

# 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

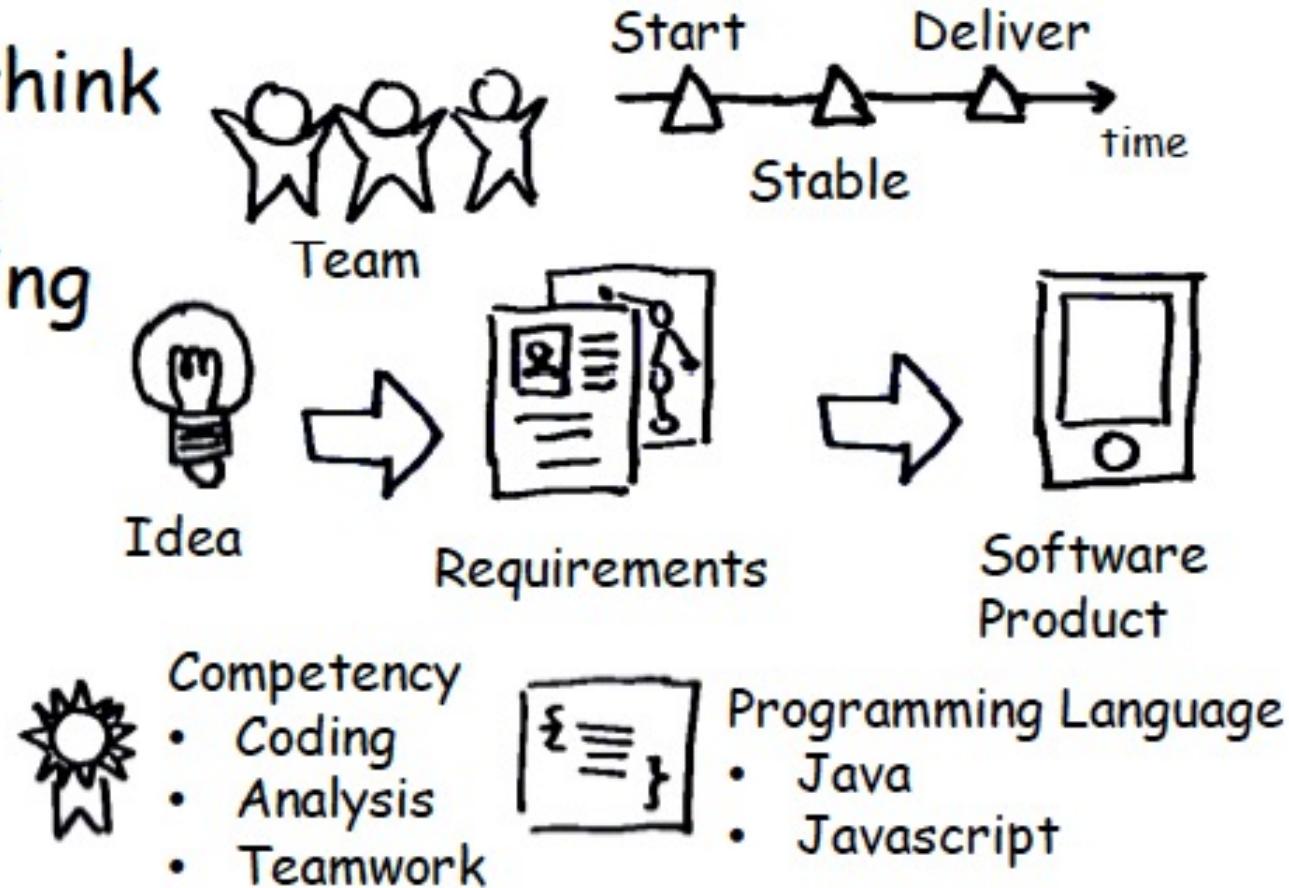
# From programming to development

Cosa è più difficile:  
scrivere software, oppure  
leggerlo (per es. per modificarlo)?

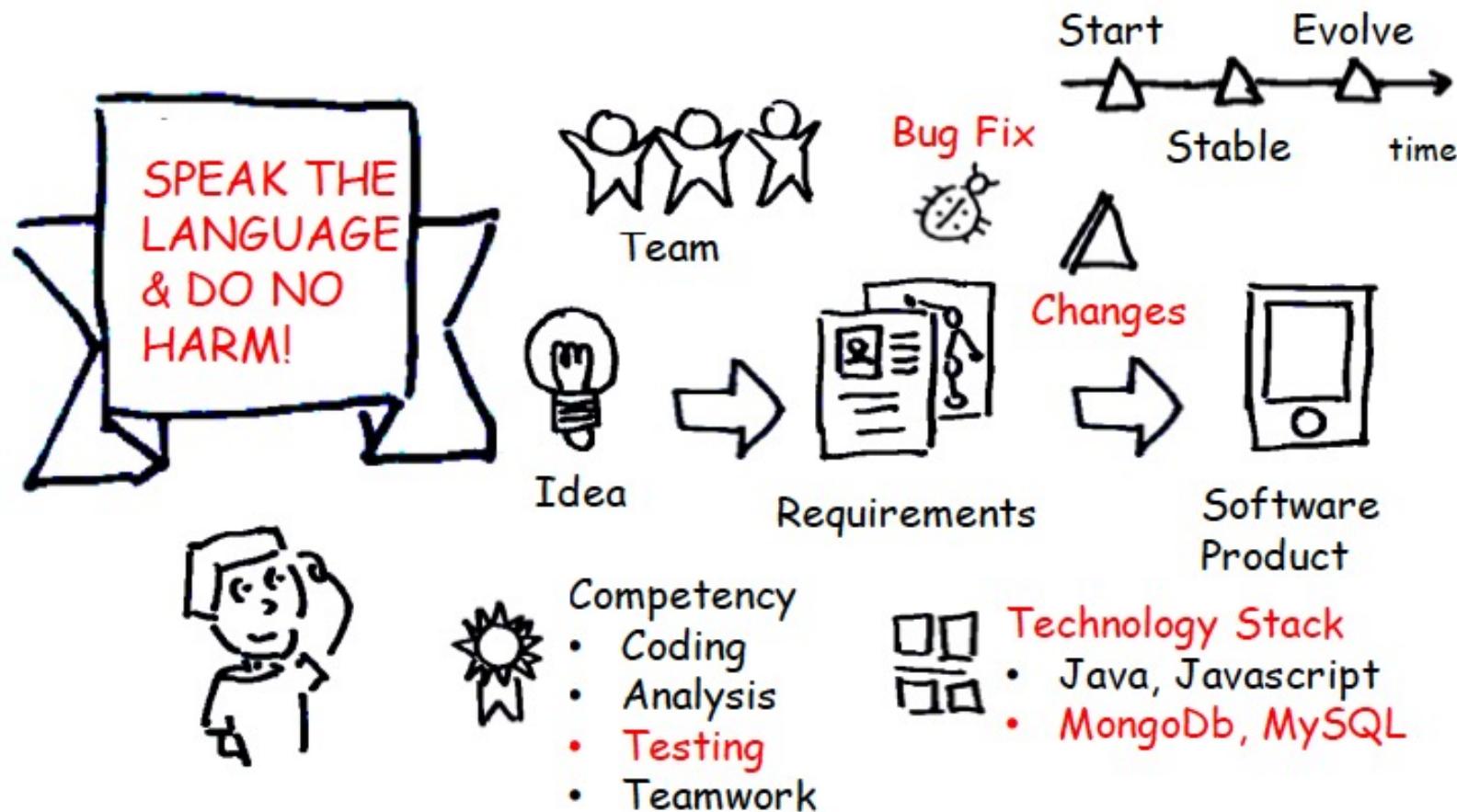
Come si **sviluppa** il software?

