
  - La valutazione dell'effort deve essere fatta per ogni elemento WBS
  - Il package B viene sviluppato dopo A e in parallelo a C
- Il WBS viene definito preliminarmente nella fase iniziale del progetto, e può essere soggetto ad aggiustamenti in base alle risultanze dell'analisi che segue

…Previsione schedule time [JPL]

1. Per ogni elemento del WBS calcolare il tempo richiesto, in base alle risorse allocate
   - Esempio se l'*effort* per l'elemento A e 10 MU, e si allocano 3 unità di personale, per A si deve prevedere 3,3 mesi di tempo di sviluppo
   - Prevedere un margine di sicurezza (1 mese su 1 anno)
2. Precisare le dipendenze e l'ordine di sviluppo dei vari elementi e quali elementi vengono sviluppati in parallelo, in modo da arrivare al piano temporale complessivo (schedule) di sviluppo
3. Identificare eventuali percorsi critici e rimettere mano allo schedule.
4. Confrontare lo schedule con i tempi previsti per il progetto ed eventualmente modificare/aumentare/riallocare le persone (questo può comportare la verifica della compatibilità col budget e la necessità di dover reiterare il procedimento)

…Previsione schedule time [JPL]

- Alla fine confrontare con i dati di esperienza (qui sotto quelli usati da JPL)

Table 8. Allocation of Schedule Time over Software Development Phases

Phase	Industry Data (mean)
Requirements Analysis	18
Software Design	22
Implementation	36
SW Integration & Test	24

Non include il progetto di dettaglio

Include il progetto di dettaglio, la codifica e lo unit testing

Come ripartire l'effort [JPL]

- Verificare anche l'allocazione della forza lavoro con i dati della pratica industriale

Table 9. Allocation of Effort for New, Modified, or Converted Software Based on Industry Data

Phase	New Software[11] %	Modify Existing Software %	Convert Software %
Requirements Analysis and Design	20%	15%	5%
Detail Design, Code and Unit Test	57%	10%	5%
SW Integration &Test	23%	40%	30%
Relative Effort	100%	65%	40%

- Ovviamente questi dati hanno senso per il contesto per il quale vengono proposti. Molto meglio disporre di dati aziendali che di dati industriali generici

Sintesi del metodo seguito in JPL (predizione)

1. Dare una valutazione del numero di PSLOC
   - Basata su dati storici di progetti similari
2. Apportare un abbattimento in base al tipo linguaggio
   - Passaggio da PSLOC a LSLOC
3. Calcolare l'Effort (LSLOC/produttività)
   - in base a dati di produttività su progetti analoghi
4. Aggiustare l'Effort in base a fattori correttivi relativi a
   - Fattore di riuso, fattori per attività addizionali (e.g., SW mangmnt)
5. Allocare l'Effort, frazionandolo, in base all'organizzazione del progetto (WBS)
   - (Per esempio: il progetto è suddiviso in packages)
   - Per ogni elemento del WBS allocare le risorse, prendendo a riferimento dati di esperienza (30% all'analisi, 30% coding, …) e calcolare il tempo richiesto.
   - Tenere conto delle precedenze tra gli elementi del WBS
   - Iterare fino a che non si trova un piano soddisfacente

Altra metrica

I Function Points

Misurano la funzionalità di un sistema software

(Interpretabili come dimensione)

FPA (Function Point Analysis)

- Introdotta da Allan Albrecht, IBM (1979)
- Misura le funzionalità di un programma rilevabili dal punto di vista dell'utente finale.
  - L'aspetto fondamentale è questo: si guarda a "cosa" fa il programma, non a come è fatto.
  - Punto di vista dell'utente NON del progettista/sviluppatore
  - Le funzionalità si misurano sulle **specifiche (sui requisiti)** non sul codice
  - Indipendente dalla tecnologia
- E' abbastanza naturale passare dai Punti Funzione all'*effort* o anche alla dimensione in SLOC.

- Si tratta di un metodo empirico
- Usato ampiamente nel mondo dell'elaborazione dati (in parte anche nella programmazione OO)

FPA (Function Point Analysis)

- E' uno standard *de facto* e probabilmente sarà anche *de jure.*
- Usata in Sogei (Finsiel), Enel, Olivetti, Rai, Banca d'Italia
- Indicata dall'AIPA (Autorità Informatica nella Pubb. Amministrazione)
  - Nei documenti AIPA (attualmente CNIPA):

"*quale metrica dimensionale per il software nei contratti di sviluppo e manutenzione",* AIPA, Indicazioni per il dimensionamento dei progetti per adeguamento dei sistemi informativi della PA all'EURO, Luglio 97

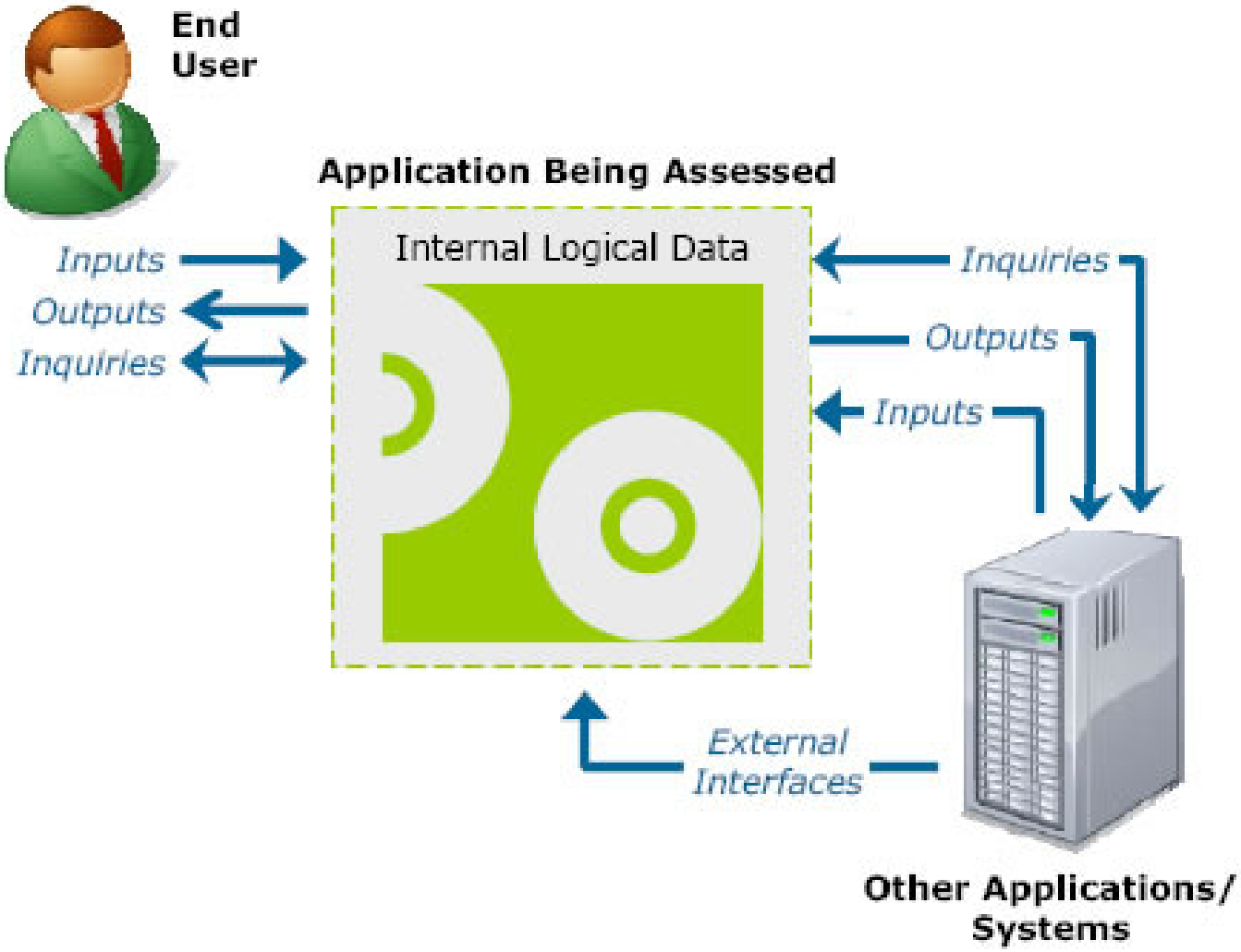
"quale strumento per stabilire la congruità tecnico-economica del progetto"

"l'Amministrazione svolgerà un'attività di monitoraggio e controllo dei punti funzione per consentire le attività di controllo e rendicontazione."

Si ritiene *"opportuno che la valutazione dei costi relativi alla produzione, allo sviluppo ed alla manutenzione del software venga effettuata utilizzando la tecnica dei punti funzione"*

Indirizzo AIPA: www.aipa.it

Schema per i FP



FPA

- Le linee guida di conteggio dei FP sono standardizzate dall'associazione **IFPUG** (***International Function Point Users Group***) nel "**Function Point Counting Practices Manual**"

- Gli elementi richiesti per il conteggio sono facilmente individuabili in qualsiasi fase del progetto. Il conteggio può essere effettuato:
  - nella fase di analisi
  - alla fine della fase di design
  - durante i processi di manutenzione del software
- Il metodo di stima attraverso i FP può essere usato:
  - per stimare la dimensione del sistema da sviluppare o già sviluppato e, in modo indiretto, per stimare il numero finale di ore/uomo necessarie a svilupparlo
  - Al fine della stima del costo occorre disporre di relazioni empiriche (di norma tabelle) tra FP e ore/uomo. Ogni produttore dovrebbe sviluppare le proprie sulla base di progetti pregressi. .

FPA

L'idea è che FPA soddisfi a questi requisiti:

- I FP devono essere indipendenti dalla tecnologia (linguaggi di programmazione, tools di sviluppo, ..)
- I FP devono misurare tutte le funzioni consegnate all'utente (la misura dei FP è la stessa indipendentemente dalla tecnica di sviluppo)
- I FP devono misurare solo le funzioni consegnate all'utente (non deve variare se i programmatori sono inesperti, i sistemi di sviluppo sono scadenti)

In sostanza si cerca una misura che sia legata solo alla **funzionalità** del software (**ciò che vede l'utente**).

Conteggio FP (IFPUG)

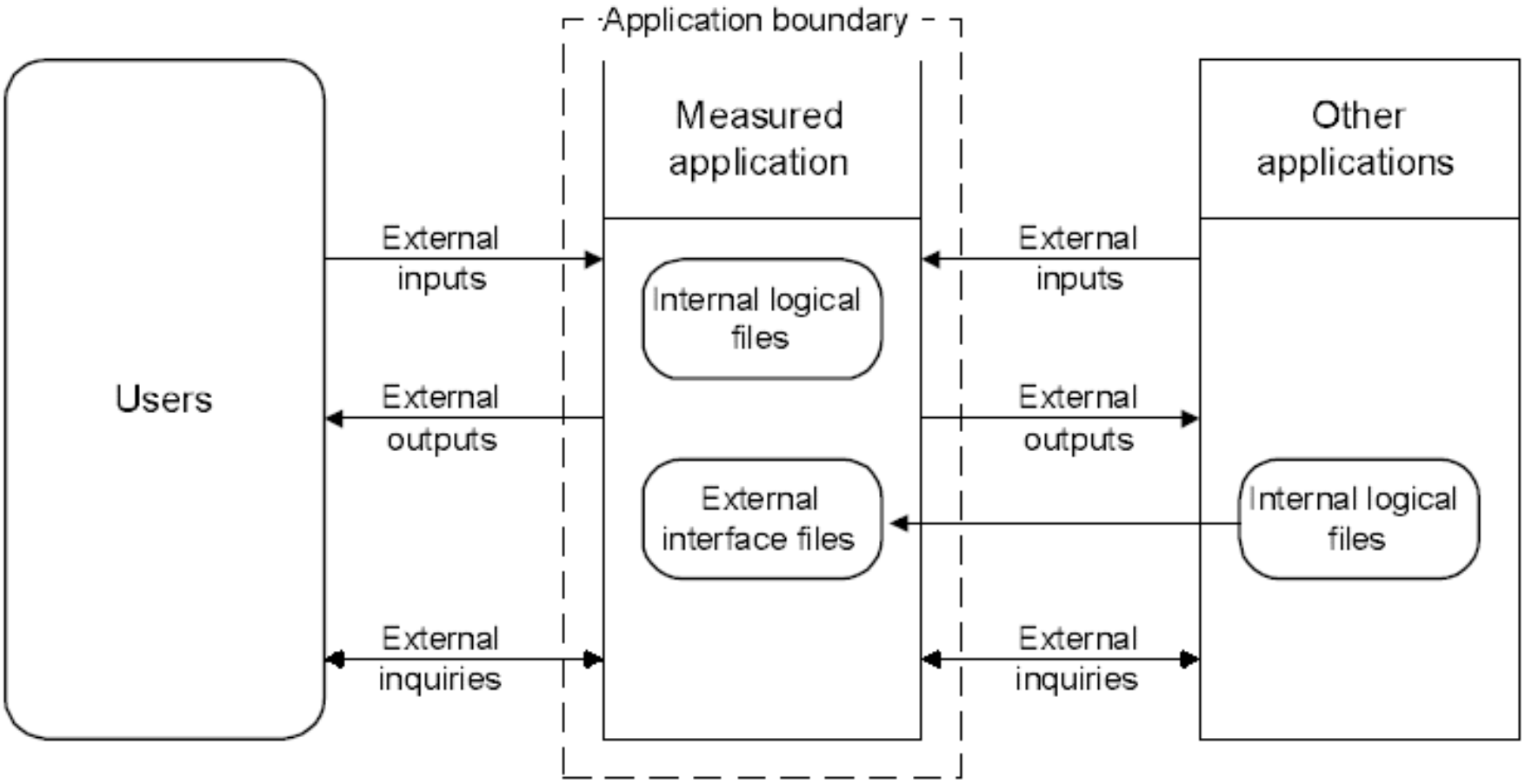
1. Si individuano e si conteggiano 5 differenti ***Function Types*** *che* sono:

   - **External Input** (EI): dati in ingresso (dall'utente o da altre applicazioni)
   - **External Output** (EO): dati restituiti in uscita
   - **External Inquiry** (EQ) interrogazioni che producono una risposta immediata del sistema (senza che il comportamento di questo sia alterato e nessun ILF modificato)

   - **Internal Logical File** (ILF): file logico di dati gestiti internamente dal sistema
   - **External Interface File** (EIF): file logico di dati riferiti o condivisi con altre applicazioni e contenuti nel confine di queste

