

# Project Management dei Progetti Software



*Prof. Paolo Ciancarini  
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
  - Linee di codice
  - Function Point
- Stima Algoritmica
  - COCOMO

# La legge di Brooks

*Aggiungere personale ad un progetto sw  
in ritardo lo farà ritardare ancora di più*

F. Brooks, *The mythical Man-Month*, 1975  
[en.wikipedia.org/wiki/The\\_Mythical\\_Man-Month](http://en.wikipedia.org/wiki/The_Mythical_Man-Month)

# Discussione

- Cos' è un progetto? come si gestisce?



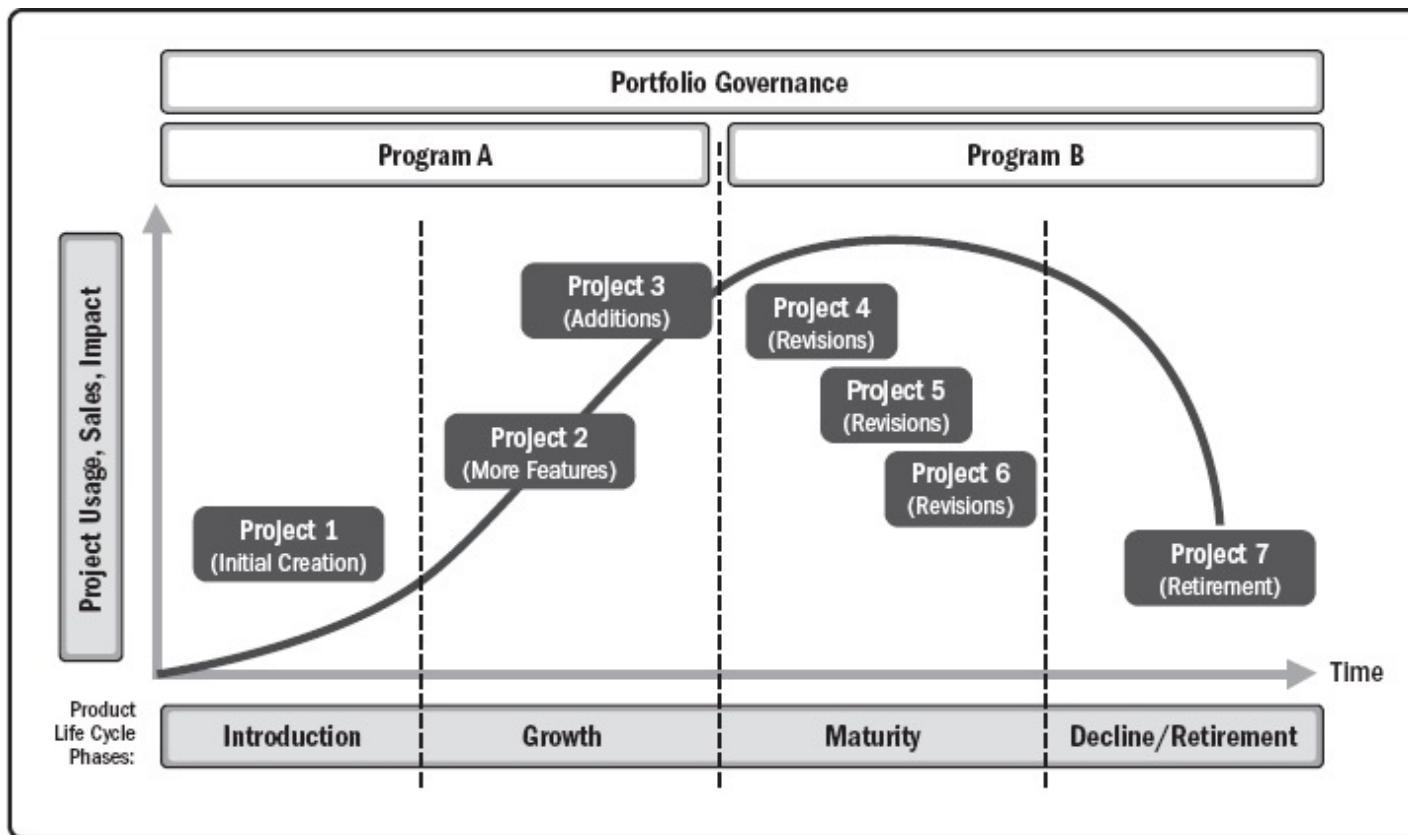
# Alcune definizioni

(fonte: PMI)

- **Progetto**: impresa temporanea intrapresa per raggiungere uno scopo
- **Project Management**: applicazione di conoscenze, tecniche e strumenti alle attività di uno specifico progetto allo scopo di portarlo a termine con successo
- **Product Management** : funzione aziendale di pianificazione, produzione e marketing di uno specifico prodotto che segue un ciclo di vita di prodotto

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

(fonte: Guida PMI 2021)



# Progetti software

(fonte:PMI)

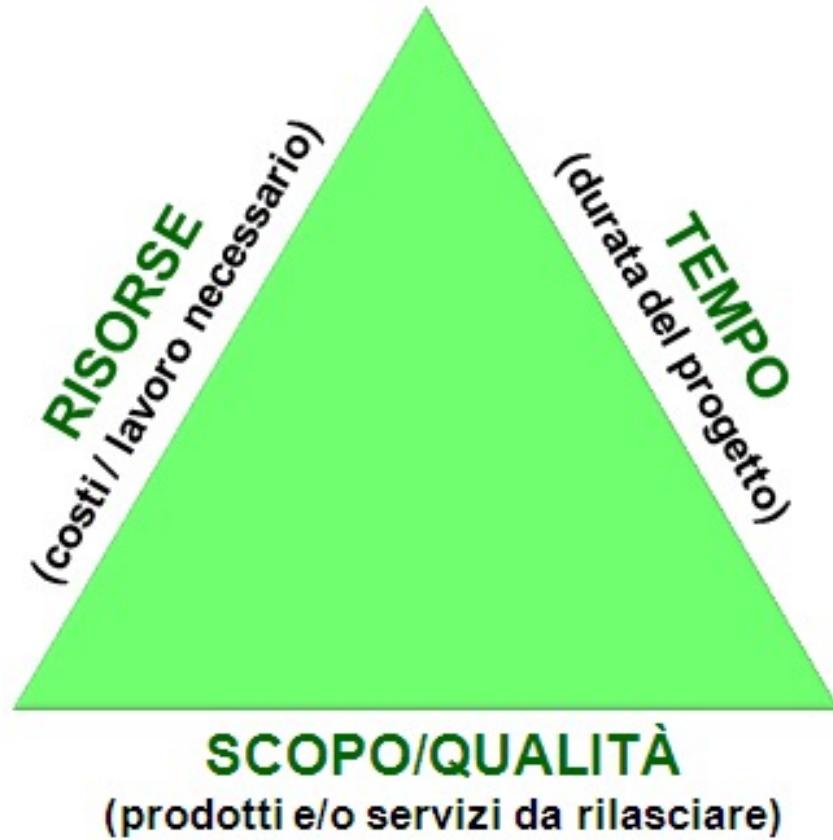
- Software projects, like all projects, are undertaken to achieve a specific objective.
- In addition to creating new products, software projects are often undertaken to **modify** an existing software product, to **integrate** a set of existing software components, to **extend** the capabilities of software products, or to modify the software **infrastructure** of an organization.
- Software projects may also be undertaken to satisfy **service** requests, **maintenance** needs, or to provide **operations** support.
- **Software product** life cycles, in contrast to project life cycles, typically involve maintenance and support activities
- **IT projects**, such as design of an enterprise information system, IT service transition to another vendor, or deploying a solution to end users are not software projects in the traditional sense

# 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

# Il triangolo del project management



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

- Incrementare lo scopo (funzioni+qualità) significa aumentare i tempi e i costi del progetto;
- ridurre 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 necessari

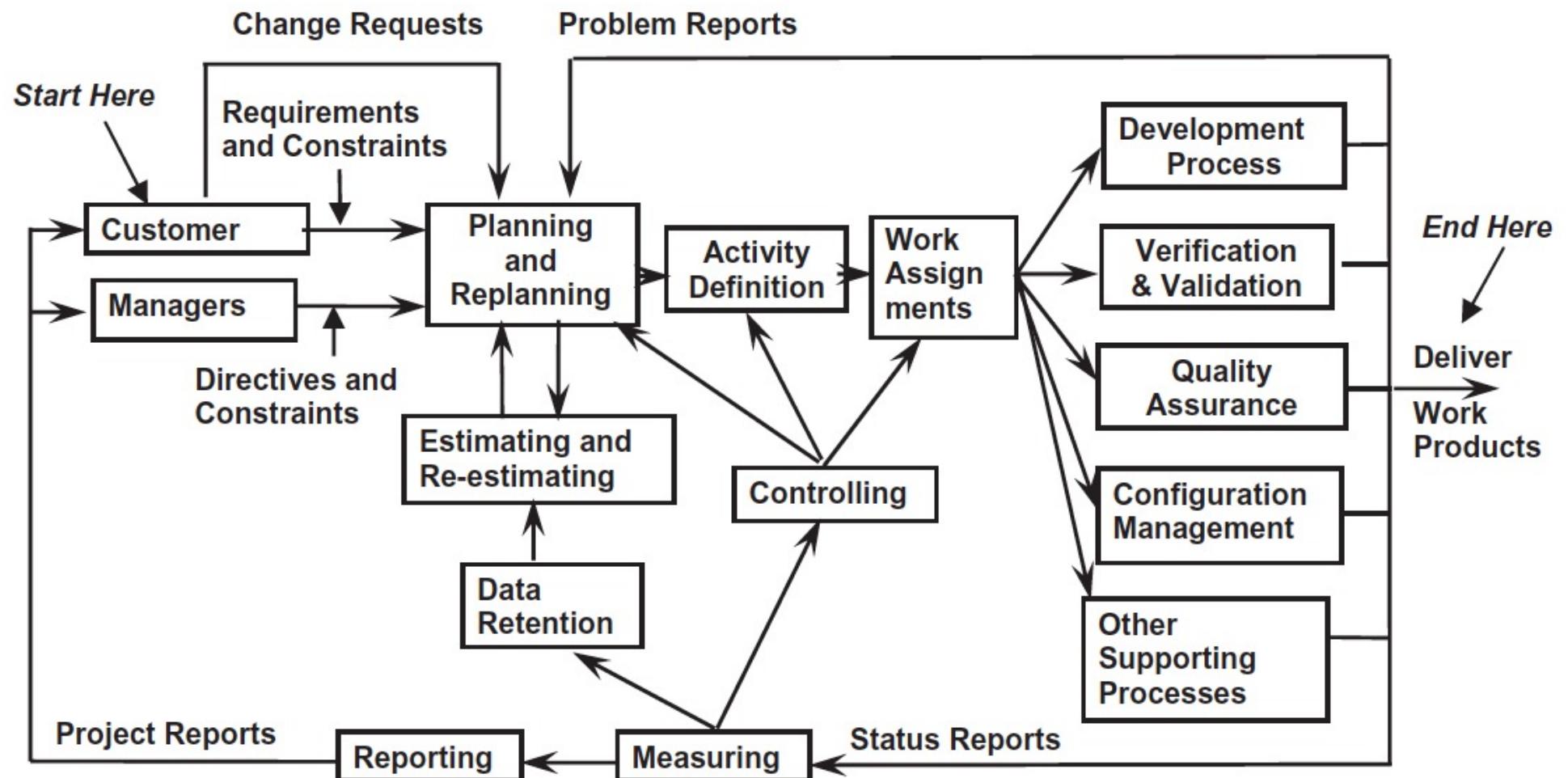
# Il Project Manager

- 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

# Compiti del PM

1. Comprendere l'obiettivo del progetto
2. Identificare il processo di sviluppo
3. Determinare la struttura del team di progetto
4. Identificare il processo manageriale
5. Sviluppare un calendario di attività del team
6. Pianificare l'acquisizione di persone nel team
7. Iniziare la gestione dei rischi
8. Identificare i documenti da produrre
9. Iniziare il processo...