# Punto di vista: studente

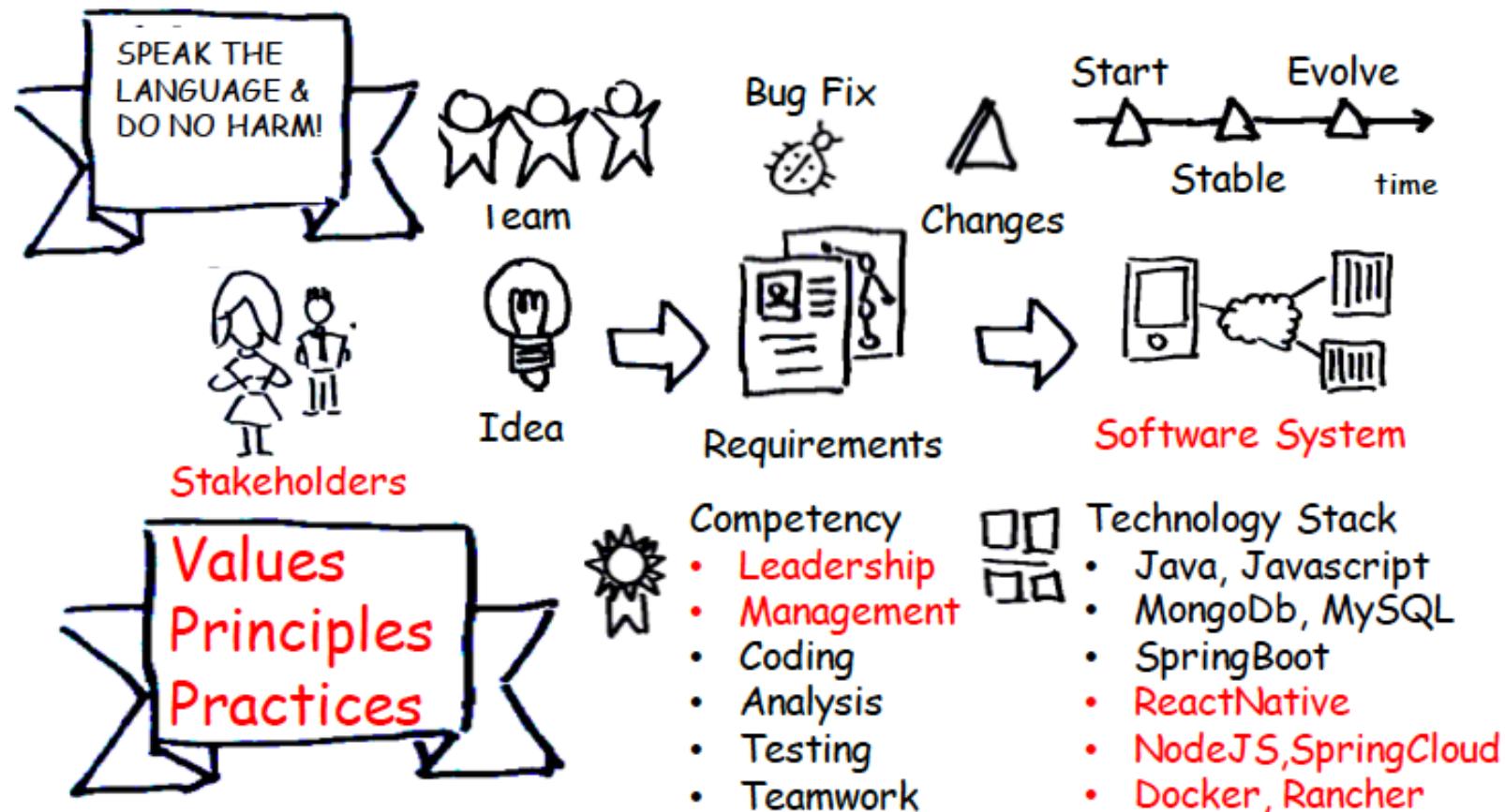
What I think  
software  
engineering  
is about



# 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 developer?

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.

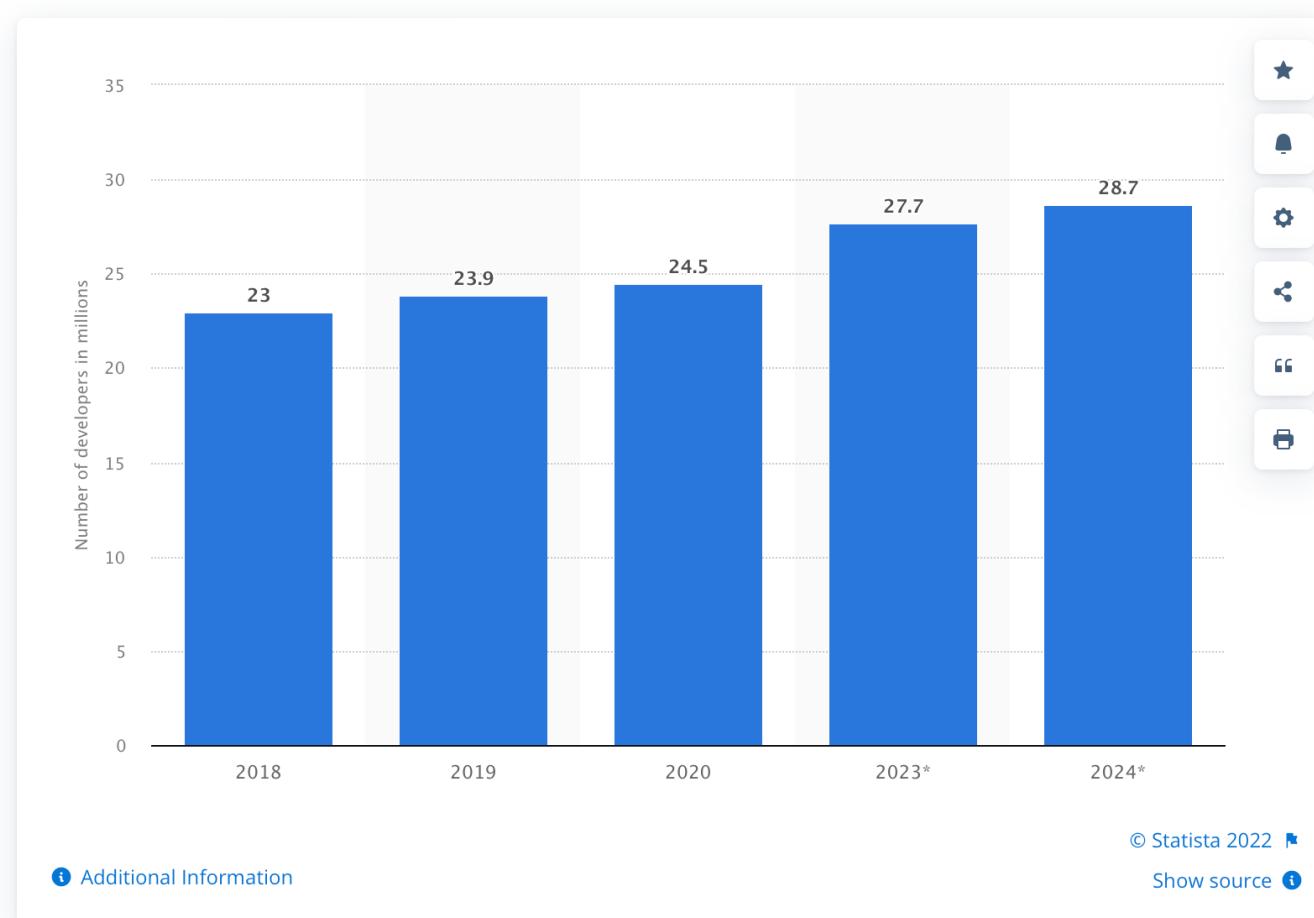
# Ingegneria del software

- 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
- NB: manca una TEORIA dell'ingegneria del sw

# La professione

Technology & Telecommunications › Software

**Number of software developers worldwide in 2018 to 2024  
(in millions)**



Ingegneria del software

9

# La professione

- 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 con altre professioni



## Information Security Analyst

### 💡 #1 in Best Technology Jobs

As concern about cybersecurity grows, so does the demand for information security analysts. It is the duty of these professionals to prepare and carry out security measures that protect a company's computer networks and systems. [READ MORE »](#)

#### PROJECTED JOBS

**47,100**

#### MEDIAN SALARY

**\$103,590**

#### EDUCATION NEEDED

**Bachelor's**



## Software Developer

### 💡 #2 in Best Technology Jobs

Software developers need to be innovative, creative and, of course, technical in order to succeed in this field. They might write new code or fix bugs in code to make it work better. [READ MORE »](#)

#### PROJECTED JOBS

**409,500**

#### MEDIAN SALARY

**\$110,140**

#### EDUCATION NEEDED

**Bachelor's**



## Data Scientist

### 💡 #3 in Best Technology Jobs

Data scientists use technology to glean insights from large amounts of data they collect. [READ MORE »](#)

#### PROJECTED JOBS

**19,800**

#### MEDIAN SALARY

**\$98,230**

#### EDUCATION NEEDED

**Bachelor's**

# Dove sono meglio pagati i SwEng

Best Cities for Software Engineers 2019: A Data-Backed Analysis					
Real Earnings (USD) Ranking					
Rank	USA Cities	Real Earnings	Rank	International Cities	Real Earnings
1	Seattle	50,438	1	Berlin	36775.68
2	San Jose	47,631	2	Tel Aviv	29072.12
3	Houston	41,830	3	Montreal	28306.92
4	Dallas	41,669	4	Vancouver	25431.88
5	Baltimore	41,387	5	Sydney	23367.08
6	Phoenix	41,097	6	Toronto	22673.2
7	Austin	39,111	7	Tokyo	21927.28
8	Raleigh	38,480	8	Melbourne	19400.8
9	San Diego	36,863	9	Oslo	18190.64
10	Detroit	36,798	10	Paris	18001.72
11	Philadelphia	36,239	11	Amsterdam	17310.76
12	Portland	35,883	12	Taipei	13914.12
13	Denver	35,845	13	London	11187.16
14	Atlanta	34,646	14	Warsaw	10286.64
15	Minneapolis	33,677	15	Sao Paulo	8608.92
16	Los Angeles	32,849	16	Singapore	7875.24
17	San Francisco	32,153	17	Seoul	5614.64
18	Chicago	31,943	18	Hong Kong	4277.68
19	Washington DC	29,429	19	Moscow	2737.6
20	Boston	29,241	20	Bangalore	807.12
21	New York	23,515	21	Beijing	-194.76
			22	Shanghai	-2302.52

