# Project Management dei Progetti Software


*Corso di Ingegneria del Software*
*CdL Informatica*
*Università di Bologna*

# Obiettivi della lezione

- Cos'è il Project Management
- Tecniche per la stima dei costi sw
- Misurare le dimensioni di un progetto sw
  - LOC o FP
- Stimare i costi
  - COCOMO

# Discussione

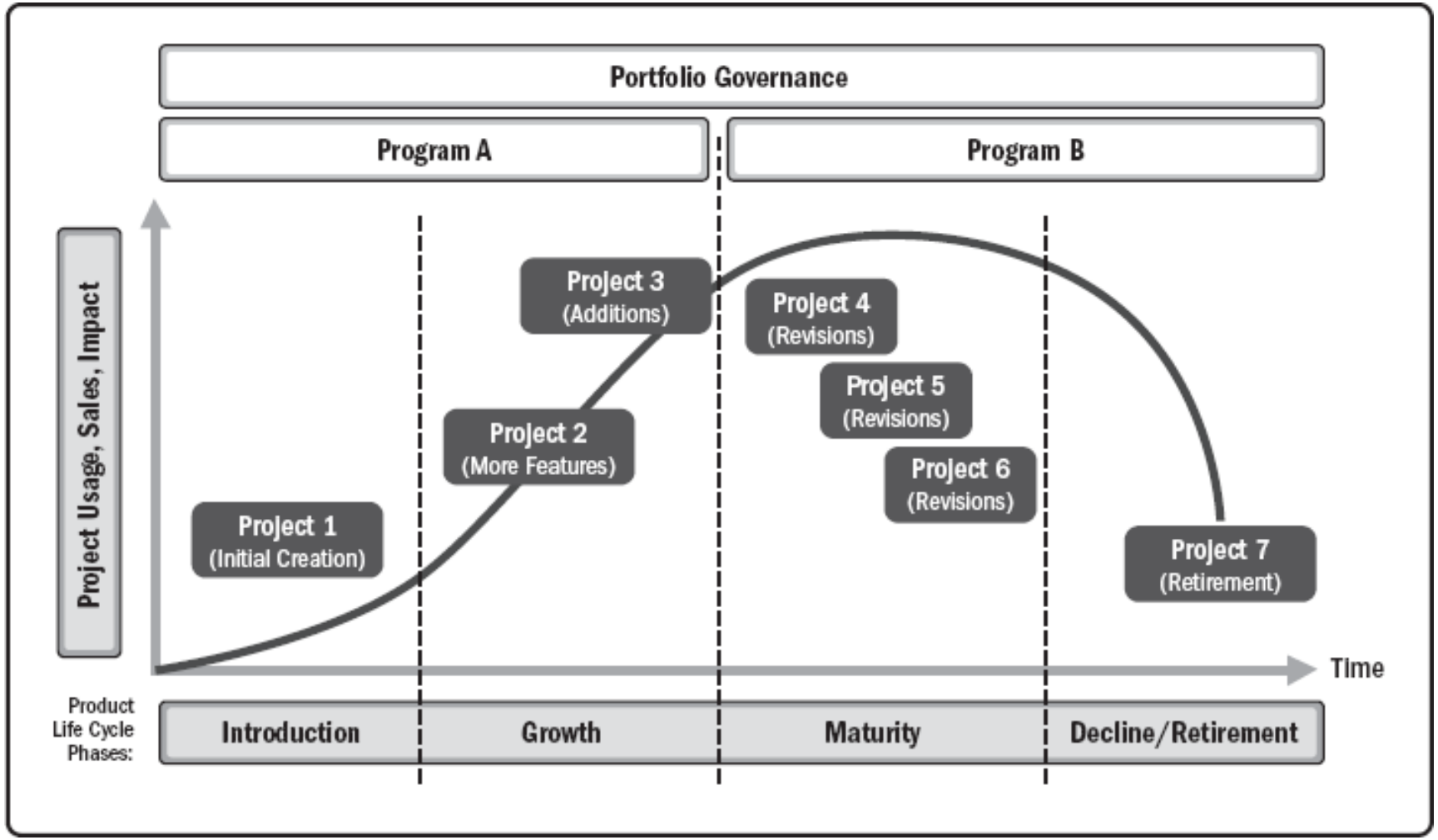
- Cos’è un progetto? come si gestisce?

# Alcune definizioni (fonte: PMI)

- **Product Management**: funzione strategica di pianificazione, produzione e marketing di un prodotto, che segue un ciclo di vita
- **Progetto**: impresa temporanea intrapresa per raggiungere uno scopo. Es: creare, vendere, e aggiornare un prodotto
- **Project Management**: Consegnare il prodotto rispettando tempi, costi e qualità stabiliti. Si concentra su iniziative specifiche e temporanee (progetti), mirate al raggiungimento di obiettivi definiti, come lo sviluppo di una nuova feature o l'aggiornamento di una versione del prodotto

# Esempio: ciclo di vita di prodotto articolato su più progetti

(fonte: Guida PMI 2021)

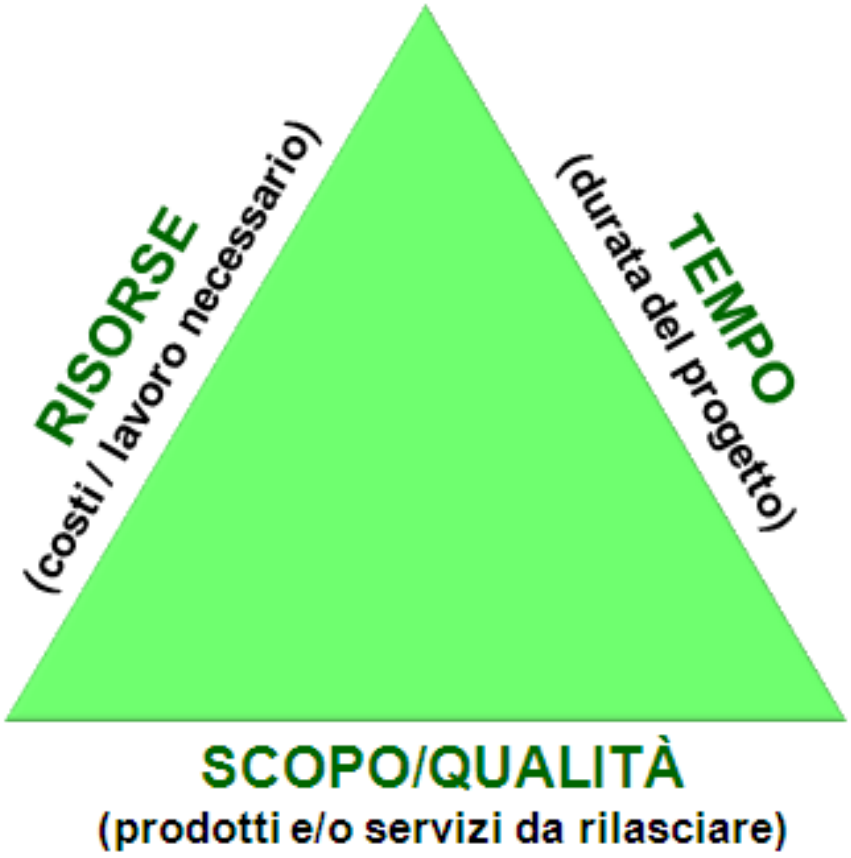


# Variabili che governano un progetto

- Scopo: requisiti di utente, specifiche di qualità, elementi da consegnare
- Costo: budget e risorse necessarie
- Tempo: inizio, fine e calendario delle attività

Un progetto si considera concluso con successo se termina in tempo, rispettando il budget, e consegna ciò che l'utente si aspetta con la qualità migliore possibile

# Il triangolo del project management




Per ogni progetto i tre vincoli: *scopo*, *risorse*, e *tempo* sono correlati

- Incrementare lo scopo e fissarlo (funzioni+qualità) significa aumentare i tempi e i costi del progetto;
- Rdurre i tempi richiede costi più alti (più risorse) o uno scopo più ristretto;
- Un budget risicato (meno risorse) può implicare tempi più lunghi o una riduzione dello scopo.

Lo scopo di solito è più importante degli altri due vincoli: nei modelli pianificati è la variabile indipendente

Nei modelli agili invece le variabili indipendenti sono la durata e i costi, lo scopo e la qualità si contrattano

# Il Project Manager

- Il Project Manager (PM, o responsabile di progetto) controlla la *pianificazione* ed il *budget* di progetto
- Durante tutto il ciclo di vita il PM deve controllare **costi** e **risorse** di un progetto:
  - **Inizialmente** per verificare la fattibilità del progetto
  - **Durante il progetto** per controllare che le risorse non vengano sprecate ed i budget vengano rispettati
  - **Alla fine** per confrontare preventivo e consuntivo

# I team agili non hanno un Project Manager

- Perché i team agili NON hanno un Project Manager?
- Nella filosofia Agile, ruolo e responsabilità tradizionalmente attribuiti al "**project manager**" sono distribuiti su più ruoli
- Esempio: nel modello agile Scrum i ruoli che hanno le responsabilità di PM sono **Product Owner** e **Team di sviluppo** (lo Scrum Master è un facilitatore, owner di processo ma non responsabile della gestione)

# Project Management Body of Knowledge

- Il manuale PMBOK è una pubblicazione del PMI che descrive le nozioni e i metodi necessari ai PM “nella maggior parte dei progetti”
- Viene aggiornato periodicamente (edizione più recente: 2021)
- Struttura il Project Management definendo nove aree di competenza
- Disponibile online: IEEE 1490-2011 “Guide adoption of PMI standard to the project management BoK”.Nota: non è più uno standard attivo

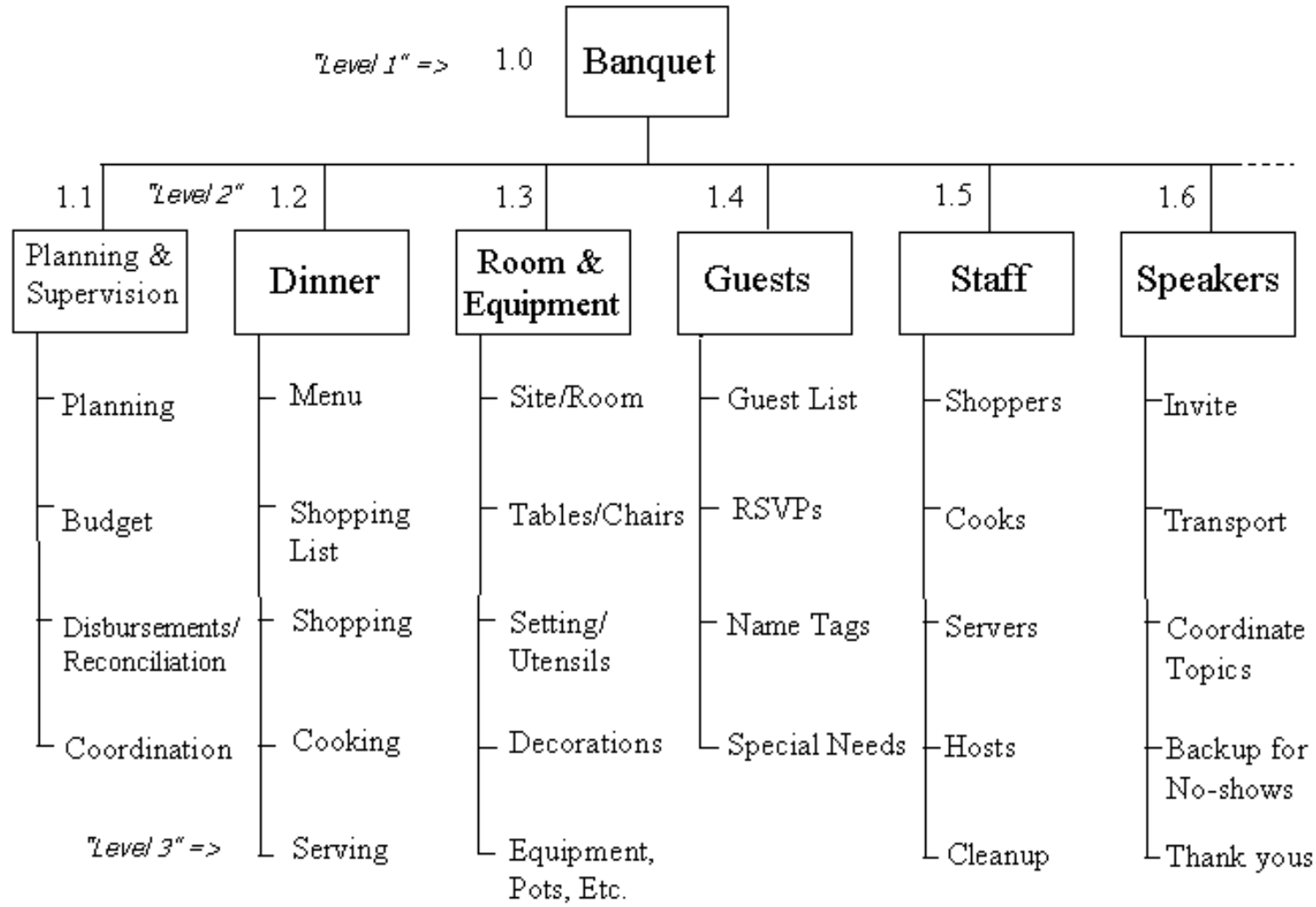
# La gestione del tempo di progetto

- Definizione delle attività
- Sequenziamento delle attività
- Stima delle risorse per le attività
- Stima della durata delle attività
- Schedulazione delle attività
- Controllo delle attività

