

# Ingegneria del Software



*Prof. Paolo Ciancarini  
Corso di Ingegneria del Software  
CdL Informatica Università di Bologna*

# Obiettivi di questa lezione

- Cos'è l'ingegneria del software?
- Il ciclo di vita del software
- Il processo di sviluppo del software
- Miti e leggende della produzione sw

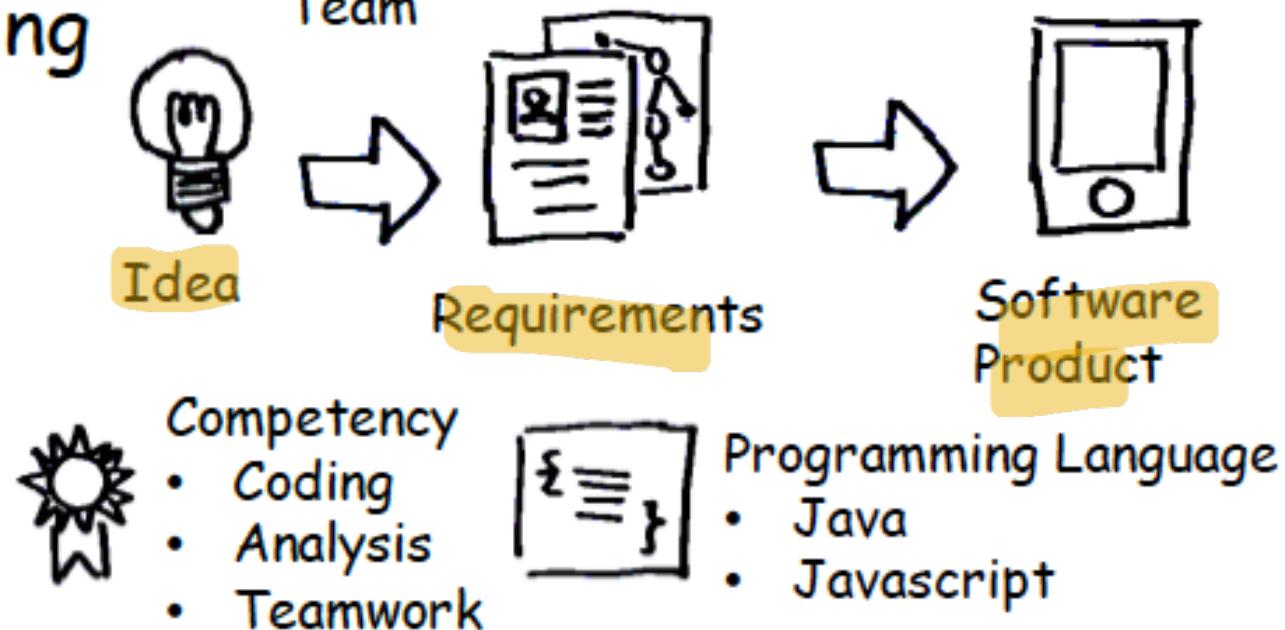
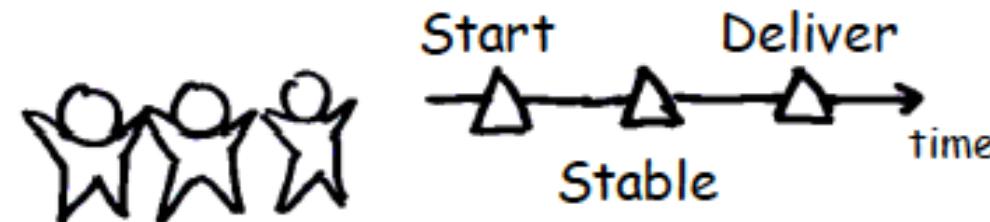
# From programming to sw engineering

Cosa è più difficile:

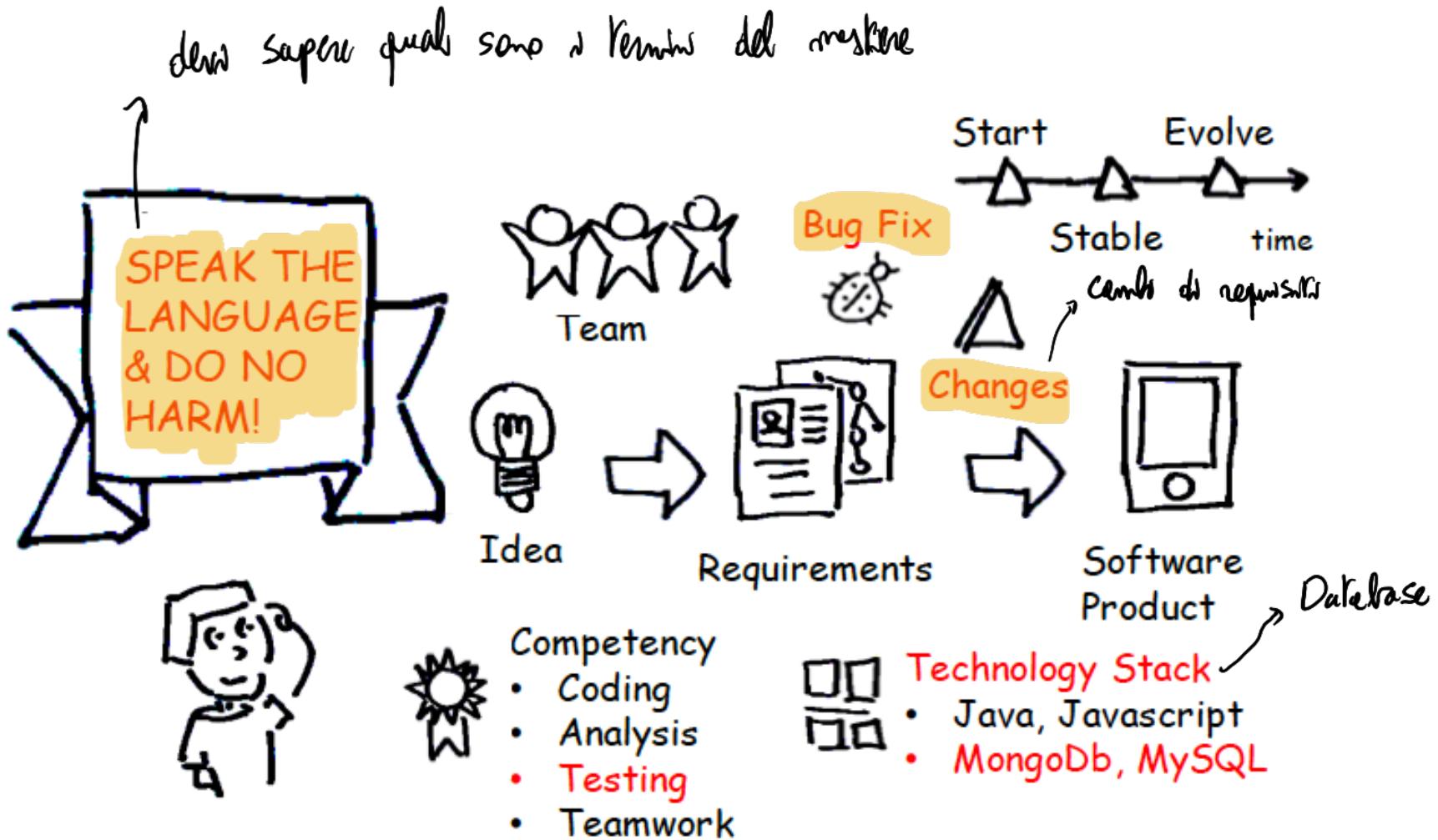
scrivere software, oppure  
leggerlo (per es. per modificarlo)?

# Punto di vista: studente

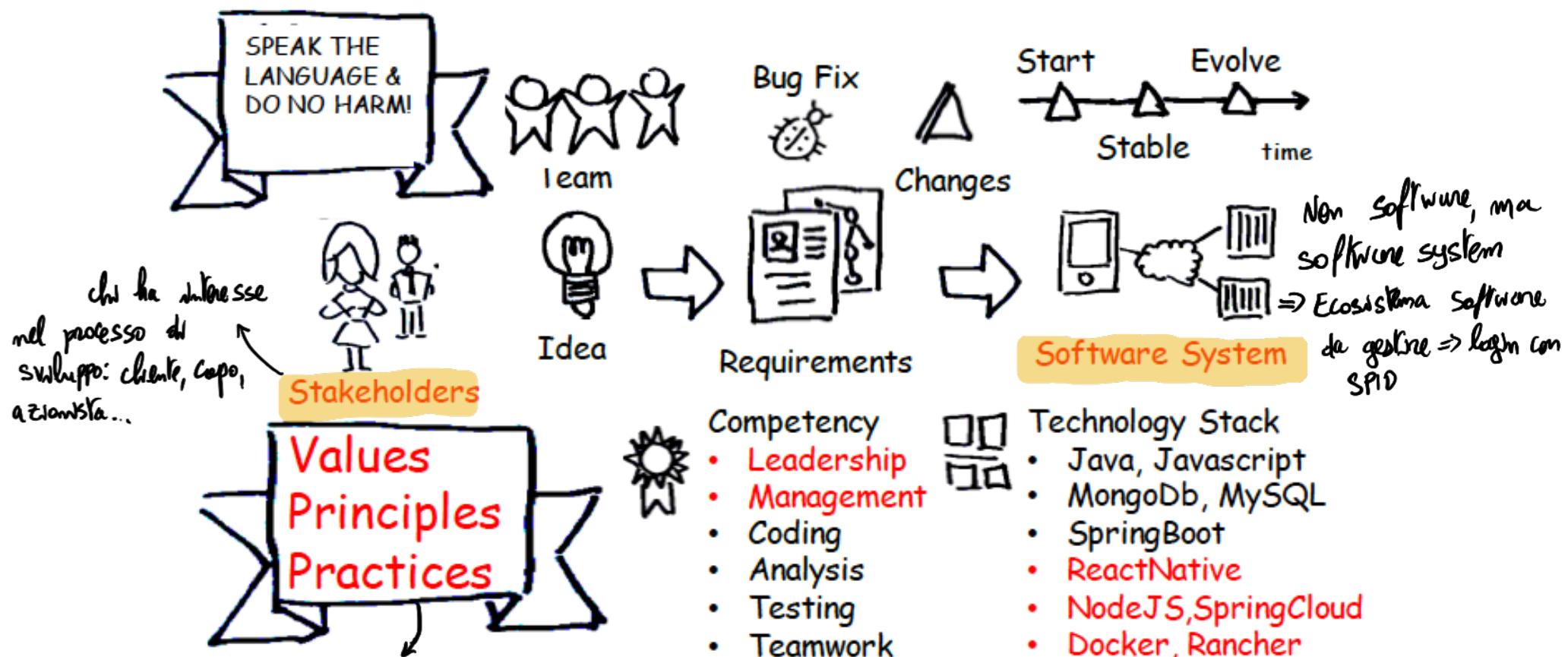
What I think  
software  
engineering  
is about



# Punto di vista: tesista in azienda



# Punto di vista: professionista



# Da Quora

(risposta di P Lovisek)

## What is something no one admits about being a sw engineer?

As a manager who is hiring and also leading mid-size team, I've observed this:

- A lot of colleagues tend to have unexpectedly huge gaps in knowledge. I would almost use the word “**dilettante**” for 30% of them. That itself is not a huge issue, as it's easy to fill gaps with some effort. Some of them however somehow survive in corporate environment with their “good enough” knowledge and carry their gaps for years and decades.
- The thing is, before 25, you learn just enough to **survive**, typically google+stackoverflow combo. In later years, you become lazier, than you've got kids and no time for in-depth study of your technologies and tools.
- There are also 2 related issues:
  - Companies are not likely to hire/pay more really extraordinary well-equipped guys. Mostly they just want someone average. Therefore, being super-deep in your topics actually can hurt you in hiring process.
  - **Things are evolving so fast no one knows any more what makes sense to learn.** Things will be different tomorrow. Old saying says, there's nothing older than yesterday's news. Well, there is: today's technology.

# Ingegneria del software

→ dà metodologia da seguire passo passo.