Best Cities for Software Engineers 2021: A Data-Backed Analysis					
Real Earnings (USD) Ranking 2021					
Rank	USA Cities	Real Earnings	Rank	International Cities	Real Earnings
1	Seattle	41,164	1	Berlin	43,904
2	Houston	35,544	2	Tel Aviv	32,698
3	Phoenix	35,534	3	Montreal	30,983
4	Austin	34,975	4	Vancouver	25,082
5	Dallas	34,950	5	Sydney	22,147
6	San Jose	34,167	6	Tokyo	21,766
7	Raleigh	32,093	7	Toronto	21,567
8	Miami	30,959	8	Melbourne	20,546
9	Baltimore	30,457	9	Amsterdam	18,082
10	Denver	30,194	10	Oslo	16,415
11	Portland	29,811	11	Paris	15,612
12	Minneapolis	27,884	12	Singapore	12,783
13	Atlanta	27,490	13	Taipei	11,657
14	Detroit	27,086	14	Warsaw	9,957
15	San Diego	26,959	15	London	7,534
16	Los Angeles	26,526	16	Moscow	6,021
17	Philadelphia	25,884	17	Sao Paulo	6,016
18	Chicago	25,589	18	Seoul	5,105
19	San Francisco	24,968	19	Beijing	1,948
20	Washington DC	22,618	20	Hong Kong	1,078
21	Boston	20,166	21	Bangalore	-170
22	New York	10,425	22	Shanghai	-983

[bit.ly/dev-salary-2019](https://bit.ly/dev-salary-2019)

arc()

[m.arc.dev/best-dev-cities-2021](https://m.arc.dev/best-dev-cities-2021)

# Come vengono pagati gli sviluppatori di sw?

# Produttività

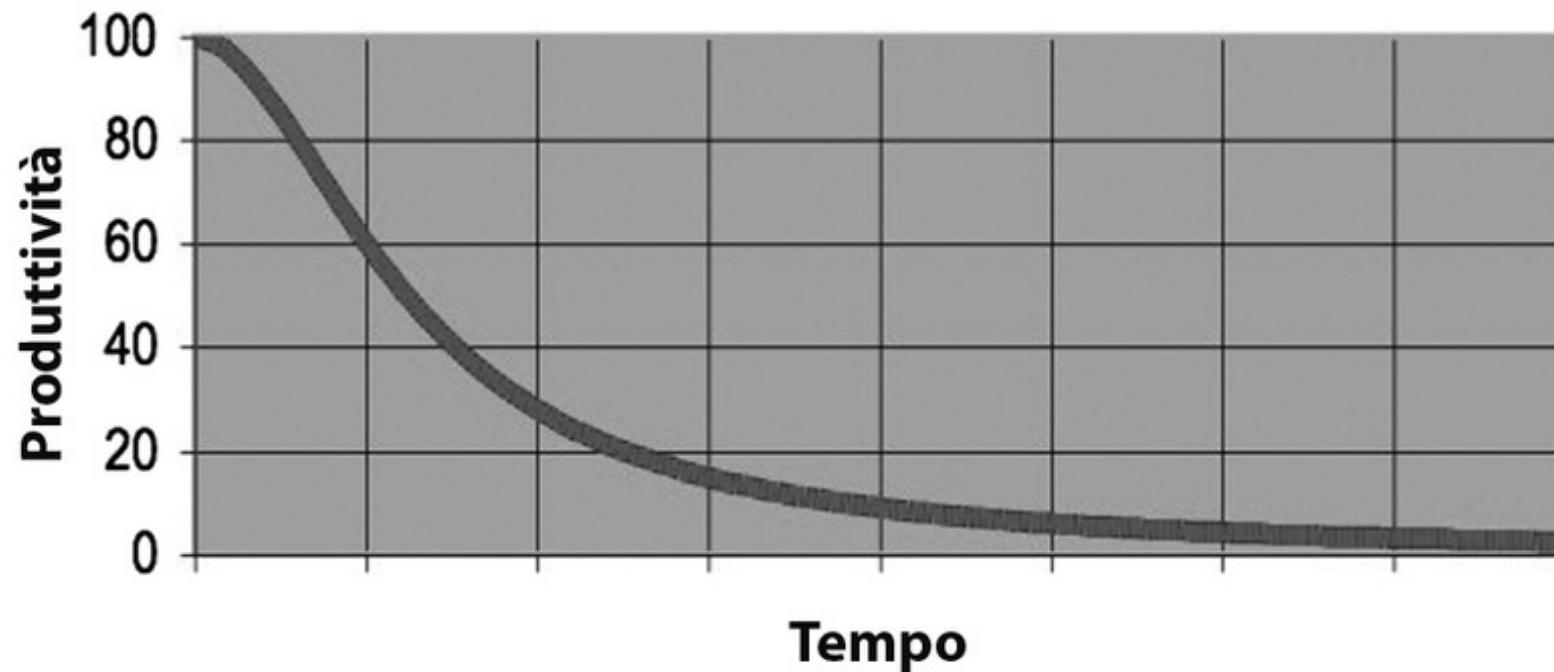
- La produttività è il rapporto tra la **quantità** di beni o servizi prodotti ed il **costo del lavoro** necessario a produrli
  - Output/Input.

Esempio: produco 1000 caramelle al costo di 100 euro.  
La produttività è: 10 caramelle per ogni euro speso

# Produttività degli sviluppatori

- La produttività dello sviluppo software è il rapporto tra **software prodotto** (misurato in LoC: Lines of Code) e il **costo dello sforzo (effort, misurato in giorni/persona)** di produrlo
  - LoC/effort (esempio: 50 LoC per giorno/persona)

# Produttività nel software



# Misurare la produttività



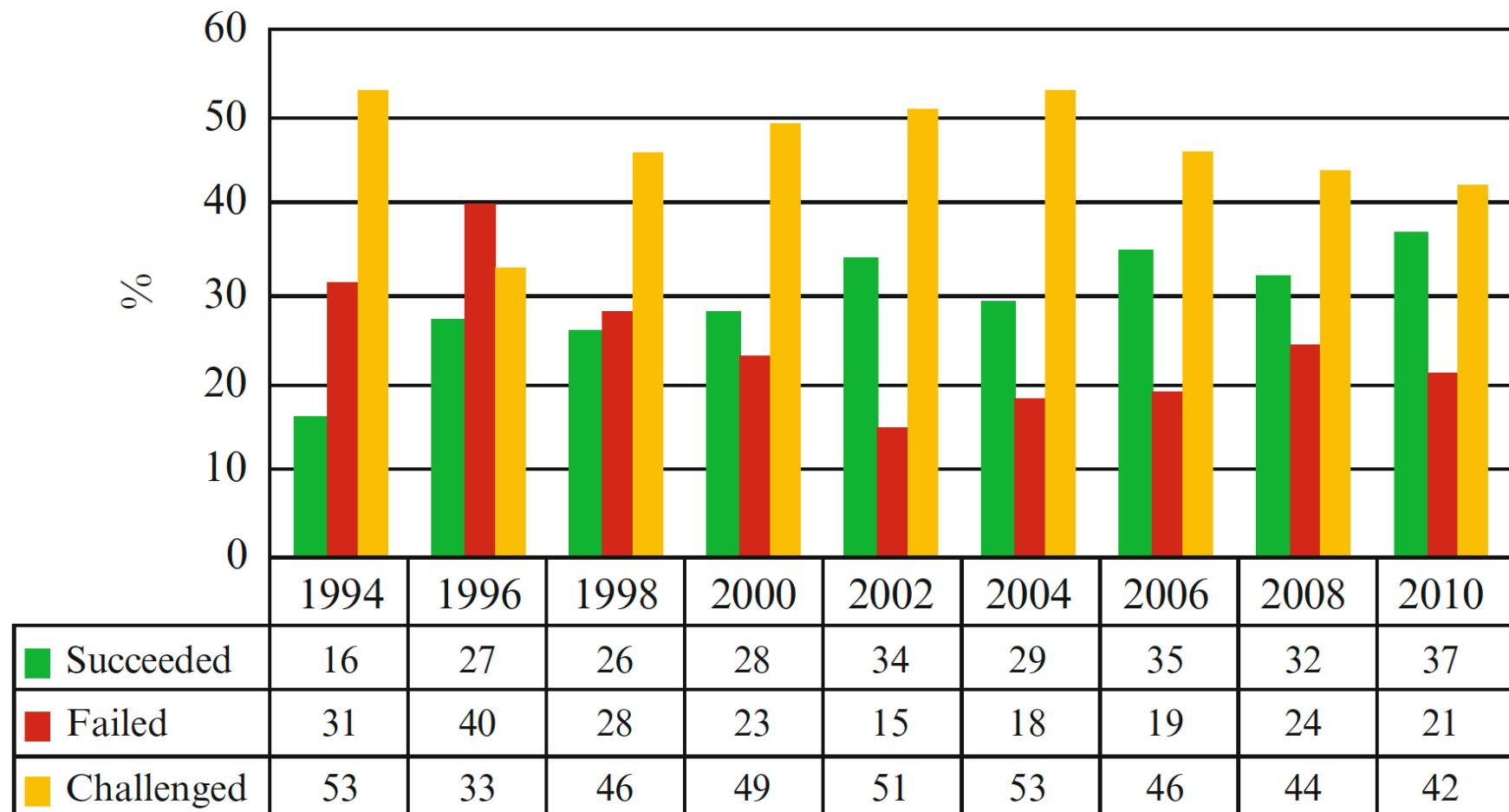
# La produttività dell'industria sw è bassa