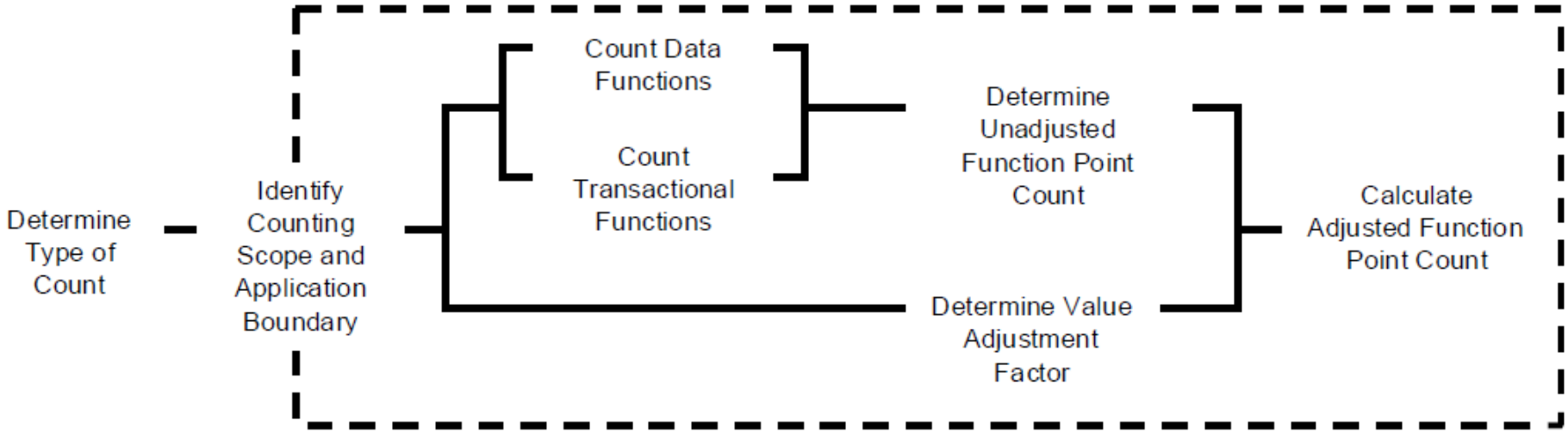
   (il primo gruppo sono i "tipi transazione", il secondo i "tipi dati")

Modello per il conteggio



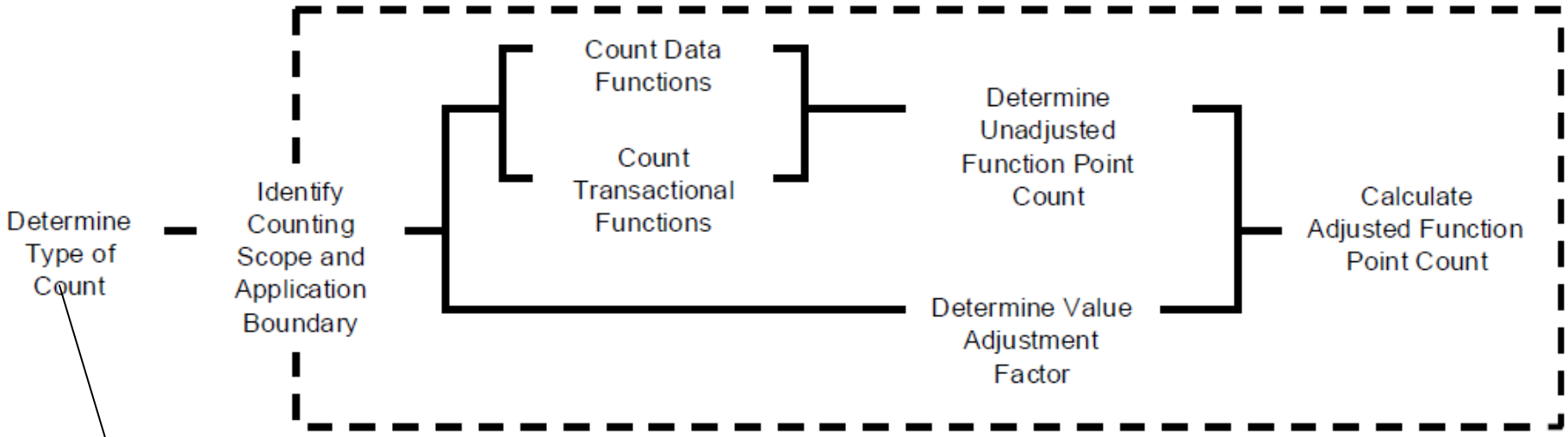
Procedura per il conteggio

- Dal Manuale IFPUG



Il manuale è diviso per capitoli ciascuno dei quali descrive una delle attività di cui si compone la procedura

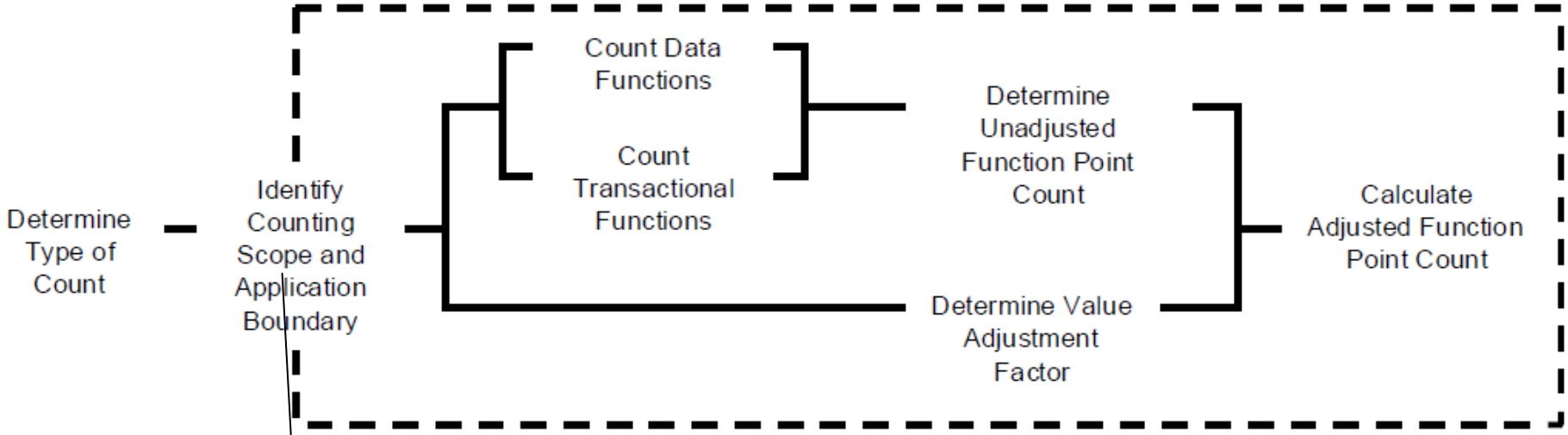
Procedura per il conteggio



- Development project
- Enhancement project
- Application

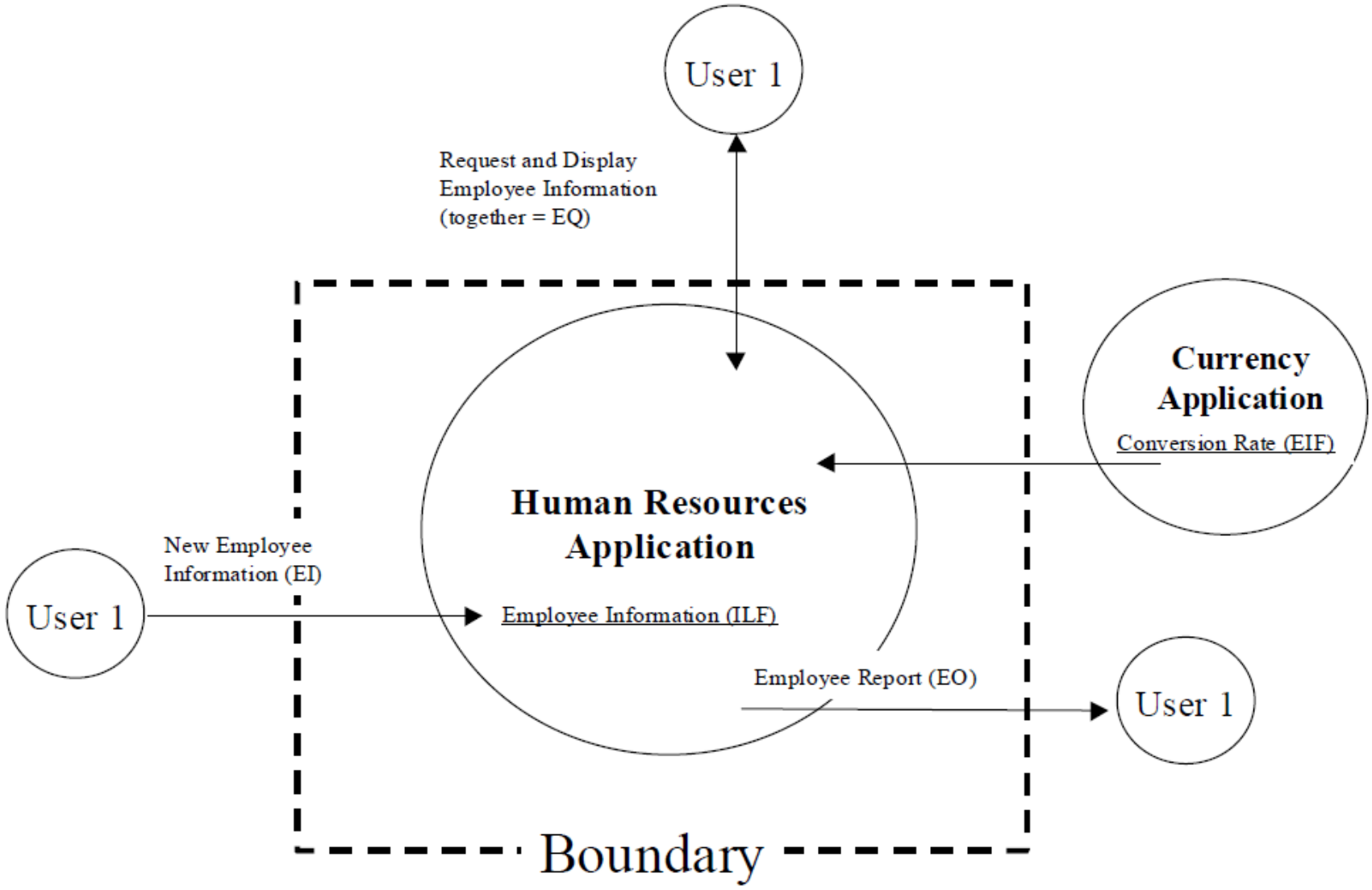
Il tipo di conteggio ha influenza sul modo di calcolo degli adjusted FP

Procedura per il conteggio

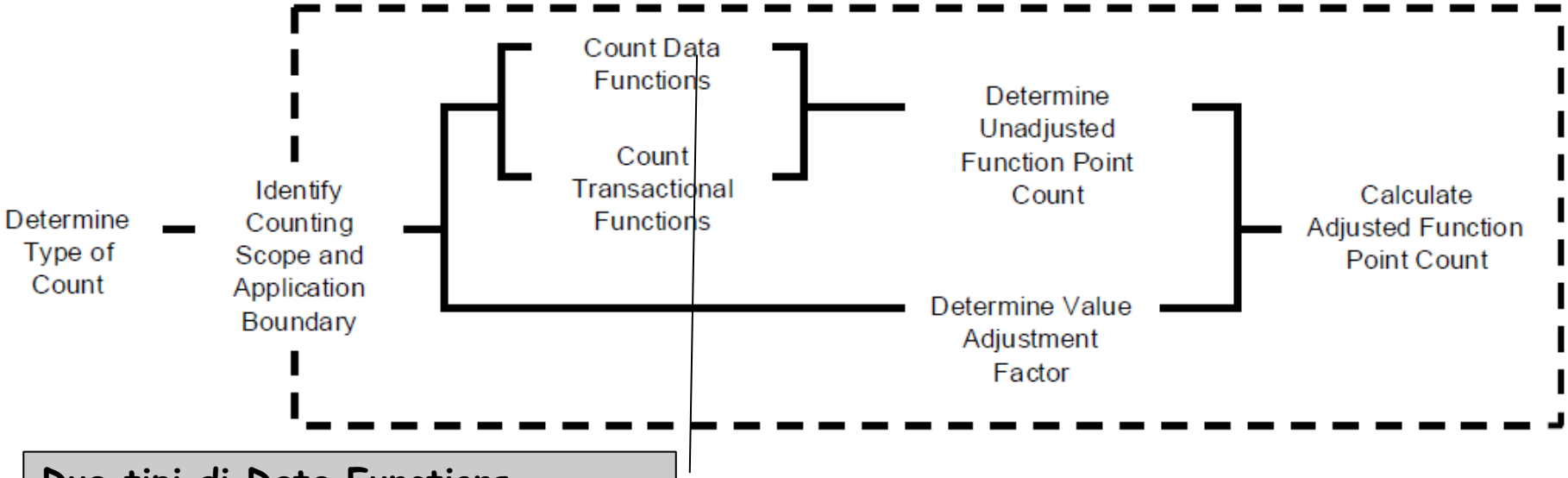


Identificazione del confine tra il sistema/progetto/applicazione che viene misurato e le applicazioni esterne e gli utenti

Esempio


User 1
Request and Display
Employee Information
(together = EQ)
Currency
Application
Conversion Rate (EIF)
Human Resources
Application
New Employee
Information (EI)
User 1
Employee Information (ILF)
Employee Report (EO)
User 1
Boundary

Procedura per il conteggio



Due tipi di Data Functions

- **Internal logical files (ILFs)**
- **External interface files (EIFs).**

Il termine *file non* indica un file in senso tradizionale

Esso indica un gruppo di dati logicamente correlati e non la loro implementazione fisica.

Identificazione ILF

- Regole di identificazione di un ILF sono:
  1. ILF è un gruppo di dati logico, o identificabile dall'utente
  2. Il gruppo è mantenuto da un processo interno all'applicazione che si sta misurando.

- Sono da considerare ILF:
  - Ogni entità logica di dati dal punto di vista utente (la struttura, la classe, la tabella, il file che rappresentano gli impiegati).
  - Ogni file o tabella interna generata o gestita dall'applicazione.
  - Ogni file o tabella gestita dall'utente.

- Non sono da considerare ILF:
  - I file intermedi o di sort.
  - I file di cui l'utente non abbia visibilità.

Identificazione EIF

- le regole di identificazione di un EIF sono:
  1. EIF è un gruppo di dati logico, o identificabile dall'utente, referenziato dall'applicazione che si sta misurando ma è ad essa **esterno**.
  2. Il gruppo di dati non è mantenuto dall'applicazione che si sta misurando.
  3. Il gruppo di dati è contato come un ILF per un'altra applicazione.
  4. Il gruppo di dati identificato non è stato contato come un ILF per l'applicazione.

- Sono da considerare EIF:
  - I files ricevuti da un'altra applicazione.
  - Gli archivi dei messaggi di help.