- L'Ingegneria del Sw (Software Engineering) è una *disciplina metodologica*, cioè studia i *metodi* di produzione, le *teorie* alla base dei metodi, e gli *strumenti* di sviluppo e misura della *qualità* dei sistemi software
- È anche una *disciplina empirica*, cioè basata sull'*esperienza* e sulla *storia* dei progetti passati

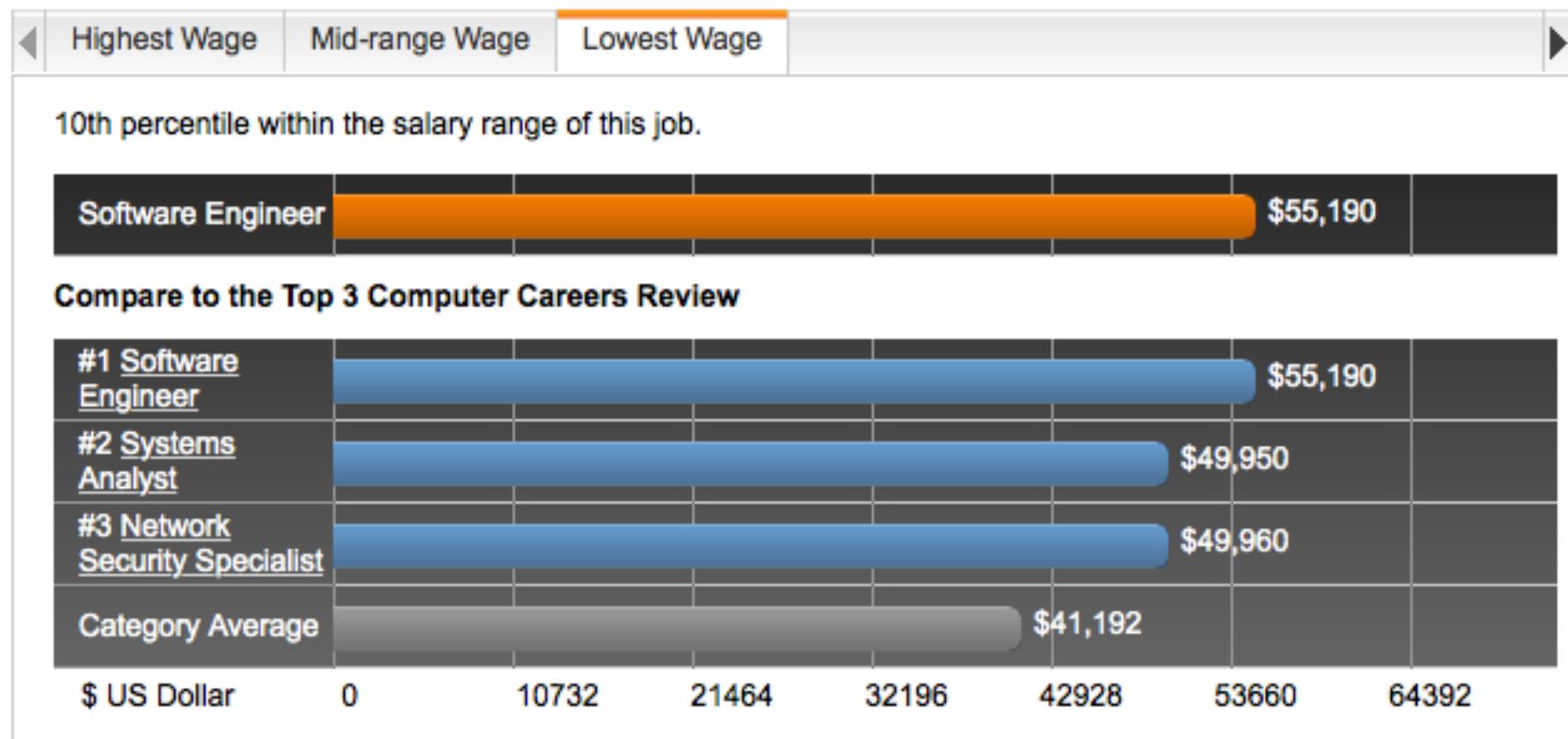
# Ingegnere del software

- Nei paesi anglosassoni “software engineer” è una professione riconosciuta
- Oltre metà di tutti gli ingegneri USA sono “sw engineers”

Fonte: [https://en.wikipedia.org/wiki/Software\\_engineering\\_demographics#United\\_States](https://en.wikipedia.org/wiki/Software_engineering_demographics#United_States)

<http://computer-careers-review.toptenreviews.com/software-engineer-review.html> dati al 2015  
<https://evansdata.com/reports/viewRelease.php?reportID=9> dati al 2018

# Confronto tra software engineers e altre professioni



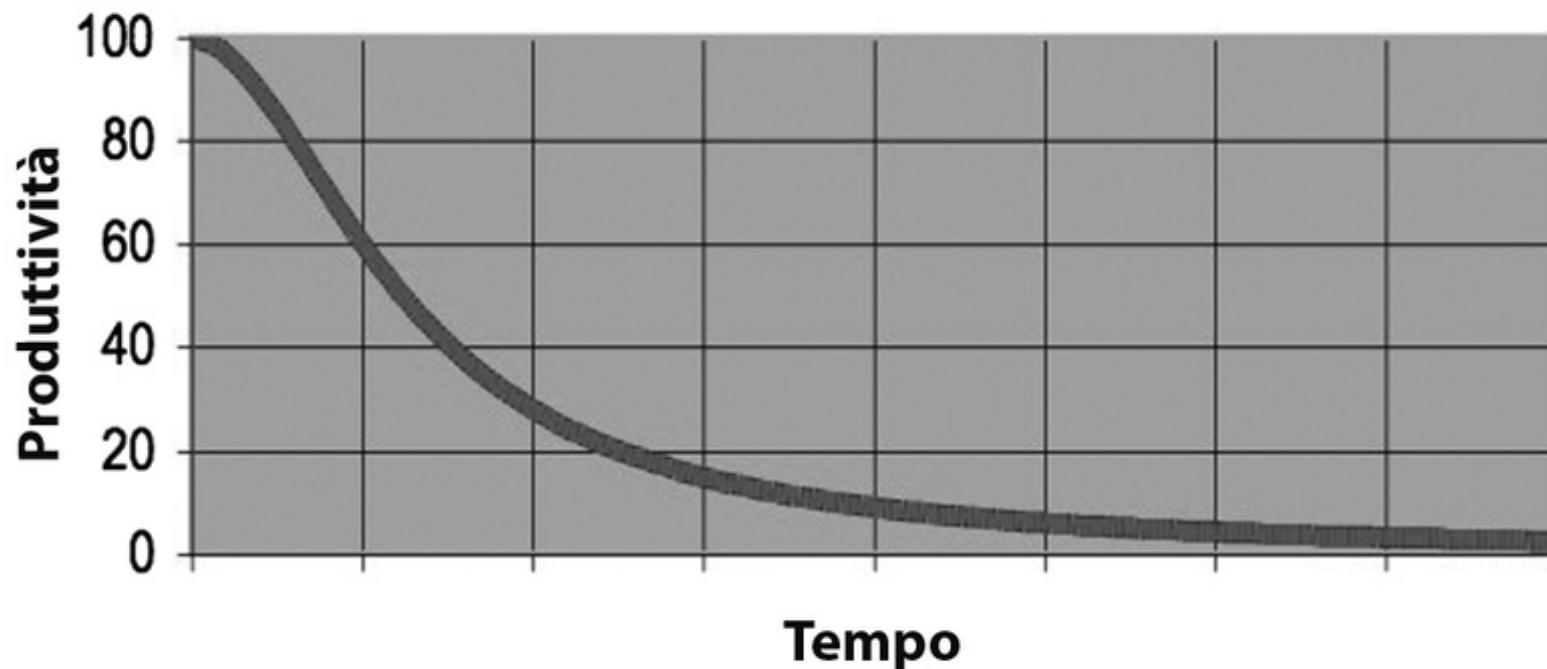
<https://money.usnews.com/careers/best-jobs/rankings/best-technology-jobs>

# Produttività

- La produttività è il rapporto tra la quantità di beni o servizi prodotti ed il costo del lavoro necessario a produrli
  - Output/Input
- La produttività dello sviluppo software è il rapporto tra software prodotto e il costo dello sforzo di produrlo
  - LOC/effort

mesi per uomo

# Produttività nel software



Più passa nel tempo meno ho bisogno di sviluppatore

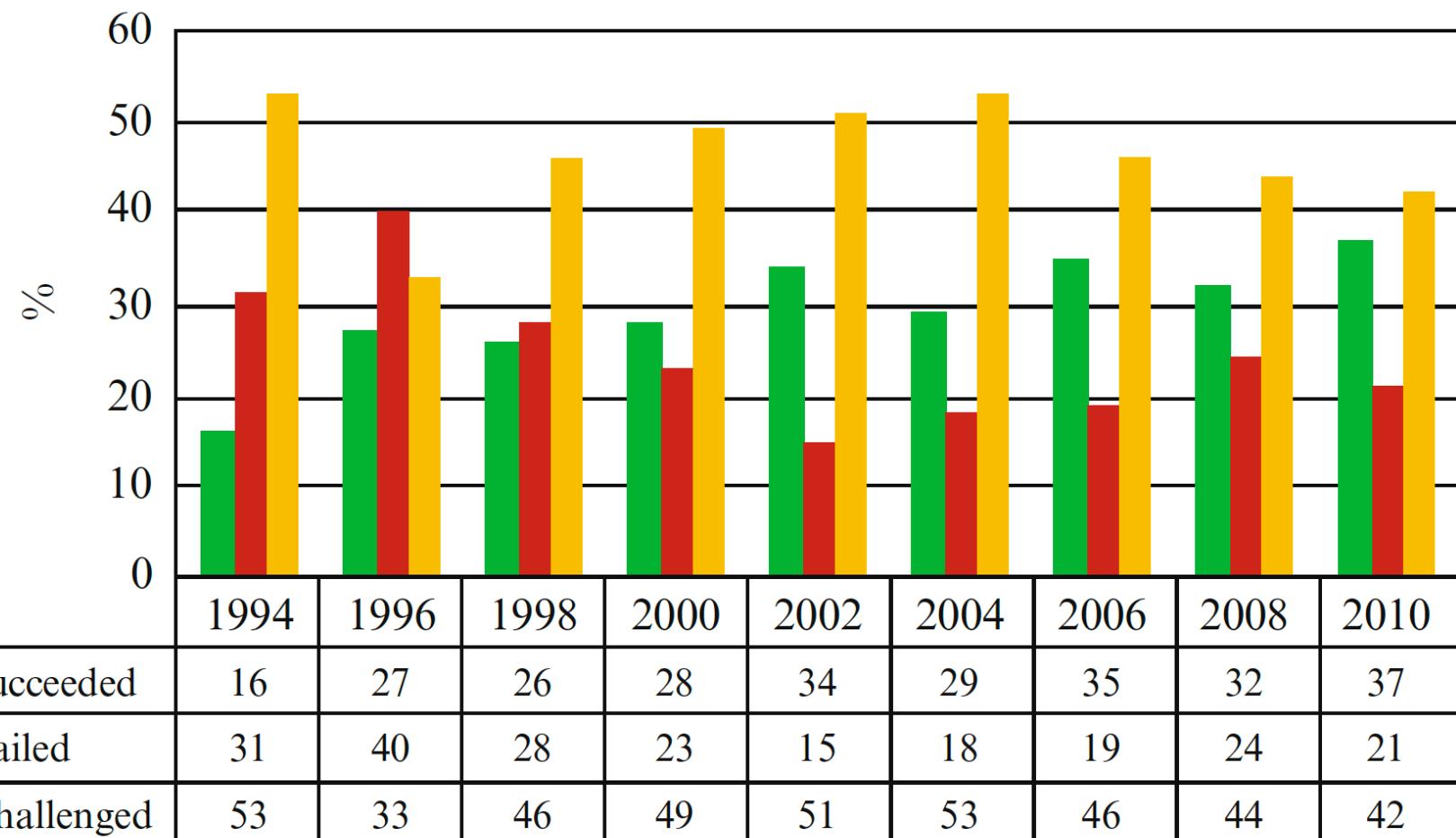
# La produttività dell'industria sw è bassa

- Da un'analisi di 13.522 progetti di costruzione sw:
  - 66% di tutti i progetti **falliscono** (non hanno risultato utile)
  - 82% dei progetti superano i tempi previsti
  - 48% dei progetti producono **sistemi senza le funzioni richieste dai clienti** → requisiti cambiati nel tempo
    - ↳ o non si erano capiti
    - ↳ o sono fatti \$ e tempo
  - 55 miliardi \$ di spreco considerando solo i progetti USA

*Standish Report 2003*

# Standish CHAOS reports

Standish figures 1994-2010



↳ Con standish o semp, Ingegneria del software aggiorn.