- 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
  - 55 miliardi \$ di spreco considerando solo i progetti USA

*Standish Report 2003*

# Standish CHAOS reports

Standish figures 1994-2010



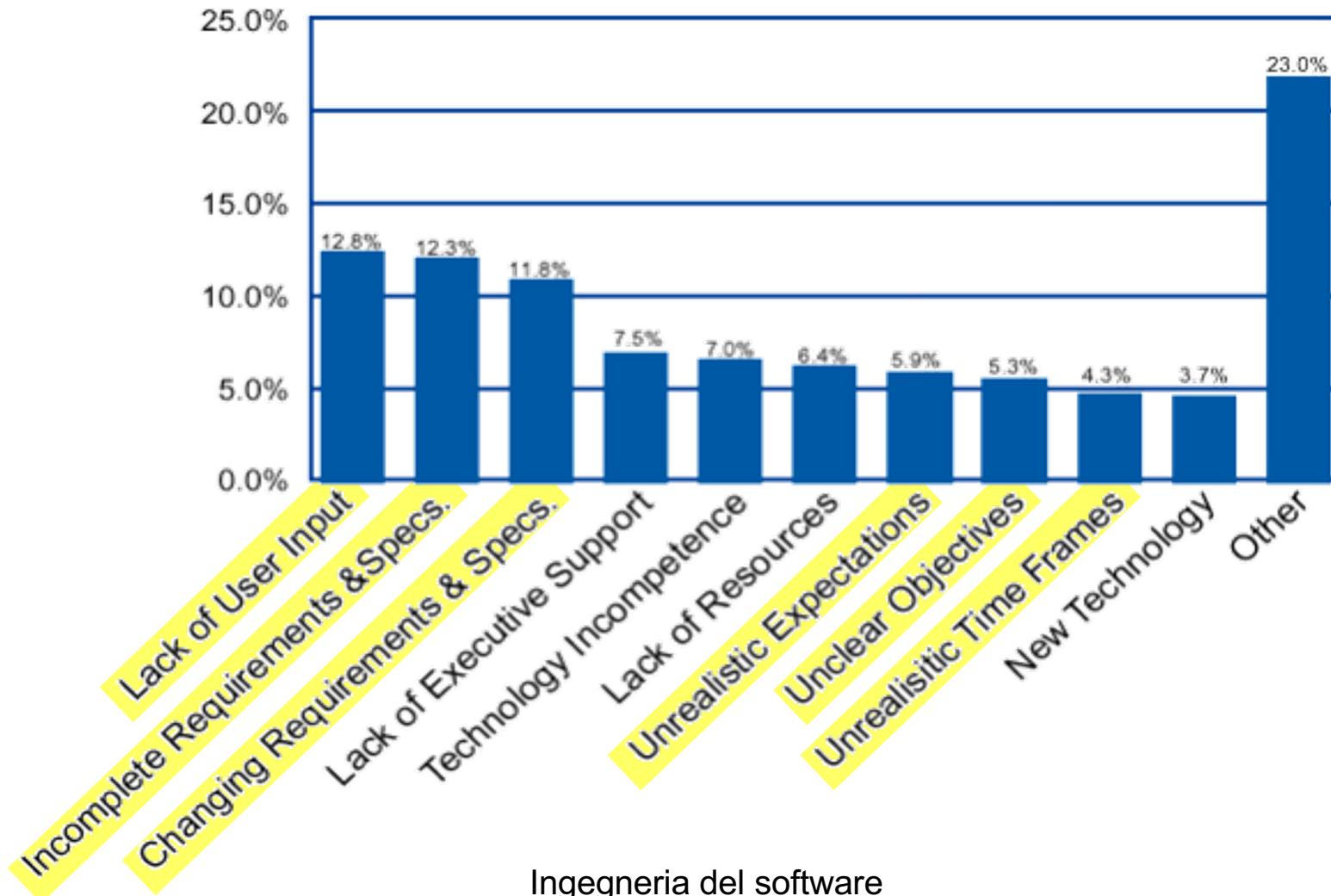
# Agile vs waterfall (CHAOS 2015)

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

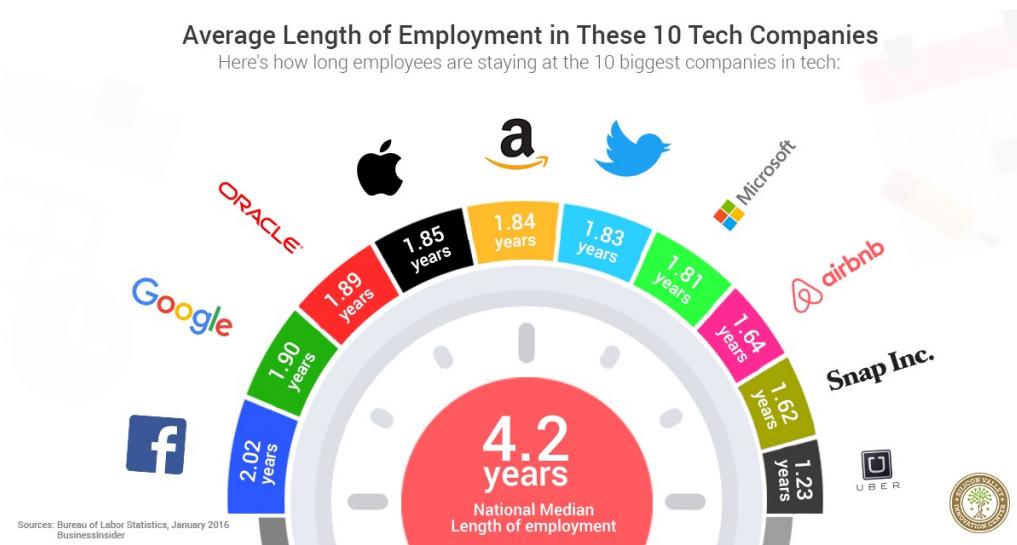
Quali sono i rischi principali di chi sviluppa software?

- 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

# Turnover dello staff

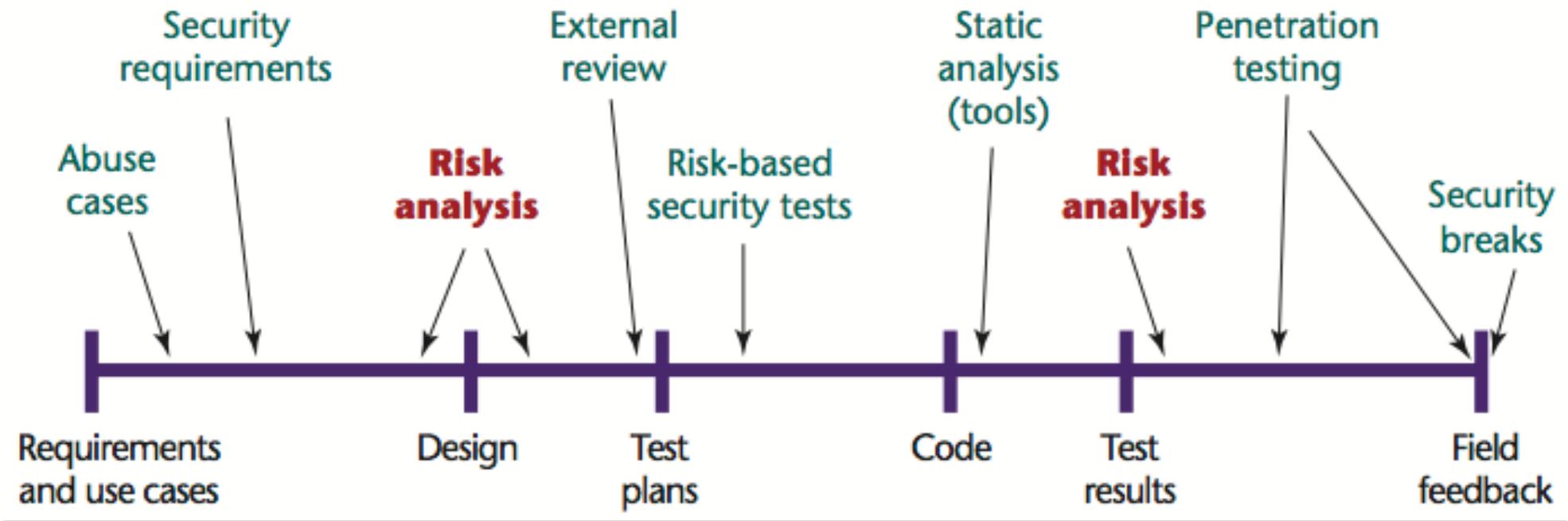
Durate media degli impieghi (2017):

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)