Complessità ILF/EIF

- **Identificati ILF e EIF deve essere calcolata per ciascuno di essi la complessità. Ciò richiede il conteggio del numero di RET e di DET**

- **Record Element Type** (RET): record non ricorsivo riconoscible dall'utente in ILF o EIF.
  - Esempio: La descrizione di un impiegato, un fattura
- **Data Element Type** (DET): campo riconoscibile all'utente (non ricorsivo).
  - Esempio: il nome, lo stipendio, l'indirizzo (ogni sottocampo è un DET); l'importo, il numero progressivo
- **Ci sono regole per conteggiare RET e DET:**
  - Se non si hanno sottogruppi, si conta l'ILF o l'EIF come un fatti di RET.
  - Si conta un DET per ogni campo utilizzato dall'utente per stabilire una relazione con un altro ILF o EIF (chiave esterna).
  - …..Svariate altre regole

Complessità ILF/EIF

Richiede il conteggio del numero di RET e di DET

- **Record Element Type** (RET): record non ricorsivo riconoscible dall'utente in ILF o EIF.
  - Esempio: La descrizione di un impiegato, un fattura
- **Data Element Type** (DET): campo riconoscibile all'utente (non ricorsivo).
  - Esempio: il nome, lo stipendio, l'indirizzo (ogni sottocampo è un DET); l'importo, il numero progressivo
- Il numero di RET si determina applicando le seguenti regole:
  - Conta un RET per ogni sottogruppo, di cui l'utente ha l'opzionalità o l'obbligo d'uso durante un processo elementare, di un ILF o EIF.
  - Se non si hanno sottogruppi, conta l'ILF o l'EIF come un RET.
- Il numero di DET si determina applicando le seguenti regole:
  - Conta un DET per ogni campo riconoscibile dall'utente e non ricorsivo.
  - Conta un DET per ogni campo utilizzato dall'utente per stabilire una relazione con un altro ILF o EIF (chiave esterna).
  - Quando due applicazioni modificano e/o si riferiscono ad uno stesso ILF/EIF, ma ognuna mantiene/riferisce separati DET, conta solamente i DET utilizzati da ogni applicazione per misurare l'ILF/EIF.

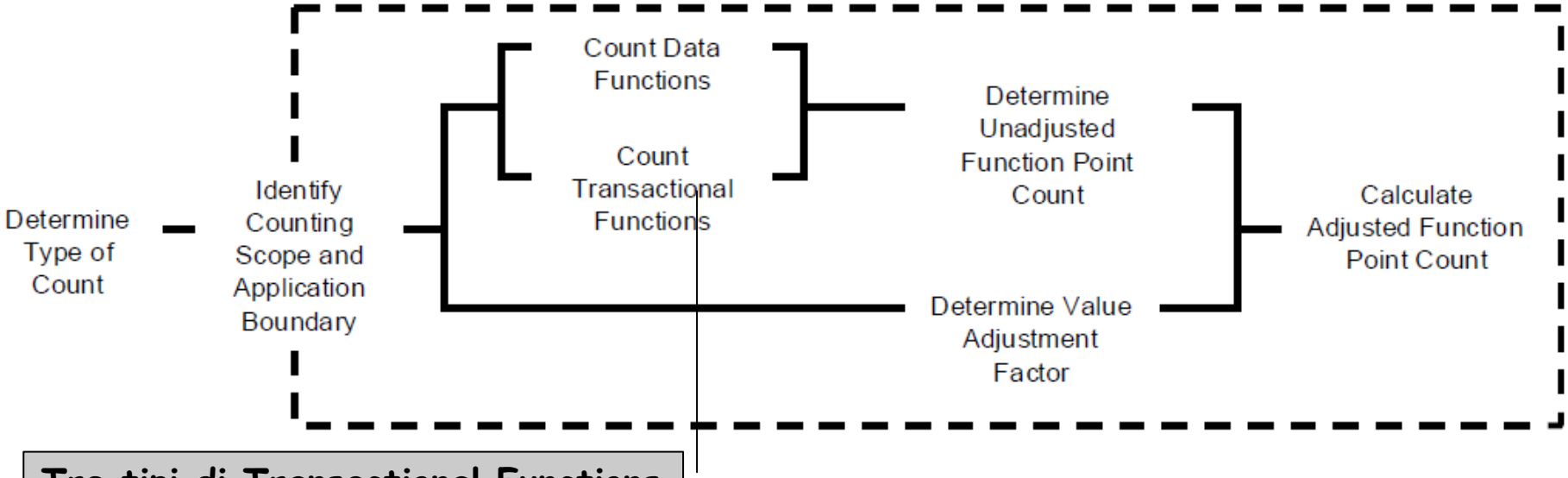
… Complessità ILF/EIF

- La quantificazione della complessità si fa in base alla tabella seguente
  - la complessità può assumere sole tre valori : **bassa, media, alta**

	1..19 DET	***20..50 DET***	***51+ DET***
1 RET	Bassa	Bassa	Media
2..5 RET	Bassa	Media	Alta
5+ RET	Media	Alta	Alta

- Analogo metodo per la complessità EI, EO e EQ.

Procedura per il conteggio



Tre tipi di Transactional Functions

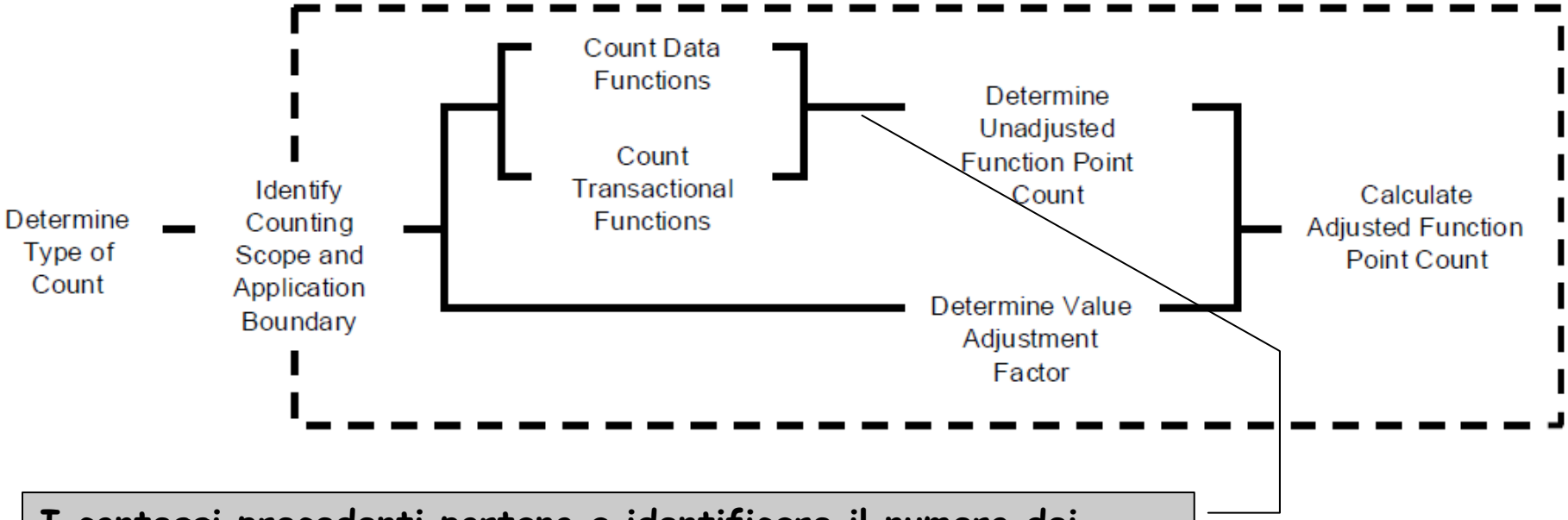
- **External Inputs (EI):**
- **External Output (EO):**
- **External Inquires (EQ)**

.

Il significato è evidente, ma ci sono delle regole precise per fare il conteggio .

Anche per questi si valuta la complessità (con criteri simili a quelli per la complessità di ILF EIF.

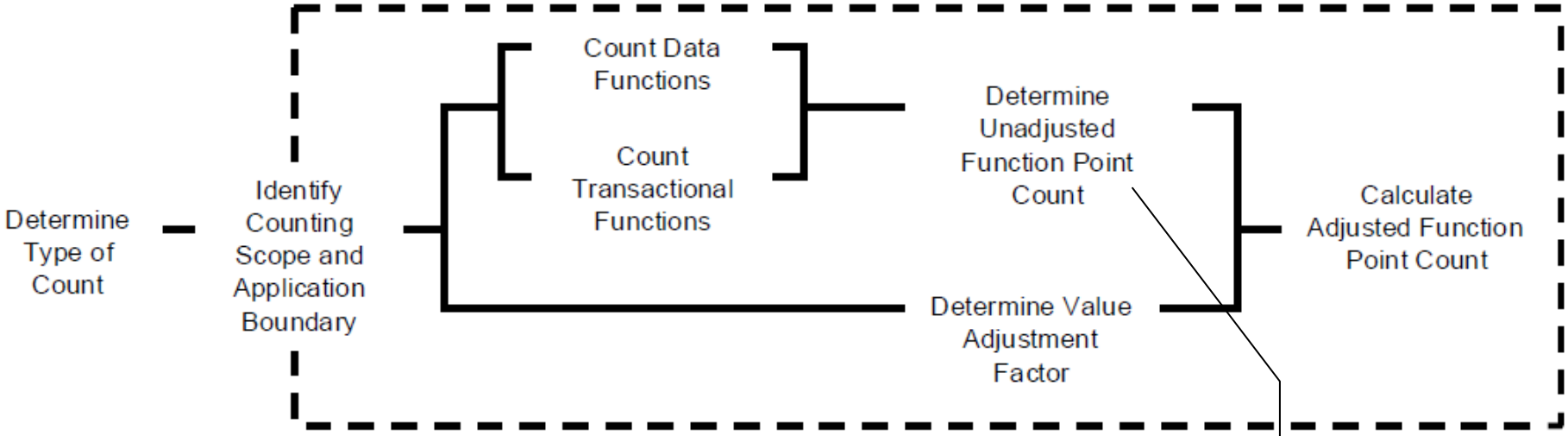
Procedura per il conteggio



I conteggi precedenti portano a identificare il numero dei Function Types suddivisi per ILF, EIF, EI, EO, EQ

Per ciascun Function Type è stata determinata la relativa complessità. (Per esempio si sono trovati 2 EI, di cui uno ad alta complessità e uno a complessità media.)

Procedura per il conteggio



Si pesano i Function Types conteggiati in base alla loro complessità ottenendo gli Unadjusted Function Points

La scala BMA fornisce i pesi in base alla tabella che segue

Tabella Complessità->peso

ILF	Bassa	7
	Media	10
	Alta	15
EIF	Bassa	5
	Media	7
	Alta	10
EI	Bassa	3
	Media	4
	Alta	6
EO	Bassa	4
	Media	5
	Alta	7
EQ	Bassa	3
	Media	4
	Alta	6

Unadjusted Function Points

ILF	Bassa	7
	Media	10
	Alta	15
EIF	Bassa	5
	Media	7
	Alta	10
EI	Bassa	3
	Media	4
	Alta	6
EO	Bassa	4
	Media	5
	Alta	7
EQ	Bassa	3
	Media	4
	Alta	6

Equivale a portare a 15 i pesi, ovvero i possibili termini della somma

$$UFP = \sum_{i=1}^{15} ((\#\ \text{elementi varietà } i) * \text{peso\_}i)$$

Unadjusted Function Points

Unadjusted Function Points

ILF	Bassa	7
	Media	10
	Alta	15
EIF	Bassa	5
	Media	7
	Alta	10
EI	Bassa	3
	Media	4
	Alta	6
EO	Bassa	4
	Media	5
	Alta	7
EQ	Bassa	3
	Media	4
	Alta	6

Esempio: supponiamo che siano stati individuati:

1 ILF, di complessità B

2 EIF, uno di complessità B e uno di complessità M

3 EI, di complessità B

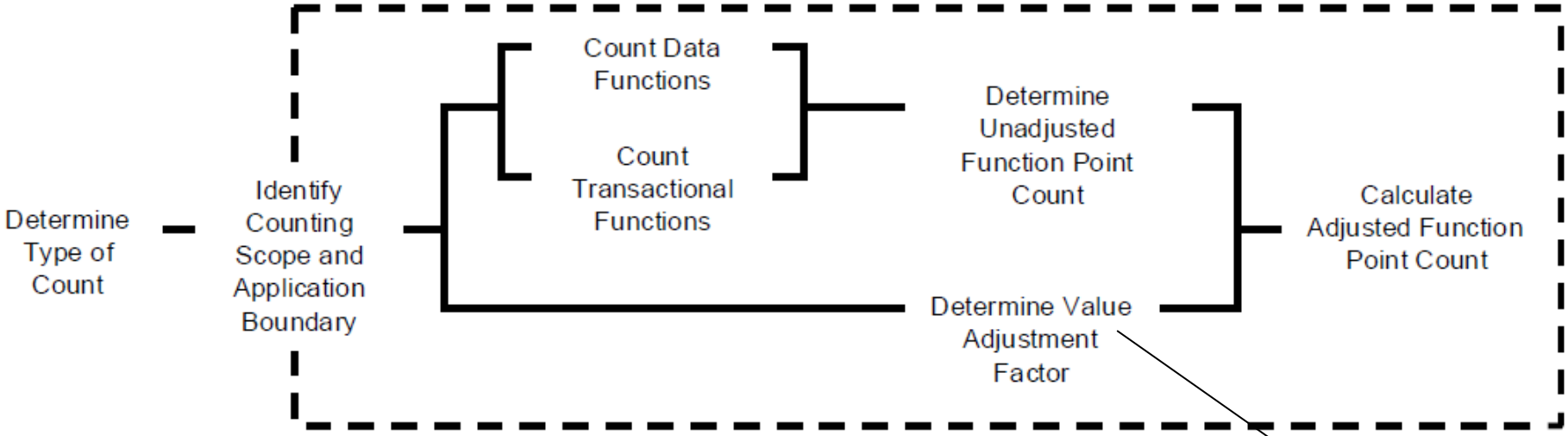
3 EO, due di complessità B e uno di complessità A

0 EQ

si ha:

UFP= 1*7 + (1*5 + 1*7) + 3*3 + (2*4 + 1*7) =
= 43

Procedura per il conteggio



Prima di arrivare a FP finali occorre calcolare il fattore di aggiustamento (VAF)

VAF si calcola in base a 14 caratteristiche di sistema, su una scala da 0 a 5

Le 14 caratteristiche

1. **Comunicazione dati.**
2. **Distribuzione dell'elaborazione.**
3. **Prestazioni.**
4. **Utilizzo estensivo della configurazione.**
5. **Frequenza delle transazioni.**
6. **Inserimento dati interattivo.**
7. **Efficienza per l'utente finale.**
8. **Aggiornamento on line.**
9. **Complessità di elaborazione.**
10. **Riusabilità.**
11. **Facilità d'installazione.**
12. **Semplicità operativa.**
13. **Molteplicità di siti.**
14. **Facilità di modifica.**

Quantificazione

Su una scala 0-5

1 **Comunicazione dati.**
2 **Distribuzione dell'elaborazione.**
3 **Prestazioni.**
4 **Utilizzo estensivo della configurazione.**
5 **Frequenza delle transazioni.**
6 **Inserimento dati interattivo.**
7 **Efficienza per l'utent**
8 **Aggiornamento on li**
9 **Complessità di elabo**
10 **Riusabilità.**
11 **Facilità d'installazion**
12 **Semplicità operativa**
13 **Molteplicità di siti.**
14 **Facilità di modifica.**

Purtroppo la quantificazione sulla scala 0-5 è soggettiva

0 Non presente, o di nessuna influenza
1 Influenza secondaria o poco significativa
2 Influenza moderata
3 Influenza media
4 Influenza significativa
5 Influenza forte generalizzata

Aggiustamento

- I function point (FP) si calcolato come:

$$\mathbf{FP = UFP * VAF}$$

- VAF: Value Adjustment Factor. Determinato in base alle 14 *caratteristiche* di sistema precedenti,

- Indicando con Fi il valore assegnato alla generica caratteristica, VAF è calcolato:
  - sommando i singoli valori assegnati alle diverse caratteristiche
  - e applicando la seguente formula:

$$\mathbf{VAF = 0{,}65 + 0{,}01} \sum_{i=1}^{14} \mathbf{F_i}$$

- VAF è compreso tra 0,65 e 1,35. Ovvero corregge UFP di un +/- 35%

Variazioni sul tema (in troppi dicono la loro!)