# Le attività di gestione di un progetto sw



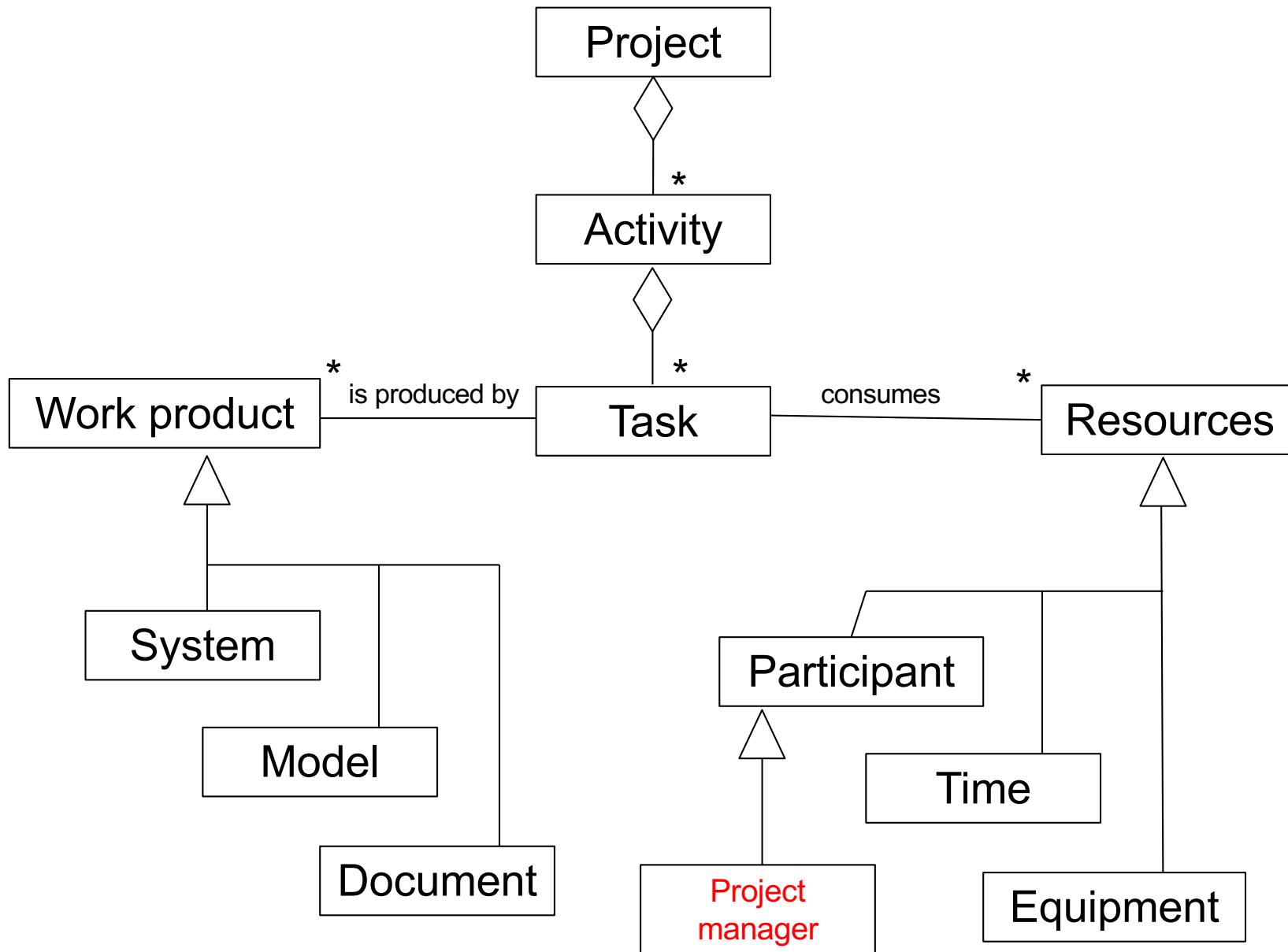
# SPMP: Sw Process Management Plan

- Documento standard IEEE 1058.1
  - Largamente usato
  - Adatto per ogni tipo di software
- Determinazione delle unità di lavoro
- Stima delle risorse
- Calcolo del budget necessario
- Costruzione di un piano temporale dettagliato

# IEEE SPMP

- Descrizione del progetto
- Organizzazione
- Gestione manageriale
- Gestione tecnica
- Articolazione
  - Work packages
  - Budget
  - Piano temporale

1.	Introduction
1.1	Project Overview
1.2	Project Deliverables
1.3	Evolution of the Software Project Management Plan
1.4	Reference Materials
1.5	Definitions and Acronyms
2.	Project Organization
2.1	Process Model
2.2	Organizational Structure
2.3	Organizational Boundaries and Interfaces
2.4	Project Responsibilities
3.	Managerial Process
3.1	Management Objectives and Priorities
3.2	Assumptions, Dependencies, and Constraints
3.3	Risk Management
3.4	Monitoring and Controlling Mechanisms
3.5	Staffing Plan
4.	Technical Process
4.1	Methods, Tools, and Techniques
4.2	Software Documentation
4.3	Project Support Functions
5.	Work Packages, Schedule, and Budget
5.1	Work Packages
5.2	Dependencies
5.3	Resources Requirements
5.4	Budget and Resource Allocation
5.5	Schedule
Additional components	



# I team agili non hanno un PM

- 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
- IEEE 1490-2011 è “Guide adoption of PMI standard to the project management BoK”

# La gestione del tempo di progetto

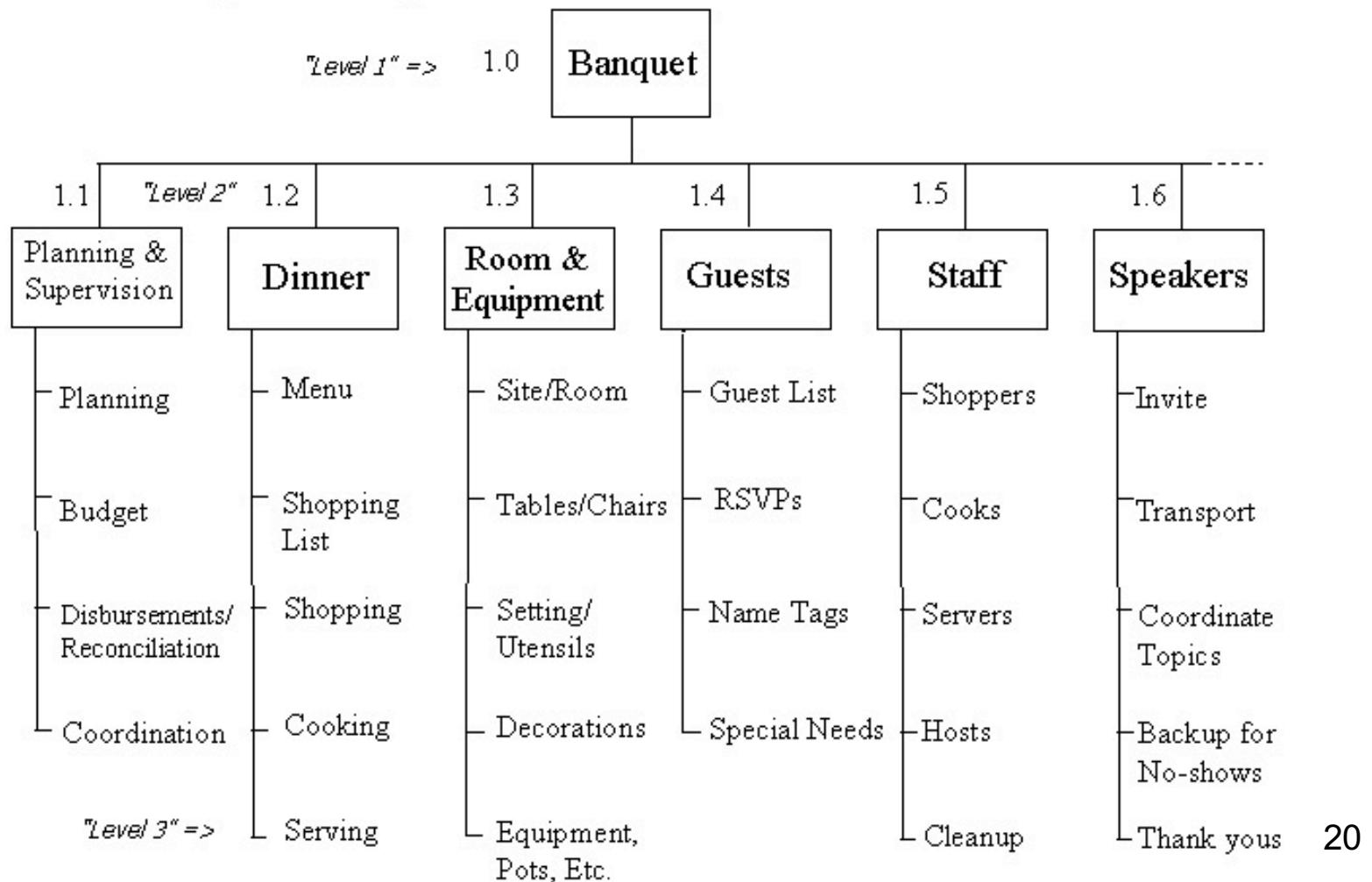
- Definizione delle attività
- Sequenziamento delle attività
- Stima delle risorse per le attività
- Stima della durata delle attività
- Schedulazione delle attività
- Controllo delle attività

# Terminologia

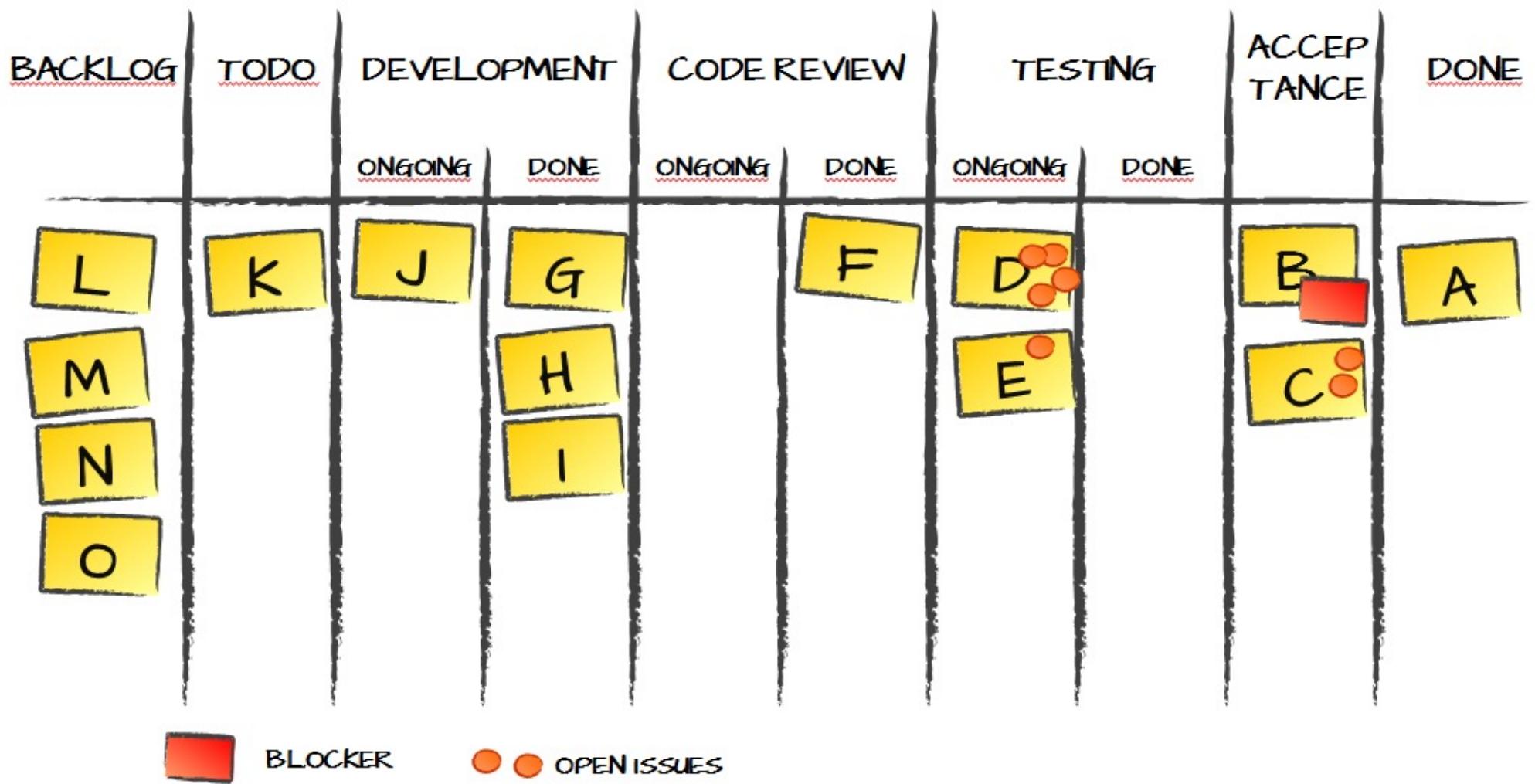
- Work breakdown: articolazione (cioè elenco e relazioni) delle attività di progetto
- Milestone: evento significativo nella vita di un progetto
- Percorso critico: sequenza delle attività la cui durata è incomprimibile

# Work Breakdown Structure (WBS)

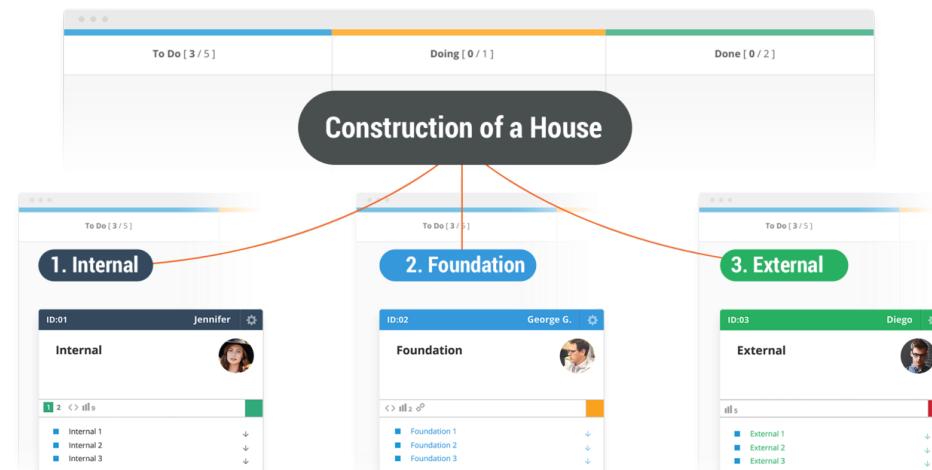
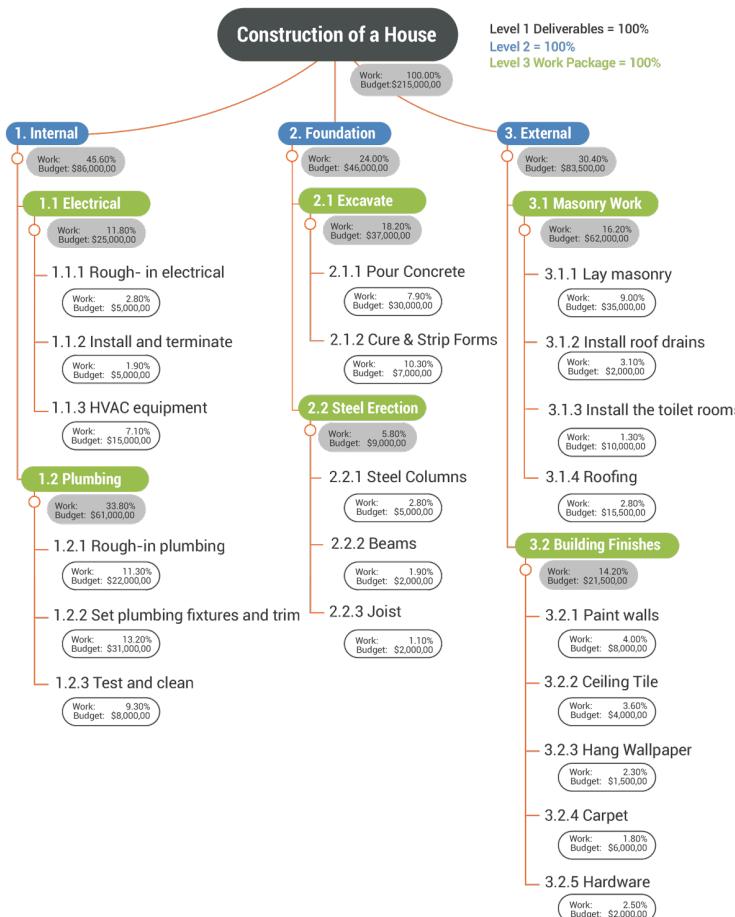
## WBS Example - Banquet