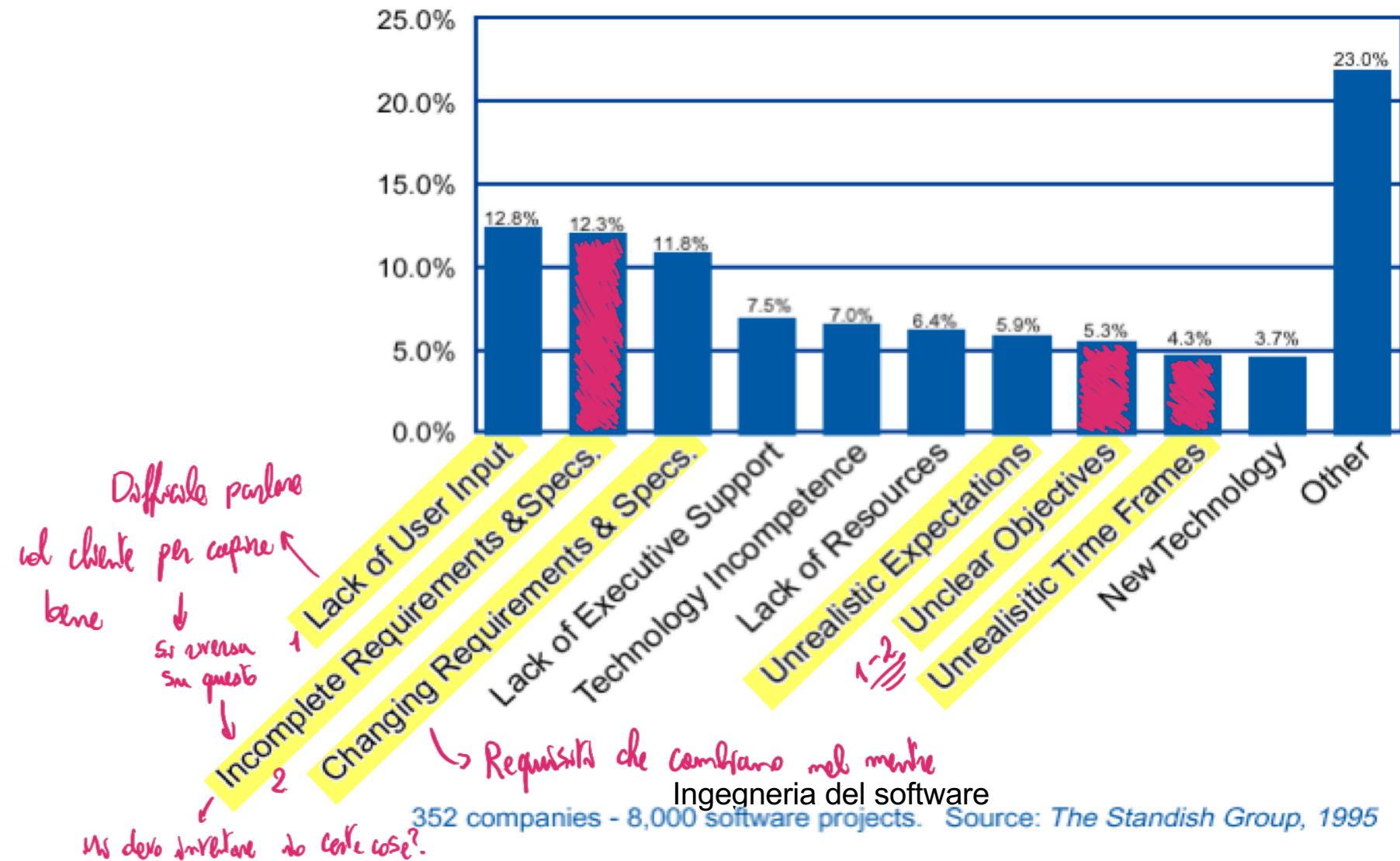
# Agile vs waterfall (CHAOS 2015)

CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%
Large Size Projects	Agile	18%	59%	23%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile	58%	38%	4%
	Waterfall	44%	45%	11%

The resolution of all software projects from FY2011–2015 within the new CHAOS database, segmented by the agile process and waterfall method. The total number of software projects is over 10,000.

# Perché falliscono i progetti sw



# Perché falliscono i progetti sw: i rischi

Quali sono i rischi **principali** di chi sviluppa software?

- Mancanza di feedback da parte del cliente/utente
  - Turnover dello staff e in particolare del team di sviluppo
  - Realizzare funzioni non richieste
  - Ritardi nella consegna
  - Superare il budget di progetto
  - Realizzare un sistema inusabile
  - Realizzare un sistema incapace di funzionare insieme con altri sistemi esistenti
- Capiro dove mettere i dubbi e capire le soluzioni ecc. ecc.*

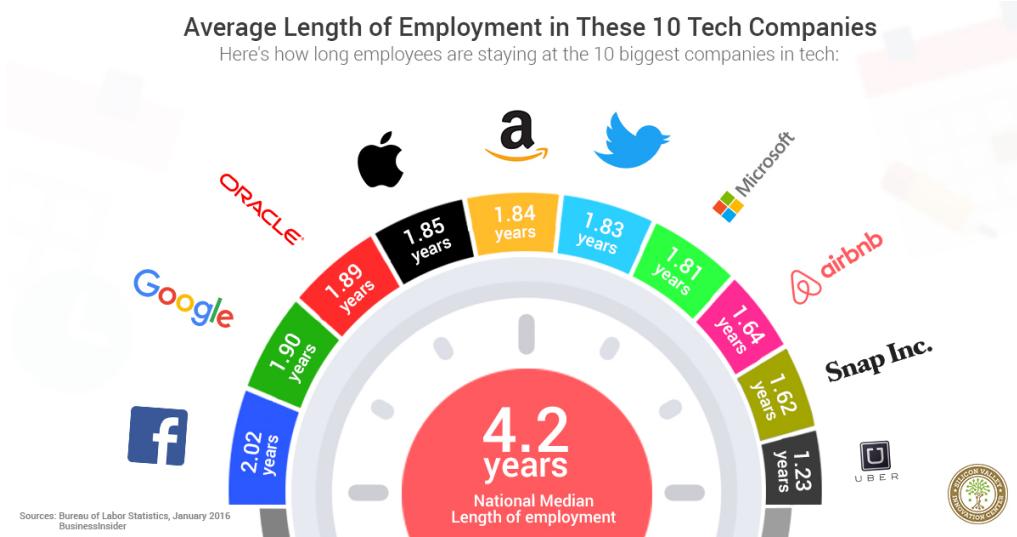


# Turnover dello staff

Durate media degli impieghi (2017):

↳ Tempo di occupazione; devo trasferire un progetto al nuovo assunto.

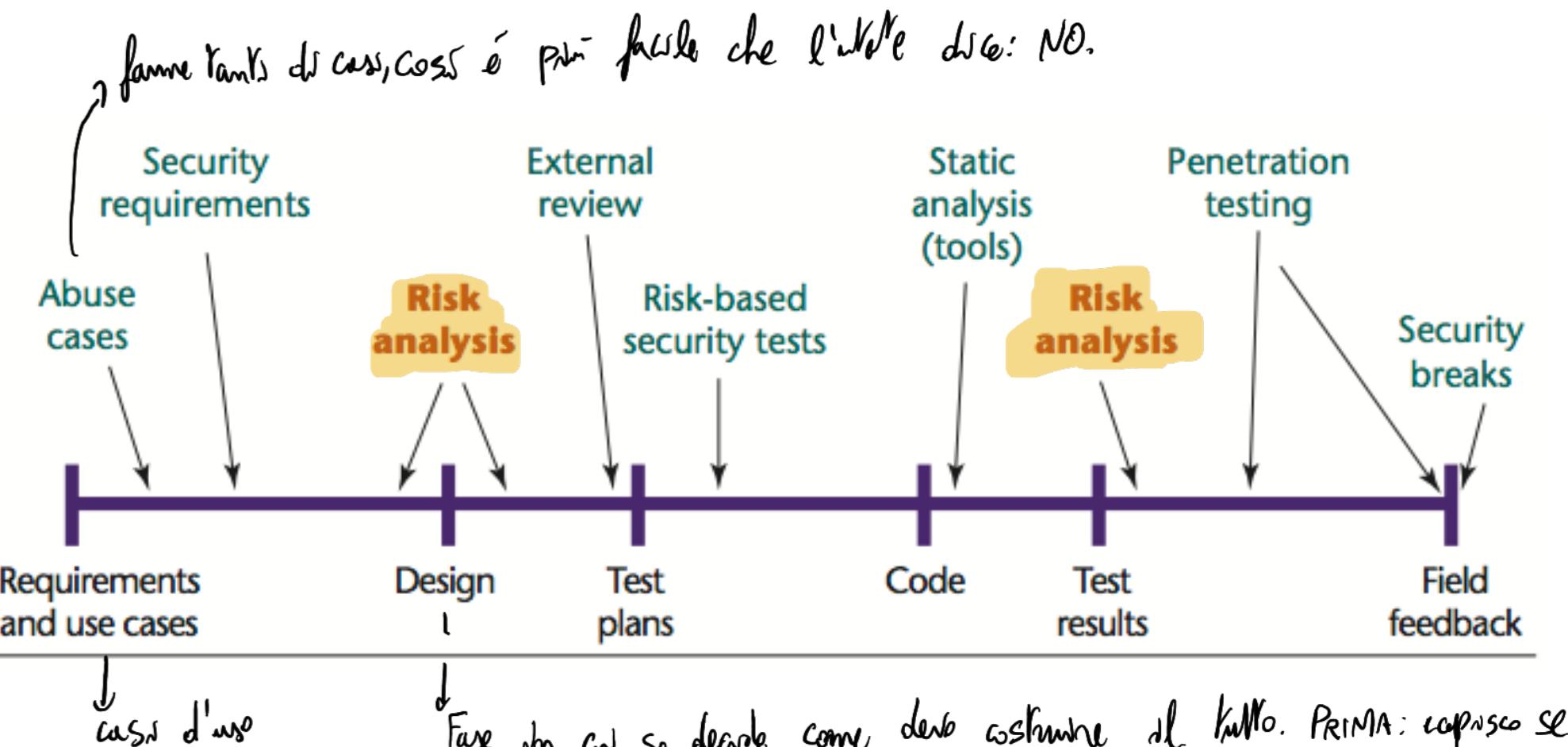
Facebook 2.02 anni  
Google 1.90 anni  
Oracle 1.89 anni  
Apple 1.85 anni  
Amazon 1.84 anni  
Twitter 1.83 anni  
Microsoft 1.81 anni  
AirBnb 1.64 anni  
Snap Inc. 1.62 anni  
Uber: 1.23 anni



Fonte: [http://www.businessinsider.com/employee-retention-rate-top-tech-companies-2017-8?IR=T&utm\\_content=bufferf5cb9&utm\\_medium=social&utm\\_source=twitter.com&utm\\_campaign=buffer](http://www.businessinsider.com/employee-retention-rate-top-tech-companies-2017-8?IR=T&utm_content=bufferf5cb9&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer)

→ Risolvo problemi con una risk analysis

# Analisi dei rischi nel ciclo di vita del software



⇒ Post design: è abbastanza semplice per essere sviluppato? O è troppo complesso?

Test plans: come verifichere il tutto? NOTA: alcuni lo fanno pre-design.  
↓  
i requisiti

Con il post design posso più essere più preciso e concreto.

External reviews: opinione esterna per capire se ha senso il tutto. Se è chiaro dall'esterno.

Security.

Code: sto sviluppando il software.

Static Analysis: guardando il codice vedo se ci sono incongruenze.

⇒ Test results → Risk analysis. Escono problematiche. Quanti peggiori? Mentre?

Penetration testing.

Field feedback: ricezione feedback dagli utenti.

# I costi del software

- A causa dell' impatto dei rischi, i costi software spesso **dominano** i costi di produzione di un sistema; in particolare, i costi sw sono spesso maggiori dei costi dell' hardware sottostante
- È più costoso mantenere il software che svilupparlo: nel caso di sistemi con vita duratura, i costi di manutenzione sono un multiplo dei costi di sviluppo (es.: 3 volte)
- L' ingegneria del software si preoccupa di produrre software con costi “accettabili”

# I problemi

I problemi principali che affronta l'IdSw riguardano

- I metodi di **analisi e progettazione** dei prodotti sw
  - *Quale metodo è il più adatto in una data situazione?*
- Lo studio del **processo di sviluppo** del sw
  - *Come posso migliorare il mio processo di sviluppo?*
- Lo sviluppo degli **strumenti di produzione** del sw
- Gli **aspetti economici** dei prodotti e dei processi
  - *Quanto costa produrre un certo sistema?*
- La **standardizzazione** di processi e tecnologie

# Le competenze richieste nello sviluppo del sw

- Software requirements – *requisiti*
- Software design – *progettazione*
- Software construction – *scrittura*
- Software testing
- Software maintenance – *evoluzione e manutenzione*
- Software configuration management
- Software engineering (project) management
- Software engineering process – *processo di sviluppo*
- Software engineering tools and methods
- Software quality

# Temi dell'ingegneria del sw

- Il ciclo di vita del software
- Il processo di sviluppo del software e gli strumenti
  - Cattura, specifica, analisi e gestione dei requisiti
  - Progettazione dell'architettura e dei moduli
  - Codifica e debugging
  - Testing
  - Deployment
- Manutenzione
- Gestione della configurazione
- Project management
- Qualità del software

→ chiunque ha a che fare col software durante la parte di sviluppo e al consumo finale.