# Analisi dei rischi nel ciclo di vita del software



# 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

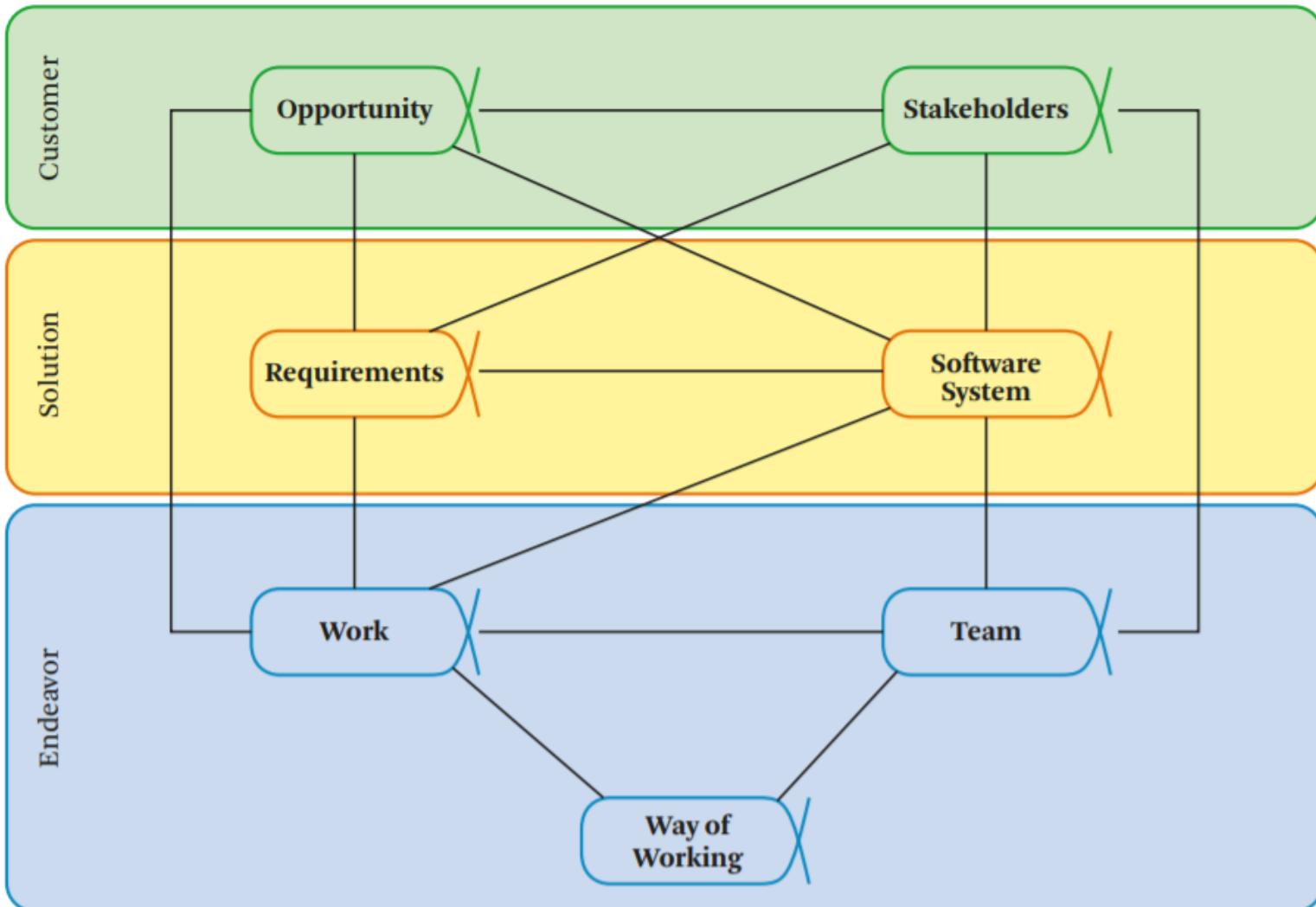
# Le competenze richieste nello sviluppo del sw

- Software requirements – analisti dei *requisiti*
- Software design – *progettisti della struttura*
- Software construction – *scrittura del codice*
- 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; ruoli e strumenti
  - Cattura, specifica, analisi e gestione dei requisiti
  - Progettazione dell'architettura e dei moduli di codice
  - Codifica e debugging del codice
  - Testing (verifica e validazione)
  - Deployment
- Manutenzione del software rilasciato
- Gestione della configurazione
- Project management
- Controlli di qualità del software

# Essence: verso una teoria dell'ingegneria del sw



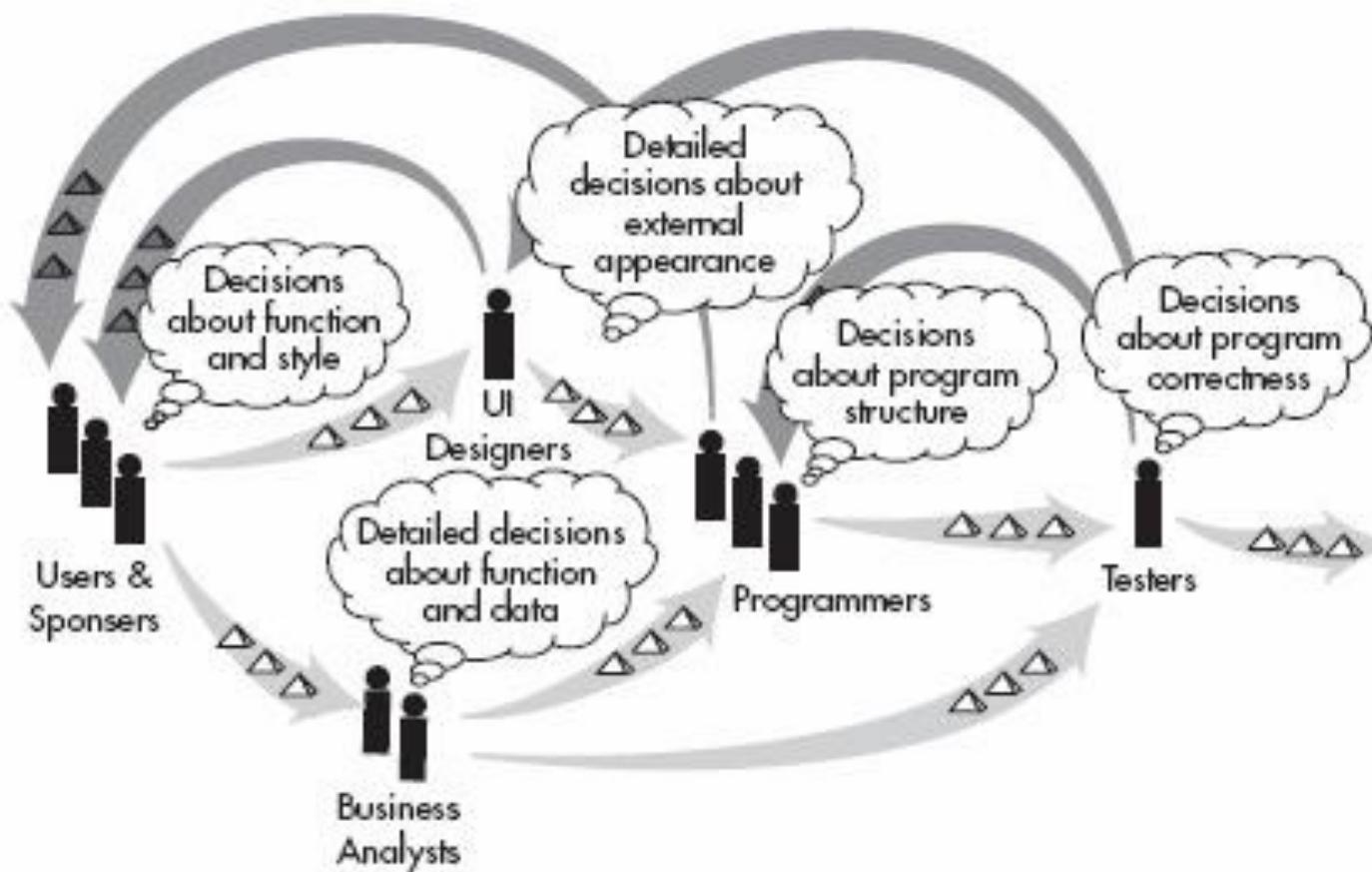
# Parti interessate (stakeholders)

## Tipi di stakeholders

- Progettisti professionisti
- Management
- Personale tecnico
- Decisori
- Utenti
- Finanziatori
- ...