- Feature Points [Caper Janes]
  - introducono un ulteriore tipo di function type: l'algoritmo, e modificano il sistema di pesature
  - sono più adatti per il sistemi di tempo reale
- Mark II [Charles Symon]
  - tiene conto di fattori ambientali: esperienza, motivazione del personale, linguaggi di programmazione, strumenti di progetto
  - 20 caratteristiche (6 riguardanti l'interfaccia con altre applicazioni); diversa pesatura (0,005 anziché 0,01 nel calcolo di FP)
- Meglio fare riferimento al metodo IFPUG

E per l'OO ?

- Si capisce (anche dalla sola terminologia) che FPA è stata pensata prima della programmazione OO
- In letteratura si trovano svariate proposte per adattarla alla programmazione OO. Molte di queste **non** sono convincenti
  - Spesso non fanno riferimento al metodo di calcolo di IFPUG ma a metodi che sembrano più convenienti per la loro tesi
  - Fanno riferimento ad aspetti di dettaglio della programmazione OO perdendo il concetto di funzionalità vista dall'utente. Mettono nel mezzo il codice.
- C'è una proposta convincente:
  - Usare il metodo IFPUG
  - Basarsi su: Casi d'uso, Modello di analisi, analisi robustezza (prendendo possibilmente in considerazione le sole ***Entity***)
  - Porta a fare una valutazione in base alle **specifiche dei requisiti non** sul codice

- T. Fetche, A. Abram, T-H Nguyen “Mapping the OO-Jacobson Approach into Function Point Analysis”, Proceedings of TOOLS97, 1997, Santa Barbara, CA

- Descrive il metodo di cui al trasparente precedente

Critiche a FPA

- Metodologia pensata per sistemi informativi convenzionali, ma si riesce ad adattare alla programmazione OO (più difficile per sistemi RT)
- Eccessiva semplificazione delle tipologie dei componenti: ad un sistema con oltre 100 DET viene attribuito il doppio dei punti di un componente con un solo DET.
- Pesi dei componenti discutibili (andrebbero aggiornati per tener conto dell'evoluzione - praticamente si è fermi all'epoca dell'introduzione)
- Intransitività dell'addizione: il conteggio per n piccoli progetti è superiore a quello di un singolo progetto che li comprenda tutti (a causa dell'inclusione degli ILF e EIF.
- Rischi di doppio conteggio
- Le caratteristiche delle applicazioni vengono giudicate soggettivamente; 14 caratteristiche possono risultare poche in certe situazioni.

- IFPUG sta affrontando alcuni di questi problemi

Relazione LOC - FP

- Ci sono in giro tabelle che danno SLOC per FP, a seconda del linguaggio di programmazione.
- Jones [*] dà questa tabella:

Linguaggio	SLOC/FP
Assembler	320
C	150
FORTRAN	106
Cobol	105
Pascal	91
Basic	64
Query languages	16
Speadsheet lang.	6

Sono dati nemmeno tanto vecchi (1995)

C'è anche una tabella con fattori di aggiustamento in funzione della complessità

[*] C.Jones "Applied Softwrae Measurement, Assuring productivity and quality", McGraw Hill, NY 1991

Language SLOC / UFP (COCOMO II)

Ada	71
APL	32
Assembly	320
Assembly (Macro)	213
ANSI/Quick/Turbo Basic	64
Basic - Compiled	91
Basic - Interpreted	128
C	128
C++	29
ANSI Cobol 85	91
Fortan 77	105
Forth	64
Jovial	105
Lisp	64
Modula 2	80
Pascal	91
Prolog	64
Report Generator	80
Spreadsheet	6

La tabella a fianco viene usata in COCOMO II (attenzione: Unadjusted!)

I FP vengono impiegati per

Valutare l'*effort* (il costo), il numero dei casi di test da effettuare, il numero dei difetti potenziali e l'effort.

In genere si tratta di formule del tipo $E = c\ FP$

oppure $E = c\ FP^{e}$

$casi\_di\_test = FP^{1,2}$

$difetti\_potenziali = FP^{1,5}$

Per il calcolo dell'effort conviene costruire relazioni empiriche basate sull'esperienza e sui progetti pregressi di una data organizzazione.

Valutazione dei costi

ovvero

Valutazione dell'*Effort*

Valutazione dei costi di un progetto

- Esistono vari modelli per la valutazione/previsione dei costi dello sviluppo software.
- Hanno a comune il fatto che essi calcolano il costo in base alla misura di attributi interni ed esterni. L'attributo fondamentale è la dimensione.
- Quasi tutti i modelli esprimono il costo in funzione della dimensione (LOC, FP, ..), mentre gli altri attributi entrano spesso come fattori correttivi.
- Si tratta di relazioni empiriche, validate attraverso il confronto con la realtà industriale. Relazioni in forma tabellare o in forma analitica.

- Nel seguito illustriamo COCOMO un modello per la previsione dei costi molto noto. Nella sua versione originale, impiega la dimensione (SLOC) come parametro fondamentale. Usa una serie di coefficienti e fattori correttivi che tengono conto di altri aspetti (p.e. la complessità)

COCOMO (Construcrive Cost Model)

- E' un metodo "composto": equazioni, parametri statistici, giudizi dell'esperto
- Risale al 1981 [*]. (Oggi è in uso la versione COCOMO II [+])

- Calcola lo sforzo E (mesi/persona) in funzione della dimensione S (migliaia di istruzioni sorgente consegnate).
- Il calcolo dello sforzo viene perfezionato attraverso un insieme di parametri detti *cost driver*
- Prevede il calcolo del tempo solare coerente, necessario per coprire lo sforzo stimato

[*] B. Boehm, Software Engineering Economics, Prentice Hall, 1981

[+] **http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html**

COCOMO

$$E = aS^{b}F$$

E: Sforzo (mesi/persona)

S: Dimensione (migliaia di linee di codice consegnate , KDSI)

a,b: costanti che variano a seconda del modello e in base alla modalità di sviluppo

F: fattore correttivo.

$$T = cE^{d}$$

T: tempo solare in mesi necessario allo sviluppo

c,d: costanti che variano a seconda del modello e in base alla modalità di sviluppo

.. COCOMO

- KDSI include tutte le istruzioni sorgente consegnate all'utente, *escludendo eventuali istruzioni usate appositamente per il debugging, testing, o altro supporto.*
- *Esclude tutte le linee di commento.*
- Sforzo e tempo solare coprono il periodo intercorrente dall'inizio dell'analisi del progetto (fine raccolta dei requisiti utente) fino alla fase di consegna del sistema.
- I valori calcolati comprendono le attività di *management* e la stesura della documentazione, *ma escludono i tempi di addestramento dell'utente, i tempi di installazione ed eventuale manutenzione.*
- La stima comprende tutti i costi diretti (programmatori, librerie, strumenti di supporto). *Sono esclusi i costi indiretti (costi segreteria, costi top management).*

.. COCOMO

- Il mese/persona corrisponde a 152 ore lavorative per 12 mesi, togliendo circa 35 giorni all'anno per ferie e malattie del personale

  (Notare che 152*12 = 1824 è un numero più alto di quello standard italiano, circa 1500)
- Il modello assume che i requisiti del sistema non subiscano sostanziali variazioni durante lo sviluppo.

COCOMO, modelli

- Sono previsti 3 modelli a seconda del livello di dettaglio delle informazioni a disposizione e di quanto debba essere dettagliata la stima del progetto in esame:
  - Modello base
  - Modello intermedio
  - Modello dettagliato

- I coefficienti sono differenziati a seconda del modello e a seconda della modalità di sviluppo (*COCOMO mode*). Tre modalità:
  - Poco strutturata (***organic mode***)
  - Mediamente strutturata (***semidetached mode***)
  - Fortemente strutturata (***embedded mode***)

COCOMO, modalità

- La modalità viene identificata essenzialmente attraverso questi parametri:
  - Dimensione del progetto
  - Organizzazione operativa del gruppo di lavoro
  - Esperienza del gruppo di lavoro

- Modalità Organica:
  - progetti di dimensione medio-piccola (SLOC<5000)
  - piccolo gruppo di lavoro
  - sviluppo prevalentemente sequenziale
  - approccio artigianale
  - ambiente tecnico-operativo molto stabile
  - esperienza del personale di sviluppo di livello medio-alto e specifica

...COCOMO, modalità

- Modalità mediamente strutturata:
  - progetti di dimensione media (5.000< SLOC<50.000)
  - gruppo di lavoro medio
  - sviluppo mediamente sequenziale
  - approccio mediamente industriale
  - ambiente tecnico-operativo mediamente stabile
  - esperienza del personale di sviluppo generica e di livello medio

- Modalità fortemente strutturata:
  - progetti di dimensione medio-grande (oltre 50.000 SLOC)
  - gruppo di lavoro grande
  - sviluppo prevalentemente in parallelo
  - approccio prevalentemente industriale
  - ambiente tecnico-operativo evolutivo e complesso
  - esperienza del personale diversificata e di livello medio

Fattori correttivi

- Il problema dei fattori correttivi nei modelli come questo è quello di evitare che diventino una pletora (alcuni studi precedenti identificavano oltre cento fattori correttivi).
- COCOMO prevede 15 fattori correttivi (Fi).
  - RELY: affidabilità richiesta
  - DATA: dimensione della base dati
  - CPLX: complessità del prodotto
  - TIME: vincoli nel tempo di esecuzione
  - STOR: vincoli nella memoria centrale
  - VIRT: instabilità della macchina virtuale
  - TURN: tempo medio di risposta macchina
  - ACAP: capacità degli analisti
  - AEXP esperienza del gruppo con il tipo di applicazione
  - PCAP: capacità programmatori
  - VEXP: esperienza del gruppo di lavoro sulla macchina virtuale
  - LEXP: esperienza sul linguaggio di programmazione
  - MODP: conoscenza tecniche moderne di programmazione
  - TOOL: utilizzo di tool di sviluppo
  - SCED: vincoli di tempificazione nello sviluppo

COCOMO base

- Non tiene conto dei fattori correttivi. Da usare come modello di approccio, quando sono poco chiare le caratteristiche del processo

- Modalità poco strutturata

$$E = 2{,}4\ S^{1.05} \qquad T = 2{,}5E^{0{,}38}$$

- Modalità mediamente strutturata

$$E = 3{,}0\ S^{1.12} \qquad T = 2{,}5E^{0{,}35}$$

- Modalità fortamente strutturata

$$E = 3{,}6\ S^{1.20} \qquad T = 2{,}5E^{0{,}32}$$

COCOMO intermedio

- Modello più usato; tiene conto di tutti i fattori correttivi, senza che si debba avere il dettaglio di come questi pesino nelle varie fasi di sviluppo.

- Modalità poco strutturata

$$E = 3{,}2\ S^{1.05}\ F \qquad T = 2{,}5E^{0{,}38}$$

- Modalità mediamente strutturata

$$E = 3{,}0\ S^{1.12}\ F \qquad T = 2{,}5E^{0{,}35}$$

- Modalità fortemente strutturata

$$E = 2{,}8\ S^{1.20}\ F \qquad T = 2{,}5E^{0{,}32}$$

- La differenza con il caso precedente sta nel fattore correttivo F (oltre che nel coefficiente a)

.. COCOMO intermedio

- F viene calcolato come $F = \prod_{i=1}^{15} Fi$

- I 15 fattori correttivi Fi sono costanti per tutte le fasi di sviluppo. Essi vengono presi da questa tabella (si riportano i primi 5)

cost driver	**m.basso**	**basso**	**nominale**	**alto**	**m.alto**	**altissimo**
RELY	0,75	0,88	1,00	1,15	1,40	
DATA		0,94	1,00	1,06	1,16	
CPLX	0,70	0,85	1,00	1,15	1,30	1,65
TIME			1,00	1,11	1,30	1,66
STOR			1,00	1,06	1,21	1,56
.....	....	.....	1,00	....	.....	.....

COCOMO dettagliato

- Si differenzia dal precedente perché la tabella dei fattori di correzione viene data per ciascuna fase (analisi, progetto, codifica, testing)
- Distingue tra
  - livello di sistema/sottosistema
  - livello di modulo
- Per il livello sistema/sottosistema vengono impiegati i 15 fattori correttivi per ciascuna fase
- Per il livello di modulo vengono impiegati solo 4 fattori correttivi, distinti per fase (CPLX, PCAP, VEXP, LEXP).

- I coefficienti a, b, c, d che compaiono nelle espressioni di E e T per le tre modalità sono identici a quelli del modello intermedio.

Osservazione

- In una relazione del tipo $E = A\ S^B$ il coefficiente B indica se ci sono economie/diseconomie di scala
  - B<1 Economie di scala
  - B=1 Parità
  - B>1 Diseconomie

- Notare che in tutte le espressioni di E date in precedenza si ha B>1.
  - I modelli COCOMO non prevedono mai economie di scala.
  - Passando da modalità di sviluppo "poco strutturato" a "fortemente strutturato" aumentano le diseconomie di scala.
  - ➔ Evidentemente la dimensione e la complessità del progetto giocano a sfavore delle economie di scala.