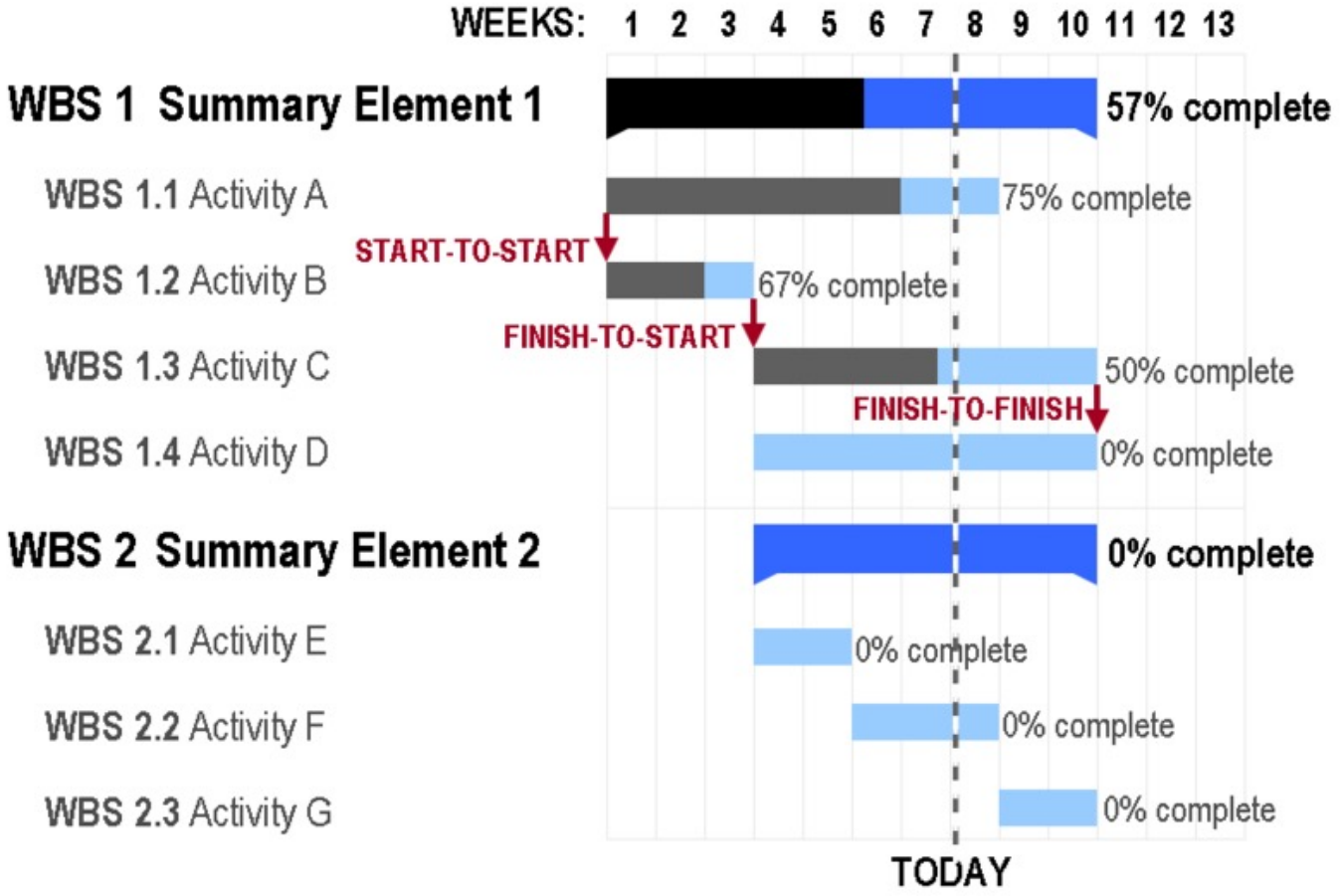
# Terminologia

- Work breakdown: articolazione (cioè elenco e relazioni) delle attività di progetto
- Milestone: evento significativo nella vita di un progetto
- Percorso critico: sequenza delle attività la cui durata è incomprimibile

# Work Breakdown Structure
## (WBS)

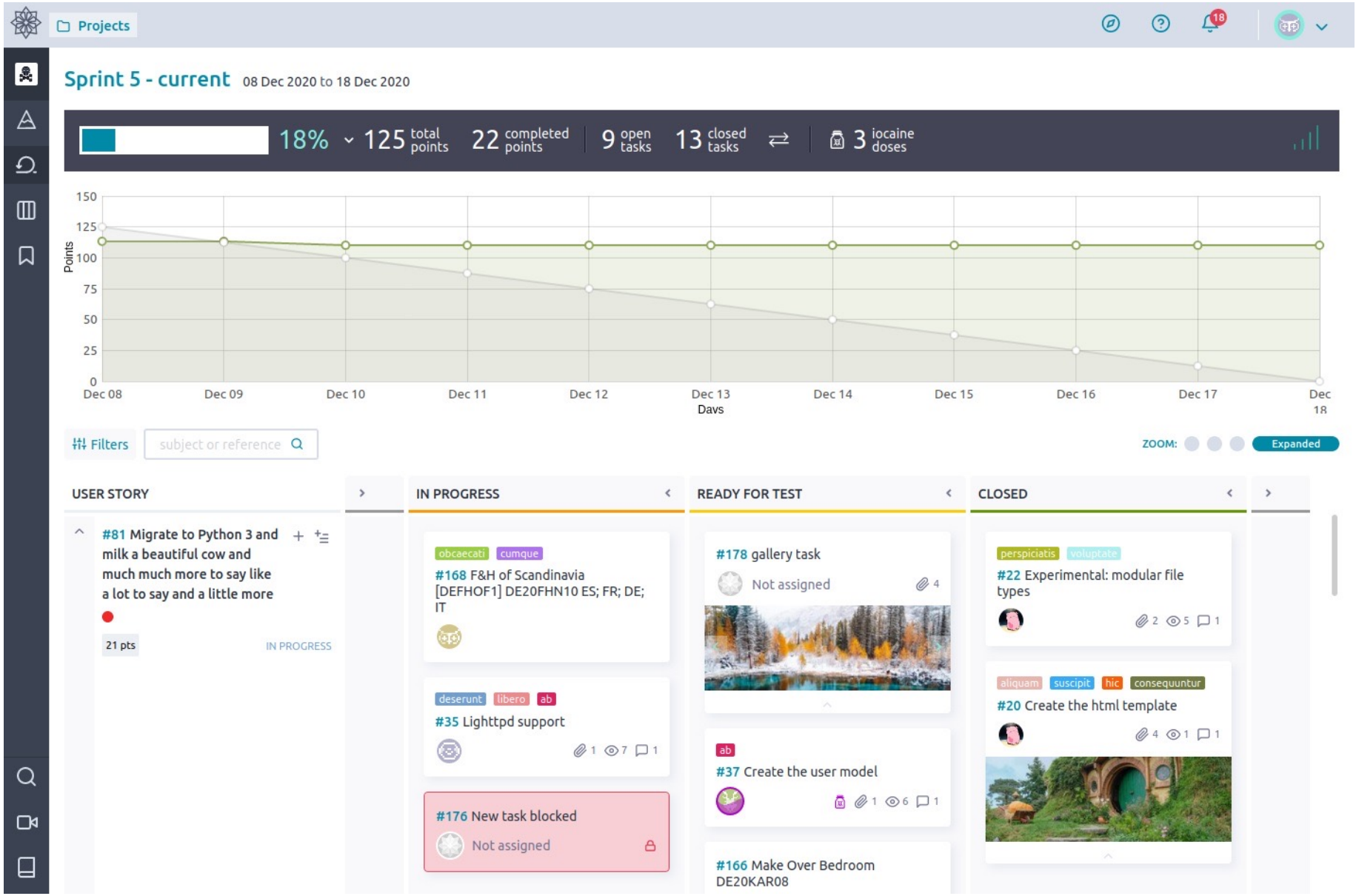


# Grafico Gantt




https://www.smartsheet.com/content/agile-gantt

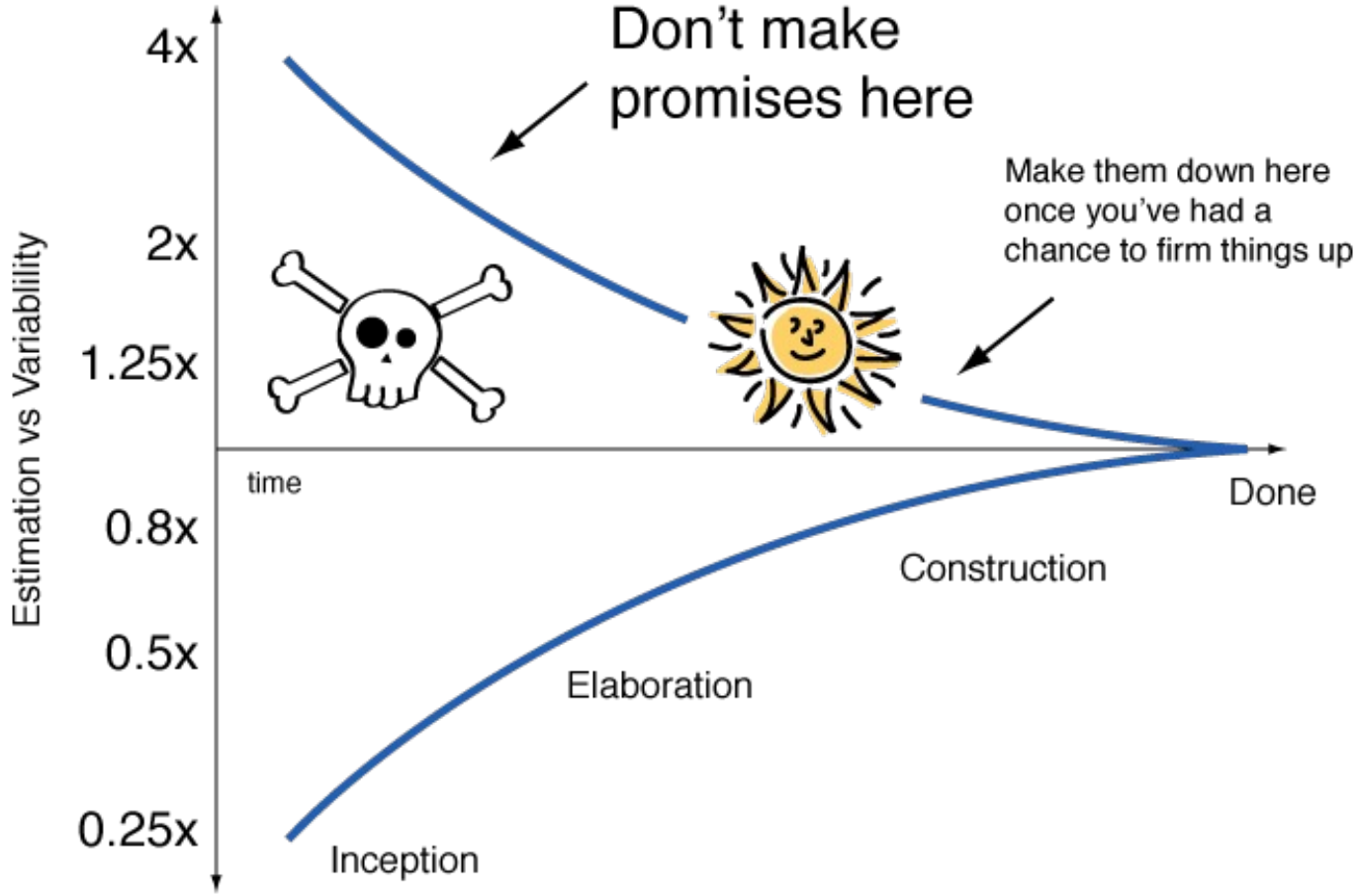
# Taiga


Projects
Sprint 5 - current 08 Dec 2020 to 18 Dec 2020
18% 125 total points 22 completed points 9 open tasks 13 closed tasks 3 iocaine doses
Points
Dec 08 Dec 09 Dec 10 Dec 11 Dec 12 Dec 13 Dec 14 Dec 15 Dec 16 Dec 17 Dec 18
Days
Filters
subject or reference
ZOOM: Expanded
USER STORY
IN PROGRESS
READY FOR TEST
CLOSED
#81 Migrate to Python 3 and milk a beautiful cow and much much more to say like a lot to say and a little more
21 pts
IN PROGRESS
obcaecati cumque
#168 F&H of Scandinavia [DEFHOF1] DE20FHN10 ES; FR; DE; IT
deserunt libero ab
#35 Lighttpd support
#176 New task blocked
Not assigned
#178 gallery task
Not assigned
ab
#37 Create the user model
#166 Make Over Bedroom DE20KAR08
perspiciatis voluptate
#22 Experimental: modular file types
aliquam suscipit hic consequuntur
#20 Create the html template

# Stimare e misurare un progetto sw

- La progettazione del software si può analizzare studiando cinque variabili: costo, durata, sforzo, produttività e numero di errori nel prodotto sw [Putnam 1992]
- Il project manager rispetto a tali variabili ha due compiti importanti: stimarle a preventivo, misurarle durante il progetto, e rendicontarle a consuntivo
- Come possiamo stimare costo, durata, sforzo, produttività e numero di errori prima che il progetto inizi?