# Kanban



# Applicare Kanban a una WBS

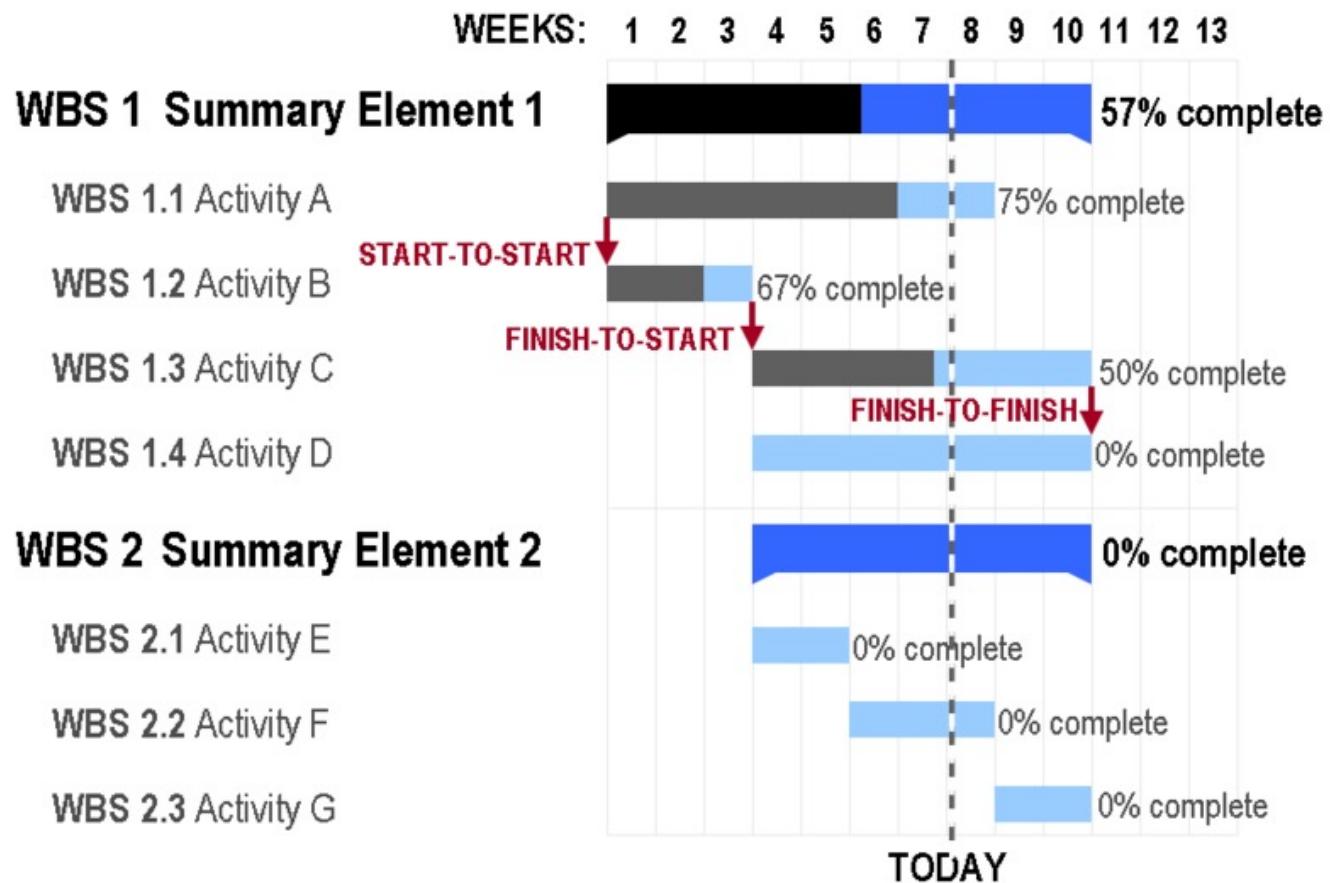


Task	Earliest start	Length	Type	Dependent on...
A. High level analysis	Week 0	1 week	Sequential	
B. Selection of hardware platform	Week 1	1 day	Sequential	A
C. Installation and commissioning of hardware	Week 1.2	2 weeks	Parallel	B
D. Detailed analysis of core modules	Week 1	2 weeks	Sequential	A
E. Detailed analysis of supporting modules	Week 3	2 weeks	Sequential	D
F. Programming of core modules	Week 3	2 weeks	Sequential	D
G. Programming of supporting modules	Week 5	3 weeks	Sequential	E
H. Quality assurance of core modules	Week 5	1 week	Sequential	F
I. Quality assurance of supporting modules	Week 8	1 week	Sequential	G
J. Core module training	Week 6	1 day	Parallel	C,H
K. Development and QA of accounting reporting	Week 5	1 week	Parallel	E
L. Development and QA of management reporting	Week 5	1 week	Parallel	E
M. Development of Management Information System	Week 6	1 week	Sequential	L
N. Detailed training	Week 9	1 week	Sequential	I, J, K, M

# Esempio: Costruzione di un GANTT e di un PERT

[http://www.mindtools.com/pages/article/newPPM\\_01.htm](http://www.mindtools.com/pages/article/newPPM_01.htm)

# Grafico Gantt



# Grafico PERT/CPM

(Program Evaluation and Review Technique)

cammino critico: sequenza di attività logicamente dipendenti tra di loro che collegano il nodo iniziale (attività di inizio) al nodo finale (attività di fine) e la cui somma delle durate è massima. Un ritardo in una di queste attività implica il ritardo dell'intero progetto

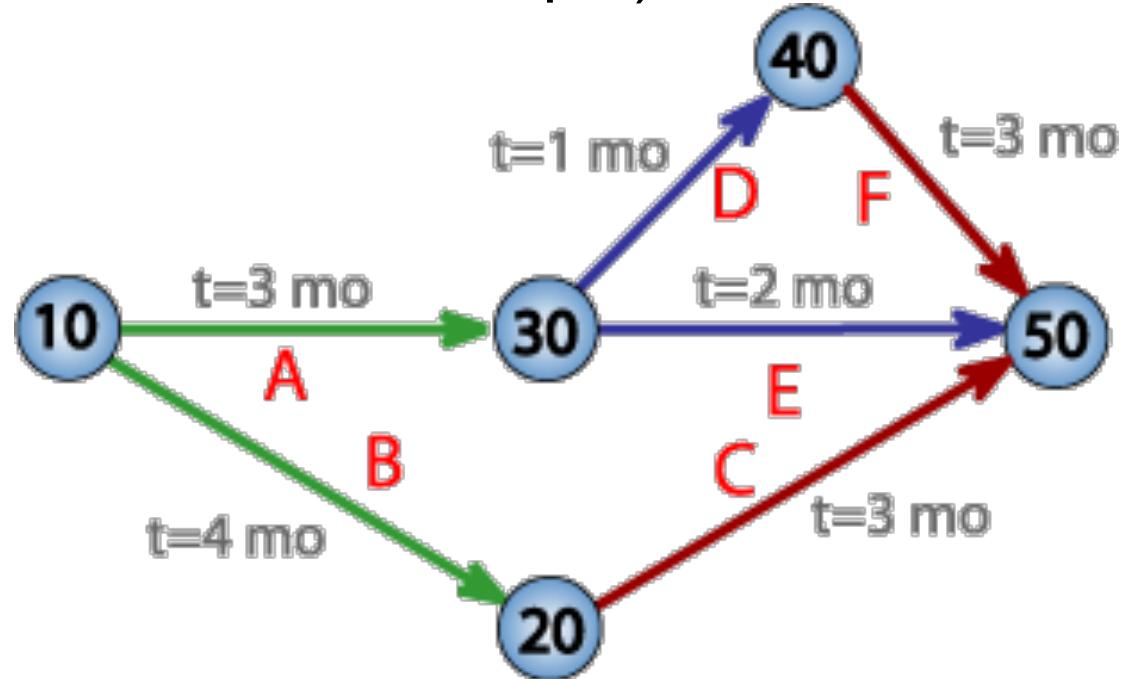
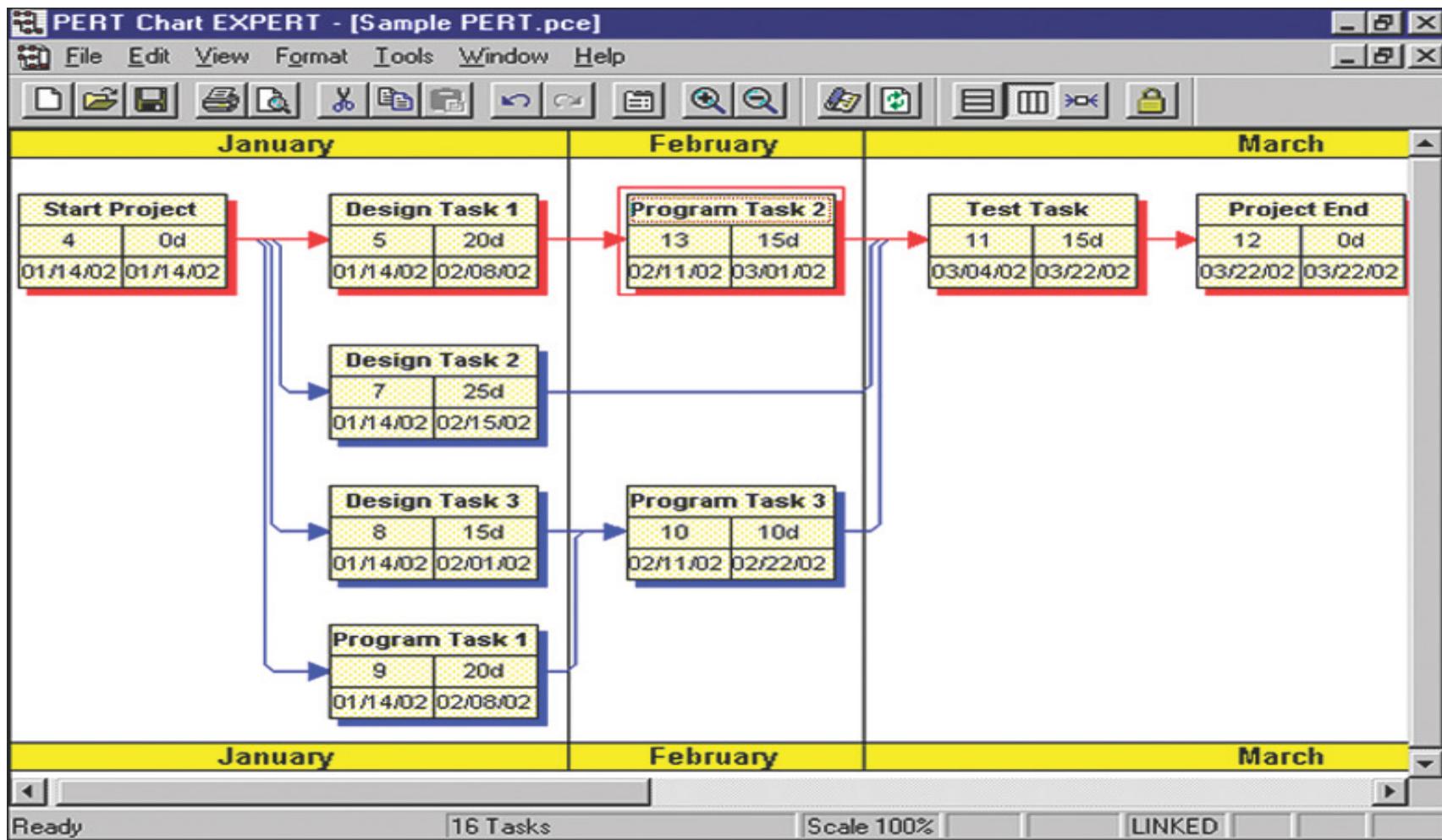


Grafico CPM per un progetto con 6 attività (sugli archi, etichettate da A a F) e 5 milestones (sui nodi, etichettate da 10 a 50).

Quanto deve durare come minimo il progetto?