Nella normale produzione manifatturiera l'economia di scala è un dato assodato (un barattolo di fagioli da 1Kg costa meno del doppio di due barattoli da 1/2 Kg)

Come si impiega il *COCOMO*

- Si stima sulla base di indicazioni teoriche/empiriche/statistiche la dimensione del progetto in KDSI
- Si stabilisce la modalità secondo cui si pensa di sviluppare il sistema (*organic, semidetached, embedded*)
- Si sceglie il modello più adeguato (*base, intermedio, dettagliato*) in base alle conoscenze circa il progetto
- Si calcola E (mesi/uomo) e T (tempo solare) in base alle scelte fatte.

- Nota: Boehm ha identificato i valori dei vari coefficienti in base allo studio di 63 progetti. Essi erano suddivisi per dimensione (in KDSI) nel modo seguente:

  Piccoli: fino a 2 Intermedi: 8

  Medi: 32 Grandi:128

  Molto grandi: 512 e oltre

COCOMO II

- E' un'evoluzione del modello precedente, ma tiene conto dello stato attuale delle tecnologie
- Obiettivi:
  - To develop a software cost and schedule estimation model tuned to the life cycle practices of the 1990's and 2000's.
  - To develop software cost database and tool support capabilities for continuous model improvement.
  - To provide a quantitative analytic framework, and set of tools and techniques for evaluating the effects of software technology improvements on software life cycle costs and schedules.

- Può essere scaricato da WWW.USC.EDU oppure ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/

…COCOMO II

- Comprende tre "stadi" (stages)
  - Stadio 1: stima dell'effort di prototipizzazione o di composizione di applicazioni
  - Stadio 2: stima dell'effort durante lo stadio iniziale del progetto (early stage), quando di esso si sa poco e i fattori di costo sono molto incerti
  - Stadio 3: stima ad architettura definita (post architecture stage), quando il progetto è ben definito

- Da un punto di vista concettuale non è dissimile dal modello originario. E' molto più dettagliato.
- Conviene usare un tool che lo implementi
- La differenza tra *embedded, semidetached* e *embedded* è catturata da una delle tante tabelle fornite

…COCOMO II

- Lo sforzo si calcola sempre secondo questa formula

$$PM_{nominal} = A \times (Size)^{B}$$

PM: Mesi uomo

$$B = 0.91 + 0.01 \times \sum W_i$$

5 termini nella somma. Per ciascuno di essi c'è almeno una tabella per la sua quantificazione (*)

- La costante A vale 2.45

- PM nominale viene poi aggiustata con 7 fattori nel modello Early Design model e con 17 nel modello Post-Architectural model.

(*) uno dei fattori che compaiono nella somma è PMAT *(Process maturity)* valutato secondo la scala CMM (veder più avanti)

…COCOMO II …Aggiustamento PM

- Early Design Model $$PM_{adjusted} = PM_{nominal} \times \left(\prod_{i=1}^{7} EM_i \right)$$
- Post-Architectural $$PM_{adjusted} = PM_{nominal} \times \left(\prod_{i=1}^{17} EM_i \right)$$

EM: Effort Multiplier

- In realtà PM nominale è più complesso da calcolare nel modello Post-Architectural. Si tiene conto del riuso, della percentuale di codice buttato via a causa della volatilità dei requisiti e di altro...

...COCOMO II ...il modello Post-Architecture

$$PM = \prod_{i=1}^{17} (EM_i) \cdot A \cdot \left[\left(1 + \frac{BRAK}{100}\right) \cdot Size\right]^{\left(0.91 + 0.01 \sum_{j=1}^{5} SF_j\right)} + \left(\frac{ASLOC \cdot \left(\frac{AT}{100}\right)}{ATPROD}\right)$$

where

$$Size = KNSLOC + \left[KASLOC \cdot \left(\frac{100 - AT}{100}\right) \cdot \frac{(AA + SU + 0.4 \cdot DM + 0.3 \cdot CM + 0.3 \cdot IM)}{100}\right]$$

$$B = 0.91 + 0.01 \sum_{j=1}^{5} SF$$

$$TDEV = \left[3.0 \times (\overline{PM})^{(0.33 + 0.2 \times (B - 1.01))}\right] \times \frac{SCED\%}{100}$$

…COCOMO II …Significato di alcuni parametri

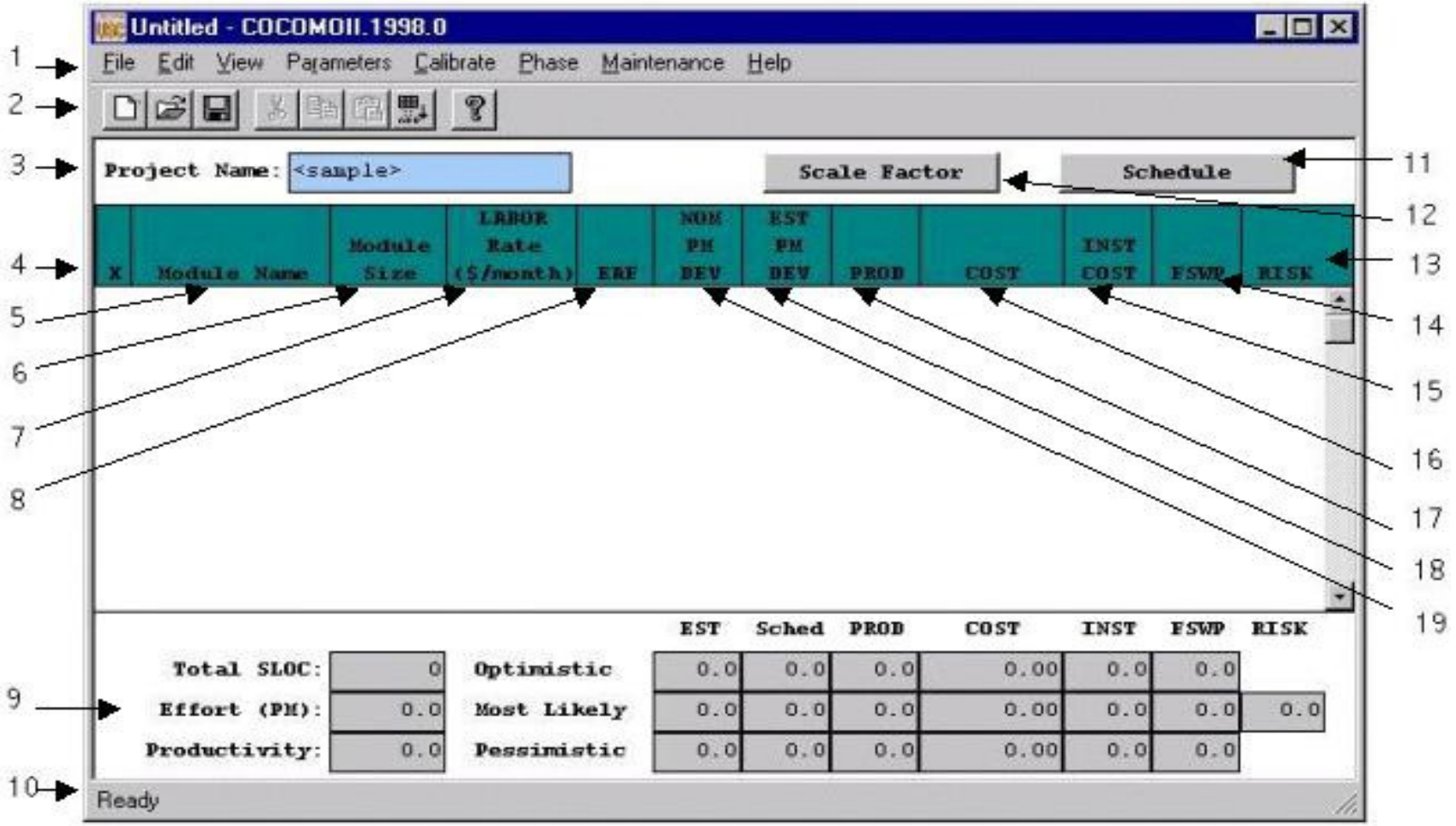
ADAPT	Percentage of components adapted (represents the effort required in understanding software)
AT	Percentage of components that are automatically translated
ATPROD	Automatic translation productivity
BRAK	Breakage: Percentage of code thrown away due to requirements volatility
CM	Percentage of code modified
EM	Effort Multipliers: RELY, DATA, CPLX, RUSE, DOCU, TIME, STOR, PVOL, ACAP, PCAP, PCON, AEXP, PEXP, LTEX, TOOL, SITE
IM	Percentage of integration and test modified
KASLOC	Size of the adapted component expressed in thousands of adapted source lines of code
KNSLOC	Size of component expressed in thousands of new source lines of code
SF	Scale Factors: PREC, FLEX, RESL, TEAM, PMAT

…COCOMO II …. Valutazione della dimensione

- Tramite SLOC e tramite FP
- SLOC: nel tentativo di dare una quantificazione che sia consistente per differenti linguaggi si misura le "linee logiche", secondo uno schema definito dal Software Engineering Institute (SEI) [*].
- FP: Calcolati secondo lo schema IFPUG. I FP vengono quindi trasformati in (migliaia di) linee di codice in base a tabelle come quelle mostrate in precedenza che esprimono la relazione tra SLOC e FP
- Ovviamente i FP sono molto indicati per le fasi iniziali del progetto

[*] Park R. (1992), "Software Size Measurement: A Framework for Counting Source Statements," CMU/SEI-92-TR-20, Software Engineering Institute, Pittsburgh, PA.

Come si presenta il tool



La complessità (attributo interno)

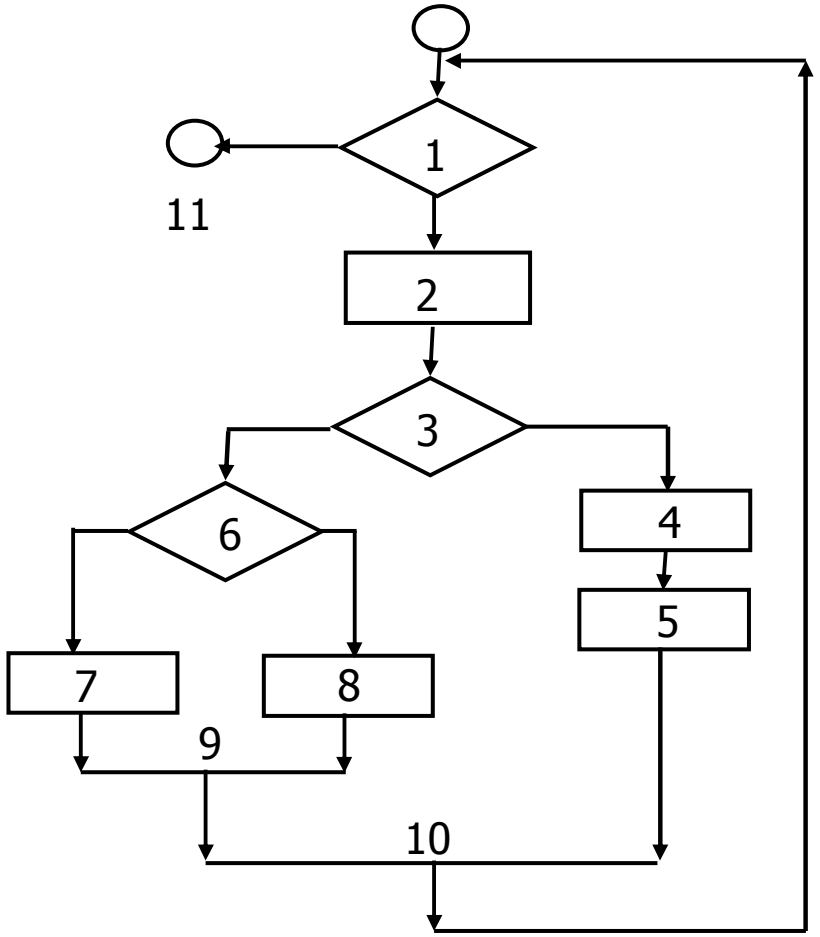
- La complessità è uno degli attributi interni che da più tempo vengono misurati
- La maniera usuale è attraverso il numero ciclomatico (o complessità ciclomatica) [McCabe]
- Per un programma con un grafo di flusso (*flowgraph*) G il numero ciclomatico è calcolato come

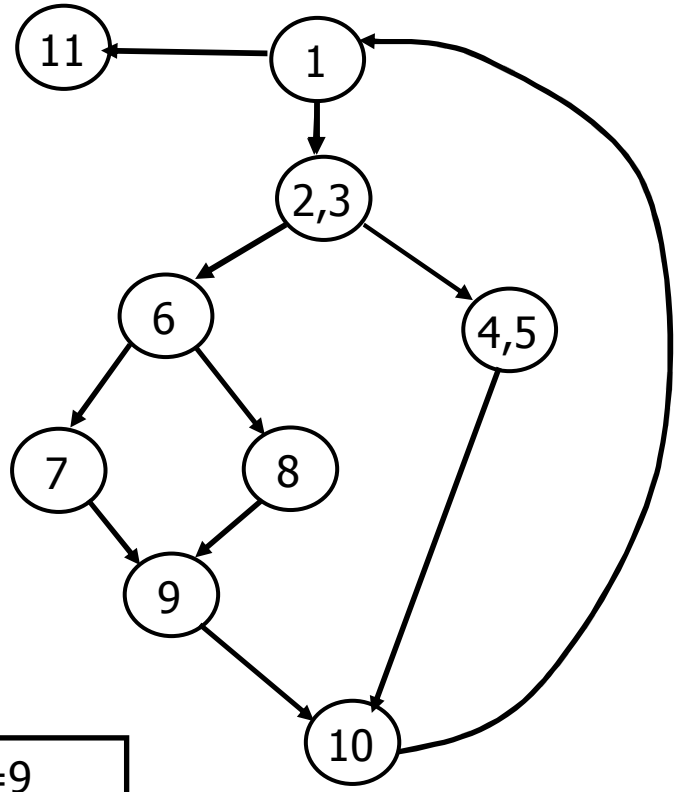
$$V(G) = a - n + 2$$

dove a: numero di archi, n: numero di nodi

- V(G) è il numero di percorsi indipendenti in G (G grafo connesso)

Grafo (di flusso) dal diagramma di flusso





a=11, n=9
V(G)= 11-9+2 = 4

Metodi di calcolo numero ciclomatico

- V(G) = n° decisioni + 1
  - if (exp) … else … 1 decisione (2 rami)
  - switch (exp) { 2 decisioni (3 rami)

```
case 1: ..
case 2:..
default: ..
}
```

- V(G) = n° regioni chiuse + 1

- Se il grafo non è connesso
  - V(G)= a - n + 2p dove p è il numero di sottografi connessi in G

Complessità ciclomatica

- Il numero ciclomatico [*] è ritenuto un indicatore di "qualità".
- *Le leggende metropolitane dicono che un modulo non dovrebbe mai presentare una complessità ciclomatica superiore a 10.*

- Uno studio Hewlett-Packard su un sistema di 850 000 linee di FORTRAN (1994) ha mostrato una relazione tra il numero di modifiche apportate ai moduli e il loro numero ciclomatico. Dall'esame dei moduli con più di 3 aggiornamenti, o studio concludeva indicando 15 come il massimo numero ciclomatico consentito
- Nella produzione del software per il tunnel sotto la Manica un modulo con un numero ciclomatico superiore a 20 veniva rifiutato (come pure se aveva più di 50 statement)

[*] McCabe T.J., "A complexity measure", IEEE Trans on Softw Eng., Dic. 1976

Complessità - struttura

- Rilevante nel caso del testing strutturale (white-box testing)
- Criteri di copertura per decidere i casi di test.