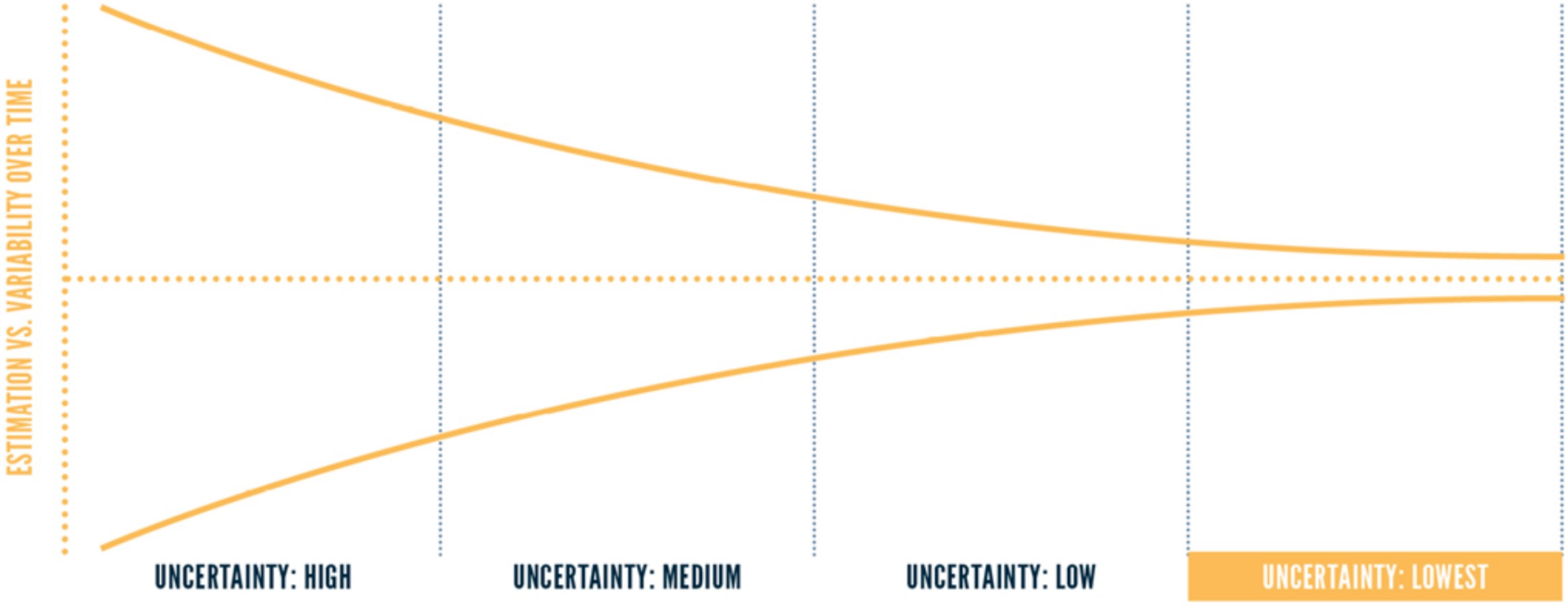
# Il cono dell'incertezza nello sviluppo sw



# L’incertezza nel sizing fonte:QSM

https://www.qsm.com/articles/sizing-matters?utm=gcaccess

| PHASE | WHAT | HOW | DO | DEPLOY/FIX |
|---|---|---|---|---|
| METHOD(S) | Order of Magnitude (T-shirt sizing) | Functional Requirements, Business Requirements, User Stories, Use Cases, ISO Standards | Functional Requirements, Business Requirements, User Stories, Use Cases, ISO Standards, RICE Objects, Technical Components, Source Code Files, SLOC Count | Functional Requirements, Business Requirements, User Stories, Use Cases, ISO Standards, RICE Objects, Technical Components, Source Code Files, SLOC Count |
| TYPE | High-level requirements | Refined requirements | Physical measurements | Refined physical measurements |



# Valore e costo del sw

Il *valore* di un sw è proporzionale al numero di utenti

Valore (sw) = #utenti*ricavo medio per utente

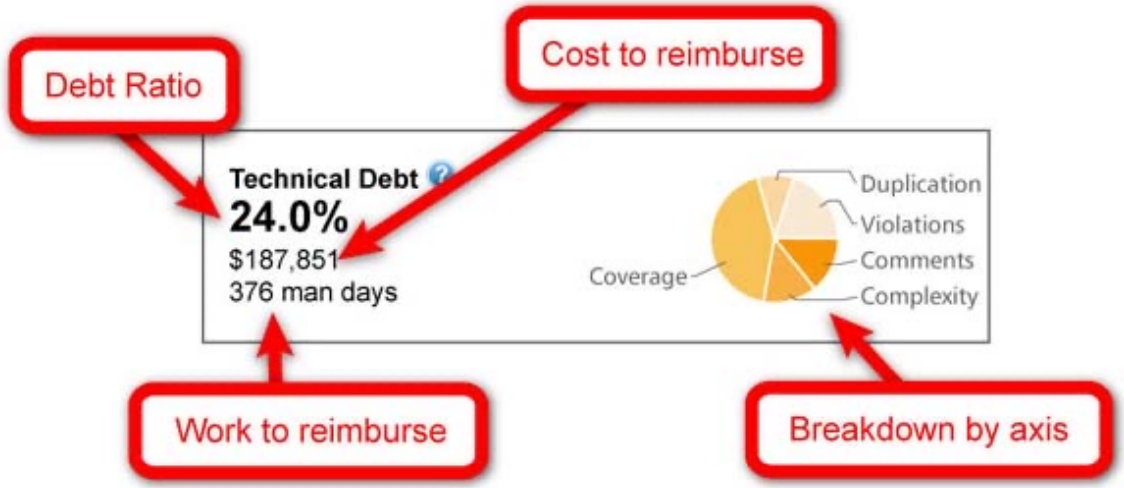
Il *costo* di un software è proporzionale allo sforzo di sviluppo (in mesi/persona), che a sua volta è proporzionale alla dimensione (*size*) del sw

- La produttività nel calcolo del costo del software viene considerata come un fattore che collega la dimensione del software (size) allo sforzo di sviluppo necessario per realizzarlo.
- Modelli come **COCOMO II** utilizzano parametri per stimare la produttività. Ad esempio, includono un "Productivity Index" che modula lo sforzo in base a vari fattori (ad esempio, la complessità del sistema, l'esperienza del personale, la qualità degli strumenti).

# Produttività

- La produttività è un'astrazione economica che si calcola col rapporto output/input
- La *produttività* di un team che ha prodotto un sw di dimensione *size* con un dato effort si calcola *size(sw)/effort*
- Esempio: «*il team ha avuto una produttività di 5K LOC/mese-persona*»

# Il debito tecnico




Il debito tecnico è una stima del costo di futuro sforzo addizionale causato da una soluzione prematura adottata oggi pur di consegnare un prodotto con qualche valore (lo sforzo futuro andrà ripagato con gli interessi)

In un software di buona qualità dovrebbe valere per ogni *versione* la disuguaglianza:

Valore (*versione*) >> Costo (*versione*) + Debito tecnico (*versione*)



https://en.wikipedia.org/wiki/Technical_debt

# Tecniche di stima dei costi di sviluppo

Le tecniche principali per stimare i costi di un progetto:

- **top-down** (o analogica): uso del costo di un progetto analogo o con componenti di costo noto come base della stima del nuovo progetto
- **bottom-up**: stima dei compiti individuali che compongono il progetto e loro somma per ottenere la stima totale
- **Design-to-cost**: uso di esperti per determinare quanta funzionalità può essere prodotta col budget disponibile
- **parametrica**: stima basata sull'uso di un modello matematico che usa una griglia di parametri

# Cosa va stimato

- **Durata**: estensione temporale del progetto, dall'inizio alla fine; dipende dalle dipendenze tra le attività di progetto; si misura di solito in mesi
- **Sforzo**: somma dei tempi di tutte le attività di progetto; si misura di solito in mesi/persona (mp)
- La **dimensione del team** si ricava dalla relazione sforzo/durata

# Esempio

- Un progetto deve durare un anno, coinvolgere quattro persone di cui due a metà tempo
- La durata è 12 mesi
- Lo sforzo è 36 mesi/persona

# Non confondere durata e sforzo!

**Esempio**:

- La *durata* legale di un corso di laurea è tre anni
- Lo *sforzo* è 180 cfu (stabilito per legge)
- Un cfu è pari a 25 h/studente

**Nota 1**: Il regolamento di CdL potrebbe definire durate diverse (es. sei anni), mantenendo però lo sforzo di 180 cfu richiesto dalla legge

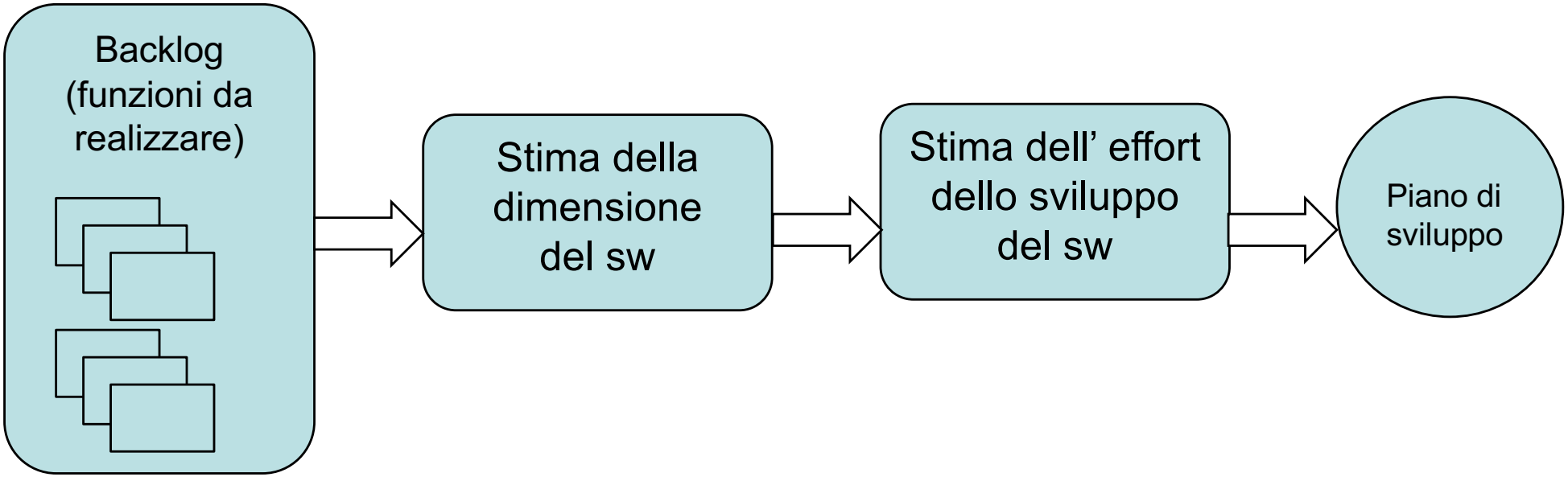
**Nota 2**: il cfu (=25h/studente) è una unità di misura dello sforzo (diversa dal mese/persona, che non è definito univocamente, ma che per esempio in COCOMO è pari a 152 h/persona)