# Esempio: strumento PERT



# 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?

# Discussione

La produttività è misurata dalla quantità di lavoro «utile» fatto in un certo periodo di tempo  
Come possiamo stimare o misurare la produttività nello sviluppo software?



# Discussione

Cos' è una stima?



# Stimare (Il problema di Fermi)

Esempio 1

Quanti accordatori di piano ci sono a Bologna?

# Il problema di Enrico Fermi

Quanti accordatori di piano ci sono a Bologna?

- Fatto: ci sono circa 400.000 persone a Bologna
- Stima: mediamente due persone per casa
- Stima: una casa su 20 ha un piano che viene accordato una volta l'anno
- Fatto: accordare un piano richiede circa 2h
- Ogni accordatore lavora 8h/g, 5gg/s, 50s/anno = 1000 piani accordati

Da queste assunzioni segue che a Bologna ci dovrebbero essere circa 10 accordatori di piano

# Esempio 2

Concerto di Springsteen, SanSiro 2016

- Quante birre sono state vendute nello stadio?
- Quali sono i principali fattori da considerare nella stima?
- Come fareste benchmark, ovvero come potreste validare la vostra stima e perché?

# Esempio 3: mousepad

## Caso 1

- Grafica: 500€ fissi
- Setup: 0.25 per unità
- Consegna: 0.01 per unità
- Materiali: 1.5 per unità
- Gomma: 0.15 per unità
- Pacchetto: 0.25 per unità

## Caso 2

- Grafica: 500€ fissi
- Setup: 0.25 per unità
- Consegna: 0.01 per unità
- Materiali: 2.5 per unità
- Gomma: 0.25 per unità
- Pacchetto: 0.35 per unità

Qual è la stima di costo medio per unità nei due casi?

In analogia, si può definire un costo medio per linea di software?

# Costo per Linea di Codice (LoC)

- Il costo **medio** di una LoC si calcola a consuntivo come 
$$\frac{Costo\ Total}{\#LoC}$$
- Il costo **marginale** è l'incremento del costo totale attribuibile ad una singola LoC in più.

**Secondo voi la  
stima del costo  
marginale di una  
LoC è costante?**



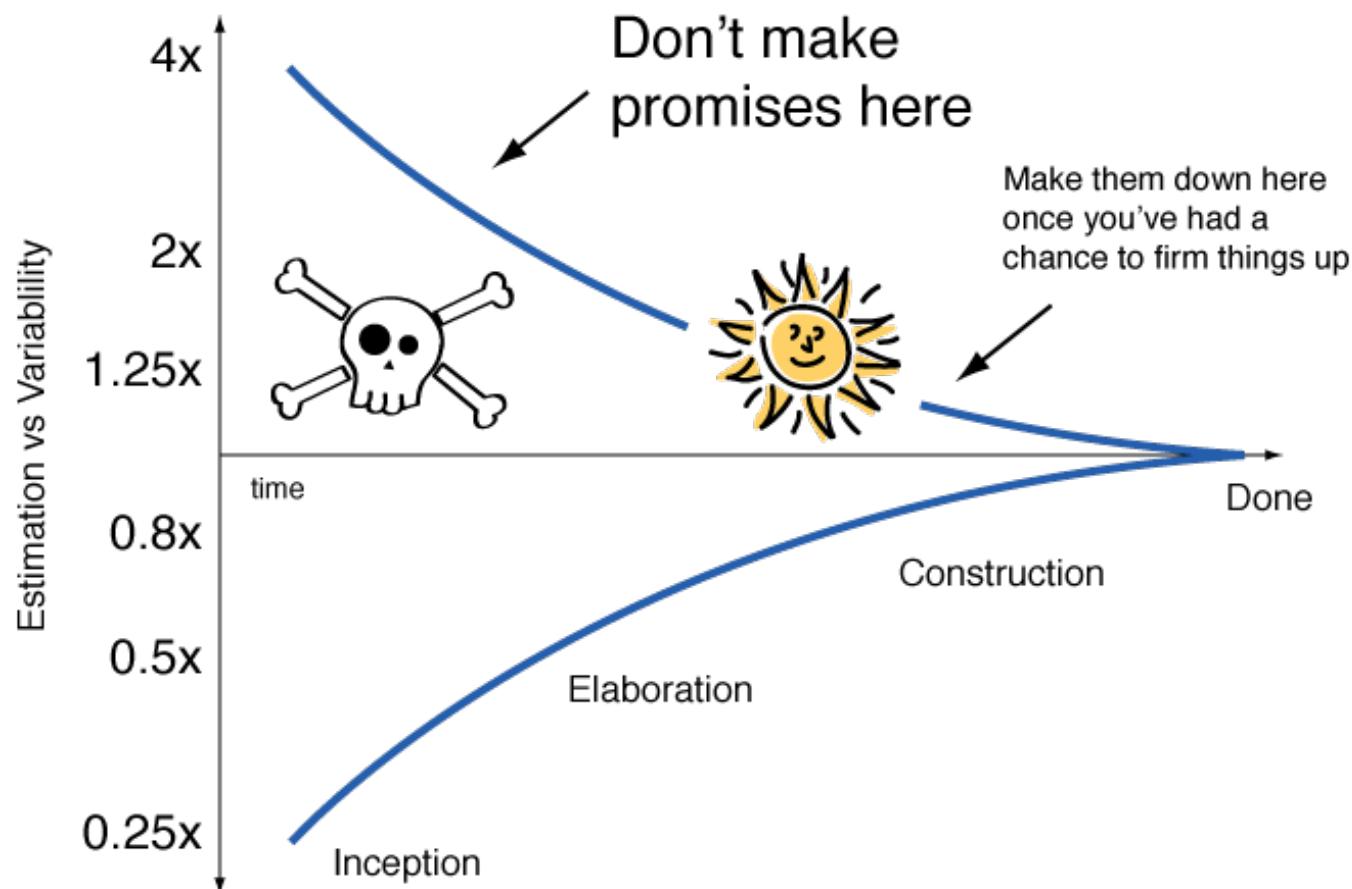
# Costo marginale per LOC

Il **costo marginale** per LOC **non è costante**:

infatti all'aumentare della dimensione in LOC

- la complessità del design del software aumenta
- occorre fare più attività (es. più testing)
- occorre più personale e aumenta la relativa difficoltà di gestione delle persone
- all'aumentare del personale aumentano le comunicazioni necessarie tra le persone coinvolte nel progetto, quindi aumentano i tempi di sviluppo

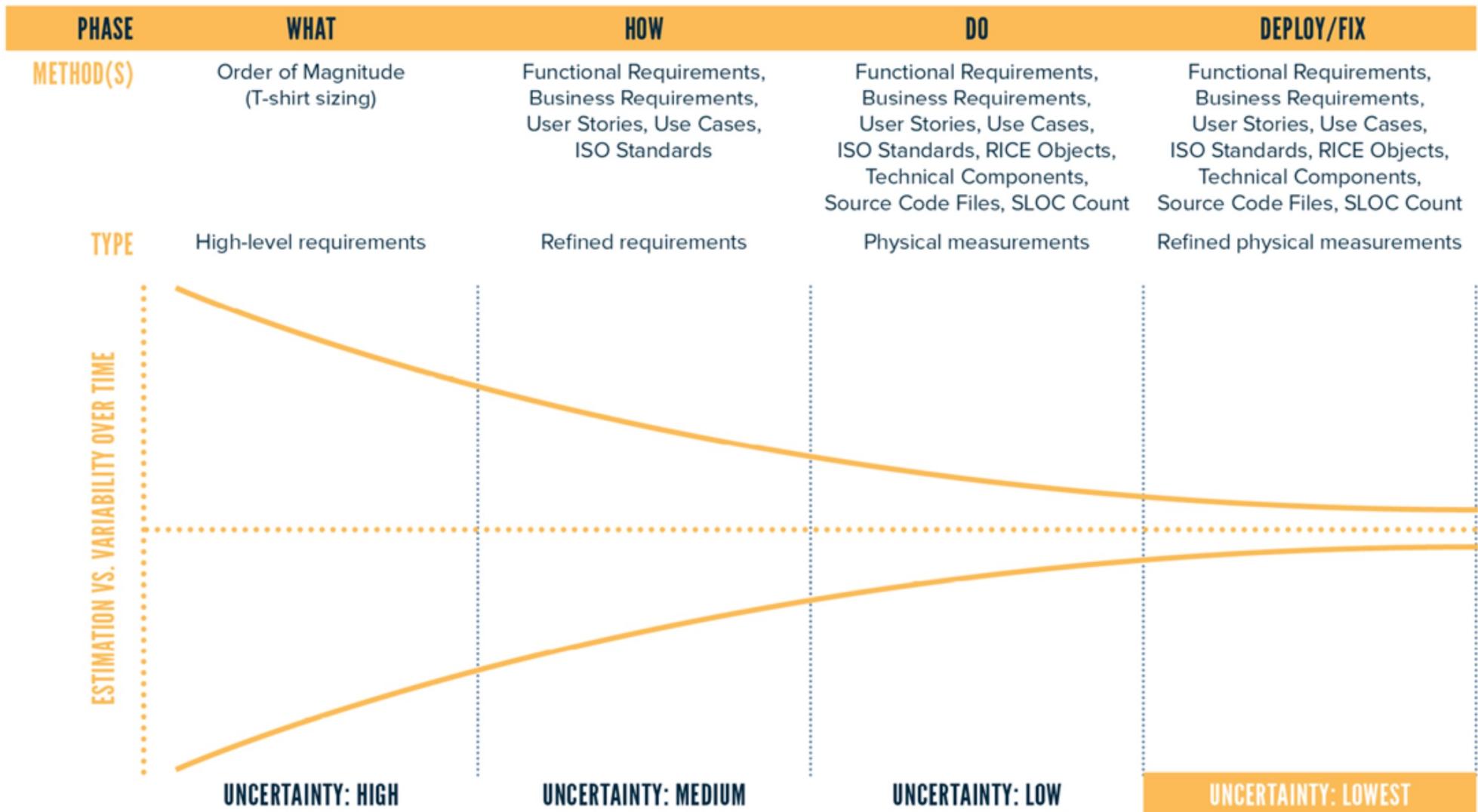
# Il cono dell'incertezza nello sviluppo sw



# L'incertezza nel sizing

fonte:QSM

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



# 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* di un team che ha prodotto un sw di dimensione *size* si calcola *effort/size(sw)*

# 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 diseguaglianza:

$$\text{Valore}(\text{versione}) >> \text{Costo}(\text{versione}) + \text{Debito tecnico}(\text{versione})$$

# 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

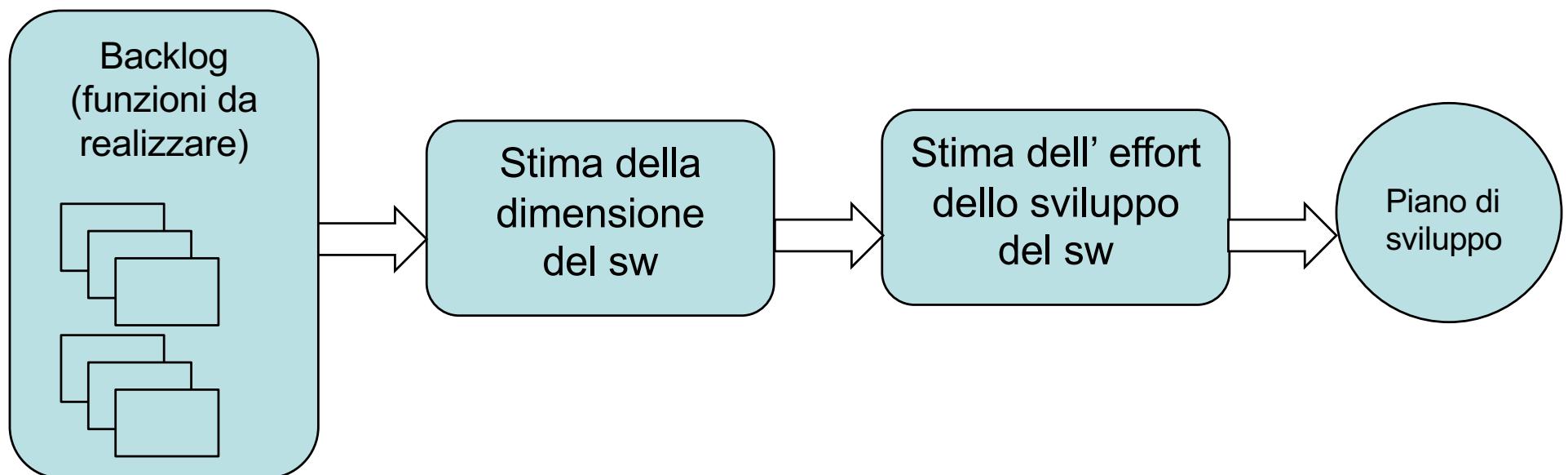
**Nota 2:** il cfu (=25h/studente) è una unità di misura dello sforzo (diversa dal mese/persona, che non è definito univocamente, ma che per esempio in COCOMO è pari a 152 h/persona)

# 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:  $\text{LoC} = \text{NCLoC} + \text{CLoC}$ , 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