# Parti interessate (stakeholders)

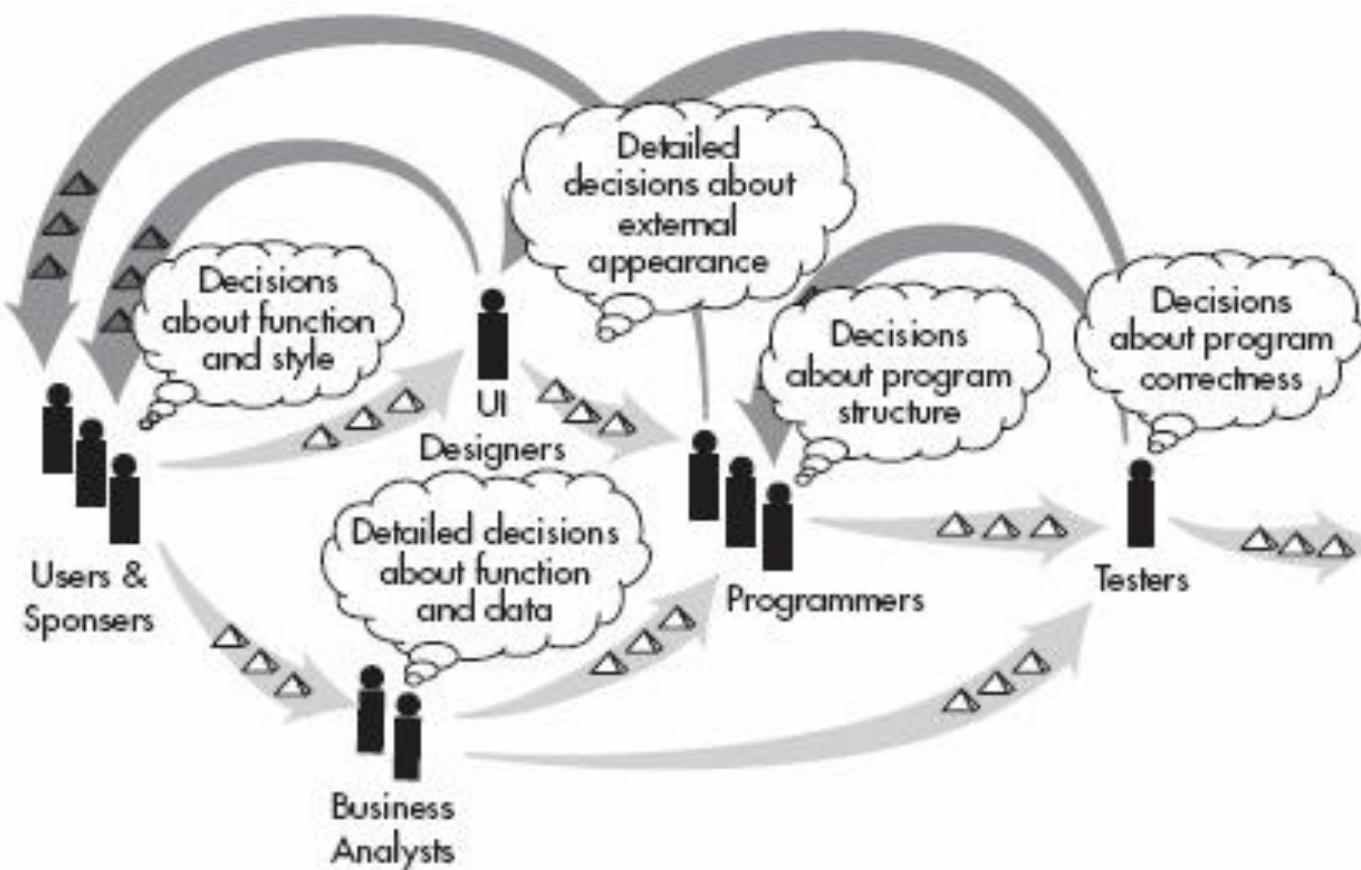
Tipi di **stakeholders** Non usato nel modello di sviluppo lineare

- Progettisti professionisti
- Management
- Personale tecnico
- Decisori ⇒ Quello che decide cosa si deve o non si deve fare
- Utenti
- Finanziatori
- ...  
soldi, tecnologia da usare ecc.

Ad ogni stakeholder corrisponde almeno uno specifico **punto di vista** (view) e varie decisioni

Tutte le decisioni influenzano tutte le decisioni

# Decisioni degli stakeholders



# Processi di produzione

- I processi di produzione si creano e poi evolvono
- Prodotti e processi possono essere descritti e valutati da un punto di vista **qualitativo**
- Processi di produzione a diversi livelli:
  - Ciclo di vita industriale
  - Ciclo di sviluppo: analisi dei requisiti, design, testing
  - Progettazione di un servizio sw (es.: e-commerce)
  - Progettazione di un modulo e del relativo test

# Discussion

Come si costruisce un prodotto software?



# Principi guida dello sviluppo software

Tutte le fasi vengono effettuate  
ciclicamente

Sviluppare iterativamente

Gestire i  
requisiti

Usare  
architetture  
di componenti

Modellare  
graficamente

Verificare  
la qualità

Tutto quello  
che noi andremo  
a fare nello  
sviluppo

Controllare le modifiche

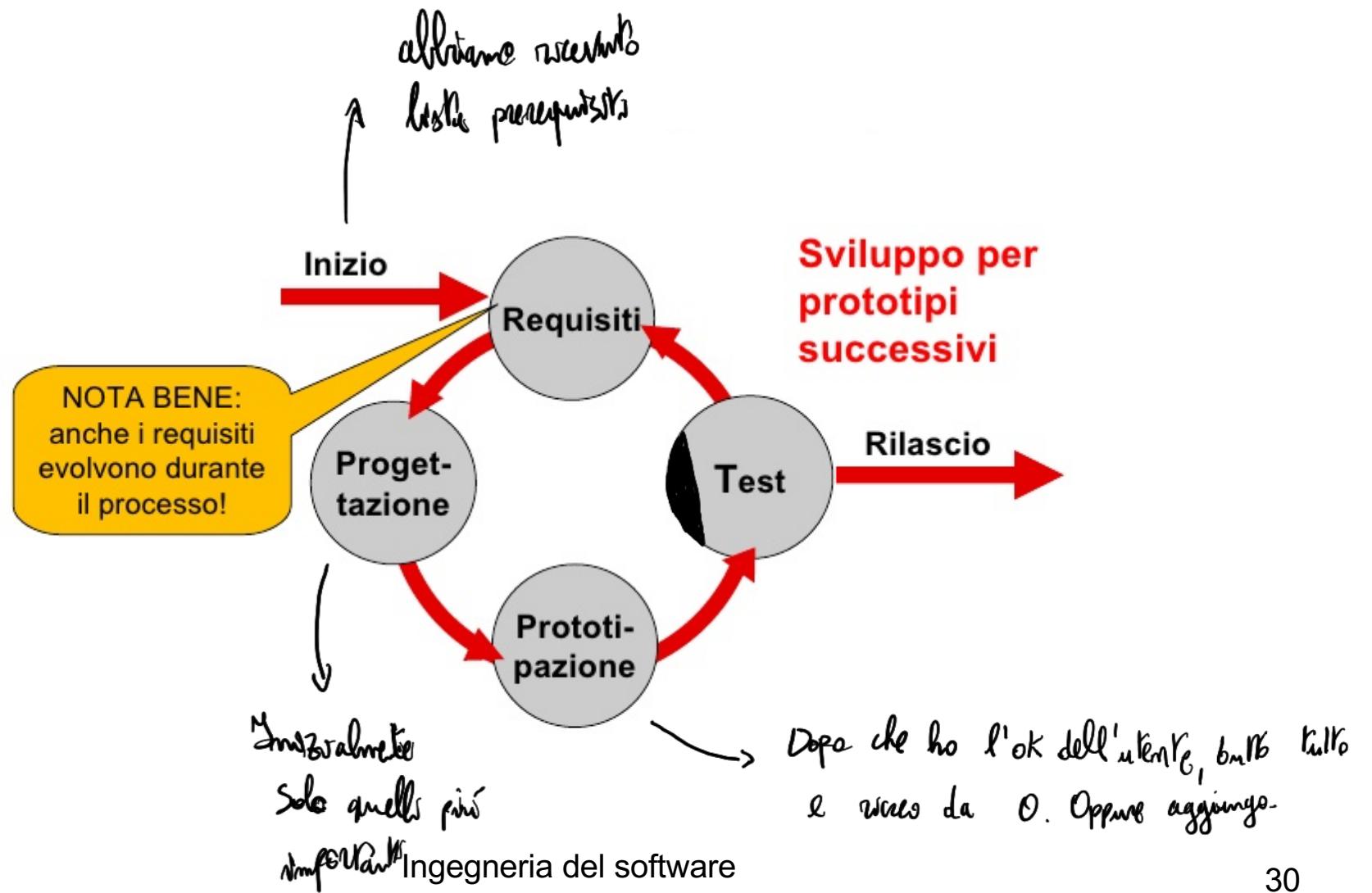
Tutte le modifiche devono essere tracciate dall'inizio  
alla fine

# Modelli del software

Un **modello** è una *descrizione* che:

- permette di studiare quali problemi possono capitare durante la costruzione di un sistema
- permette a tutte le parti interessate al sistema di comunicare tra loro usando una terminologia comune