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

# Decisioni degli stakeholders



# Processi di produzione

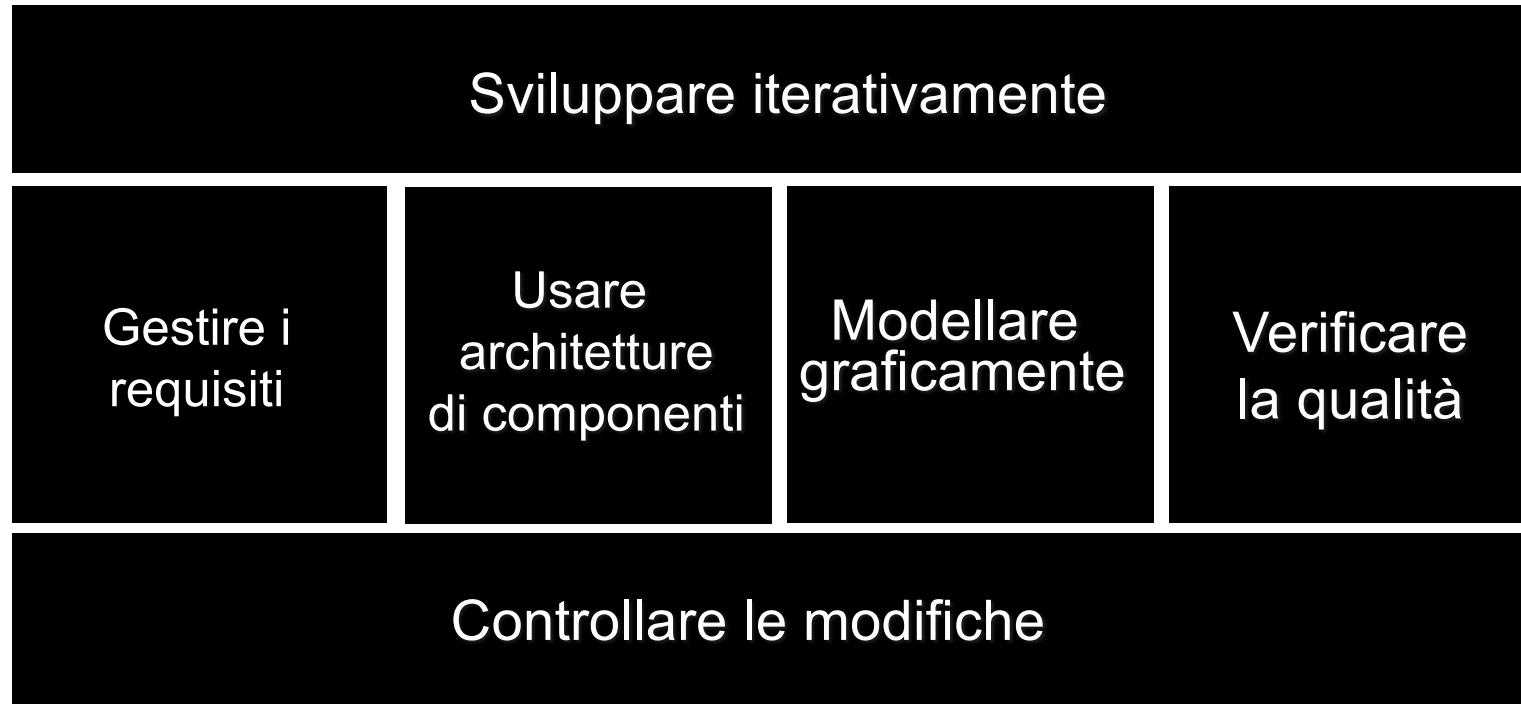
- I processi di produzione si creano e poi evolvono
- Prodotti e processi possono essere descritti e valutati da un punto di vista **qualitativo**
- Processi di produzione a diversi livelli:
  - Ciclo di vita industriale
  - Ciclo di sviluppo: analisi dei requisiti, design, testing
  - Progettazione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

# Discussione

Come si costruisce un prodotto software?  
Come si misura?