# I componenti del costo del Sw

I principali componenti di costo sw sono:

- Costo dell'hardware di sviluppo
- Costo del software di sviluppo
- Costo delle risorse umane (**sforzo**)
- **Durata** complessiva

# La pianificazione inizia con la stima della dimensione del software



# Misurare le dimensioni del progetto

- La ***dimensione*** (size) di un progetto sw è il primo coefficiente di costo in molti modelli che stimano durata e sforzo
- Esistono tre misure che stimano la dimensione di un software da produrre:
  - Le linee di codice
  - I punti funzione (Function Points)
  - I punti-storia (history points) nei processi agili
- Queste misure hanno bisogno di dati storici per poter essere "calibrate" rispetto all'organizzazione che le usa

# Linee di codice

- La misura più comune della dimensione di un progetto software è il numero di linee di codice *sorgente* (*Lines of Code*, LoC;
  1 KLoC = mille **linee di codice** sorgente)
- LoC può tener conto delle linee vuote o dei commenti (CLoC); in generale si ha: LoC = NCLoC + CLoC, cioè i commenti si contano
- La distinzione più comune è tra LoC fisiche e LoC logiche

# Esercizio: contare le LOC

```
 /* Strncat() appends up to count characters from string
src to string dest, and then appends a terminating null
character. If copying takes place between objects that
overlap, the behavior is undefined. */
char *strncat (char *dest, const char *src, size_t count)
 {
            char *temp.dest;
            if (count ) {
                        while (*dest )
                                    dest++;
                        while  ((*dest.. .*src.. )) {
                                    if (count ..  0) {
                                    *dest.'\0';
                                     break;
                                    }
            }
             }   return temp;
 }
```

Quante LOC?

# LoC fisiche e logiche

```
for (i=0; i<100; ++i) printf("hello");
```

Una LoC fisica, due LoC logiche

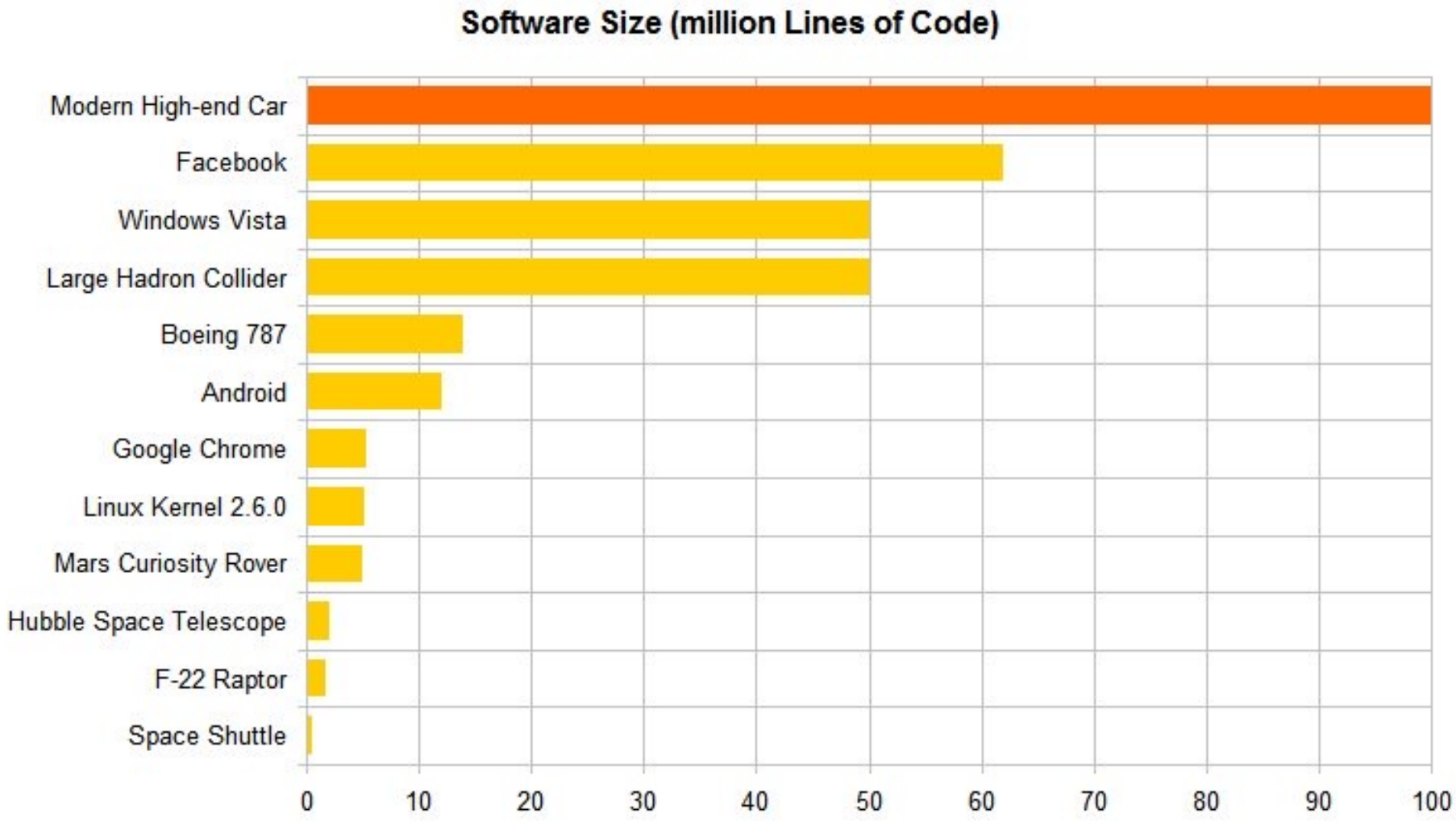
```
for (i=0; i<100; ++i)
{
printf("hello");
}
/* Now how many lines of code is this? */
```

Cinque LoC fisiche, due LoC logiche

# Esempio: Linux LOC

- March 1994, Linux 1.0.0 - 176,250 lines of code.
- March 1995, Linux 1.2.0 - 310,950 lines of code
- January 1999 - Linux - 1,800,847 lines of code
- January 2001 - Linux 2.4.0 - 3,377,902 lines of code
- December 2003 - Linux 2.6.0 - 5,929,913 lines of code
- 2012, Linux 3.2 - 14,998,651 lines of code.

- Fonte: Wikipedia- Linux kernel




https://www.linkedin.com/pulse/20140626152045-3625632-car-software-100m-lines-of-code-and-counting/

# Strumento per contare LoC

- https://dwheeler.com/sloccount/

# Esempio: stima LOC (da Pressman)

| | |
|---|---|
| • Interfaccia utente | 2.300 |
| • Gestione dati bidimensionali | 5.300 |
| • Gestione dati tridimensionali | 6.800 |
| • Gestione del db | 3.350 |
| • Visualizzazione grafica | 4.950 |
| • Controllo dispositivi | 2.100 |
| • Moduli di analisi del progetto | 8.400 |
| Totale LOC stimate | 33.200 |

Produttività "storica" per sistemi di questo tipo = 620 LOC/mp.

Costo del lavoro =€8000 /mese, quindi ogni LOC costa €13.

La stima di costo totale è €431,000 mentre la stima di sforzo è 54 mp

# LOC: pro e contro

- Metriche derivate:
  - $ per KLOC
  - errori o difetti per KLOC
  - LOC per mese/persona
  - pagine di documentazione per KLOC
  - Errori per mese-persona
  - $/pagina di documentazione
- Il codice sorgente è il prodotto tangibile del processo di sviluppo, ed esiste già parecchia letteratura sulla sua misurazione LOC
- Però, la misura delle LOC dipende dal linguaggio di programmazione
- Inoltre, penalizza (sottovaluta la produttività) programmi scritti bene e concisi

# Aspetti critici delle stime dimensionali

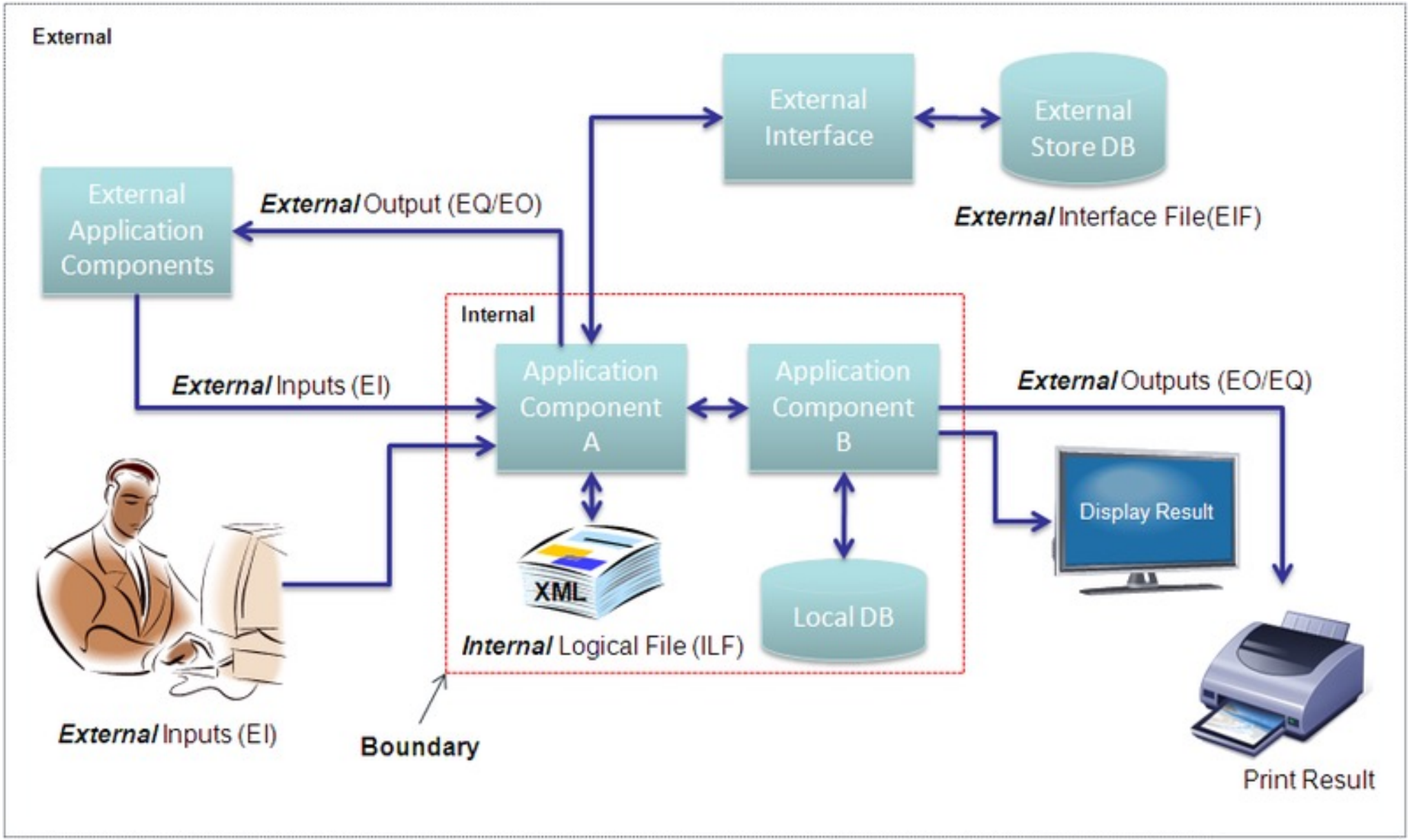
1. È difficile stimare la dimensione in LOC di una nuova applicazione
2. La stima LOC non tiene conto della diversa complessità e potenza delle istruzioni (di uno stesso linguaggio o di linguaggi diversi)
3. È difficile definire in modo preciso il criterio di conteggio (istruzioni spezzate su più righe, più istruzioni su una riga...)
4. Una maggior produttività in LOC potrebbe comportare più codice di cattiva qualità?
5. Il codice non consegnato (es. test) va contato?