# Sviluppare iterativamente

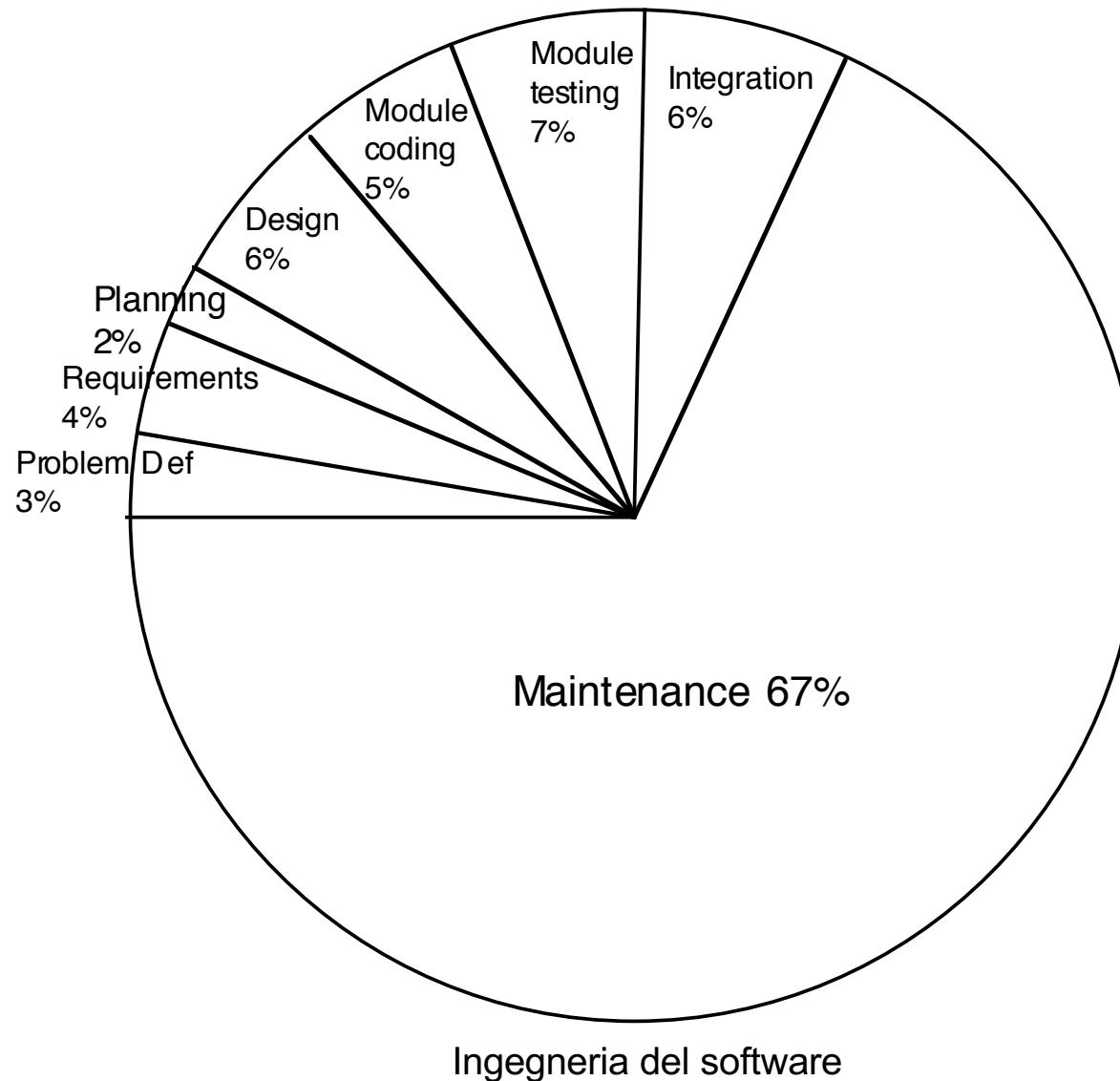


# Il ciclo di vita del software

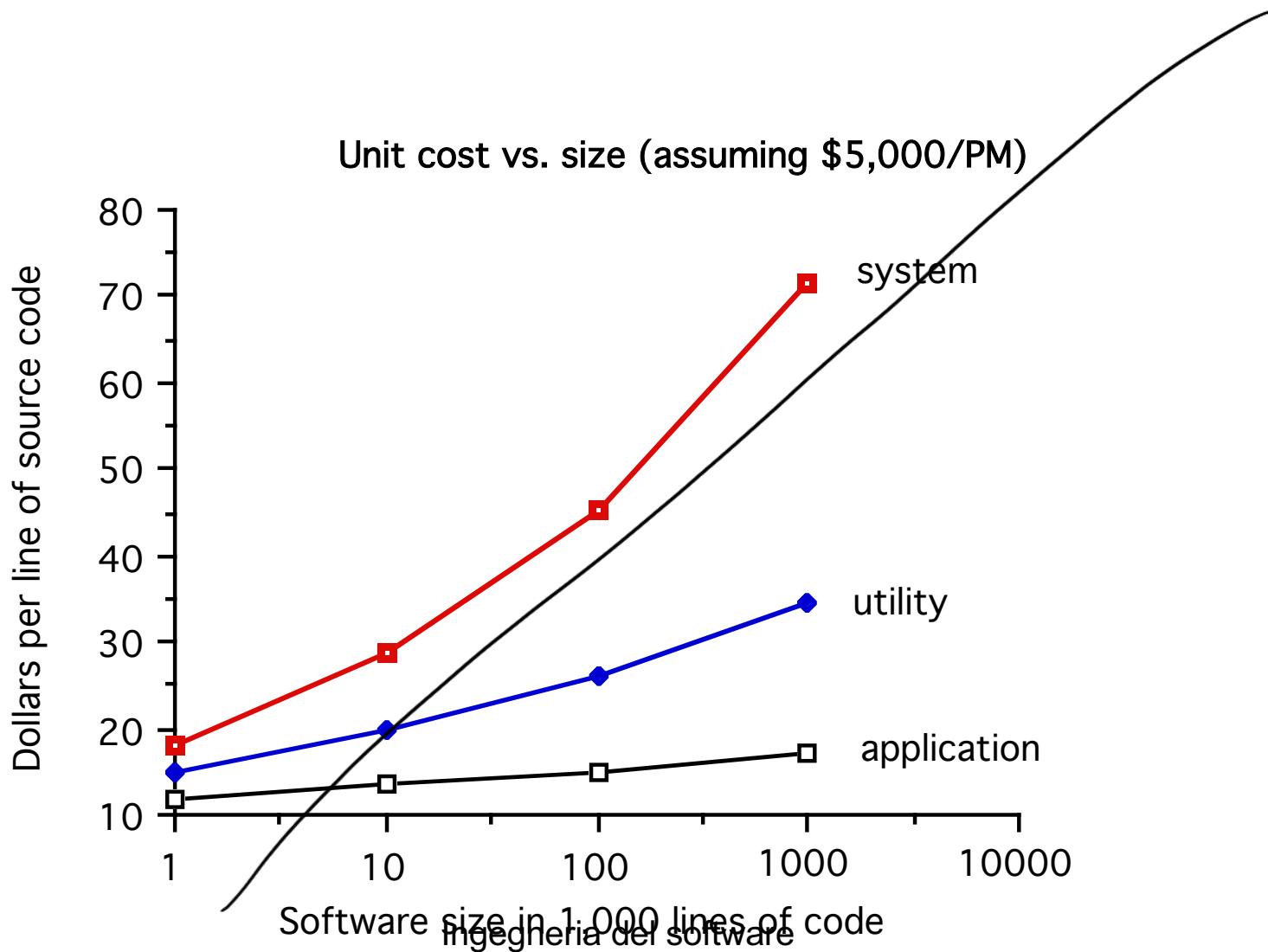
- **Requisiti: analisi e specifica** → Prima di base una specifica, poi una svolta  
Si modella architettura generale: di che ha bisogno? Database, Server...  
Implementation
- La progettazione: modellazione dell'architettura e dei singoli componenti
- La codifica ed il debugging  
↳ Come implemento carrello?  
In che parte del server devo salvare nota?
- Il testing e la verifica
- Il deployment (= la messa in opera)
- La manutenzione  
↳ Rilascio oppure se commissariata devo installarlo lì

# Costi di Sviluppo

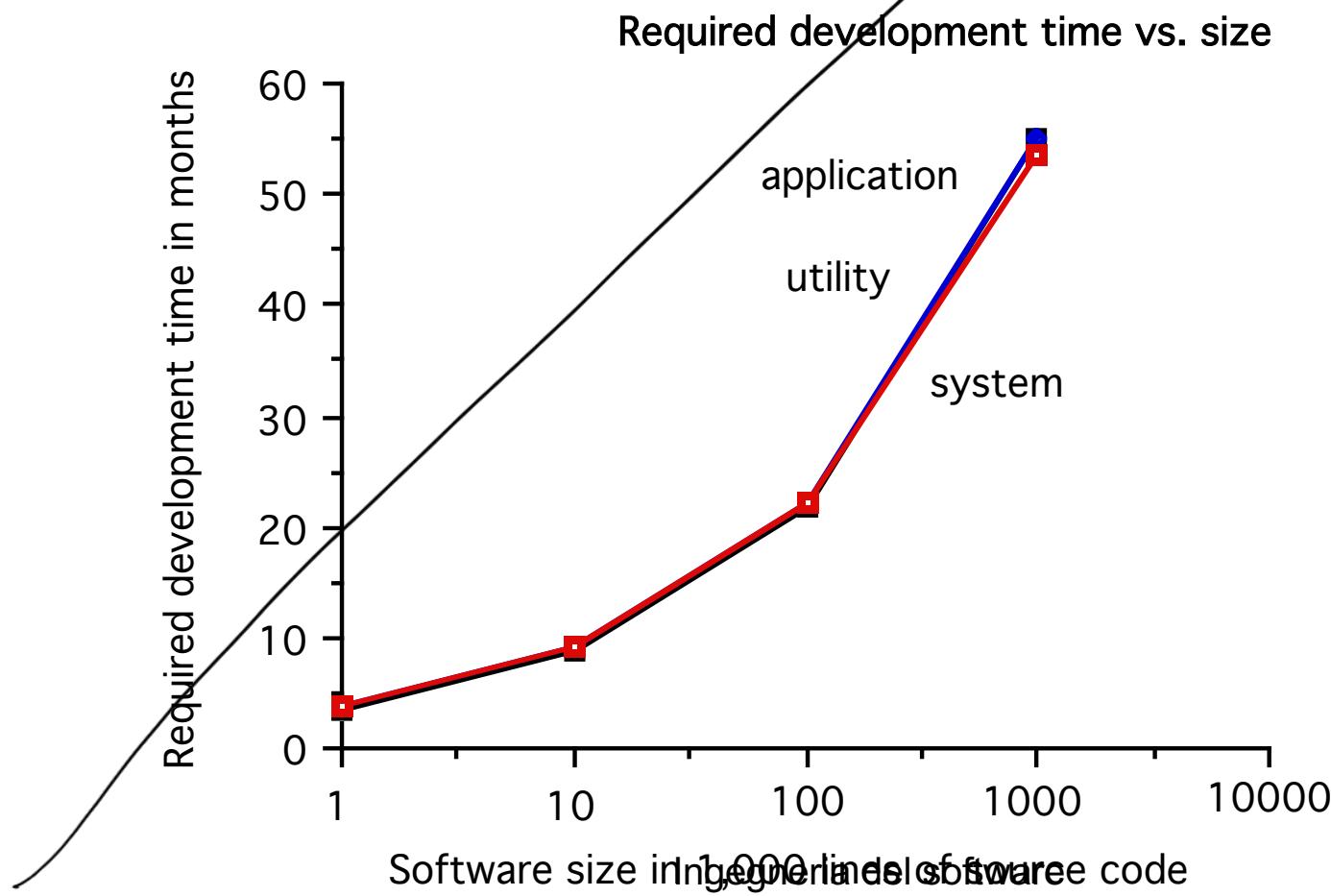
(Boehm citato da Schach)



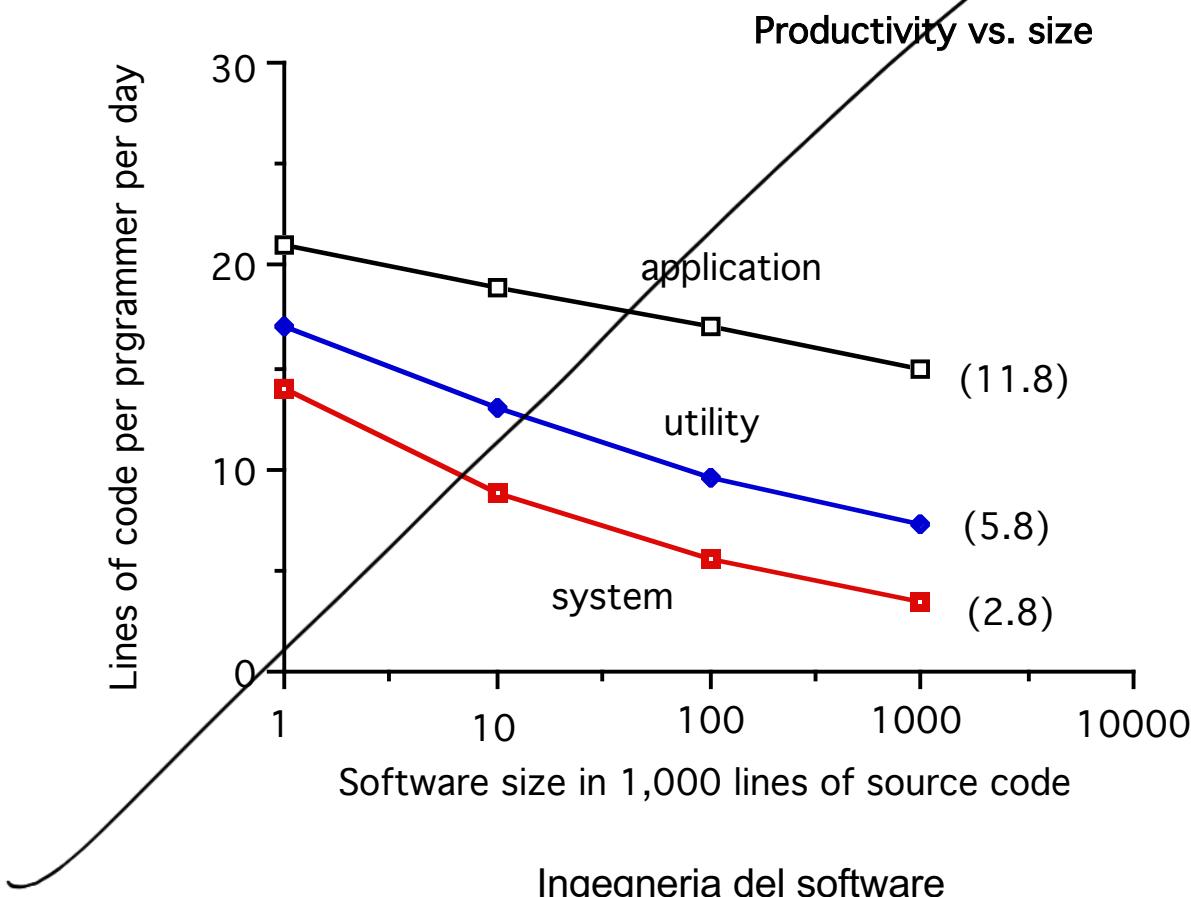
# Costo per linea di codice



# Durata



# Produttività



# Fare la cosa giusta

Di tutte le funzionalità di un'applicazione sw:

- Il 7% è usato continuamente
- Il 13% è usato spesso
- Il 16% è usato saltuariamente
- Il 19% è usato raramente
- Il 45% non è mai usato = più difficile trovare i bug

Fonte: Standish Group, Chaos report 2002

# La manutenzione

- Tutti i prodotti hanno bisogno di manutenzione a causa del cambiamento
  - I tipi principali di manutenzione:
    - Perfettiva o preventiva (65%): migliorare il prodotto
    - Adattiva (18%): rispondere a modifiche ambientali
    - Correttiva (17%): correggere errori trovati dopo la consegna
- errore trovato prima  
→ Cambia nel GDPR ecc.  
↳ errore trovato dopo

É la più costosa:  
• danno di domenica  
• non so dove sta errore  
• Bisogna correggere subito,  
soprattutto per sicurezza.

Il mondo cambia continuamente  
La manutenzione è “normale”

# Attributi dei prodotti software

- Attributi esterni (visibili all'utente)
  - Costo (e tipo di licenza)
  - Prestazioni
  - Garanzia
- Attributi interni (visibili ai progettisti)
  - Dimensione (size)
  - Sforzo di produzione (effort)
  - Durata della produzione (dall'inizio alla consegna)
  - Mantenibilità
  - Modularità → ogni comp. software si occupa di uno specifico obiettivo e ha metodi specifici per quello.