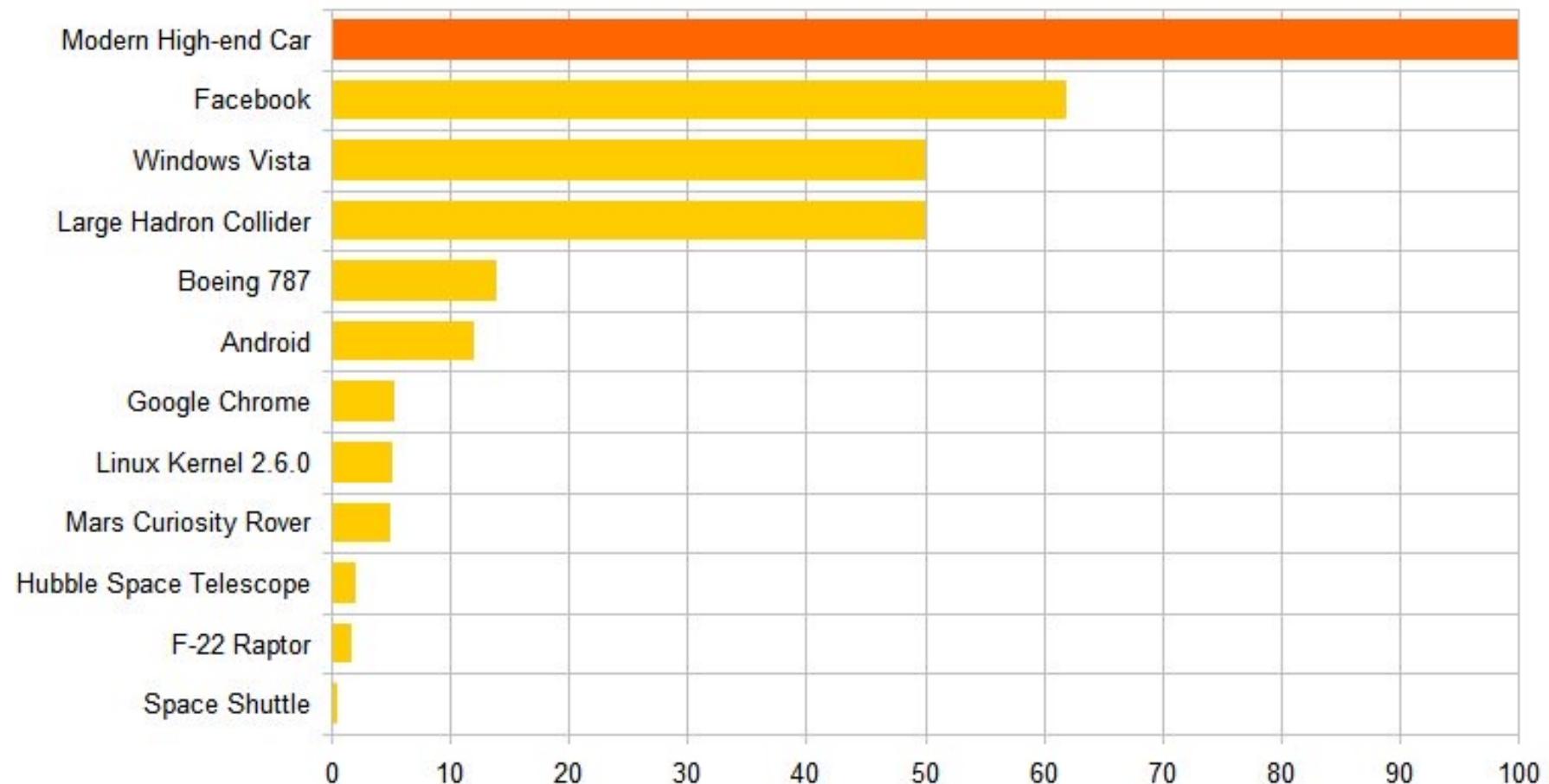
```
for ( i=0; i<100; ++i)  
{  
printf("hello");  
}  
/* Now how many lines of code is this? */
```

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

### Software Size (million Lines of Code)



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

# 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
Total 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

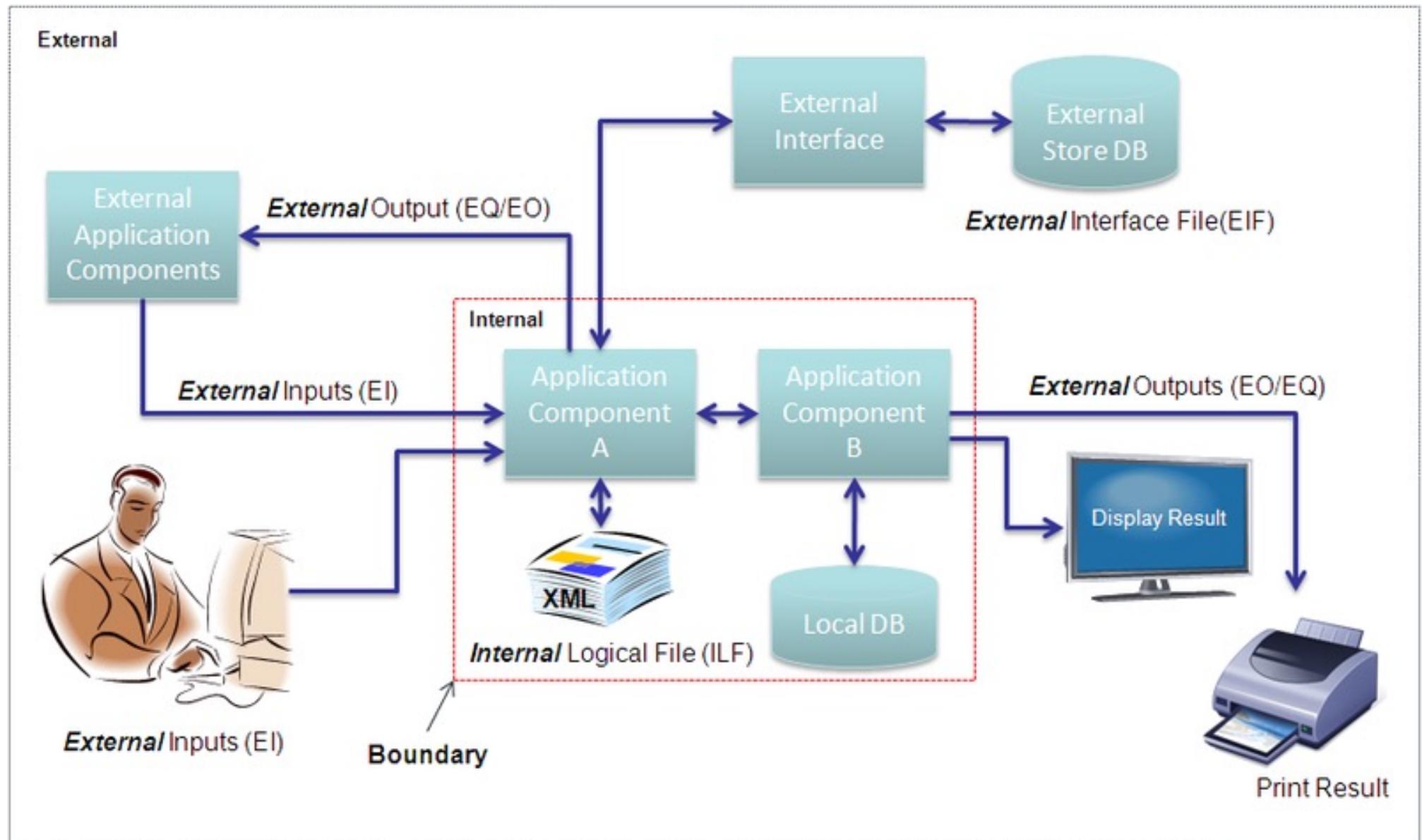
1. È difficile stimare la dimensione in LOC di una nuova applicazione
2. La stima LOC non tiene conto della diversa complessità e potenza delle istruzioni (di uno stesso linguaggio o di linguaggi diversi)
3. È difficile definire in modo preciso il criterio di conteggio (istruzioni spezzate su più righe, 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**

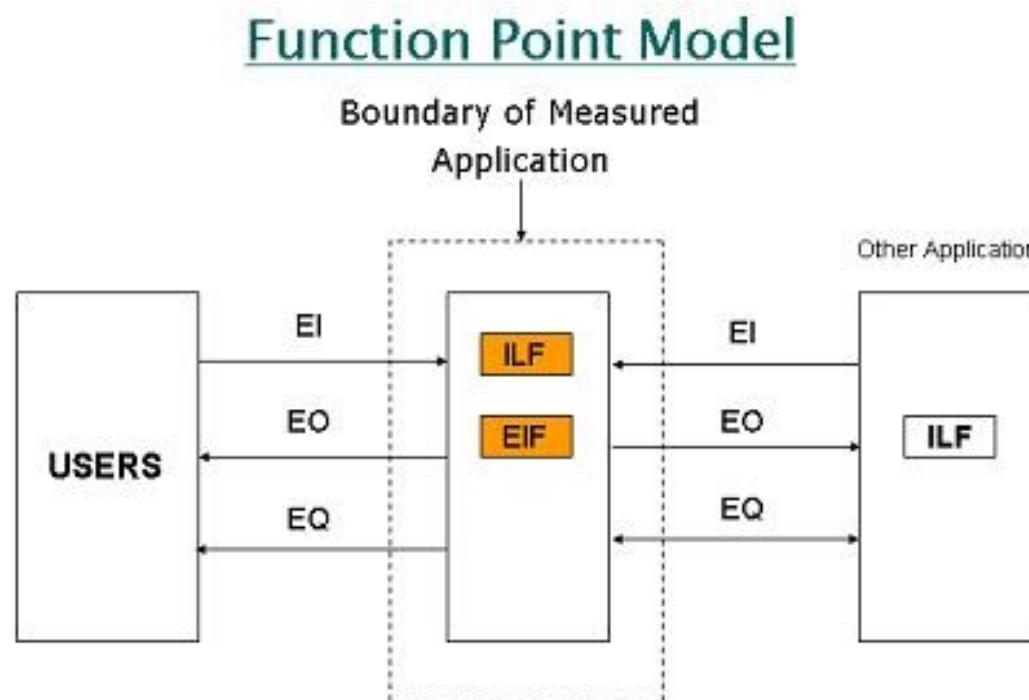
*“The original objective of the Function Points work was to define a measure [that would] help managers analyze application development and maintenance work and highlight productivity improvement opportunities.”*

*“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.”*

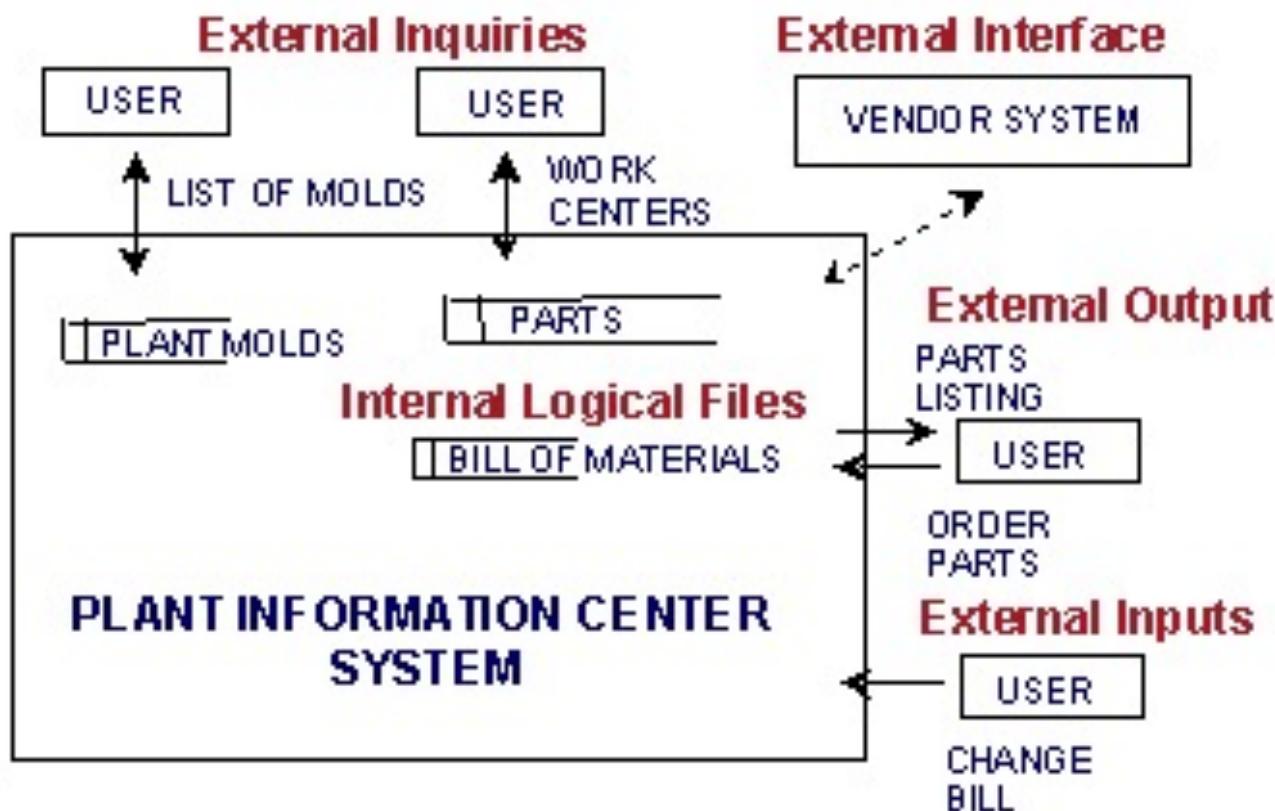


# Calcolo di FP: passo 1

Si descrive l'applicazione da realizzare  
Si parte da una **descrizione** del sistema  
**(specifica)** di natura **funzionale**