- *Statement coverage*: ogni statement eseguito almeno una volta
- *Edge coverage*: ogni ramo del grafo deve essere percoroso almeno una volta (la complessità ciclomatica rappresenta il numero minimo di percorosi indipendenti nel programma ovvero il numero minimo di test che consente di percorrere tutti gli archi)
- *Condition coverage*: ogni ramo deve essere percorso almeno una volta e tutti i possibili valori delle sottoespressioni costituenti le condizioni composte vengono valutati almeno una volta (if (c1&c2)..)

Struttura - Modularità

- Una misura intuitiva: la dimensione media del modulo
- Altri aspetti: profondità del grafo (della struttura del programma), rapporto n° rami/n° nodi.
- Accoppiamento
- Coesione

- Metriche OO:
  - WMC: Weighted Methods per class. Se la classe C ha i metodi M1, M2, .. Mn aventi complessità c1, c2 , ..cn, allora

$$WMC = \sum ci$$

  - Più decine di altre metriche (è in corso molta attività di ricerca), anche se spesso si osserva un eccessivo parcellizzazione, che rendere difficile collegare le molteplici metriche in un quadro sintetico e comprensivo.

QUALITA' DEL SOFTWARE

Misura di attributi esterni

- Obiettivo: migliorare la qualità

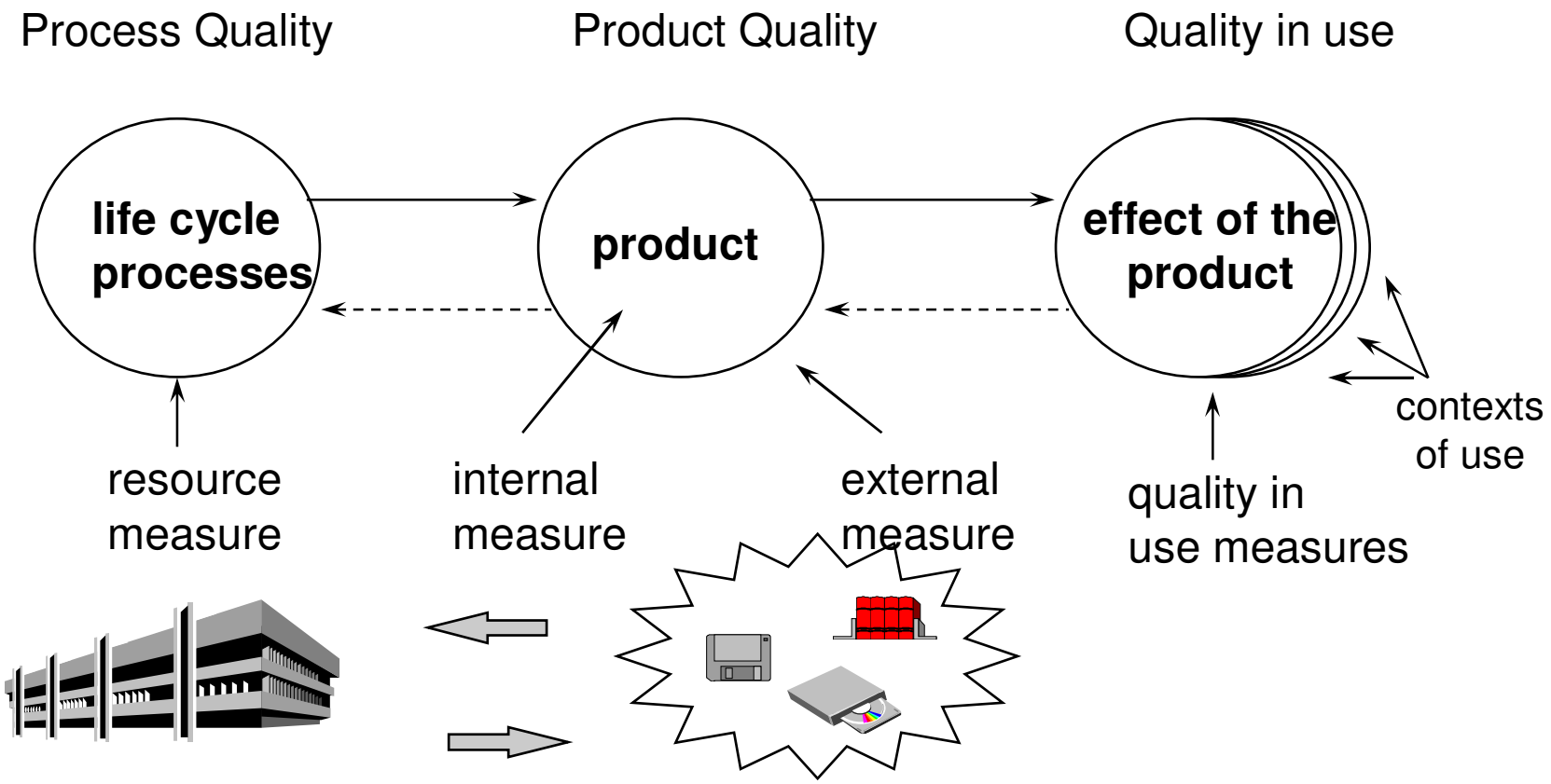
Qualità: Adeguatezza all'uso

ISO 8402 (glossario):

"The totality of the features and characteristics of a product or service that bear on its ability to satify stated or implied needs"

- Occorre identificare degli attributi del prodotto software di interesse per l'utilizzatore.
- E' ovvio che se non si trova una standardizzazione si possono costruire infiniti modelli di qualità (la scelta degli attributi)
- La standardizzazione è venuta nel 1992 con le ISO 9126 (adottano la definizione ISO 8402)

La qualità nel ciclo di vita

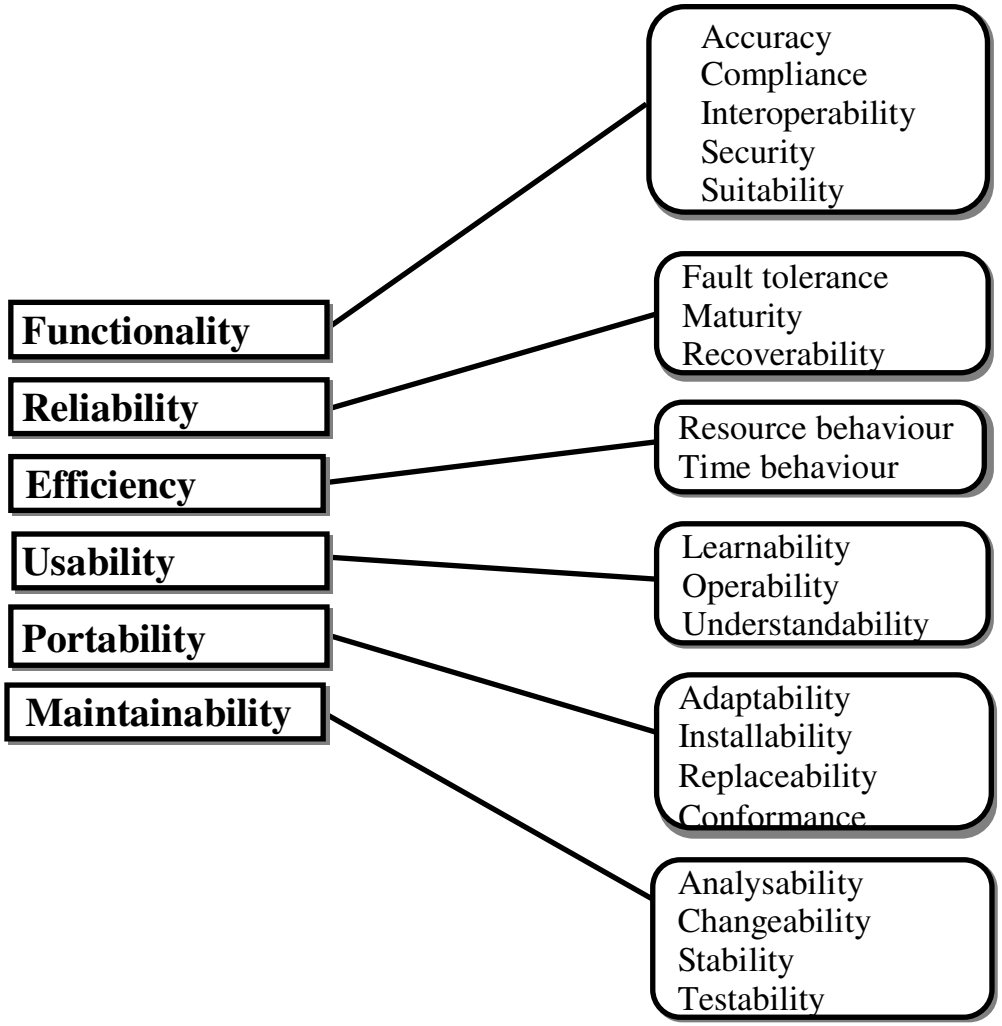


Qualità di prodotto

- ISO 9126 "Information Technology - Software Product Evaluation - Quality characteristics and guidelines for their use"
- Definisce 7 caratteristiche di qualità e 21 subcaratteristiche
- E' l'attuale standard di riferimento per la valutazione dei prodotti software.
- Comprende:
  - un modello per la definizione delle caratteristiche di qualita' del software;
  - il modello e' proposto come base per successivi raffinamenti per la descrizione della qualita' del software;
  - una linea guida per la valutazione delle caratteristiche di qualita' di un prodotto software.

Caratteristiche e sottocaratteristiche

- ISO 9129 (1991)



Caratteristiche 9126

- Functionality - Funzionalità
  - Insieme di attributi che sono in relazione con l'esistenza di un insieme di funzioni e di loro specifiche proprietà. Le funzioni sono quelle che soddisfano esigenze espresse (requisiti) o implicite.
- Reliability - Affidabilità
  - Insieme di attributi che sono in relazione con la capacità del software di mantenere il suo livello di prestazioni in determinate condizioni e per un determinato periodo di tempo.
- Usability - Usabilità
  - Insieme di attributi che sono in relazione con lo sforzo richiesto per usare (il prodotto software) da parte di un determinato insieme di utenti.
- Efficiency - Efficienza
  - Insieme di attributi che sono in relazione con il livello di prestazioni del software e la quantità di risorse usate, in determinate condizioni.
- Maintainability - Manutenibilità
  - Insieme di attributi che sono in relazione con lo sforzo richiesto per apportare determinate modifiche.
- Portability - Portabilità
  - Insieme di attributi che sono in relazione con la possibilità di un software di essere trasferito da un ambiente ad un altro.

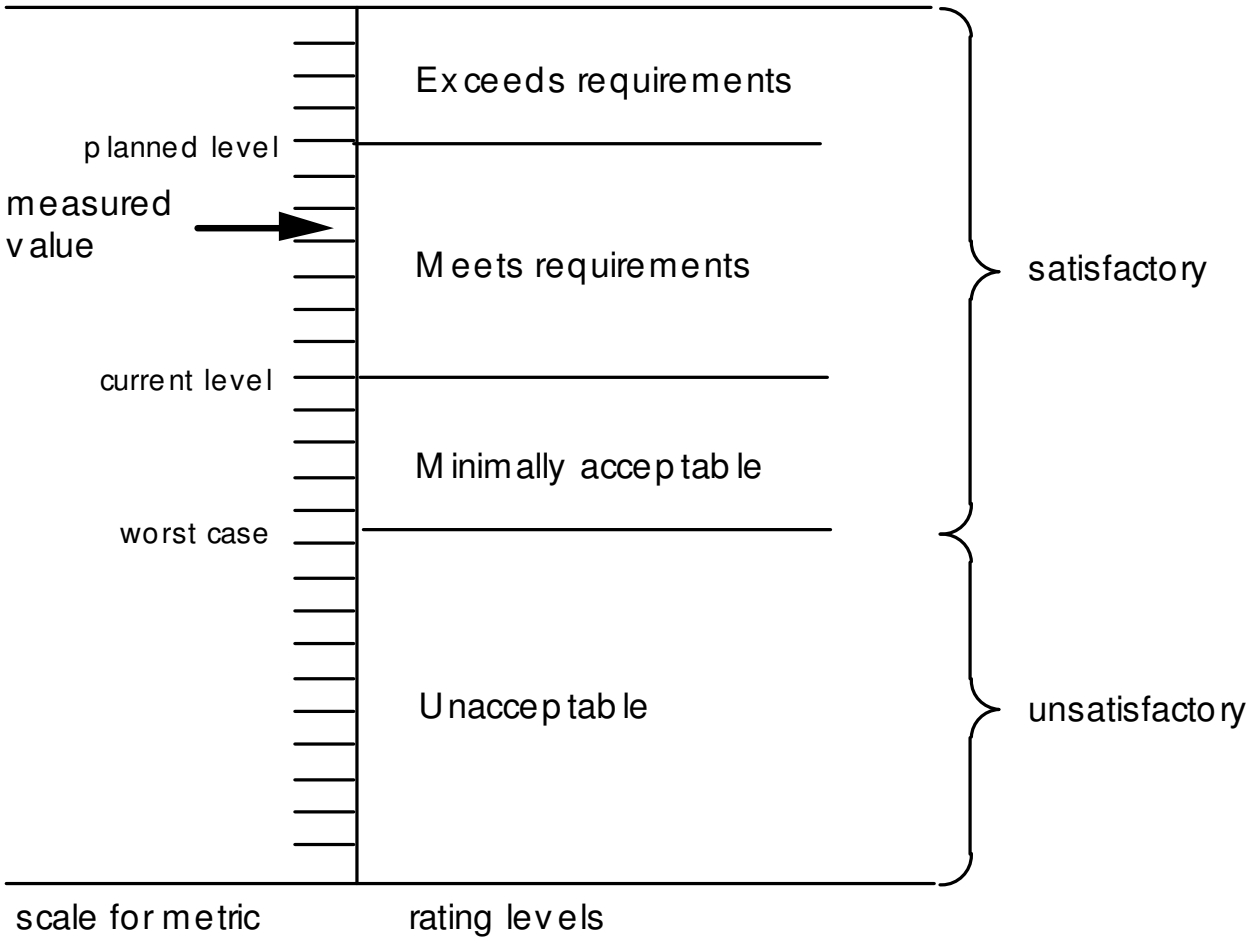
Funzionalità: sottocaratteristiche

- Aderenza - Compliance
  - Attributo del software che rende il software aderente a norme, convenzioni, leggi o prescrizioni similari.
- Accuratezza - Accuracy
  - Attributo del software che riguarda la sua capacità di fornire risultati (o effetti) giusti o accettati.
- Adeguatezza - Suitability
  - Attributo del software che riguarda la presenza e l'appropiatezza di un set di funzioni.
- Interoperabilità - Interoperability
  - Attributo del software che riguarda la sua capacità di interagire con altri sistemi.
- Sicurezza - Security
  - Attributo del software che riguarda la sua capacità di prevenire accessi non autorizzati (accidentali o intenzionali) ai dati e ai programmi.