# Function Point [Albrecht]

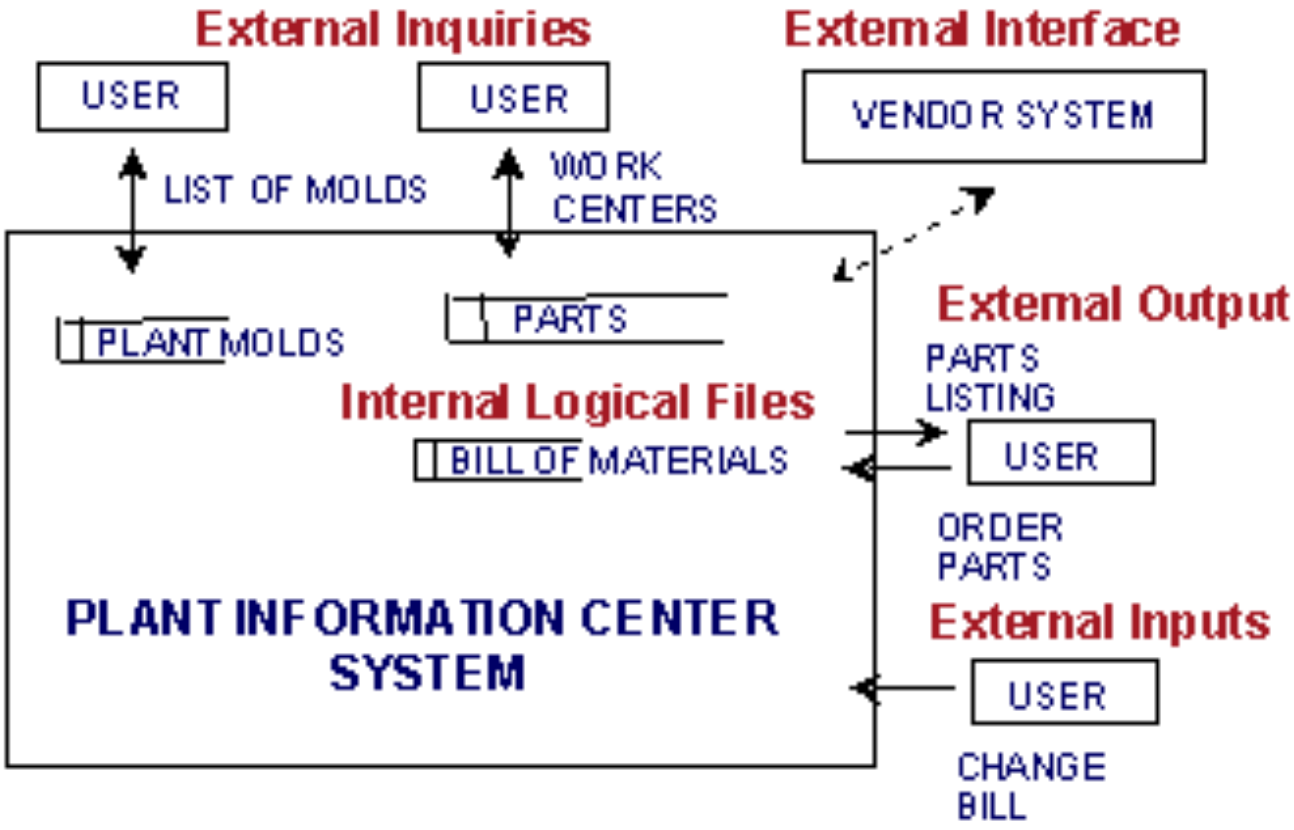
L'analisi Function Point enumera le funzionalità di un sistema dal punto di vista **utente**

*"The original objective of the Function Points work was to define a measure [that would] help managers analyze application development and maintenance work and highlight productivity improvement opportunities."*

*"The Function Points method measures the equivalent functions of end-user applications regardless of the language, technology, or development environment used to create the application."*


External
External Interface
External Store DB
External Application Components
External Output (EQ/EO)
External Interface File(EIF)
Internal
External Inputs (EI)
Application Component A
Application Component B
External Outputs (EO/EQ)
Display Result
XML
Local DB
Internal Logical File (ILF)
External Inputs (EI)
Boundary
Print Result

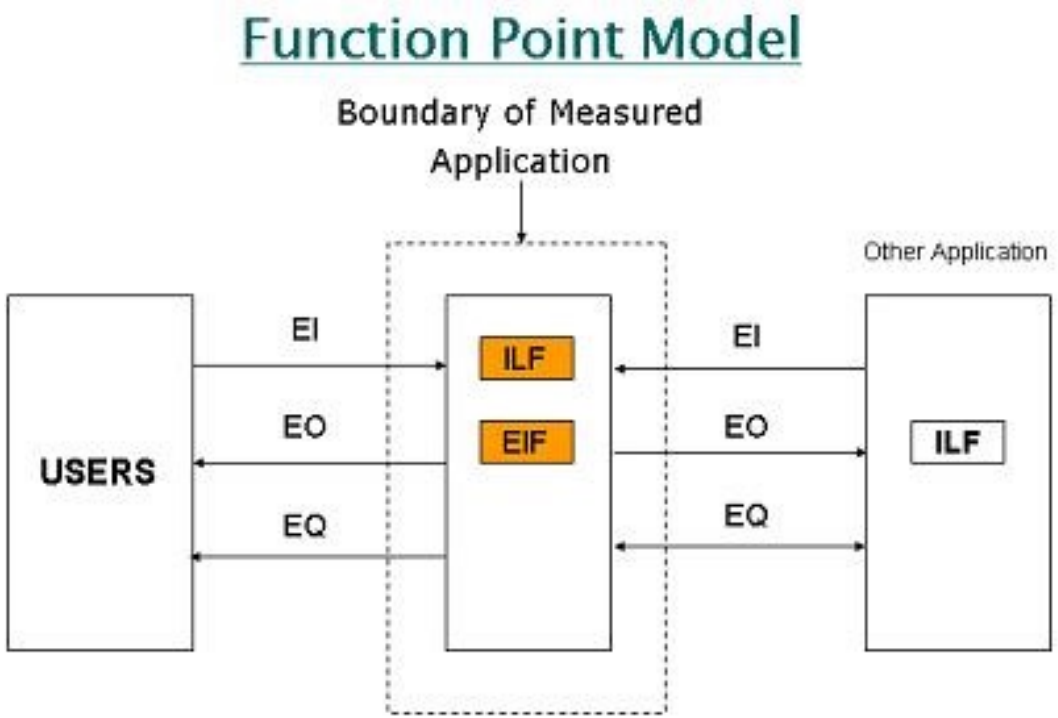
# Esempio: sistema informativo d'impianto



# Calcolo di FP: passo 1

Si descrive l'applicazione da realizzare

Si parte da una **descrizione** del sistema (**specifica**) di natura funzionale



# Calcolo di FP: passo 2

Individuare nella **descrizione dei requisiti utente** i seguenti elementi:

- External Input: informazioni, dati forniti dall'utente (es. nome di file, scelte di menù, campi di input)
- External Output: informazioni, dati forniti all'utente dell'applicazione (es. report, messaggi d'errore)
- External Inquiry: sequenze interattive di richieste – risposte
- External Interface File: interfacce con altri sistemi informativi esterni
- Internal Logical File: file principali logici gestiti nel sistema

# Calcolo di FP: passo 3

Occorre classificare i singoli elementi funzionali per complessità di progetto, usando la seguente tabella di pesi

| **Tipo Elementi** | Semplice | Medio | Complesso |
|---|---|---|---|
| External inputs | 3 | 4 | 6 |
| External outputs | 4 | 5 | 7 |
| External inquiries | 3 | 4 | 6 |
| External files | 7 | 10 | 15 |
| Internal files | 5 | 7 | 10 |

# Calcolo di FP: passo 4

Censire gli elementi di ciascun tipo, moltiplicare per il lor “peso” e sommare:

$UFC = \Sigma_{[i=1\_5]} (\Sigma_{[j=1\_3]} ( \text{elemento } i,j * \text{peso } i,j))$

Dove i = tipo elemento

j = complessità (semplice o media o complessa)

UFC = Unajdjusted Function Count

# Calcolo di FP: passo 5

Definire il fattore di complessità tecnica dell' applicazione (TFC)

Calcolo di TFC:

**TFC = 0.65+0.01* $\Sigma_{[i=1-14]}$ Fi**

Ciascun fattore Fi viene valutato tra 0 (irrilevante) e 5 (massimo)

Il valore complessivo di TFC varia nell'intervallo da 0.65 (sviluppo facile) a 1.35 (sviluppo difficile)

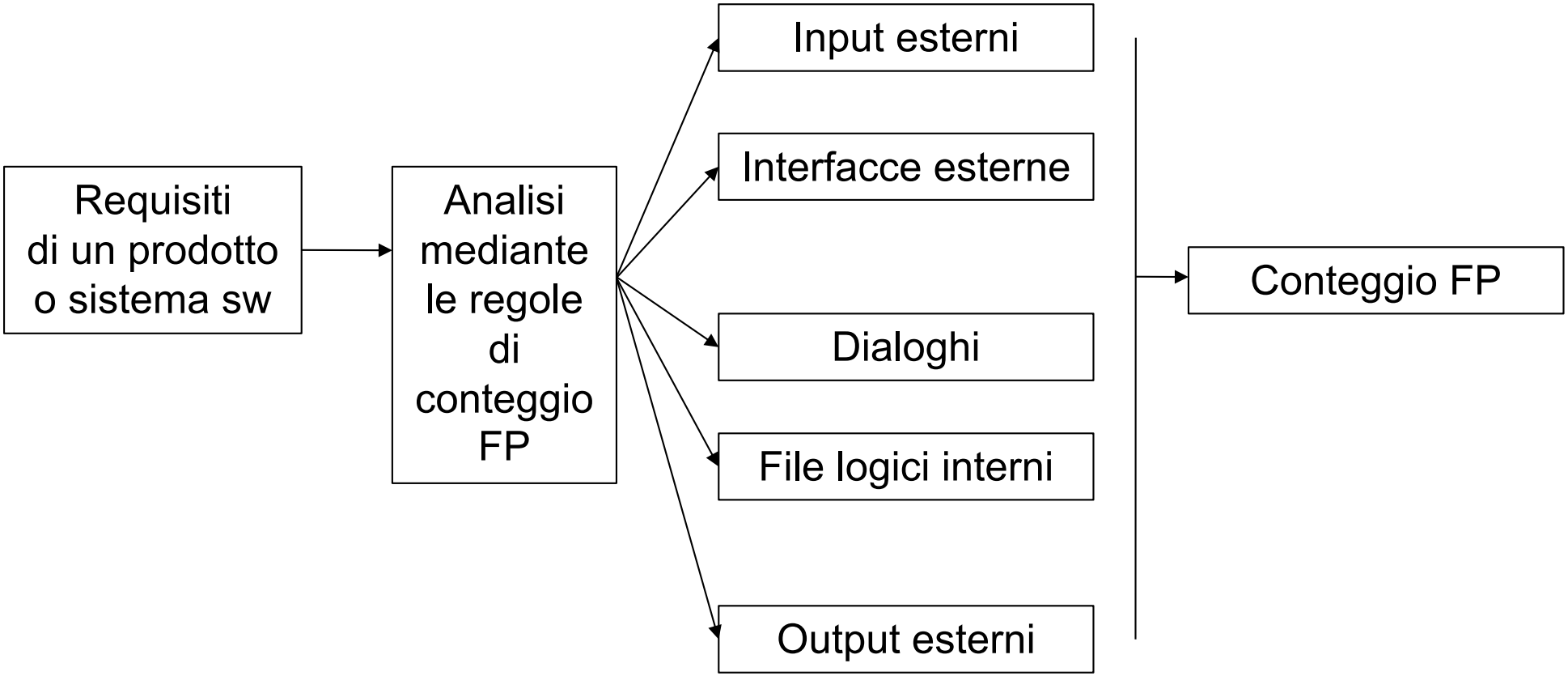
| | | | |
|---|---|---|---|
| F1 | Reliable backup and recovery | F2 | Data Communications |
| F3 | Distributed functions | F4 | Performance |
| F5 | Heavily used configuration | F6 | Online data entry |
| F7 | Operational ease | F8 | Online update |
| F9 | Complex interface | F10 | Complex processing |
| F11 | Reusability | F12 | Installation ease |
| F12 | Multiple sites | F14 | Facilitate change |

# Calcolo di FP: passo finale

Alla fine la formula risulta:

FP = UFC * TFC

# Riassunto del metodo



# Esempio

- File interni logici
  - DB Clienti
  - DB conti correnti
  - DB Pagamenti
  - DB Banche
- File esterni di interfaccia
  - GUI responsabile gestione clienti
  - GUI responsabile commerciale
  - Input esterni
  - Creazione account
  - Eliminazione account
  - Ricarica tessera
  - Saldo tessera
  - Richiesta pagamento importo
- Interrogazioni esterne
  - Visualizzazione stato cliente
  - Visualizzazione stato pagamenti
  - Visualizzazione saldi
- Input esterni
  - Notifica pagamenti
  - Gestione invio tessera