# Esempio: sistema informativo d'impianto

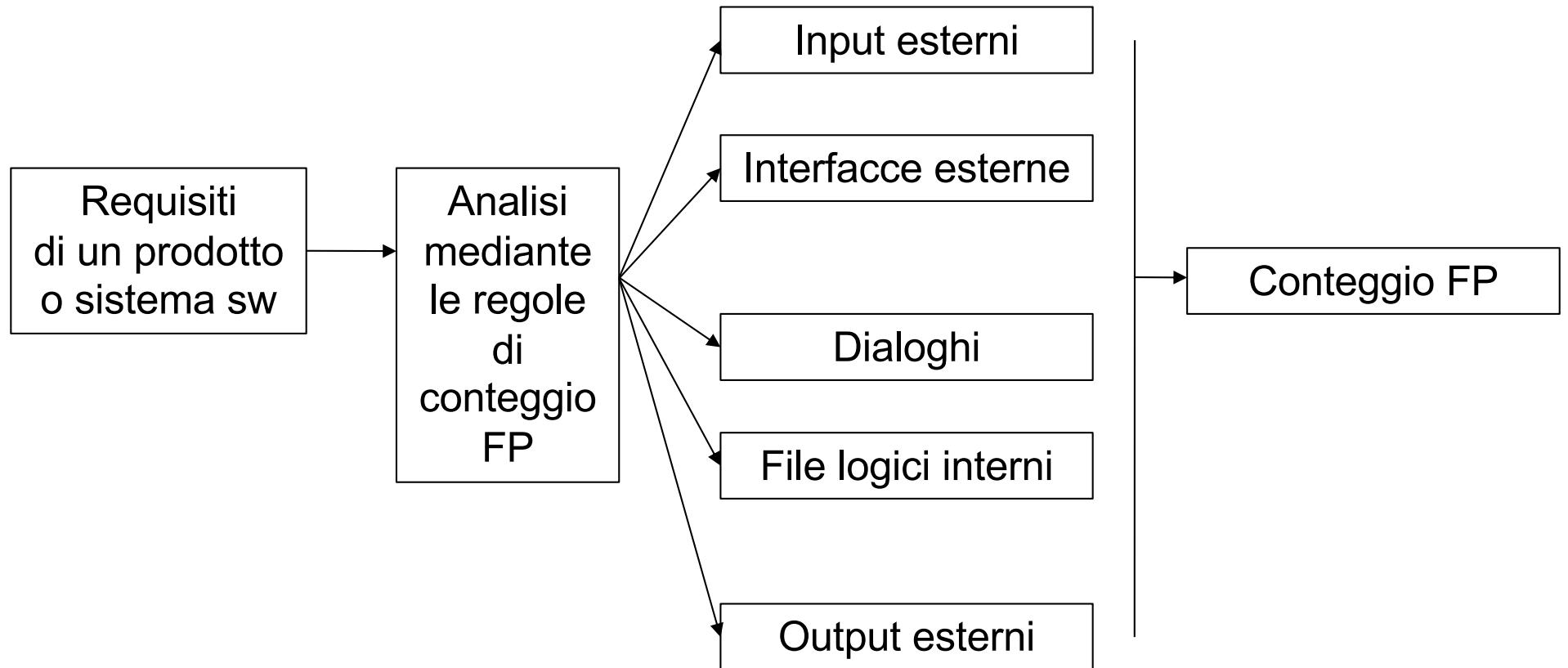


# Calcolo di FP: passo 2

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

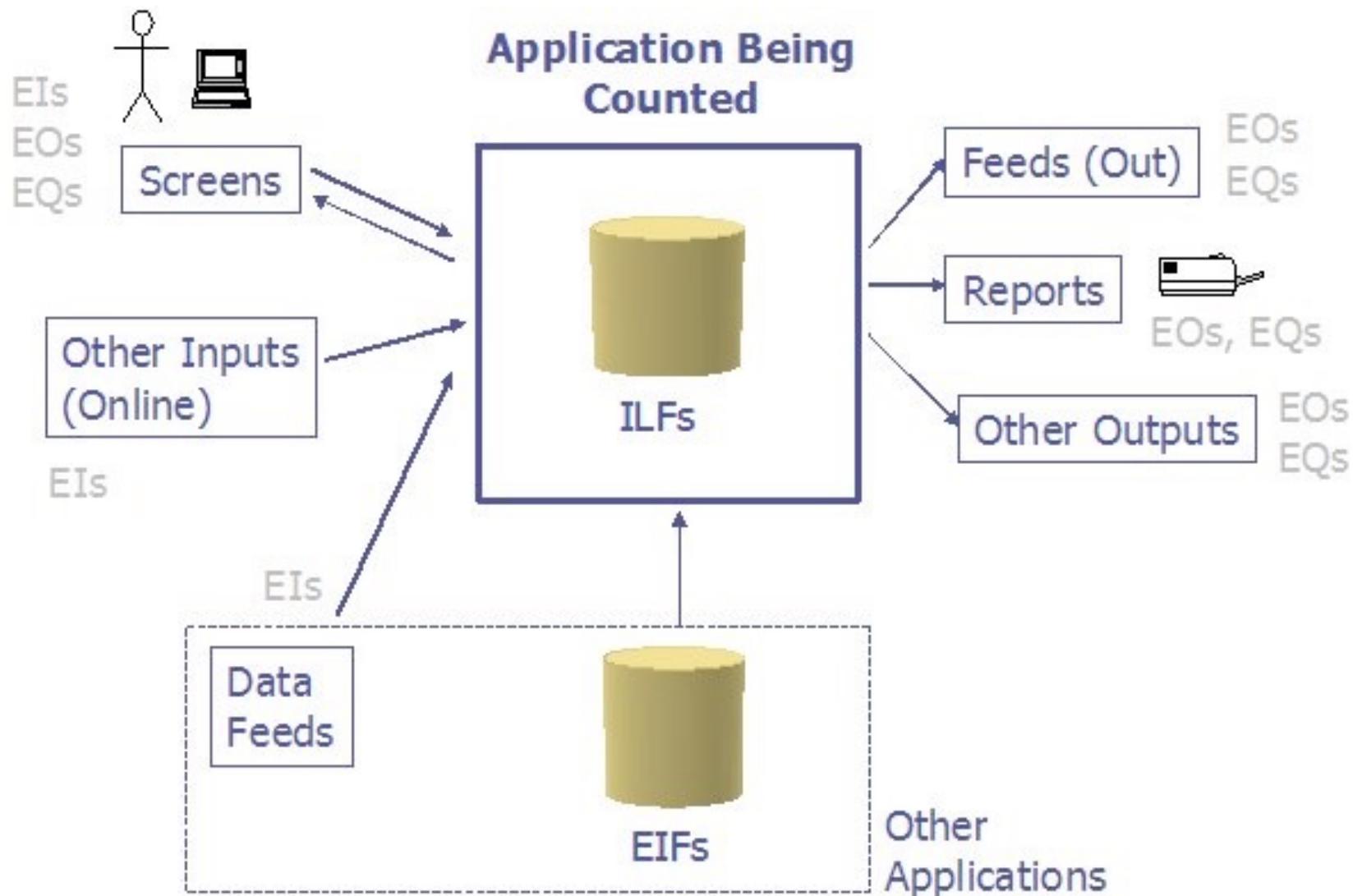
# Il metodo



# 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

# Cosa “conta” chi usa i FP



# 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:

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

Dove i = tipo elemento

j = complessità (semplice o media o complessa)

UFC = Unadjusted Function Count

# Calcolo di FP: passo 5

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

Calcolo di TFC:

$$\text{TFC} = 0.65 + 0.01 * \sum_{[i=1-14]} F_i$$

Ciascun fattore  $F_i$  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:

$$FP = UFC * TFC$$

# 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)

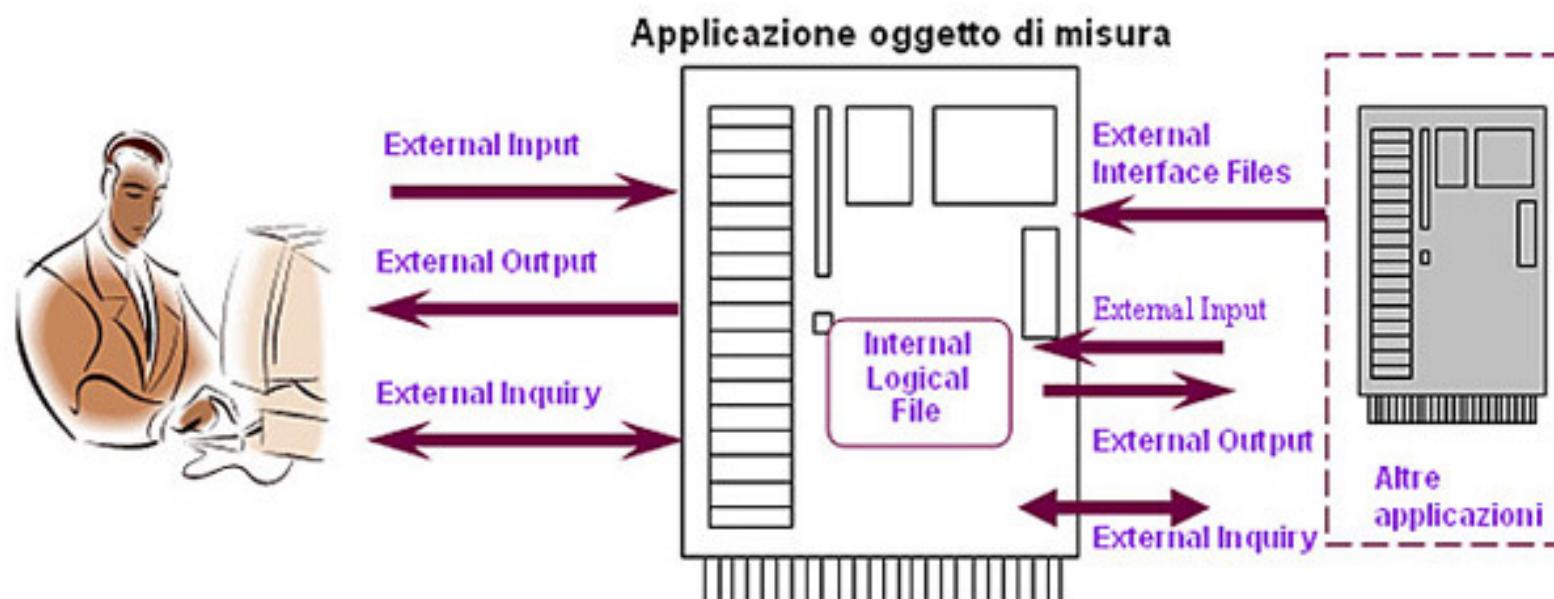
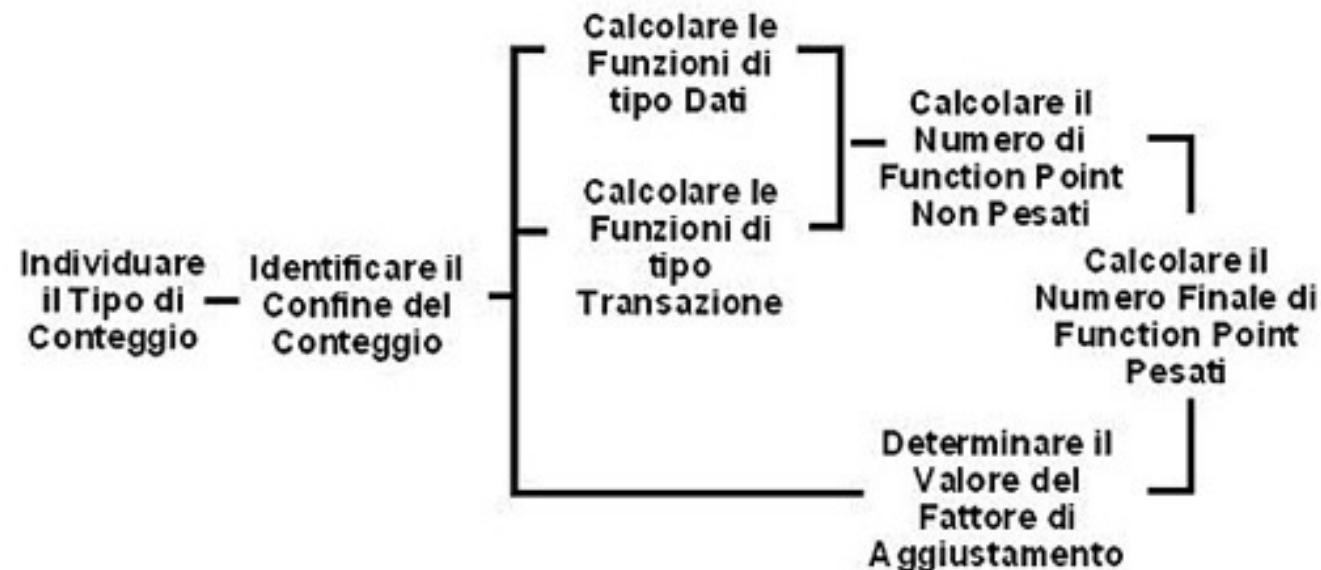
Se si sa, per esempio a causa di esperienze passate, che il numero medio di FP per mese-persona è pari a 18, allora si può fare la stima che segue:

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

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



# Calibrazione

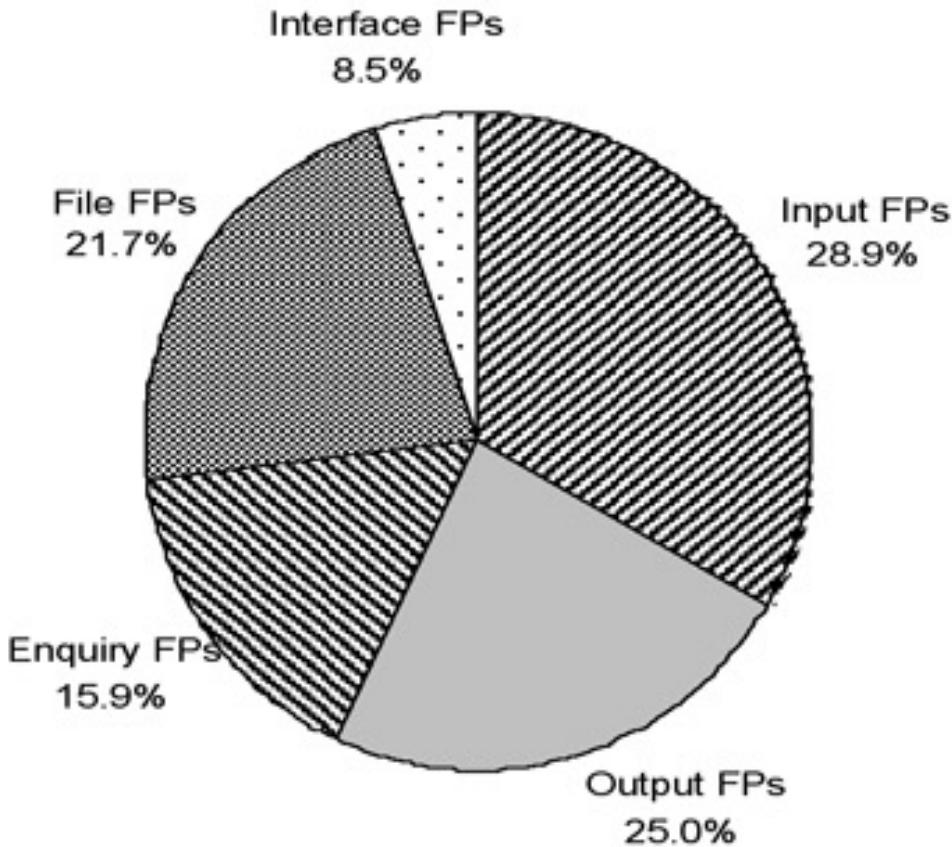
- Il conteggio dei FP si basa su giudizi **soggettivi**, quindi persone diverse possono raggiungere risultati diversi
- Quando si introduce l'Analisi FP (in sigla: FPA) in una organizzazione, è necessaria una fase di **calibrazione**, usando i progetti sviluppati in passato come base del sistema di conteggio

# Misurazioni di alcuni sistemi

(Capers Jones 2010)

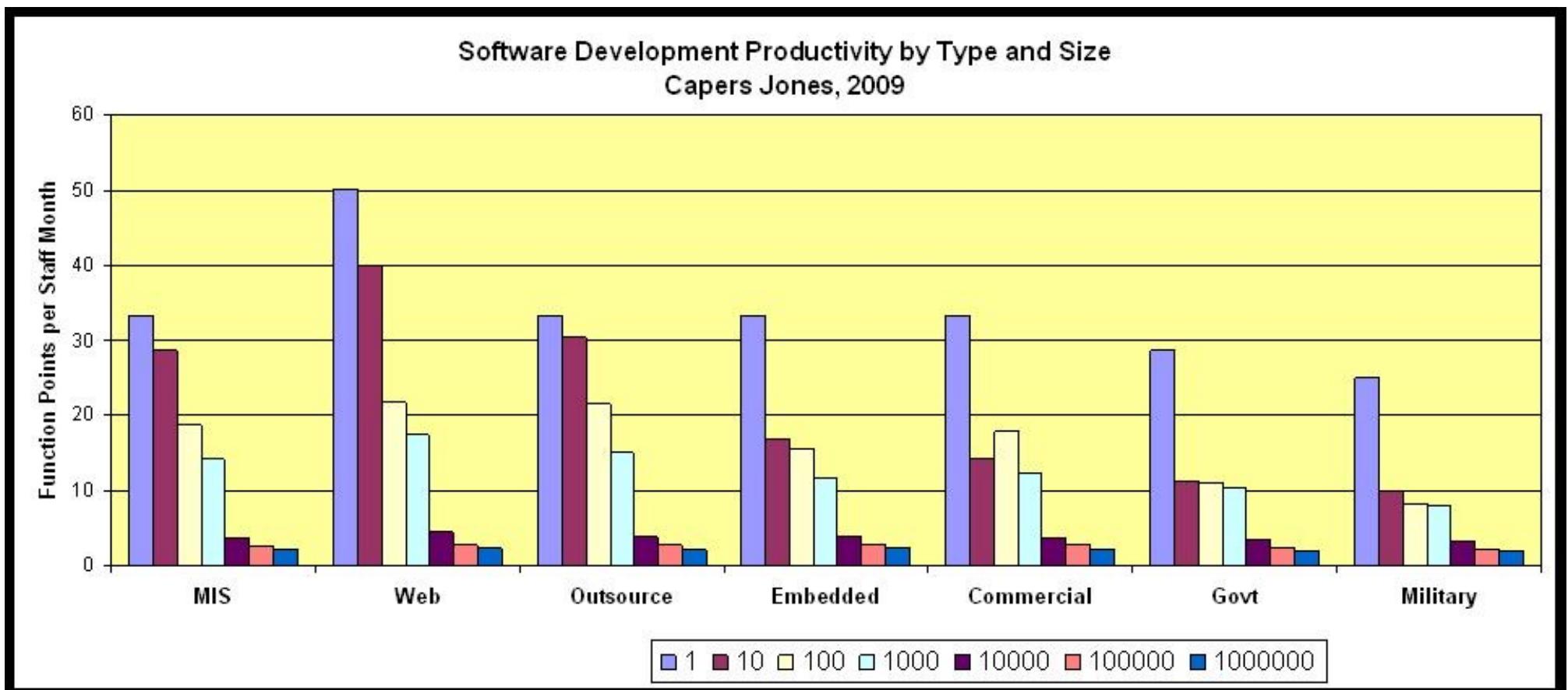
Sistema	Size in FP	KLOC	LOC/FP
US Air traffic control	306.324	65.000	213
Israeli Air Defense	300.655	24.000	80
SAP	296.764	24.000	80
Iran air defense	260.100	24.000	91
MS Vista	157.658	10.090	64
MS Office Pro	93.500	6.000	64
Iphone IOS	19.300	516	27
Google search	18.640	1.193	64
Amazon Website	18.080	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 ai requisiti 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**:
    - $E = A + B \cdot S^C$   
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:

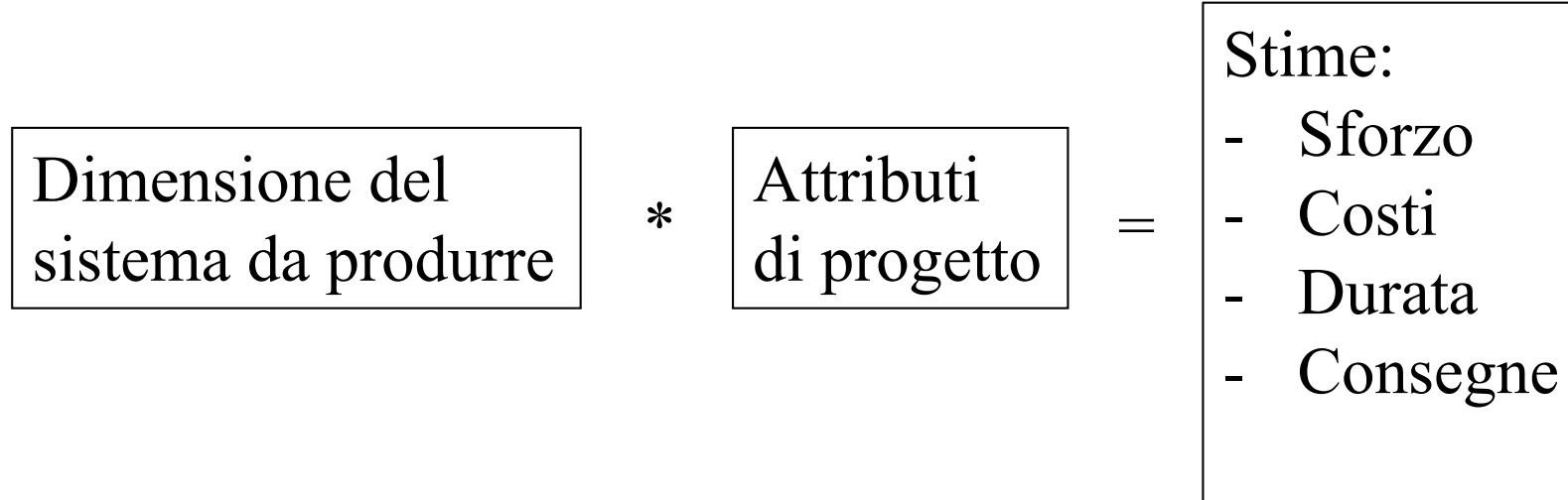
- incapacità di dimensionare accuratamente un progetto;
- incapacità di definire nel suo complesso il processo e l'ambiente operativo di un progetto;
- valutazione inadeguata del personale, in quantità e qualità;
- mancanza di requisiti di qualità per la stima di attività specifiche nell'ambito del progetto

# Modelli dei costi del software

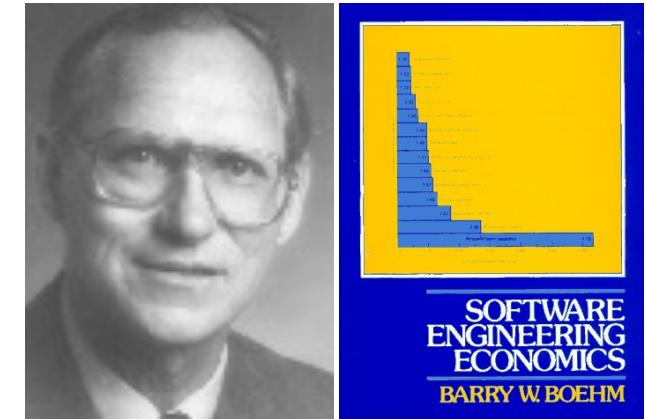
## Esempi di modelli commerciali

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

# Uso di parametri per le stime



# Modelli di costi sw: COCOMO



- Il **Constructive Cost Model (COCOMO)** è uno dei modelli parametrici più diffusi per fare stime nei progetti software
- COCOMO 1 è descritto nel libro *Software Engineering Economics* di Barry Boehm, 1981
- 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

# Modelli di costi sw: COCOMO

- Il principale calcolo di COCOMO si basa sull' Equazione dello Sforzo per stimare il numero di mesi-persona necessari per un progetto  
$$\# \text{ mp} * \text{costo\_lavoro} = \text{Costo\_Stimato}$$
- La maggior parte delle altre grandezze stimate (la durata, la dimensione del personale, ecc.) vengono poi derivate da questa equazione

# Aspetti economici del software

**Performance** = **(Complexity)** <sup>(Process)</sup> \* **(Team)** \* **(Tools)**

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

# Constructive Cost Model

- 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)

# CoCoMo81

- I tre modelli hanno forma equazionale:

$$\text{Effort} = a \cdot S^b \cdot EAF$$

- 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 COCOMO81

- **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<sup>b</sup>

**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 azzetta i valori reali con approssimazione  $\pm 20\%$  circa il 68% delle volte
- (Modello più usato con COCOMO 1)

# Fattori di costo (COCOMO81, Intermediate)

<b>Cost Drivers</b>	<b>Rating</b>					
	<b>Very Low</b>	<b>Low</b>	<b>Nominal</b>	<b>High</b>	<b>Very High</b>	<b>Extra High</b>
Product attributes						
Required software reliability	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<sup>b</sup> \* 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      91

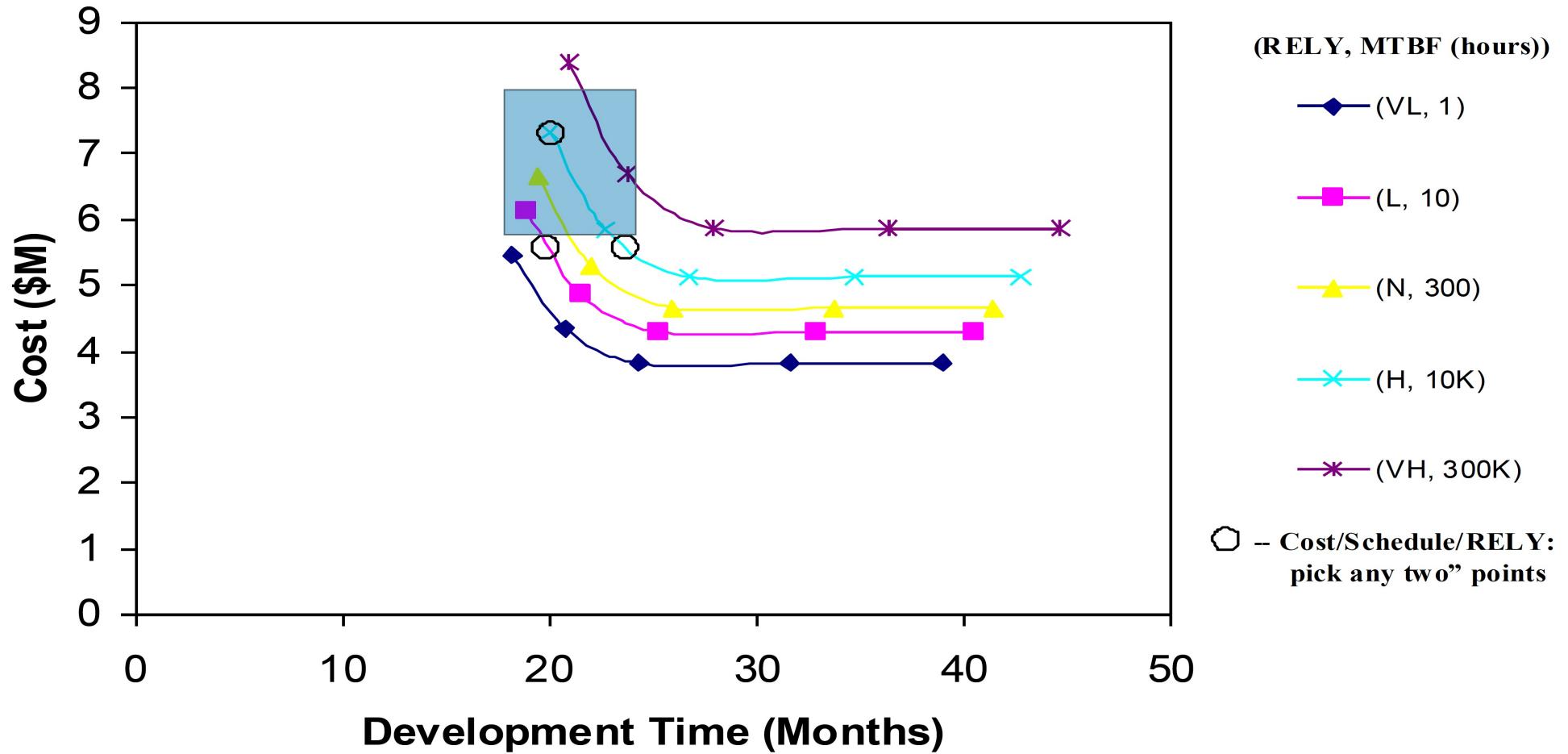
# Esempio 2

- 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
Virtual 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 =  $2.8 \times (10)^{1.20} = 44$  mp
- Moltiplicatori di effort = 1.35
- Sforzo stimato reale  $44 \times 1.35 = 59$  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.