Usabilità: sottocaratteristiche

- Comprensibilità - Understandability
  - Attributo del software che riguarda lo sforzo necessario per comprendere la logica e le modalità d'uso del prodotto.
- Apprendibilità - Learnability
  - Attributo del software che riguarda lo sforzo per imparare ad utilizzare l'applicazione.
- Operabilità - Operability
  - Attributo del software che riguarda lo sforzo necessario ad utilizzare il prodotto e controllarne il funzionamento.

Definire un criterio di rating



Come si fa a misurare

- Gli attributi esterni dipendono sicuramente da quelli interni, ma la relazione tra di essi è praticamente non esprimibile
- Occorre identificare gli attributi esterni e trovare il modo di quantificarli; in certi casi si possono dare delle definizioni (abbastanza) rigorose. Per esempio

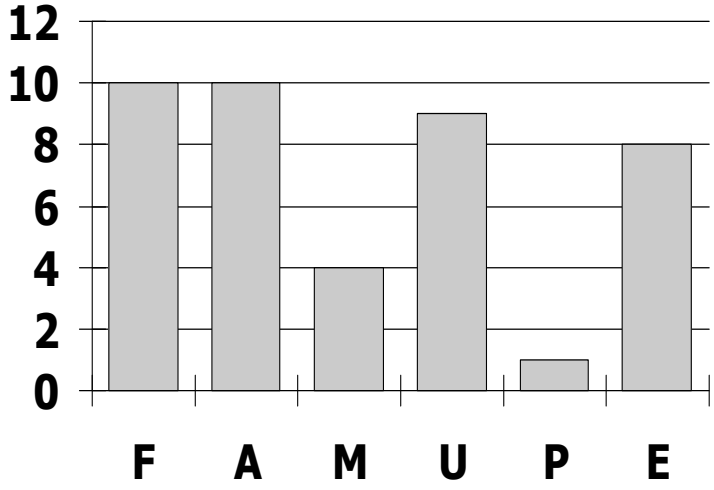
  $$\text{Portabilità} = 1 - CD/CS$$

  dove CS è il costo del software originale e CD quello necessario a partalo su un differente ambiente

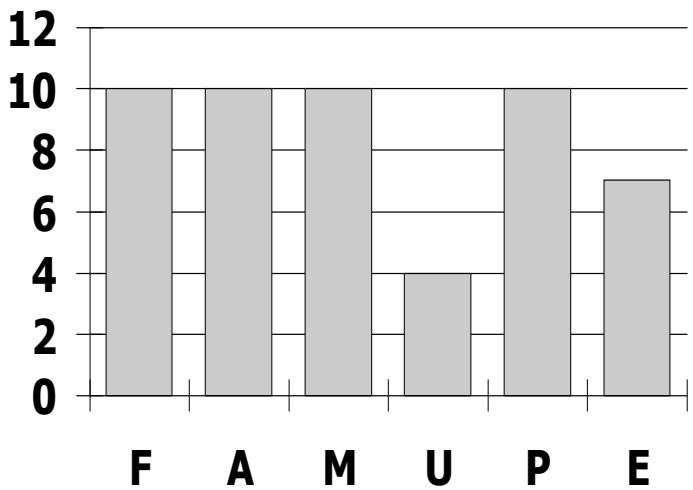
- Di norma occorre definire delle scale ad hoc, e assegnare dei valori su di esse (per esempio da 1 a 5) in base alle risposte a questionari
- L'obiettivo è definire un profilo di qualità atteso (PQA) da confrontare con un profilo di qualità misurato (PQM)

Profili di qualità

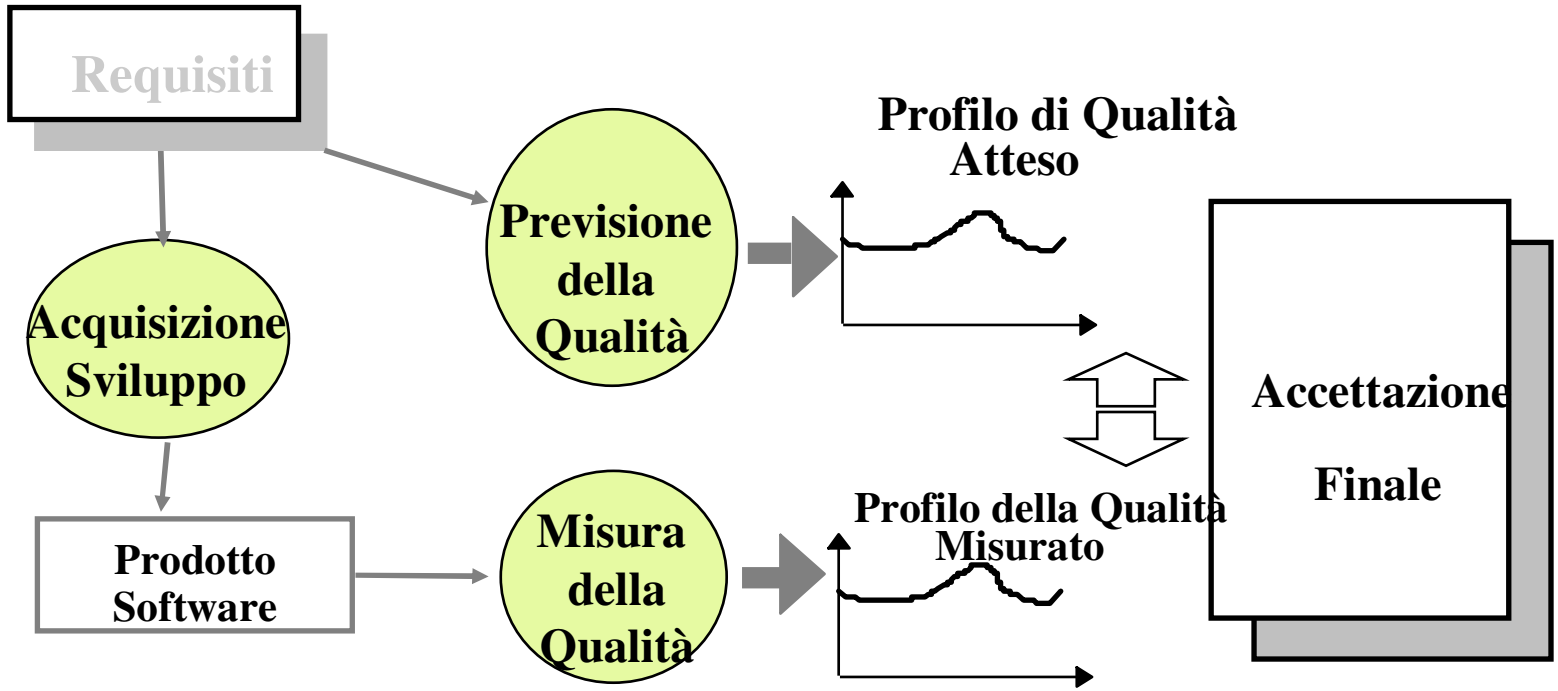
Atteso



Misurato



Accettazione del prodotto



Esempio di questionario (PQM)

		Soddisfazione			
N.	Domanda	Piena	G	M	N/A
_ITA'					
	Adeguatezza				
1	Esistono manuali di specifica e documentazione delle funzionalità ?				
2	Il livello di dettaglio dei documenti di specifica è congruente con la complessità delle funzioni ?				
3	I manuali utente riflettono le funzioni offerte ?				

	Accuratezza				
i	L'accuratezza del calcolo corrisponde a quanto specificato nei documenti di specifica ?				
i+1	L'utente incontra casi in cui il risultato di una operazione è diverso da quanto atteso ?				
i.2	Le tecniche numeriche adottate garantiscono la precisione richiesta ?				

	Interoperabilità				
k	Sono definite tutte le applicazioni con cui il sottosistema interagisce ?				

Standard di processo

- Il processo software è l'altra faccia della qualità del software
- La qualità è il risultato del processo di sviluppo (è l'assunzione di base dell'ingegneria del software. L'assunto è che un buon processo darà luogo a un buon prodotto.
- I processi di produzione sono ben stabiliti nel campo dell'ingegneria. La produzione di beni richiese sistemi di qualità (specialmente se si vuole competere sul mercato):
  - Organizzazione
  - Responsabilità
  - Procedure
  - Risorse

  } Sintetizzate nel manuale di qualità
- "Software engineering standards are heavy on process and light on product, while other engineering standards are the reverse"

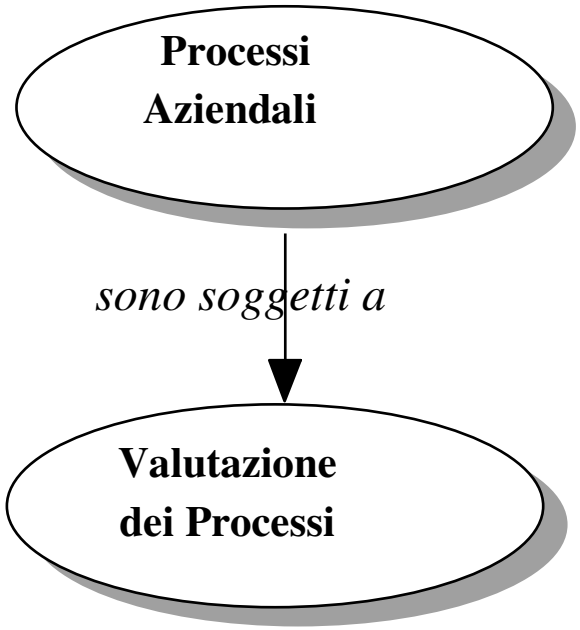
  [**Pfleeger, Fenton & Page, "Evaluation Software Eng. Standards", IEEE Computer , Sept. 1994]**

ISO 9000

- Standard di processo (non solo software). Si applicano in particolare al caso in cui un produttore deve mostrare la bontà del processo di produzione.
- ISO 9001 è lo standard generale per i processi (di qualunque natura). Esso richiede che l'organizzazione abbia un sistema di qualità documentato, un manuale di qualità che descriva le procedure aziendali.
- ISO 9003 è un ulteriore standard che fornisce le interpretazioni della 9001 nel contesto della produzione del software.
- Non è un modello di processo, ma di *Quality Assurance*
- Come è noto oggi è pratica comune leggere su etichette commerciali che "l'azienda è certificata ISO 9001". Va interpretato nel senso che l'azienda ha superato le verifica di certificazione rispetto alla standard
- NOTA: il certificatore non entra nel merito della bontà del prodotto. Si limita a verificare che il sistema di qualità aziendale sia formalmente coerente rispetto a quanto previsto dallo standard.

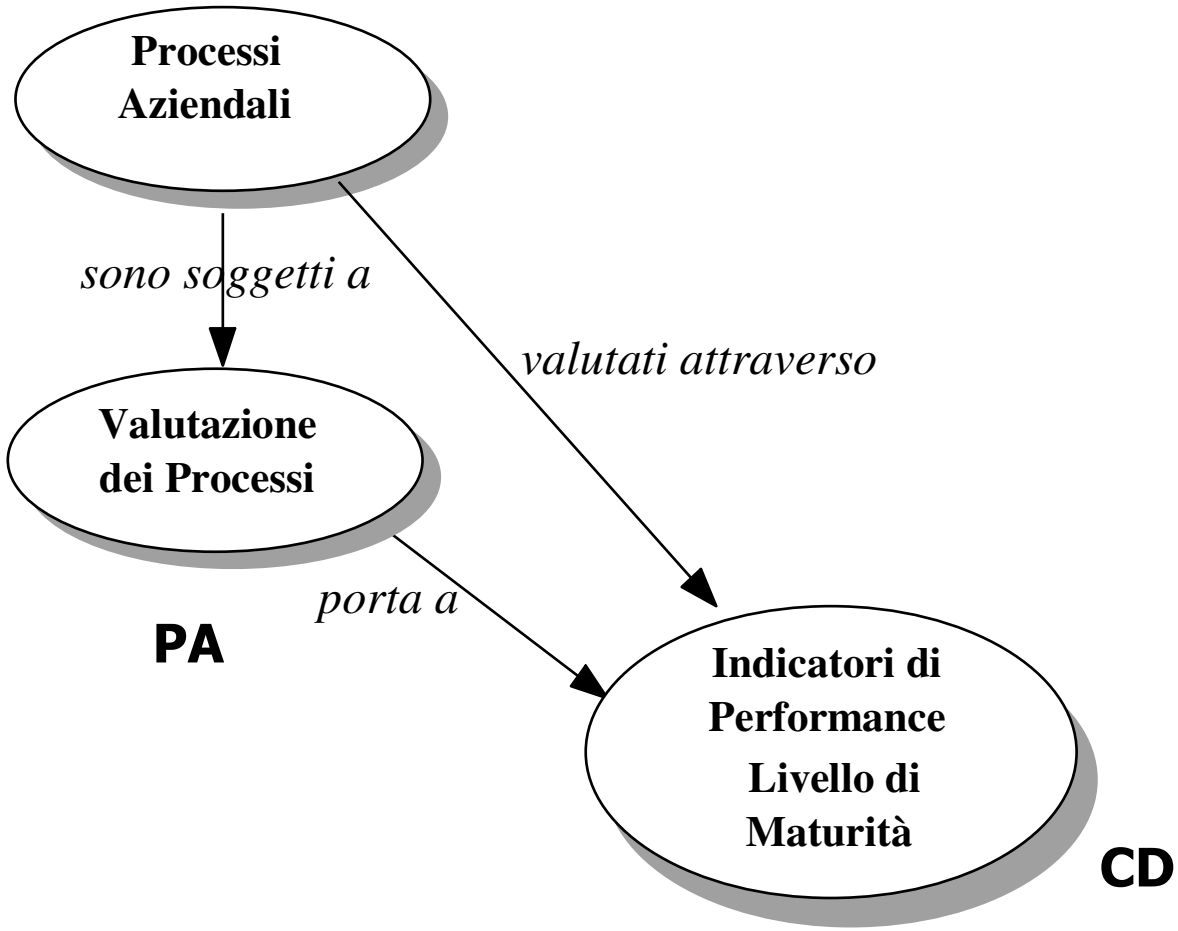
Assessment di processo (PA)