# Esempio

La descrizione funzionale di un sistema contiene

| | | | | | |
|---|---|---|---|---|---|
| 6 | Input “medi” | x | 4 | = | 24 |
| 6 | Output “complessi” | x | 7 | = | 42 |
| 2 | File “medi” | x | 10 | = | 20 |
| 3 | Inquiries “semplici” | x | 2 | = | 6 |
| 2 | Interfacce “complesse” | x | 10 | = | 20 |
| | Unadjusted FC | | | = | 112 |

# Esempio (continua)

| | |
|---|---|
| Data communications | 3 |
| Distributed processing | 2 |
| Online processing | 4 |
| Complex internal processing | 5 |
| Multiple sites | 3 |
| TFC | 17 |

Calcolo finale: UFC * TFC = 112 * (.65+.17) = 92 FP

# Esempio (continua)

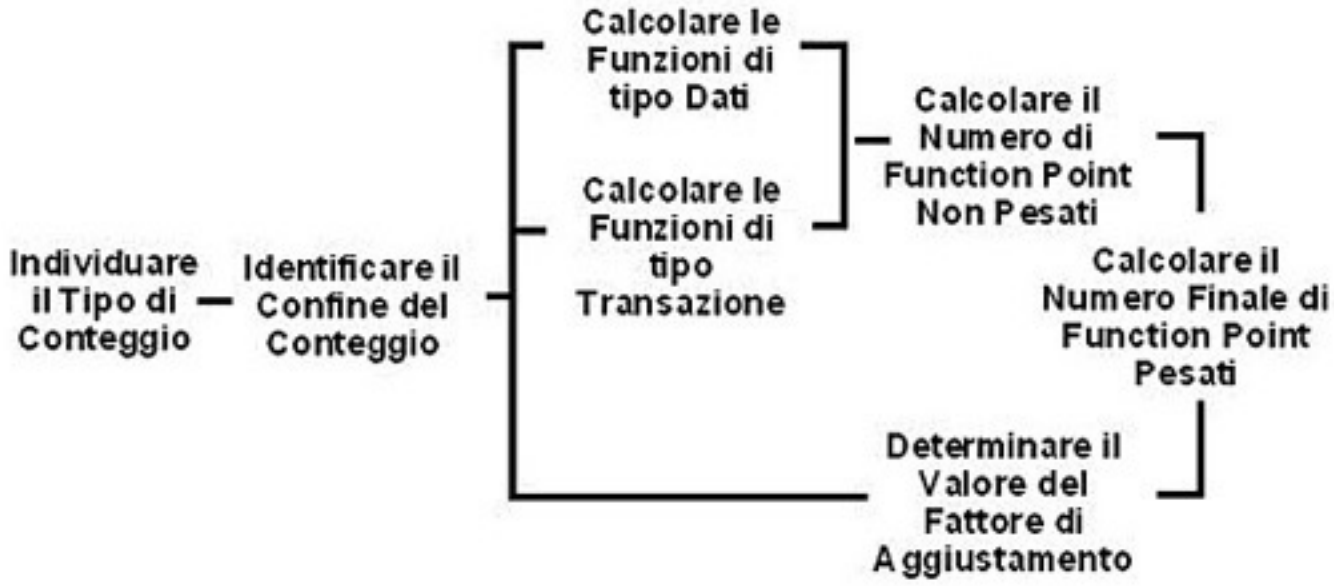
Se si sa, per esempio a causa di esperienze passate, che il numero medio di FP per mese-persona è pari a 18, allora si può fare la stima che segue:

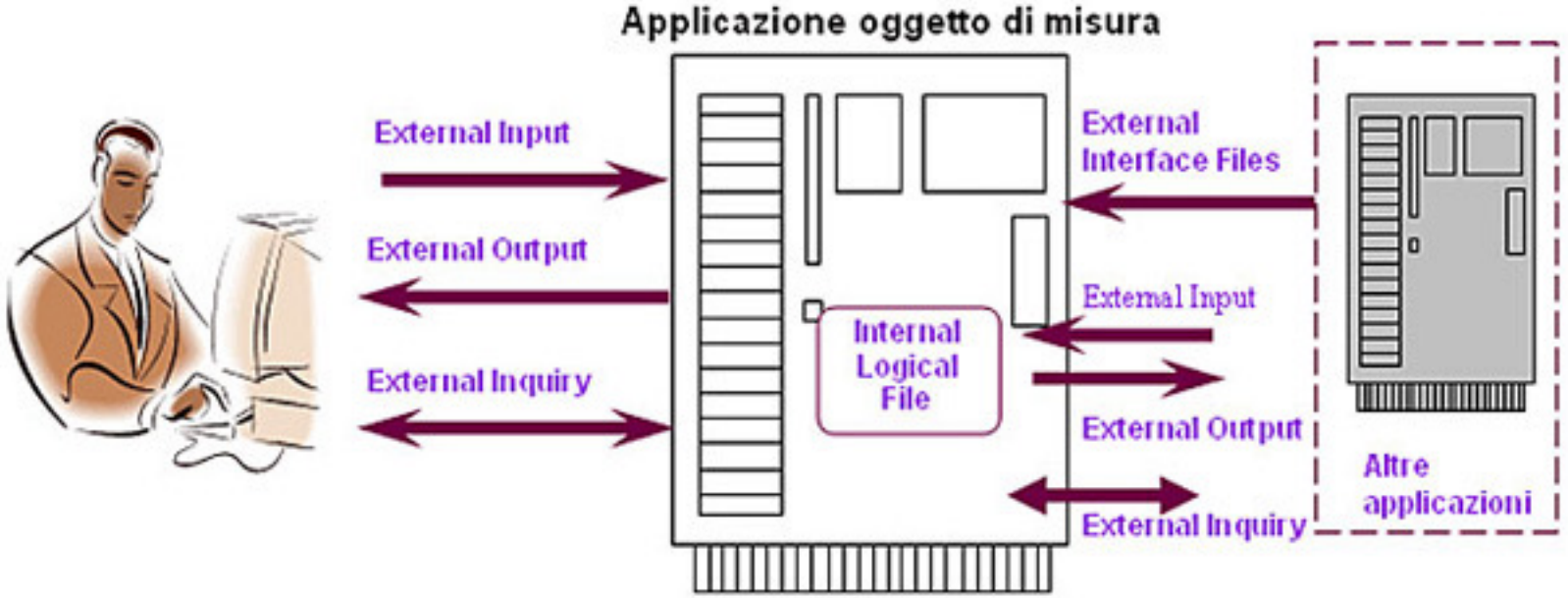
$$92 \text{ F.P.} \div 18 \text{ F.P./mp} = 5.1 \text{ mp}$$

Se lo stipendio medio mensile per lo sviluppatore è di is €6.500, allora il costo [dello sforzo] del progetto è

$$5.1\text{mp} \times €6.500 = €33.150$$

# Riassunto


Individuare il Tipo di Conteggio
Identificare il Confine del Conteggio
Calcolare le Funzioni di tipo Dati
Calcolare le Funzioni di tipo Transazione
Calcolare il Numero di Function Point Non Pesati
Calcolare il Numero Finale di Function Point Pesati
Determinare il Valore del Fattore di Aggiustamento


Applicazione oggetto di misura
External Input
External Output
External Inquiry
Internal Logical File
External Interface Files
External Input
External Output
External Inquiry
Altre applicazioni

# Calibrazione

- Il conteggio dei FP si basa su giudizi **soggettivi**, quindi persone diverse possono raggiungere risultati diversi

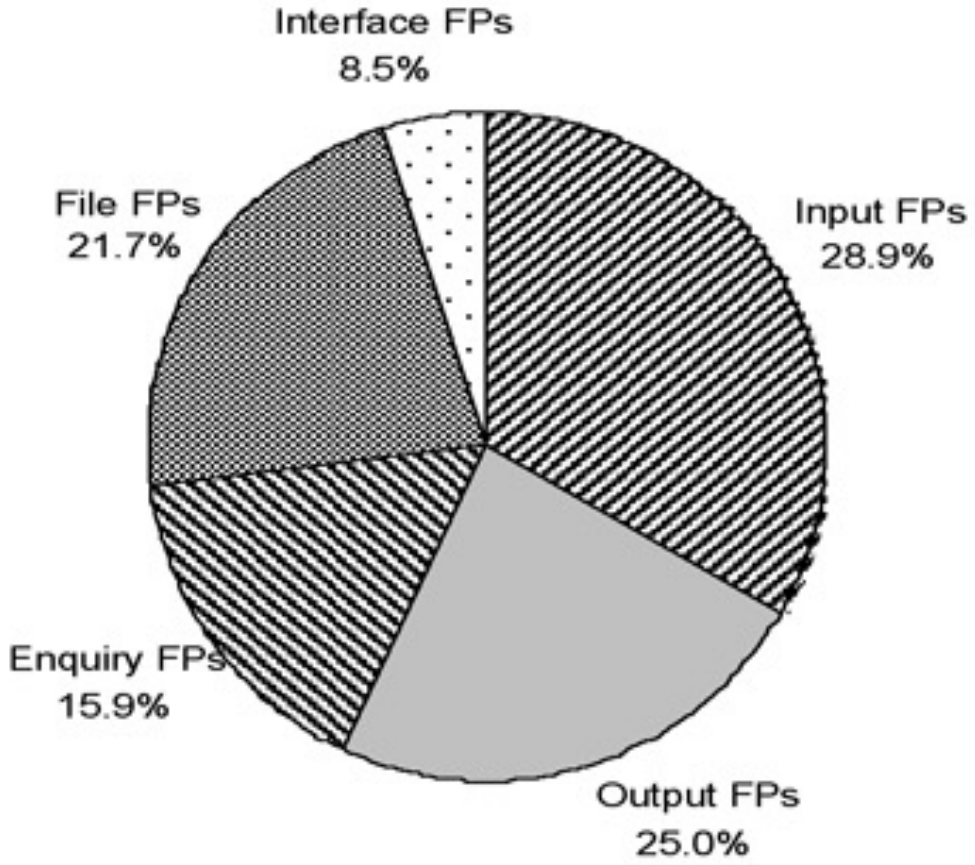
- Quando si introduce l'Analisi FP (in sigla: FPA) in una organizzazione, è necessaria una fase di **calibrazione**, usando i progetti sviluppati in passato come base del sistema di conteggio

# Misurazioni di alcuni sistemi

(Capers Jones 2010)

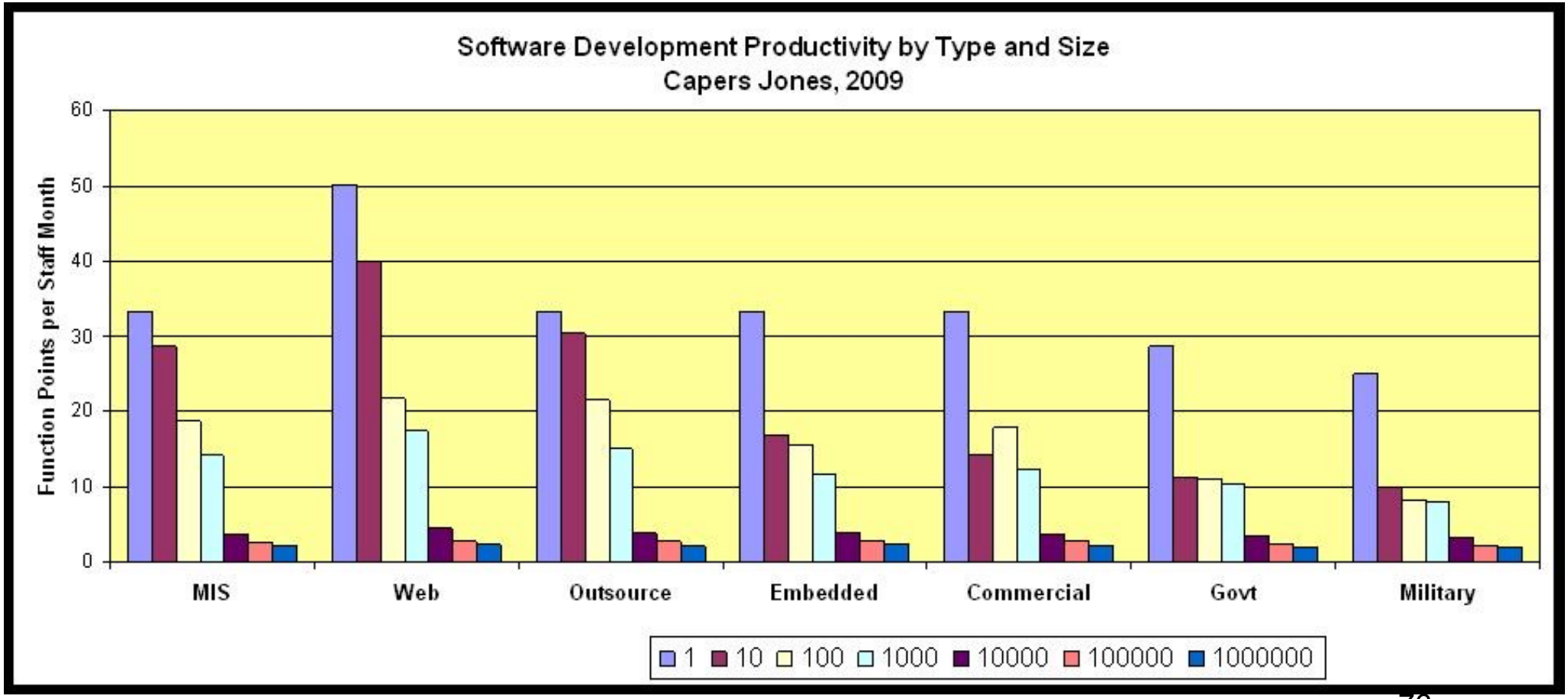
| Sistema | Size in FP | KLOC | LOC/FP |
|---|---|---|---|
| US Air traffic control | 306.324 | 65.000 | 213 |
| Israeli Air Defense | 300.655 | 24.000 | 80 |
| SAP | 296.764 | 24.000 | 80 |
| Iran air defense | 260.100 | 24.000 | 91 |
| MS Vista | 157.658 | 10.090 | 64 |
| MS Office Pro | 93.500 | 6.000 | 64 |
| Iphone IOS | 19.300 | 516 | 27 |
| Google search | 18.640 | 1.193 | 64 |
| Amazon Website | 18.080 | 482 | 27 |
| Apple Leopard | 17.884 | 477 | 27 |
| Linux | 17.505 | 700 | 40 |