# Principi guida dello sviluppo software

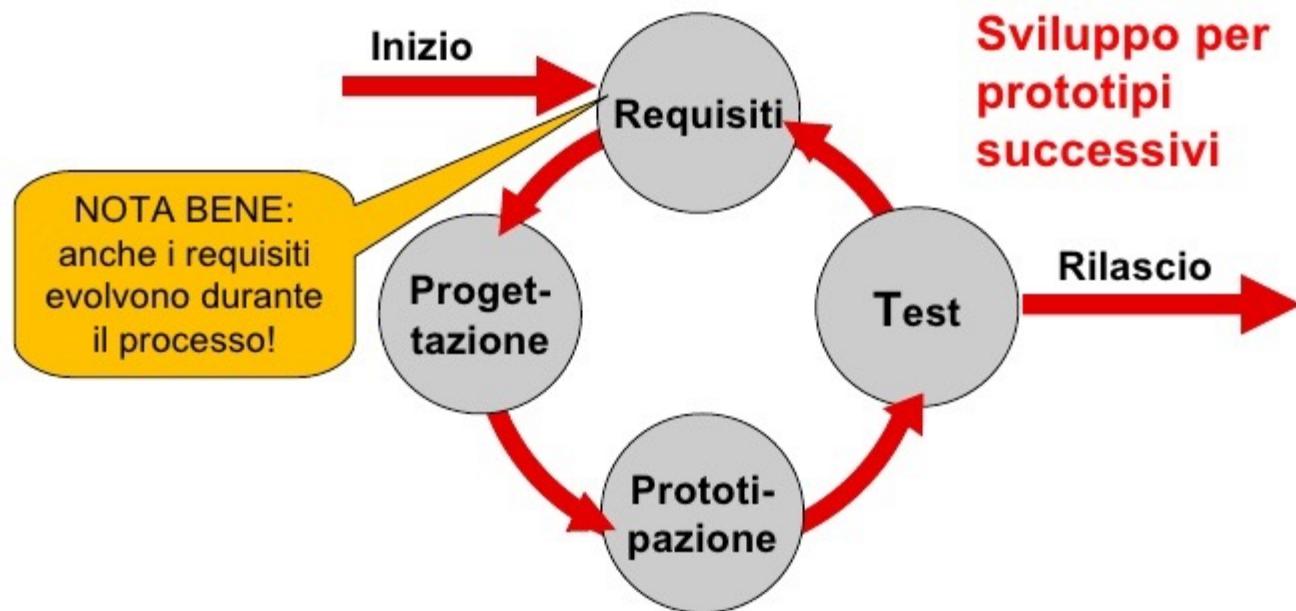


# Modelli del software

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

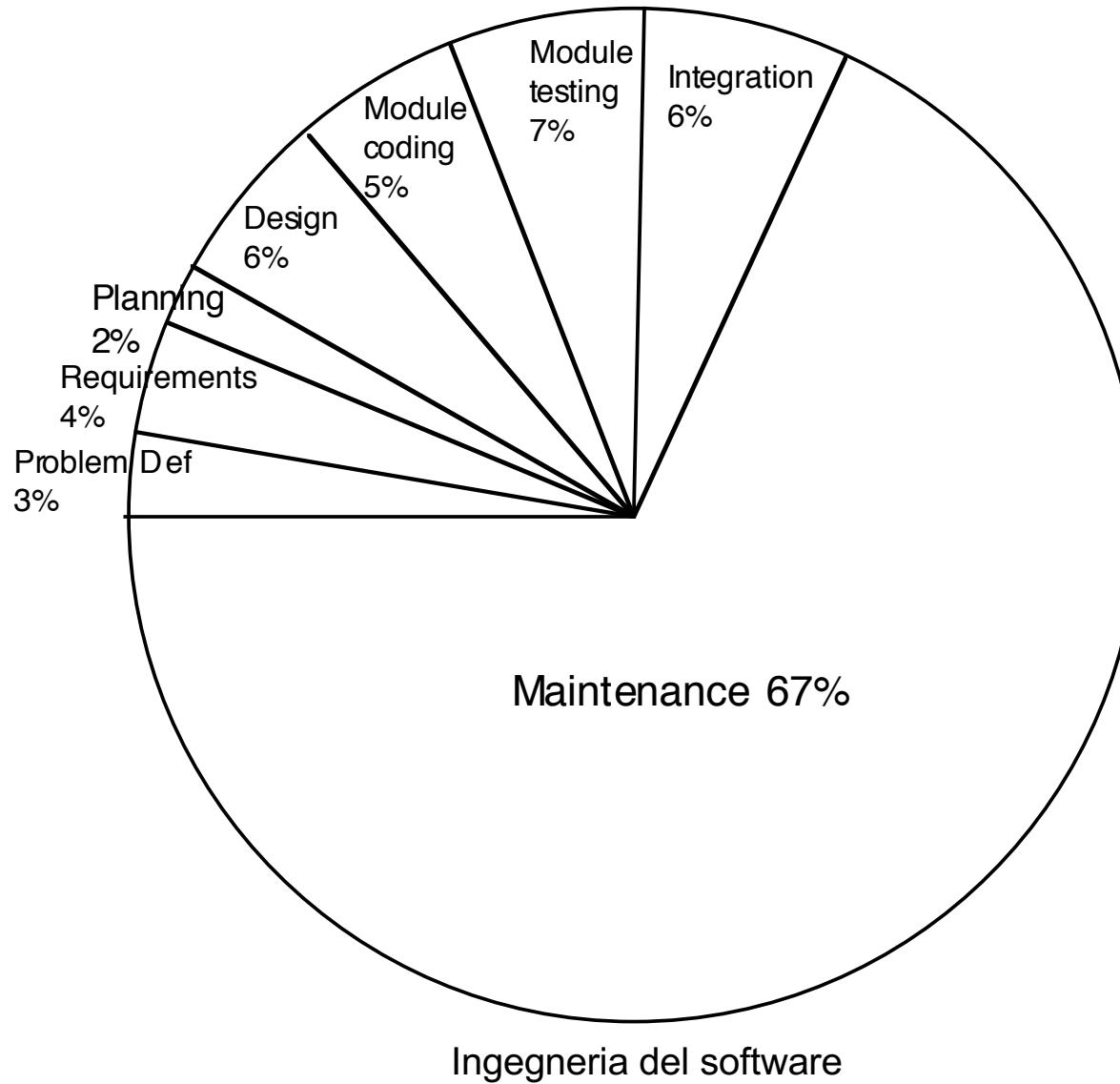


# Il ciclo di vita del software

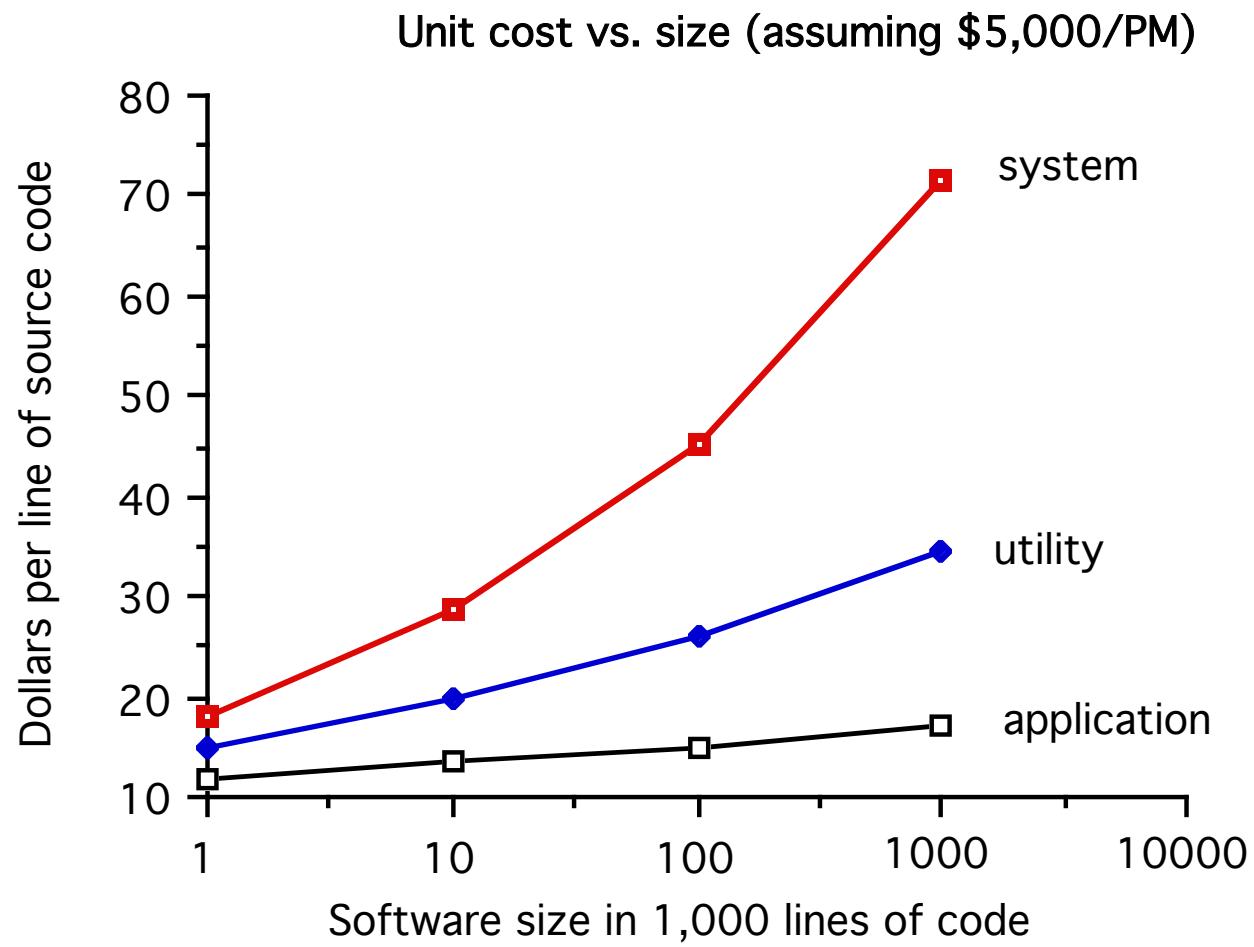
- Requisiti: analisi e specifica
- La progettazione: modellazione dell'architettura e dei singoli componenti
- La codifica ed il debugging
- Il testing e la verifica
- Il deployment (= la messa in opera)
- La manutenzione

# Costi di Sviluppo

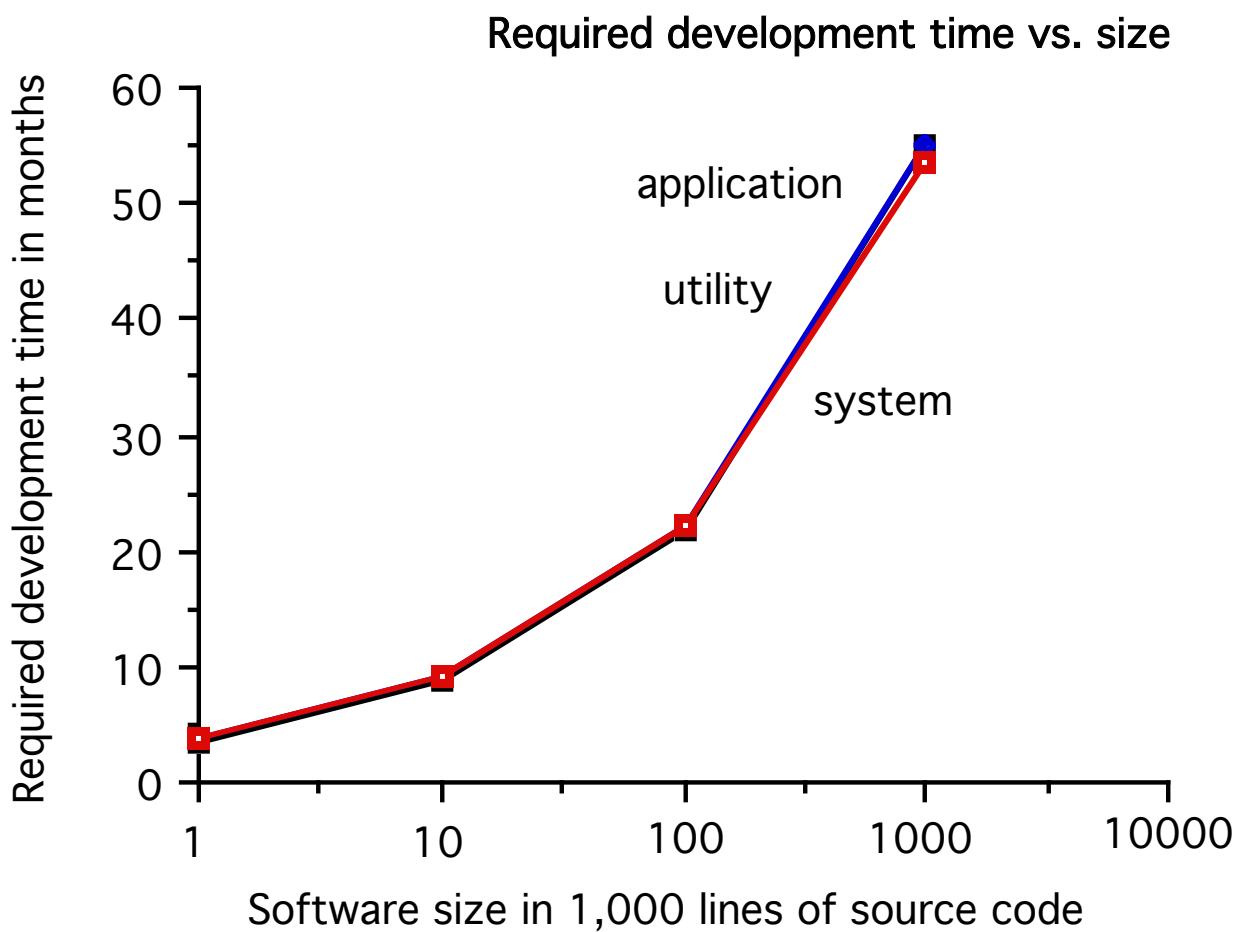
(Boehm citato da Schach)