- Metodi e metriche per analizzare il processo di sviluppo ed evidenziare aree di miglioramento

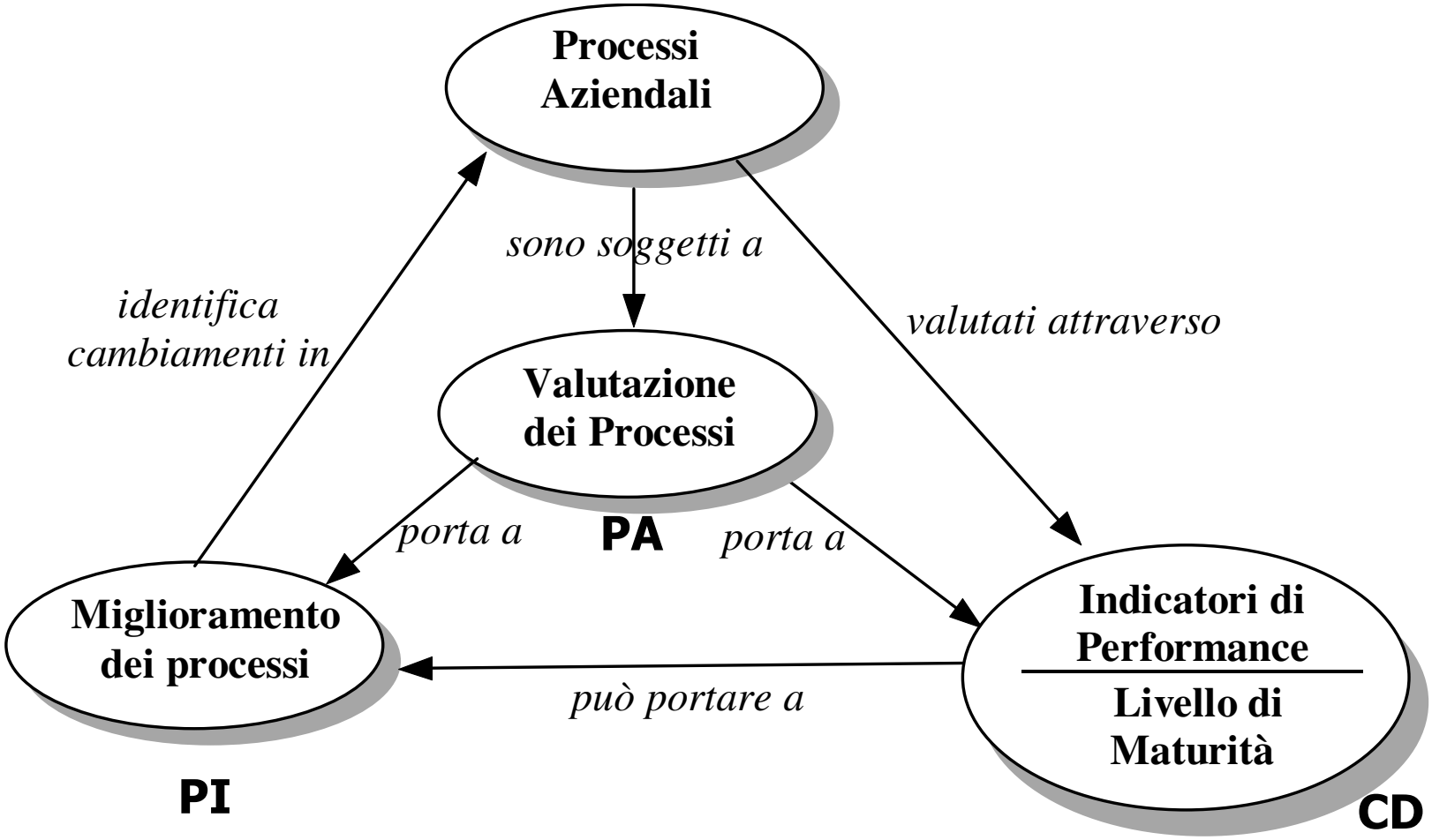


PA

.. Capability Detemination



…Process Improvement

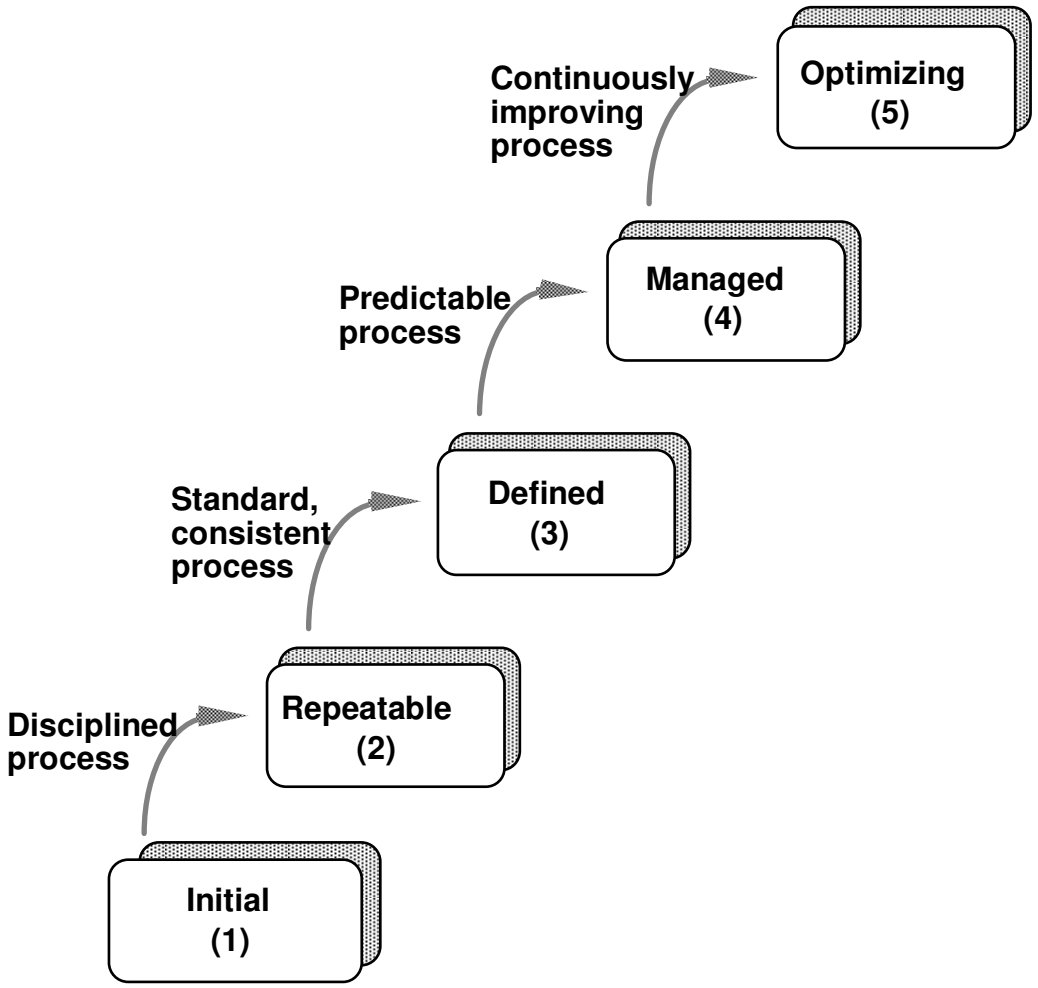


IL CMM (Capability Maturity Model)

- Un modello sviluppato dal SEI (Software Engineering Institute) per assistere il DOD (Dep.t of Defense) nel valutare la qualità dei fornitori di software
- La maturità del processo software è valutata su una scala da 1 a 5
- E' predisposta una lista di domande.
- Per poter essere valutato ad un dato livello occorre che tutte le risposte alle domande della relativa lista siano affermative (se una sola risposta è negativa il processo è automaticamente considerato di livello inferiore.
- La lista prevede 12 domande per l'assessment del livello 2. (una di queste è: "E' usata una procedura formale per stimare la dimensione del software"

- Presuppone un processo evolutivo attraverso cicli di SPA e SPI

IL CMM (Capability Maturity Model)

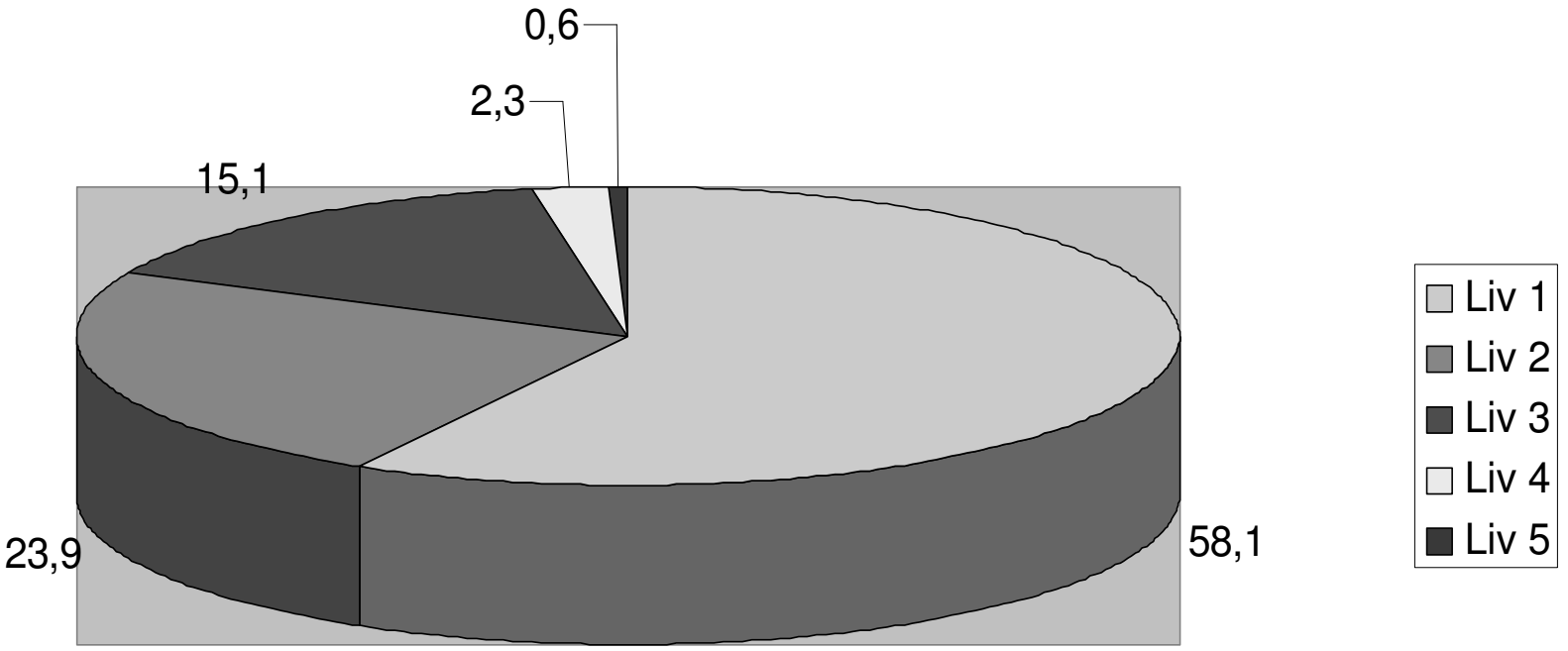


- I 5 livelli di CMM

Livelli CMM

- 1. *Iniziale.* Processo di sviluppo ad hoc o caotico. Pochi processi definiti, risultato dello sviluppo dipendente dall'impegno individuale e non del team.
- 2. *Ripetibile.* Sono in uso processi minimi di gestione. L'attività è disciplinata e viene tenuta traccia di costi, tempi e funzionalità.
- 3. *Definito.* Processi documentati e standardizzati. Integrazione di attività di manageriali e tecniche. Capacità di previsione dei costi.
- 4. *Gestito*. Si raccolgono misure dettagliate del processo e della qualità del prodotto. Processi e prodotti controllati e gestiti attraverso tecniche quantitative.
- 5. *Ottimizzato*. E' attivato un miglioramento continuo sulla base dei dati quantitativi raccolti e monitorando tecniche e strumenti in modo da poter quantificare il loro impatto sul processo.

Assessment al maggio 1998



Altri modelli per SPI

- CMM ha favorito la proliferazione di altri modelli (che ad esso sono fortemente ispirati):
  - Trillium, prodotto dalle compagnie telefoniche canadesi)
  - Bootstrap, esetensione di CMM sviluppato nel contesto di un progetto ESPRIT
  - AMI (Application of Metrics in Industry) sempre nel contesto di ESPRIT
  - SPICE (Software Process Improvement and Capability Determination) è nato nel contesto ISO/IEC
  - Paradigma SEL (Sofware Engineering Laboratory della NASA)

Altri riferimenti

- N.E. Fenton e S.L. Pfleeger "Software Metrics, a rigorous approach, second edition", PWS publishing company, 1997
- L. Buglione, "Misurare il software, seconda edizione" Franco Angeli, 2003
- Putnam, L.H., Myers, W., "Measures for excellence, reliable software on time, within budget" Prentice Hall, 1992
- Park, R. (1992), "Software Size Measurement: A Framework for Counting Source Statements," CMU/SEI-92-TR-20, Software Engineering Institute, Pittsburgh, PA.
- Goethert, W., E. Bailey, M. Busby (1992), "Software Effort and Schedule Measurement: A Framework for Counting Staff Hours and Reporting Schedule Information." CMU/SEI-92-TR-21, Software Engineering Institute, Pittsburgh, PA.
- Amadeus (1994), Amadeus Measurement System User's Guide, Version 2.3a, Amadeus Software Research, Inc., Irvine, California, July 1994

- http://sunset.usc.edu/research/COCOMOII/

La Software Science (Halstead)

- Metrica di carattere "teorico". Ha avuto una certa notorietà (anni settanta). Ora è sostanzialmente abbandonata
- Si basa sul conteggio dei "token" che compaiono in un programma:

  μ1 = numero di operatori distinti

  μ2 = numero di operandi distinti

  N1 = numero totale di operatori

  N2 = numero totale di operandi

  Per esempio lo statement F = A(K) ha un operatore (=) e due operandi (F e A(K))
- Vengono definiti: volume, livello di difficoltà e altri parametri.
- Viene data una stime della lunghezza del programma, dello sforzo, etc.
- Ha degli aspetti di inconsistenza

Attributi (e relative metriche sw)

- Esterni (visibili all'utilizzatore)
  - Prestazioni (tempo risposta)
  - Costo (€)
  - Affidabilità (MTBF)
  - Usabilità (???)
- Interni (visibili al progettista/sviluppatore)
  - Dimensione (SLOC)
  - Effort di produzione (MU) (interno?)
  - Funzionalità (FP)
  - Complessità (Num. ciclomatico)
  - Modularità (???)