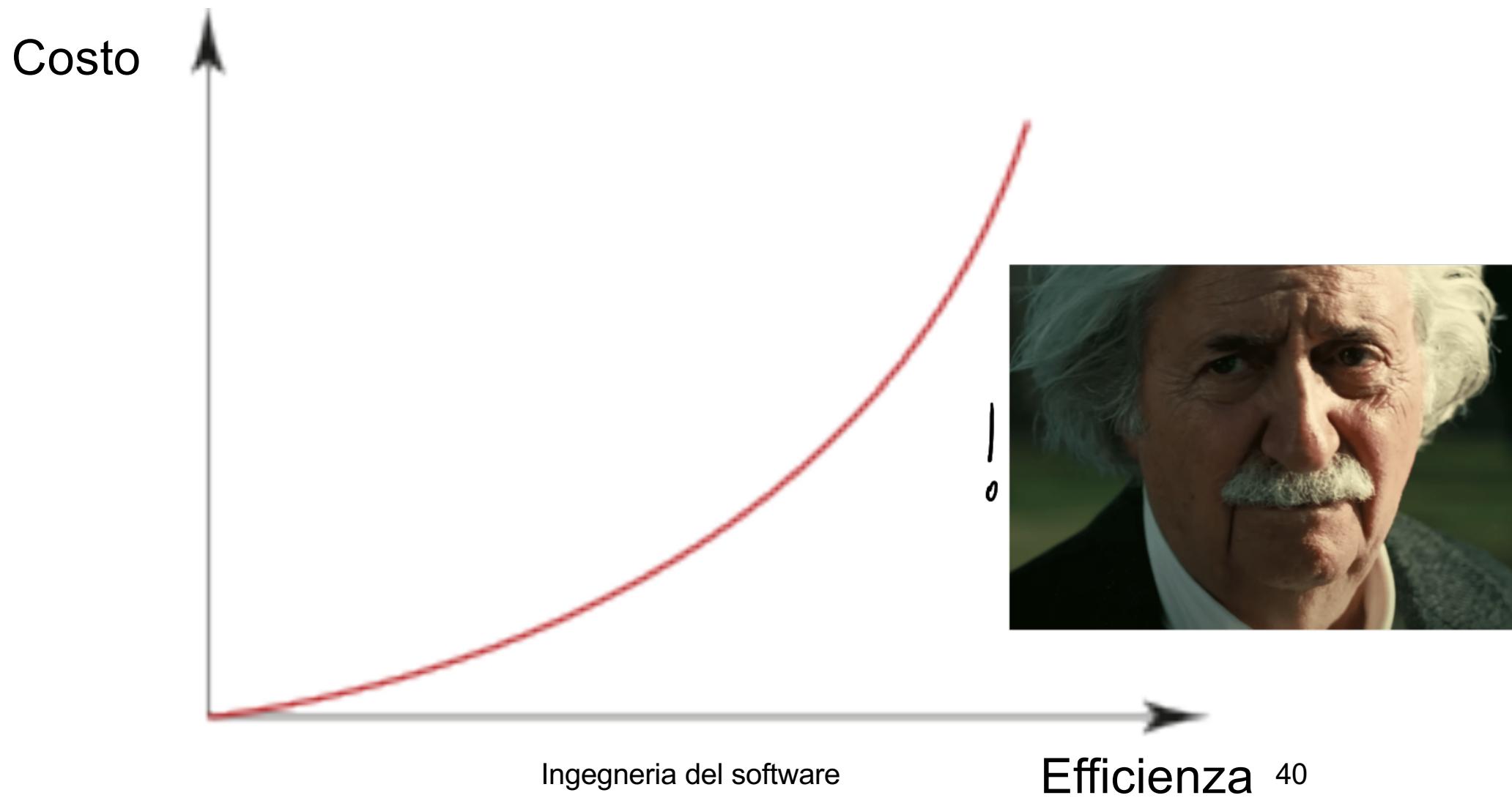
Ingegneria del software  
Molto modulare = facilmente mantenibile

→ ma può anche funz

# Bilanciamento degli attributi

- L'importanza relativa degli attributi di prodotto dipende dal prodotto e dall'ambiente in cui verrà usato
- A volte certi attributi sono più importanti
  - Nei sistemi in tempo reale con requisiti di sicurezza, gli attributi chiave sono l'affidabilità e l'efficienza
- Se un attributo dev'essere particolarmente curato e "spinto", i costi di sviluppo tenderanno a crescere esponenzialmente

# Il costo dell'efficienza

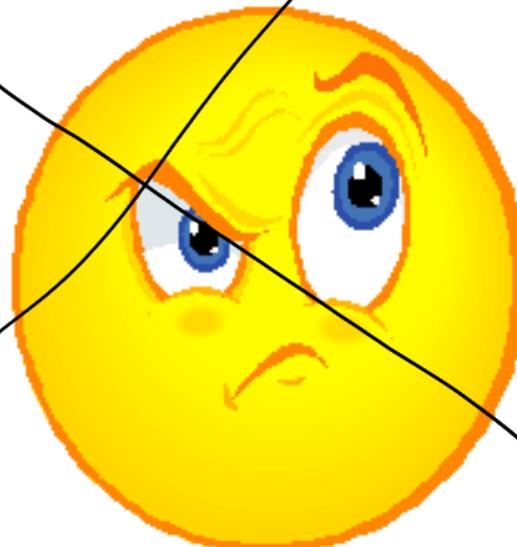


# Il costo della qualità

<b>Prodotto</b>	<b>Dimensione del team di sviluppo</b>	<b>Dimensione del team di testing</b>
NT 3.1	200	140
NT 3.5	300	230
NT 3.51	450	325
NT 4.0	800	700
Win2k	1400	1700

# Discussione

- Come si organizza un processo di sviluppo del software?



# Gli standard per lo sviluppo del software

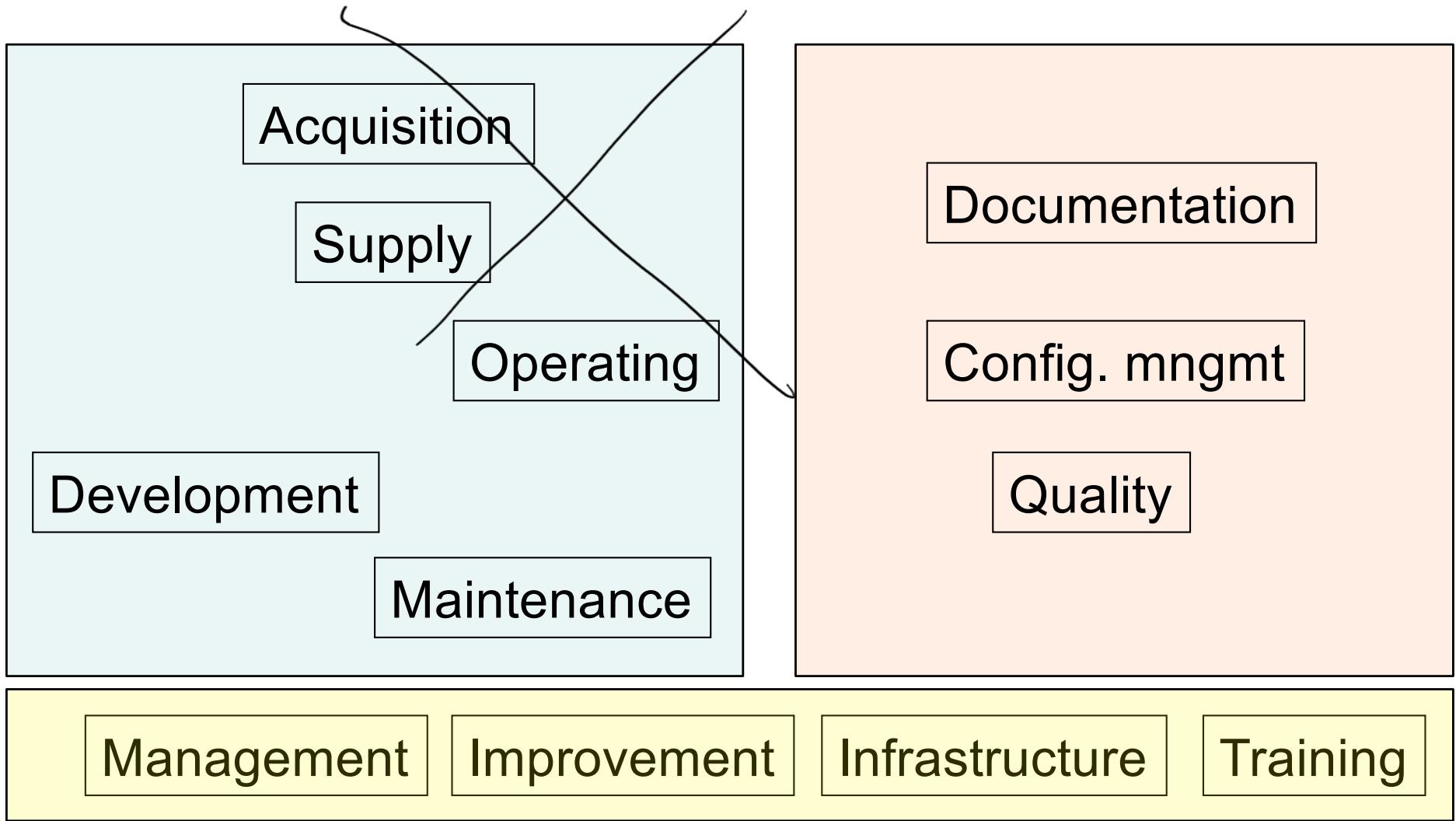
## Standard principali software engineering IEEE

- IEEE 610 Standard glossary sw engineering
- IEEE 828 Sw configuration management
- IEEE 829 Sw test documentation
- IEEE 830 Recommended practice for sw Requirements Specifications
- IEEE 1008 Sw unit testing
- IEEE 1219 Sw maintenance
- IEEE 1471 Recommended practice for sw Architectural Descriptions
- IEEE 1517 Sw reuse processes

# Processi a ciclo di vita

- Lo standard IEEE12207 definisce le fasi principali dei processi a ciclo di vita:
  - **Primarie**: Acquisition, supply, development, operation, maintenance *(o che realizza software)*
  - **Supporto**: audit, configuration *(rende più agile software)*, management, documentation, quality assurance, verification, validation
  - **Organizzative**: management, infrastructure, improvement, training

# Lo standard IEEE12207

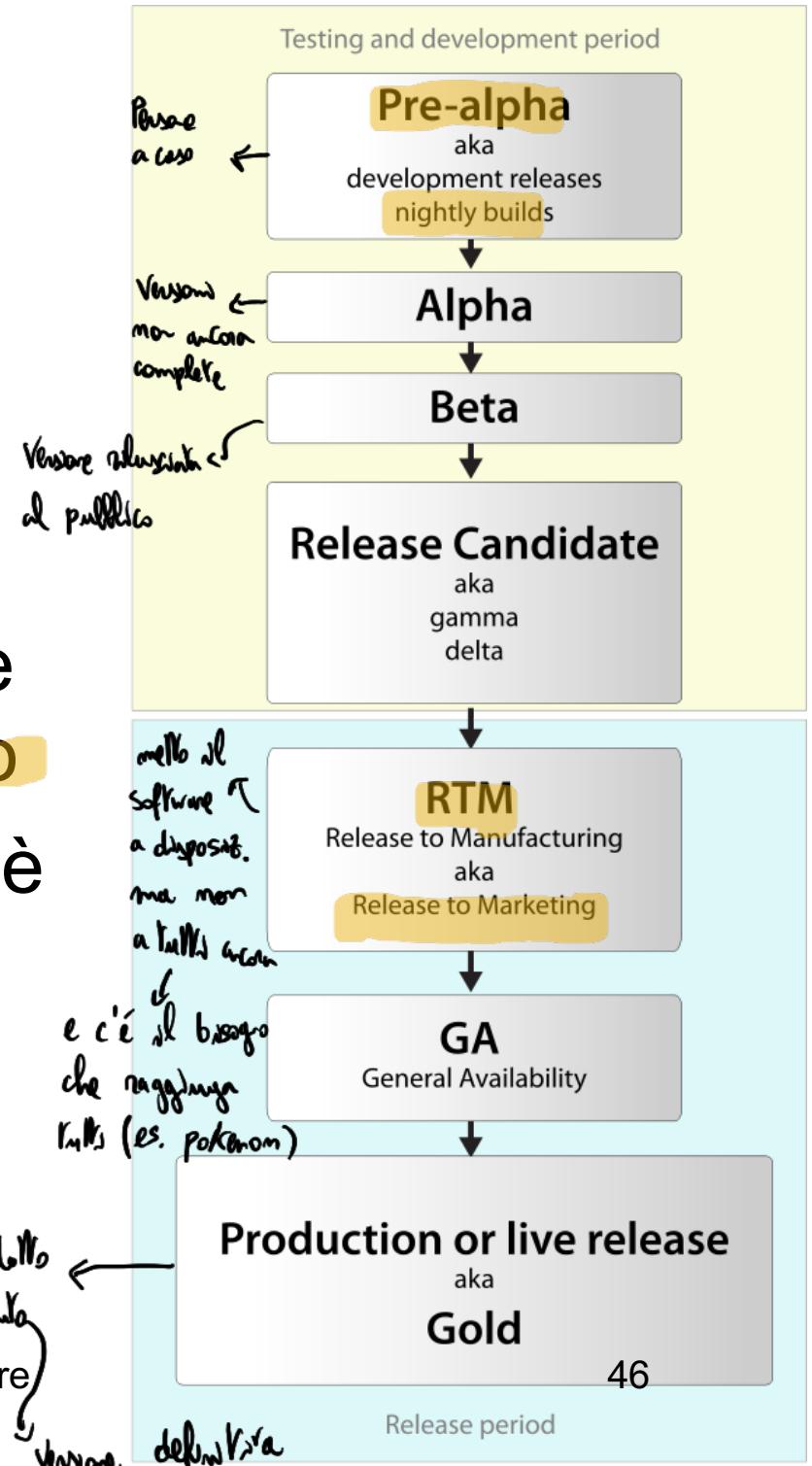


# Ciclo di vita di un rilascio sw

- Un **rilascio software** (*software release*) è una **versione** di un prodotto che viene **immessa sul mercato**
- Il **ciclo di vita della release** è l'**insieme delle fasi** del suo **sviluppo e della sua vita operativa**