# Function Point Mix
# New Developments




Source: Estimating, Benchmarking & Research Suite Release 9
[209 projects - FPA METHOD: IFPUG 4]

# Produttività FP/mesepersona per diversi mercati di sw



# Costo per FP

Alcuni sviluppatori usano scale di costo basate su FP per i contratti

- Requisiti iniziali = $500 per FP;
- Modifiche nei primi tre mesi = $600 per FP;
- Modifiche successive = $1000 per FP

# Stima basata su Modelli di Costo

- I modelli di costo permettono una stima «rapida» dello sforzo, utile durante le primissime fasi di un progetto
- Questa prima stima viene poi raffinata, più avanti nel ciclo di vita, mediante dei fattori detti **cost driver**
  - Il calcolo si basa sull'**equazione dello sforzo**:
    - $E = A + B*S^C$

    dove E è lo sforzo (in mesi-persona),
    A, B, C sono parametri dipendenti dal progetto e dall'organizzazione che lo esegue,
    S è la dimensione del prodotto stimata in KLOC o FP.

# Problemi dei costi del software

I costi del software sono dominanti, in percentuale dei costi totali dei sistemi informativi.

I problemi di stima dei costi sw sono causati da:

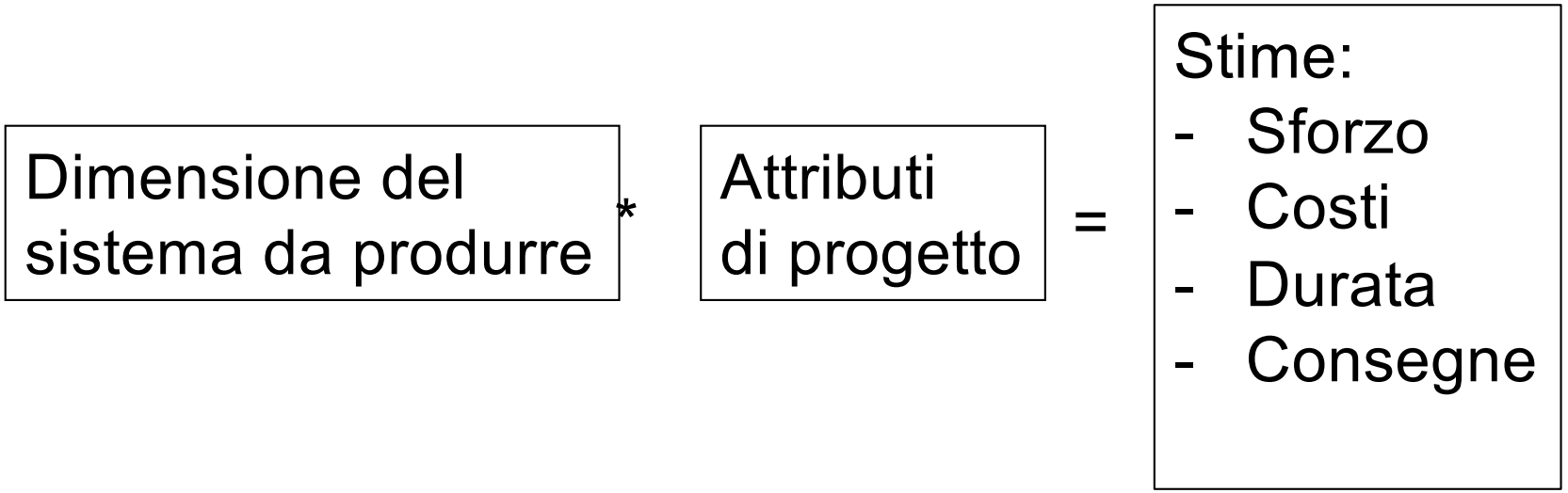
- incapacità di dimensionare accuratamente un progetto;
- incapacità di definire nel suo complesso il processo e l'ambiente operativo di un progetto;
- valutazione inadeguata del personale, in quantità e qualità;
- mancanza di requisiti di qualità per la stima di attività specifiche nell'ambito del progetto

# Modelli dei costi del software

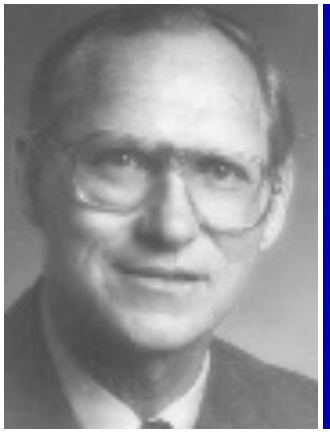
## Esempi di modelli commerciali

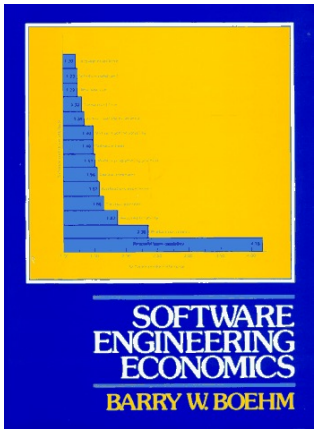
- **COCOMO**
- COSYSMO
- COSTXPERT
- SLIM
- SEER
- Costar, REVIC, etc.

# Uso di parametri per le stime



# Modelli di costi sw: COCOMO






- Il **Constructive Cost Model** (**COCOMO**) di Barry Boehm è uno dei modelli parametrici più diffusi per fare stime nei progetti software
- COCOMO 1 è descritto nel libro *Software Engineering Economics*, 1981
- COCOMO 2 è descritto nel libro *Software Cost Estimation*, 2000
- COCOMO è un modello basato su regressione (cioè su un archivio storico) che considera vari parametri del prodotto e dell'organizzazione che lo produce, pesati mediante una griglia di valutazione

# COCOMO: forma equazionale

- Il principale calcolo di COCOMO si basa sull'Equazione dello Sforzo per stimare il numero di mesi-persona mp necessari per un progetto

  Costo_Stimato = # mp * costo_lavoro

- La maggior parte delle altre grandezze stimate (la durata, la dimensione del personale, ecc.) vengono poi derivate da questa equazione

# COCOMO: forma equazionale

$$\text{Performance} = (\text{Complexity})^{(\text{Process})} * (\text{Team}) * (\text{Tools})$$

- **dove:**
  - **Performance** = Sforzo
  - **Complexity** = Dimensione del codice generato
  - **Process** = Maturità del processo e metodo
  - **Team** = Abilità, esperienza, motivazione
  - **Tools** = Automazione del processo

**Boehm, *Software Engineering Economics***

# COCOMO 1

- Boehm costruì la prima versione di un modello di costo chiamato CoCoMo 1 nel 1981
- CoCoMo 1 è una collezione di tre modelli:
  - **Basic** (applicato all'inizio del ciclo di vita del progetto)
  - **Intermediate** (applicato dopo la specifica dei requisiti)
  - **Advance** (applicato al termine della fase di design)

# COCOMO1

- I tre modelli hanno forma equazionale:

  **Effort = $a*S^{b}*EAF$**

  – Effort è lo sforzo in mesi-persona

  – EAF è il *coefficiente di assestamento*

  – S è la dimensione stimata del codice sorgente da consegnare, misurata in migliaia di linee di codice (KLOC)

  – a e b sono dei coefficienti che dipendono dal tipo di progetto

# Tipi di progetti in COCOMO1

- **Organic mode** (progetto semplice, sviluppato in un piccolo team)
- **Semidetached mode** (progetto di difficoltà intermedia)
- **Embedded** (progetto con requisiti molto vincolanti e in campo non ben conosciuto)

# Formule per il Modello Base

| Tipo fase basic | A | B | EAF | FORMULA RISULTANTE |
|---|---|---|---|---|
| Organic | 2.4 | 1.05 | 1 | $E = 2.4 * S^{1.05}$ |
| Semi detached | 3.0 | 1.12 | 1 | $E = 3.0 * S^{1.12}$ |
| Embedded | 3.6 | 1.20 | 1 | $E = 3.6 * S^{1.20}$ |

# Un esempio

Dimensione = 200 KLOC
Sforzo(in mp) = a * Dimensione $^{b}$

**Organic**: Sforzo = 2.4*(200 $^{1.05}$) = 626 mp
**Semidetached**: Sforzo = 3.0*(200 $^{1.12}$) = 1133 mp
**Embedded**: Sforzo = 3.6*(200 $^{1.20}$) = 2077 mp

# Modello Intermedio

- Prende il modello basic come riferimento
- Identifica un insieme di attributi che influenzano il costo (detti *cost driver*)
- Moltiplica il costo di base per un fattore che lo può accrescere o decrescere
- La stima di questo modello azzecca i valori reali con approssimazione ±20% circa il 68% delle volte
- (Modello più usato con COCOMO 1)

# Fattori di costo

(COCOMO1, Intermediate)

| Cost Drivers | Rating | | | | | |
|---|---|---|---|---|---|---|
| | Very Low | Low | Nominal | High | Very High | Extra High |
| Product attributes | | | | | | |
| Required software reliablity | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 | |
| Database size | | 0.94 | 1.00 | 1.08 | 1.16 | |
| Product complexity | 0.70 | 0.85 | 1.00 | 1.15 | 1.30 | 1.65 |
| Computer attributes | | | | | | |
| Execution time constraint | | | 1.00 | 1.11 | 1.30 | 1.66 |
| Main storage constraint | | | 1.00 | 1.06 | 1.21 | 1.56 |
| Virtual machine volatility* | | 0.87 | 1.00 | 1.15 | 1.30 | |
| Computer turnaround time | | 0.87 | 1.00 | 1.07 | 1.15 | |
| Personnel attributes | | | | | | |
| Analyst capabilities | 1.46 | 1.19 | 1.00 | 0.86 | 0.71 | |
| Applications experience | 1.29 | 1.13 | 1.00 | 0.91 | 0.82 | |
| Programmer capability | 1.42 | 1.17 | 1.00 | 0.86 | 0.70 | |
| Virtual machine experience* | 1.21 | 1.10 | 1.00 | 0.90 | | |
| Programming language experience | 1.14 | 1.07 | 1.00 | 0.95 | | |
| Project attributes | | | | | | |
| Use of modern programming practices | 1.24 | 1.10 | 1.00 | 0.91 | 0.82 | |
| Use of software tools | 1.24 | 1.10 | 1.00 | 0.91 | 0.83 | |
| Required development schedule | 1.23 | 1.08 | 1.00 | 1.04 | 1.10 | |

*For a given software product, the underlying virtual machine is the complex of hardware and software (operating system, database management system) it calls on to accomplish its task.