# Stime coi metodi agili

- Anche i **metodi agili** devono stimare lo sforzo necessario per le users' stories
- Tecnica del **planning poker**: i membri del team eseguono stime indipendenti su più turni; ad ogni turno si può cambiare la stima sulla base dei risultati del turno precedente, cercando il consenso (**metodo Delphi**)

[https://it.wikipedia.org/wiki/Metodo\\_Delphi](https://it.wikipedia.org/wiki/Metodo_Delphi)

# Esempio (I)

- User story: “qualcuno vuole gestire ordini d’acquisto usando un carrello elettronico”
- Team di sviluppo con 7 membri
- Primo turno di stime (valori history points):



# Esempio (II)



- I partecipanti 3 e 6 spiegano le loro stime
- Secondo turno:

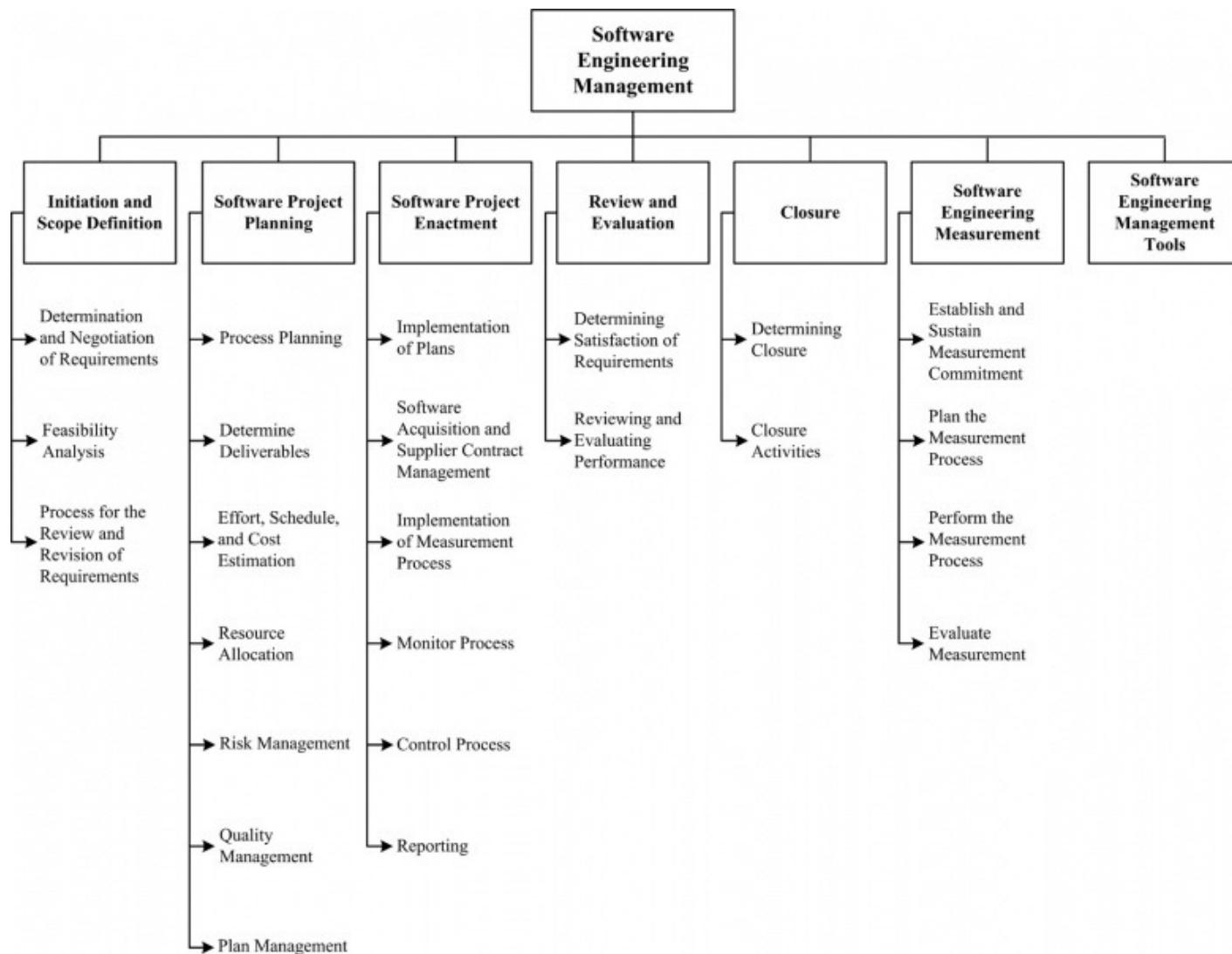


# Esempio (III)



- Tutte le stime convergono eccetto la terza:
  - Altro turno, oppure
  - Usare 3 o 5 come stima

# 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](http://www.pmi.org)
- [www.mindtools.com/pages/article/newPPM\\_01.htm](http://www.mindtools.com/pages/article/newPPM_01.htm)
- [sunset.usc.edu/csse/research/COCOMOII/cocomo\\_main.html](http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html)
- [www.ifpug.org](http://www.ifpug.org) (**associazione FP**)
- [www.gufpi-isma.org](http://www.gufpi-isma.org)
- [www.dwheeler.com/sloc/](http://www.dwheeler.com/sloc/)
- [www.devdaily.com/FunctionPoints/FunctionPoints.shtml](http://www.devdaily.com/FunctionPoints/FunctionPoints.shtml)
- [www.cosmic.com](http://www.cosmic.com)
- [brodzinski.com](http://brodzinski.com) (**blog su sw project management**)
- [gispict.wikispaces.com/Analisi+dei+costi+software](http://gispict.wikispaces.com/Analisi+dei+costi+software)
- [alvinalexander.com/FunctionPoints/](http://alvinalexander.com/FunctionPoints/) (**esempio FP**)
- [conferences.embarcadero.com/article/32094#Eqs](http://conferences.embarcadero.com/article/32094#Eqs) (**stesso esempio, FP**)

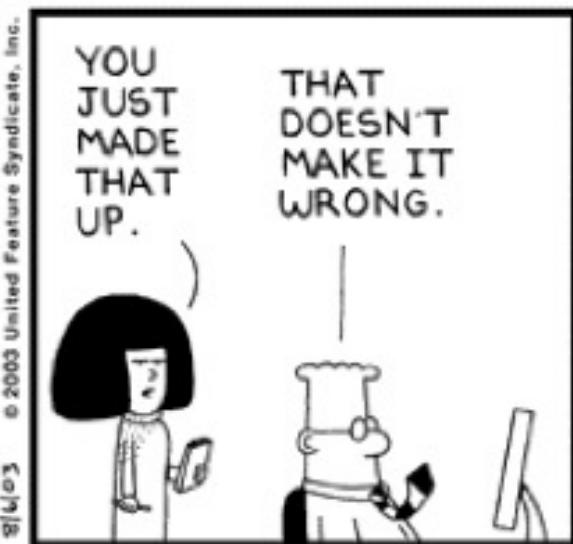
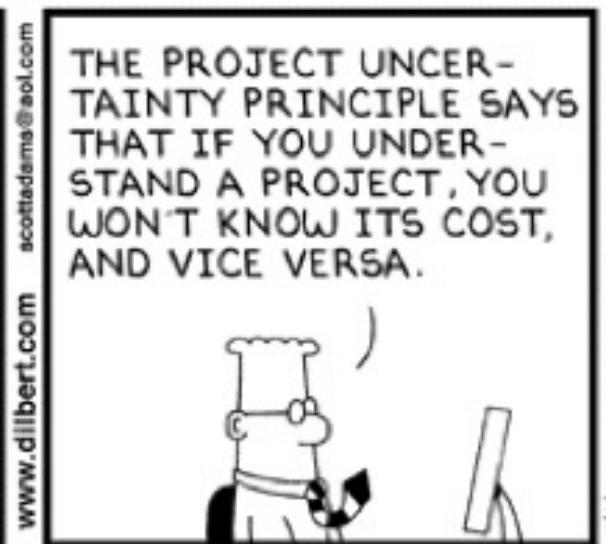
# Strumenti

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

# Pubblicazioni scientifiche

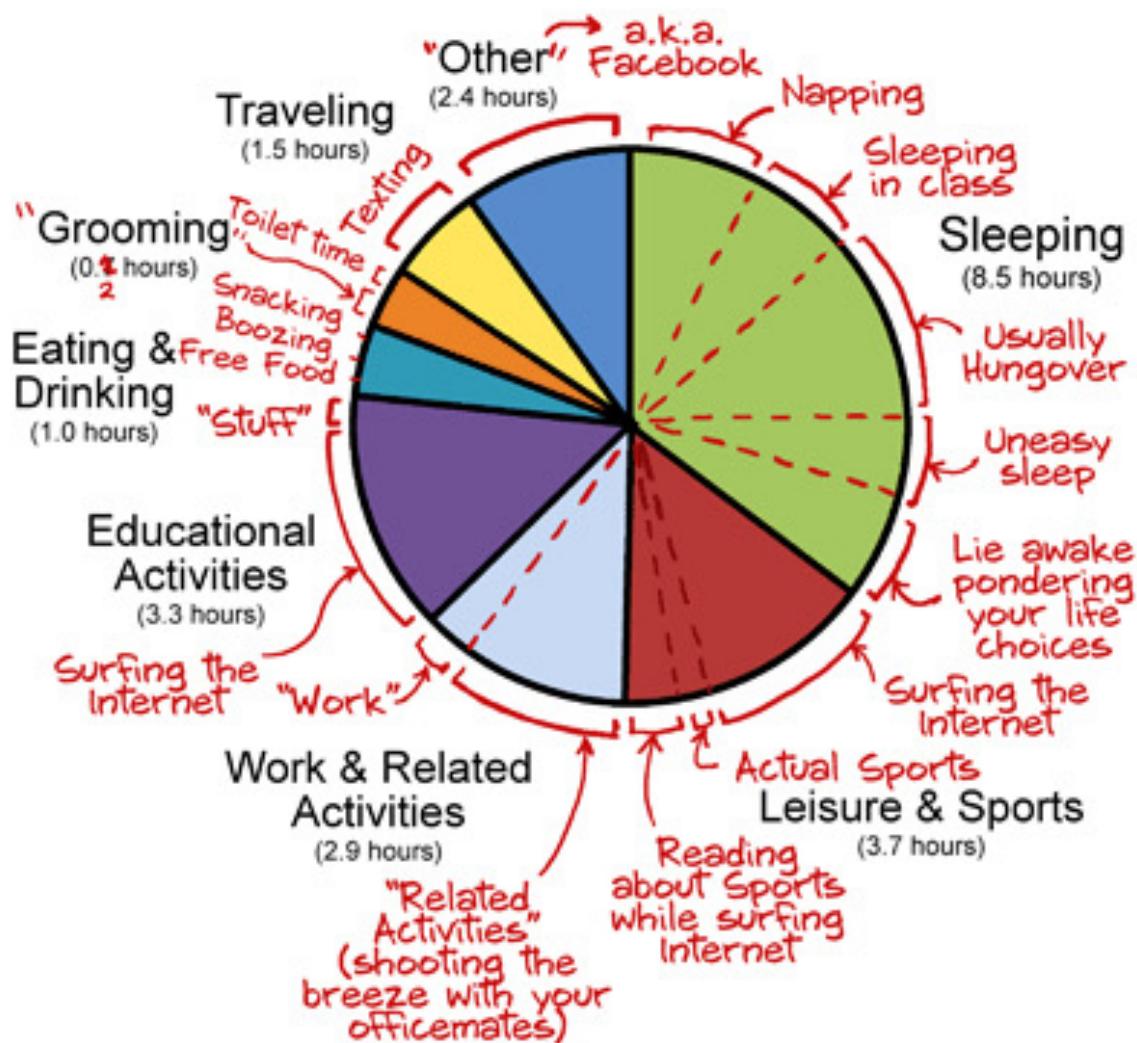
- IEEE International Software Metrics Symposium
- IEEE Int. Conf on Empirical Software Engineering and Measurement
- Software Quality Journal

# Domande?



© 2003 United Feature Syndicate, Inc.

# How you spend your time:



SOURCE: Bureau of Labor Statistics, American Time Use Survey of Full Time University Students.