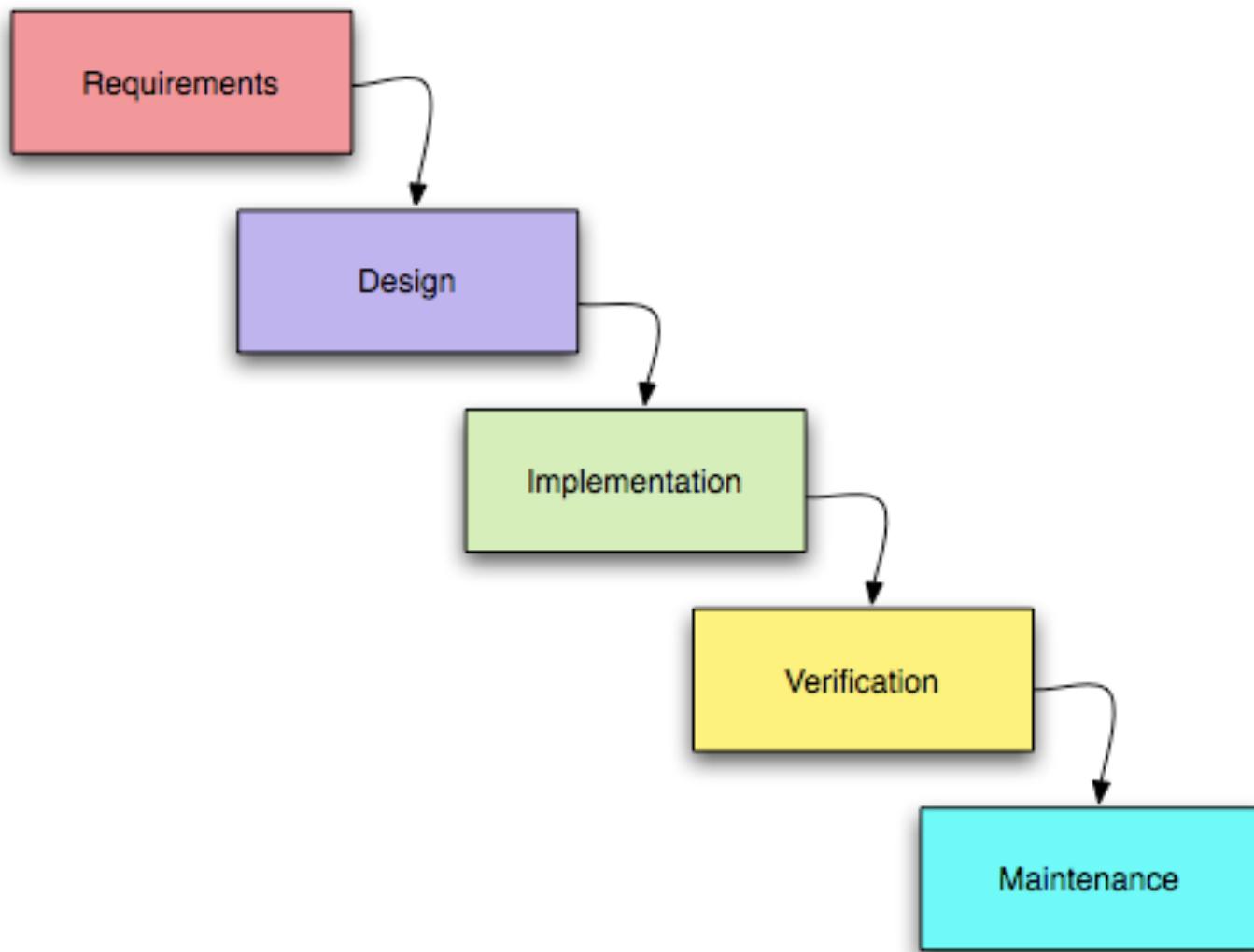
*che avvengono ciclicamente*

Ingegneria del software



# Le attività di sviluppo

- Le attività di sviluppo del software differiscono in funzione dell'organizzazione che sviluppa e del sw da produrre, ma di solito includono:
  - **Specific**a delle funzionalità richieste (requisiti)
  - **Progetto** della struttura modulare e delle interfacce
  - **Implementazione**: codifica moduli e integrazione
  - **Verifica e validazione**
  - **Evoluzione** e manutenzione
- Per poterle gestire vanno **esplicitamente** modellate



# Il processo di sviluppo del sw

- **Processo software**: insieme dei ruoli, delle attività e dei documenti necessari per creare un sistema software
- Esempi di **ruoli**: stakeholder, progettista, sviluppatore, tester, manutentore, ecc.
- Esempi di **documenti**: codice sorgente, codice eseguibile, specifica, commenti, risultati di test, ecc.

# Le attività di sviluppo nel mondo

- Cusumano e altri nel 2003 hanno analizzato 104 progetti software in quattro regioni

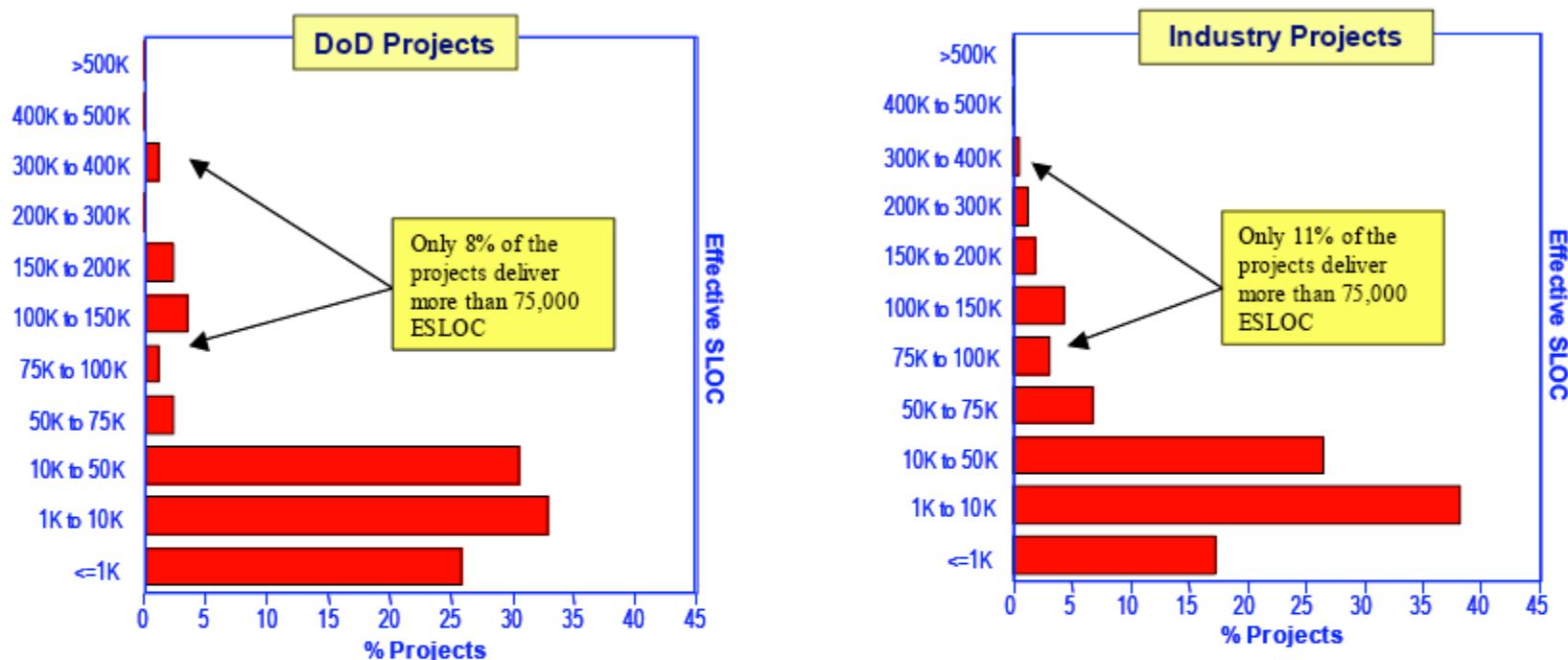
Practice / No. of Projects	India	Japan	US	Europe	Total
Architectural Specification	24	27	31	22	104
Functional Specification	83.3%	70.4%	54.8%	72.7%	69.2%
Detailed Design	95.8%	92.6%	74.2%	81.8%	85.6%
Code Generation	100.0%	85.2%	32.3%	68.2%	69.2%
Design Review	62.5%	40.7%	51.6%	54.5%	51.9%
Code Review	100.0%	100.0%	77.4%	77.3%	88.5%
Subcycles	95.8%	74.1%	71.0%	81.8%	79.8%
Beta Testing	79.2%	44.4%	54.8%	86.4%	64.4%
Pair Testing	66.7%	66.7%	77.4%	81.8%	73.1%
Pair Programming	54.2%	44.4%	35.5%	31.8%	41.3%
Daily Builds					
At the Start	16.7%	22.2%	35.5%	9.1%	22.1%
In the Build	12.5%	25.9%	29.0%	27.3%	24.0%
At the End	29.2%	37.0%	35.5%	40.9%	35.6%
Regression Testing	91.7%	96.3%	71.0%	77.3%	83.7%

# La produttività nel mondo

- Sugli stessi 104 progetti Cusumano raccolse i seguenti dati di produttività e qualità:

	India	Japan	US	Europe	Total
No. of Projects	24	27	31	22	104
LOC/programmer month	209	469	270	436	374
Defects/KLOC (12 mon. after delivery)	0.263	0.020	0.400	0.225	0.150

# Characteristics of Projects Completed in 12 Months



- Approximately 10% of the projects built more than 75,000 SLOC in 12 months
- Average staff required for projects less than 75,000 ESLOC is approximately 5-10
- Staff required for projects exceeding 75,000 ESLOC is approximately 20-100, depending on size

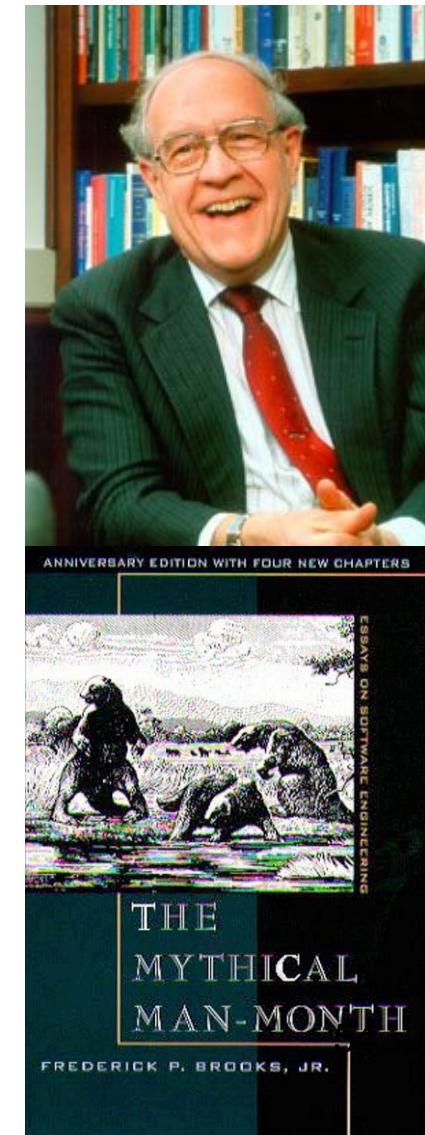
Figure 1. Frequency Distribution of new engineering software developed with a 12-month time period for DoD and private industry.

# Miti e leggende dell'ingegneria del sw

- Il "silver bullet" è il proiettile d'argento che uccide i lupi mannari
- "*Trovare un silver bullet*" è sinonimo di "trovare una soluzione finale" ad un problema
- Costruire software è difficile: qual è il silver bullet dell'ingegneria del sw?