# Calcolo del fattore moltiplicativo

- I fattori moltiplicativi dovuti agli attributi hanno ciascuno un valore che indica lo spostamento dal valore normale di quel determinato attributo
- Il valore dei diversi fattori si determina tenendo conto dei progetti passati (calibrazione)
- Il prodotto della valutazione degli attributi rilevanti forma EAF

# Esempio

Dimensione = 200 KLOC

Sforzo = a * Dimensione $^{b}$ * EAF

Cost drivers:

| | |
|---|---|
| Low reliability | 0.88 |
| High product complexity | 1.15 |
| Low application experience | 1.13 |
| High programming language experience | 0.95 |

EAF= 0.88 * 1.15 * 1.13* 0.95 = 1.086

**Organic**: Sforzo = 2.4*(200 $^{1.05}$) * 1.086= 906 mp

**Semidetached**: Sforzo = 3.0*(200 $^{1.12}$) * 1.086 = 1231 mp

**Embedded**: Sforzo = 3.6*(200 $^{1.20}$) * 1.086 = 2256 mp

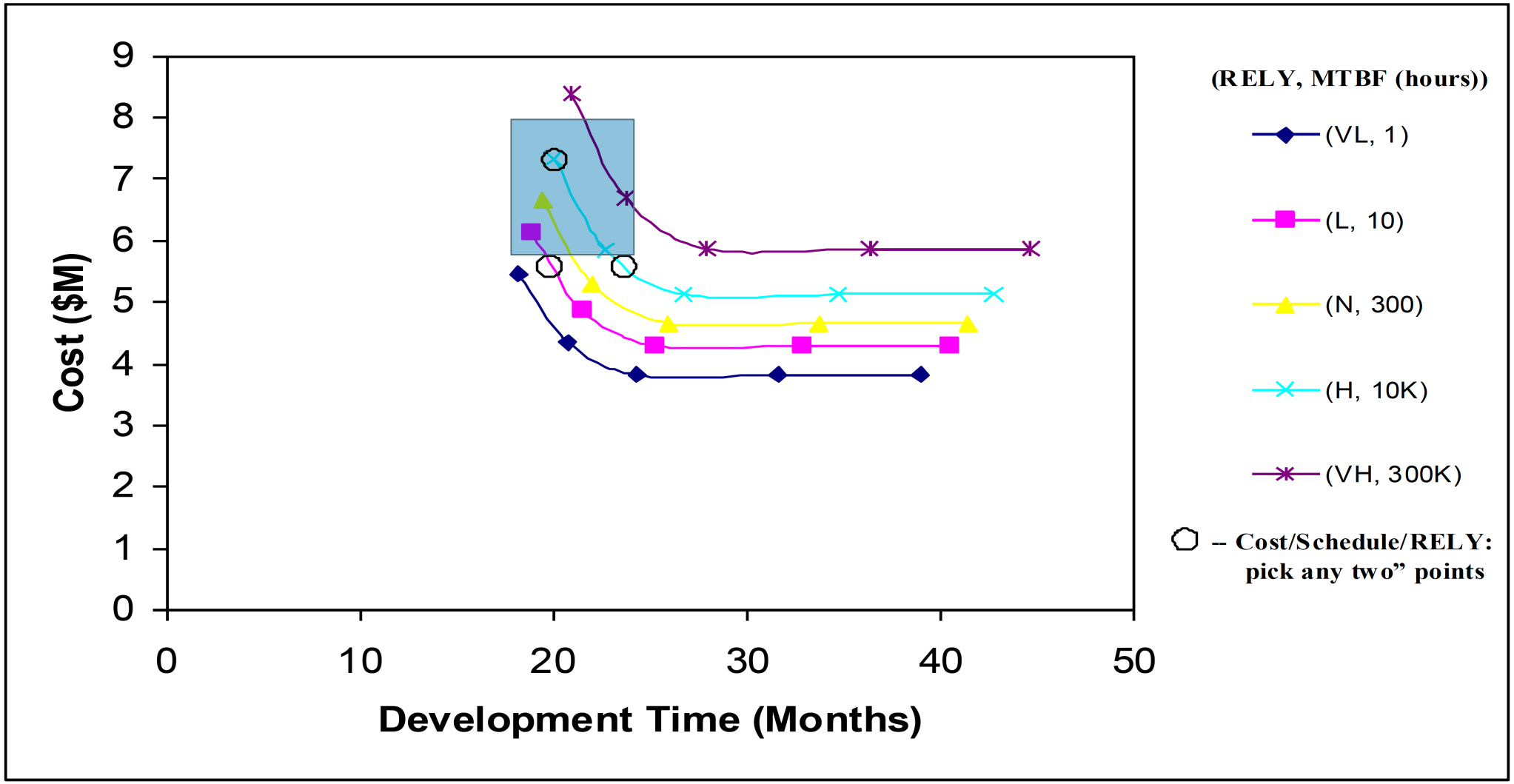
# Esempio 2

- Sw di comunicazione per trasferimento fondi (per es. Bancomat)
- Embedded mode
- 10 KLOC

| Cost Drivers | Situation | Rating | Effort Multiplier |
|---|---|---|---|
| Required software reliability | Serious financial consequences of software fault | High | 1.15 |
| Database size | 20,000 bytes | Low | 0.94 |
| Product complexity | Communications processing | Very high | 1.30 |
| Execution time constraint | Will use 70% of available time | High | 1.11 |
| Main storage constraint | 45K of 64K store (70%) | High | 1.06 |
| Virtrual machine volatility | Based on commercial microprocessor hardware | Nominal | 1.00 |
| Computer turnaround time | Two hour average turnaround time | Nominal | 1.00 |
| Analyst capabilities | Good senior analysts | High | 0.86 |
| Applications experience | Three years | Nominal | 1.00 |
| Programmer capability | Good senior programmers | High | 0.86 |
| Virtual machines experience | Six months | Low | 1.10 |
| Programming language experience | Twelve months | Nominal | 1.00 |
| Use of modern programming practices | Most techniques in use over one year | High | 0.91 |
| Use of software tools | At basic minicomputer tool level | Low | 1.10 |
| Required development schedule | Nine months | Nominal | 1.00 |

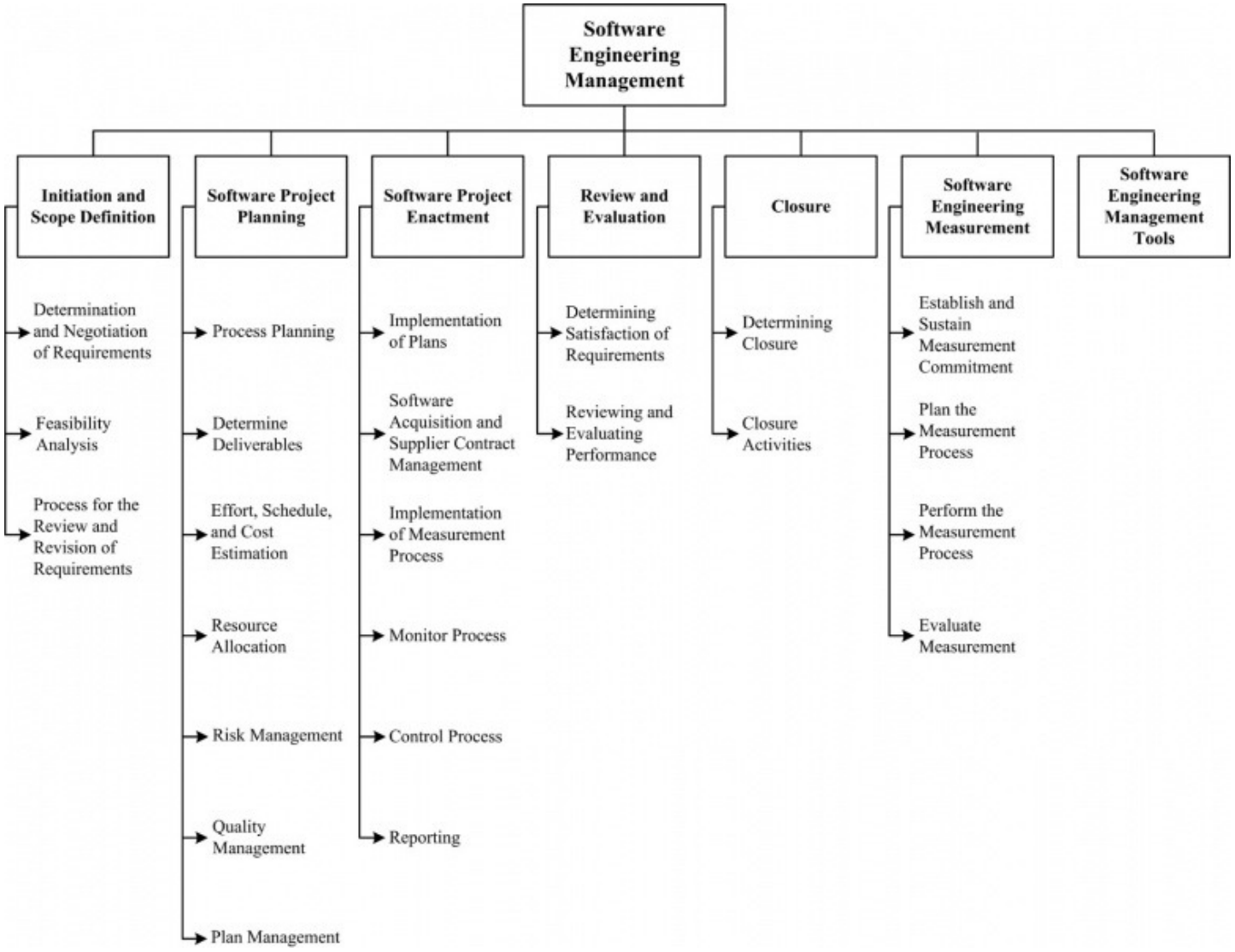
# Esempio 2

- Effort = 2.8 x $(10)^{1.20}$ = 44 mp
- Moltiplicatori di effort = 1.35
- Sforzo stimato reale 44 x 1.35 = 59 mp
- Questo indice (59 pm) si può usare in altre formule per calcolare o stimare
  - Costo in Euro
  - Piani di sviluppo
  - Distribuzioni di attività
  - Costi dell'hw
  - Costi di manutenzione annui
  - Ecc.




La figura mostra il bilanciamento di costo, durata, e affidabilità (RELY cost driver) per 100 KSLOC
Si supponga che il progetto richieda alto livello RELY (10K-ore MTBF), una durata 20 mesi e un budget di $5.5M.
Non si possono avere le tre cose insieme: Il budget deve salire a $7.33M.
Per un costo di $5.5.M, si può avere un livello alto RELY e una durata di 23.5 mesi, oppure una durata di 20 mesi e un basso livello RELY level.
Per avere le tre cose insieme occorre ridurre la dimensione a 77 KSLOC.

# Project Management nel SWEBOK



# Domande di autotest

- Quali sono i compiti di un project manager?
- Cos’è il cono dell’incertezza?
- Cos’è un mese-persona? Quante ore-persona vale?
- Il processo influenza il costo di sviluppo del sw?
- Cos'è una linea di codice? Cos'è un function point?
- Un programmatore preferisce essere pagato a LOC oppure a FP realizzati?
- Cosa calcola il modello COCOMO?
- Cos’è uno history point?

# Riferimenti


- Wysocki, *Effective Project Management* (Traditional, Agile, Extreme), Wiley 2014
- Fairley, *Managing and Leading Software Projects*, Wiley, 2009
- McConnell, *Professional Software Development*, AW 2004
- IEEE Standard 1058-1998 for Software Project Management Plans
- PMI, *A Guide to the Project Management Body of Knowledge, 6ed, 2017*
- Longstreet*, Function Points Analysis Training Course*, 2004
- Bohem, *Software Cost Estimation with COCOMO 2*, PrenticeHall, 2000
- Putnam & Myers, *Measures for Excellence*, YourdonPress, 1992

# Siti

- www.pmi.org
- www.mindtools.com/pages/article/newPPM_01.htm
- sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html
- www.ifpug.org (associazione FP)
- www.gufpi-isma.org
- www.dwheeler.com/sloc/
- www.devdaily.com/FunctionPoints/FunctionPoints.shtml
- www.cosmic.com
- brodzinski.com (blog su sw project management)
- gispict.wikispaces.com/Analisi+dei+costi+software
- alvinalexander.com/FunctionPoints/ (esempio FP)
- conferences.embarcadero.com/article/32094#Eqs (stesso esempio, FP)

# Strumenti

- http://softwarecost.org/tools/COCOMO/
- www.dotproject.net
- sourceforge.net/projects/projectlibre
- www.planningpoker.com

# Domande?