# Fred Brooks

- Fred Brooks, premio Turing 1999, fu progettista del sistema IBM 360
- Dalle sue esperienze trasse spunto per scrivere il libro “The Mythical Man Month” e l’articolo “No silver bullet”



Ingegneria del software

# Miti e leggende

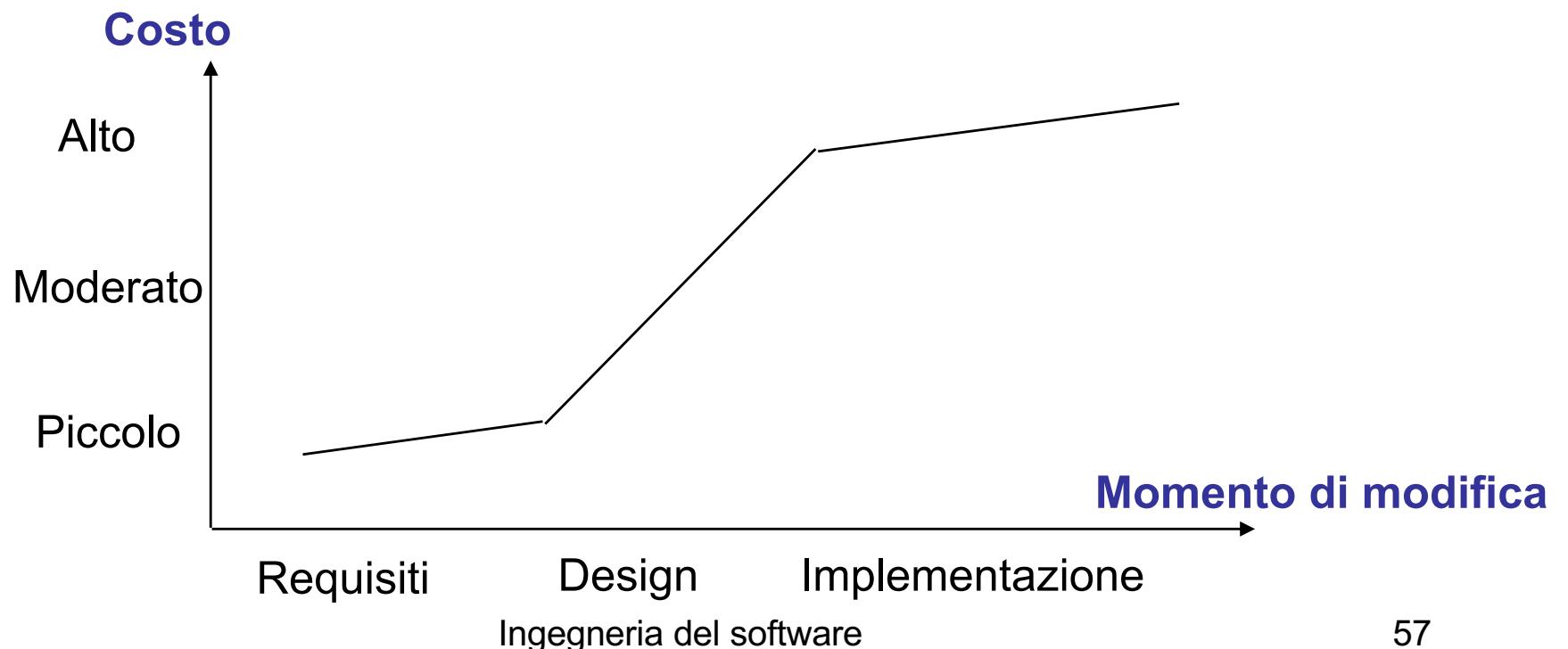
- Se il progetto ritarda, **possiamo aggiungere programmatori e rispettare la consegna**
  - **Legge di Brooks:** “Aggiungere personale ad un progetto in ritardo lo fa ritardare ancor di più”

# Miti e leggende

- Per cominciare a scrivere un programma, basta un' idea generica dei suoi obiettivi - **ai dettagli si pensa dopo**
  - **La cattiva definizione della specifica dei requisiti è la maggior causa di fallimenti progettuali**

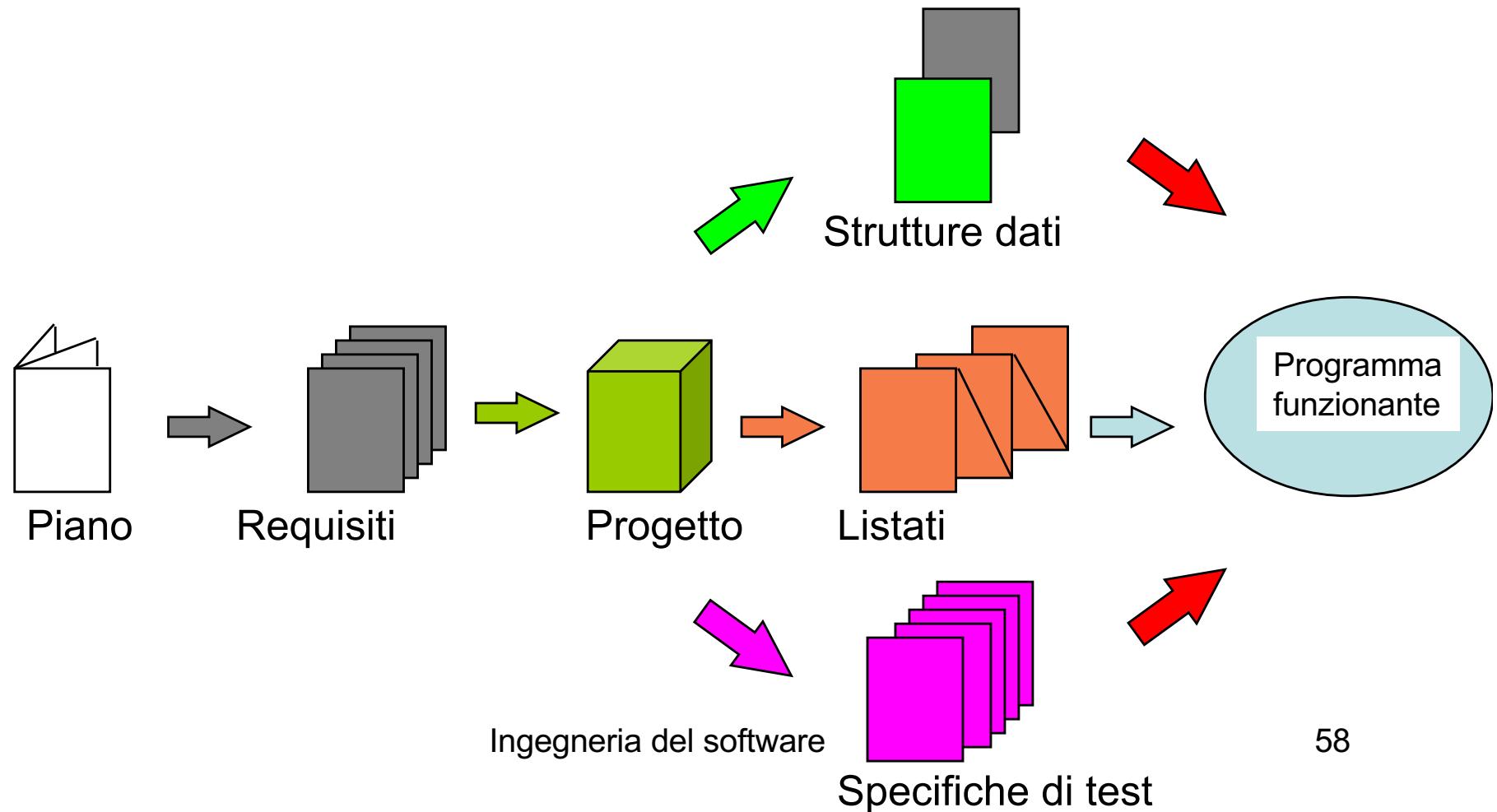
# Miti e leggende

- Se i requisiti di un progetto cambiano, non è un problema tenerne conto perché il **software è flessibile**



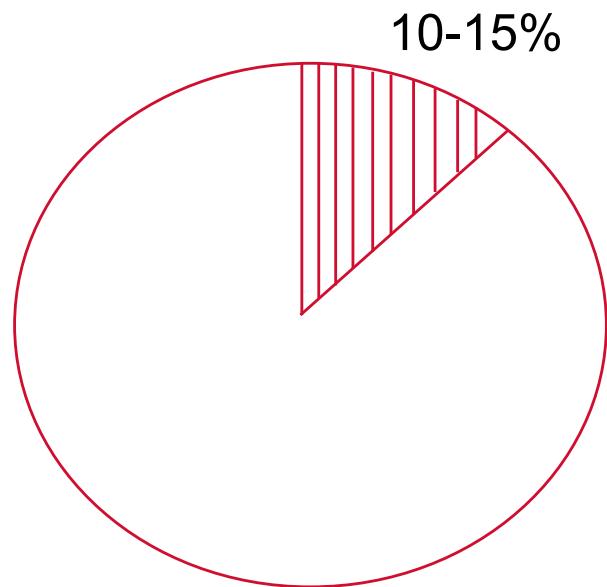
# Miti e leggende

- L'unico prodotto (**deliverable**) di un progetto di successo è un programma funzionante

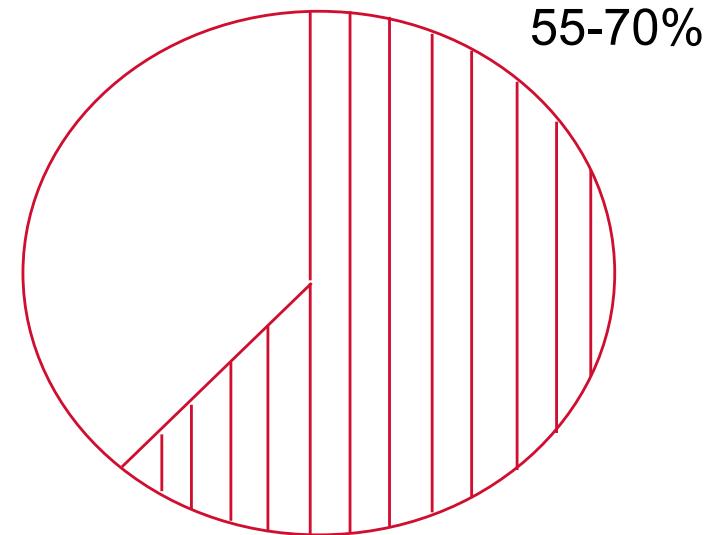


# Miti e leggende

- Se il software “funziona”, la manutenzione è **minima** e si può gestire errore per errore, quando capita di trovarne uno



Costi di manutenzione ~~pregettati~~<sup>previsti</sup> del software



Costi di manutenzione reali<sup>5</sup>

# Sommario

- Produrre software è costoso
- La produttività dell' industria del sw è bassa
  - Le consegne sono spesso in ritardo
  - I costi software spesso sforano il budget
  - La documentazione è inadeguata
  - Il software è spesso difficile da usare
- Soluzione: migliorare il processo software

# Domande di autotest

- Quali sono le fasi tipiche del ciclo di vita di un sistema software? E quelle dello sviluppo?
- Qual è la fase solitamente più costosa?
- In quale fase dello sviluppo è più pericoloso commettere un errore?
- In quale fase dello sviluppo è più semplice correggere un errore?
- Cos'è un processo di sviluppo del software?
- Quali sono i tipici documenti prodotti durante un processo software?
- Cos'è la legge di Brooks?

# Lettura consigliata

F.Brooks, No Silver Bullets, *IEEE Computer*,  
20:4, 1987

# Riferimenti

- *Software Engineering Body of Knowledge*, IEEE, 2014
- F.Brooks, *The Mythical Man Month*, AddisonWesley, 1995
- M.Cusumano, *The Business of Software: What Every Manager, Programmer, and Entrepreneur Must Know in Good Times and Bad*, Free Press, 2004
- IEEE/EIA 12207.0, "Standard for Information Technology – Software Life Cycle Processes"

# Principali pubblicazioni scientifiche

- IEEE Transactions on Software Engineering
- ACM Transactions on Software Engineering and Methodology
- Int. Conference on Software Engineering

# Pubblicazioni di ricerca sul sw engineering

## Riviste:

- IEEE Transactions on sw engineering
- ACM Transactions on software engineering and methodology
- IEEE Software
- Empirical Software Engineering
- Automated Software Engineering
- Journal of Object Technology
- ACM SIGSOFT

## Conferenze:

- International Conference of Software Engineering
- Fundamentals of Software engineering
- SPLASH
- International Conference on Software and System Process

# Siti utili

- [www.sigsoft.org/seworld](http://www.sigsoft.org/seworld)
- [www.computer.org/web/swebok](http://www.computer.org/web/swebok)
- [swebokwiki.org/Main\\_Page](http://swebokwiki.org/Main_Page)

# Domande?