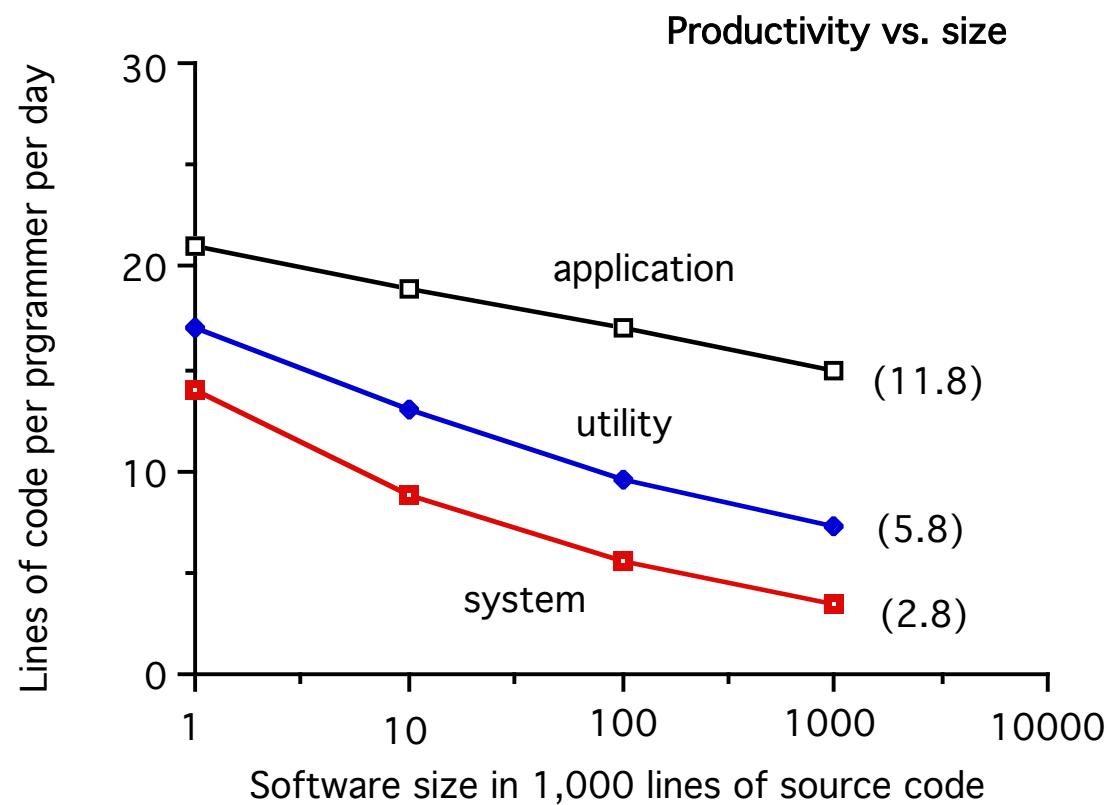
# Costo per linea di codice



# 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

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

**Il mondo cambia continuamente  
La manutenzione è “normale”**

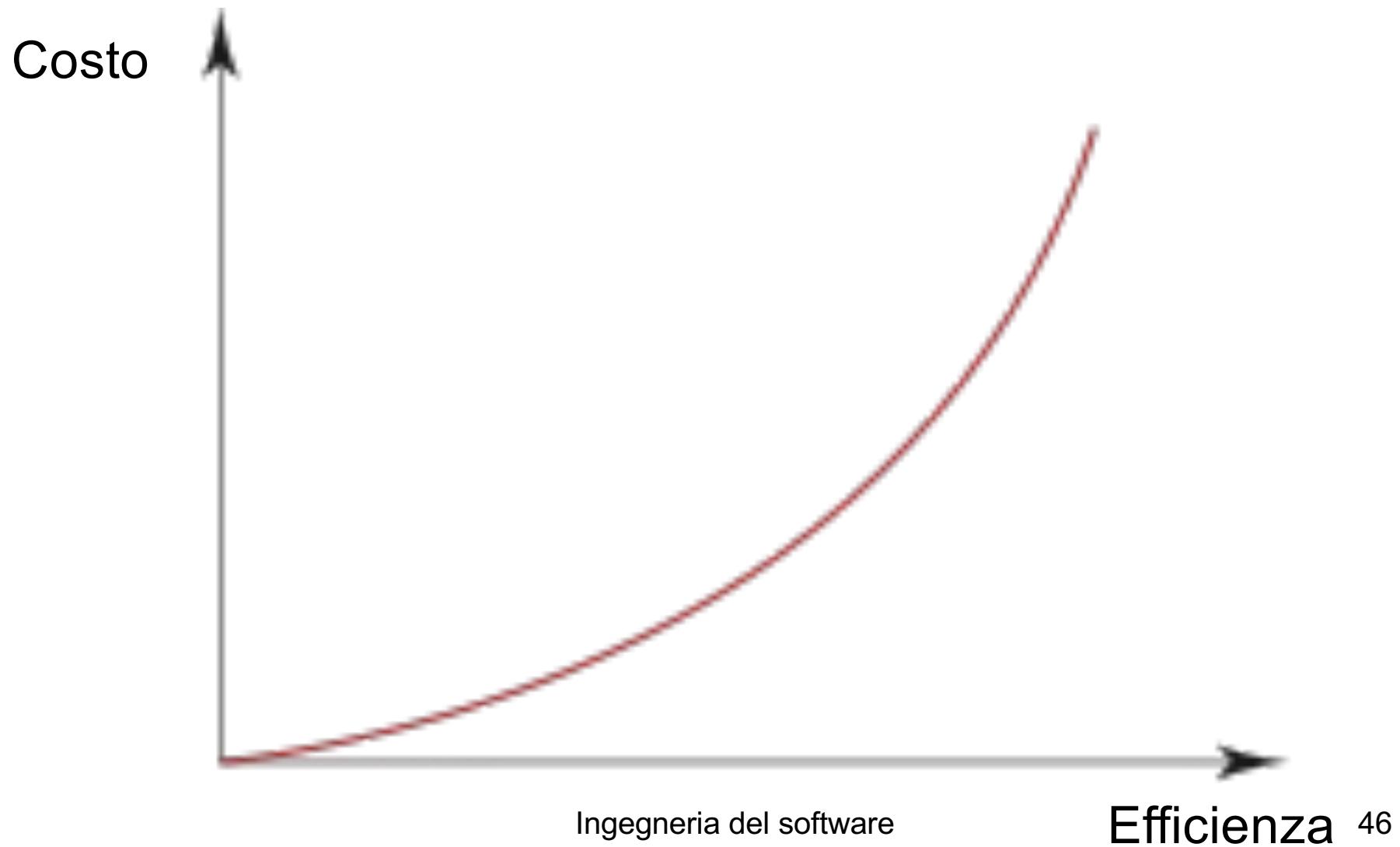
# Attributi dei prodotti software

- 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

# 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

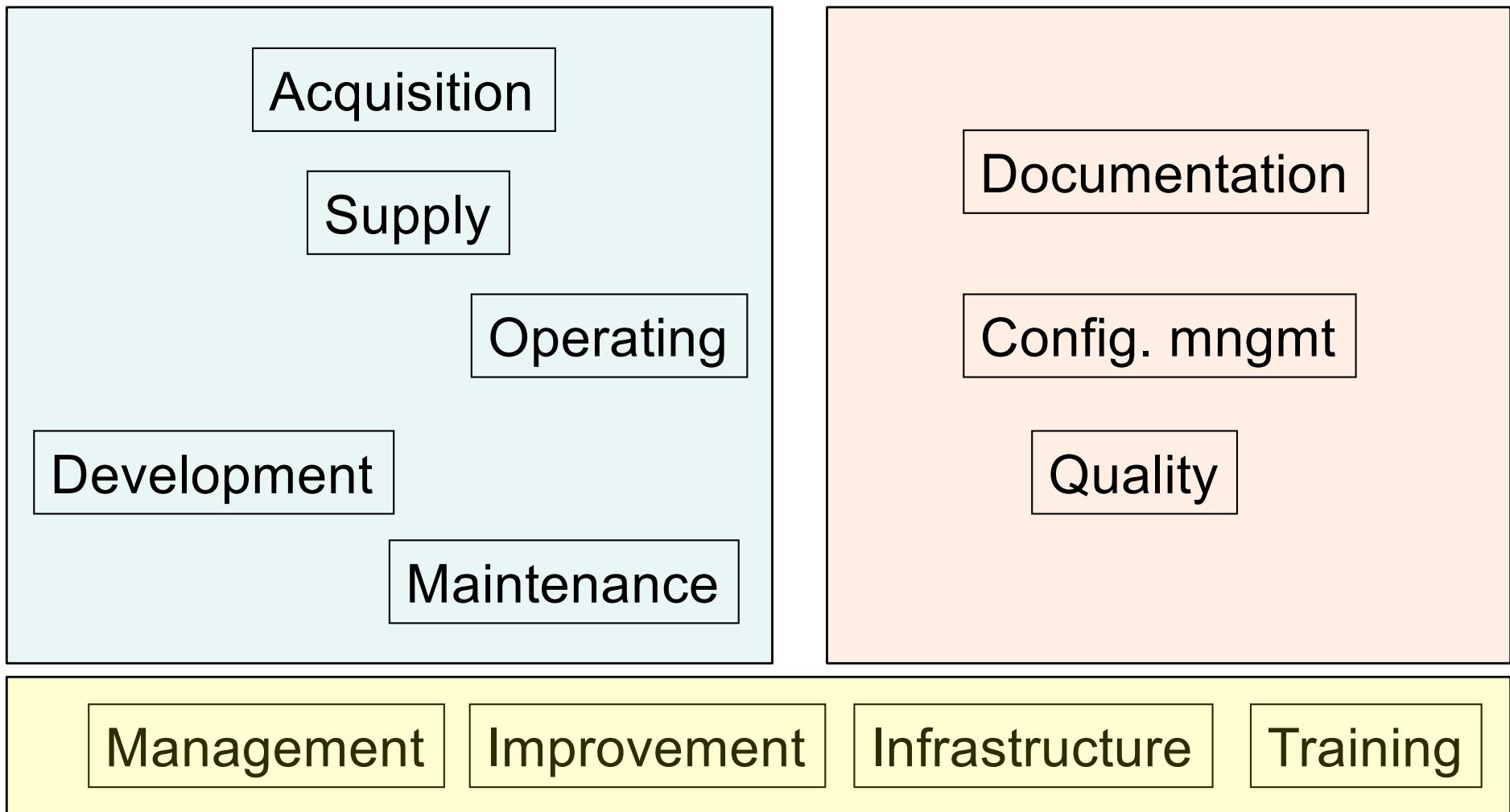
## Standard principali software engineering IEEE

- 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
  - **Supporto**: audit, configuration management, documentation, quality assurance, verification, validation
  - **Organizzative**: management, infrastructure, improvement, training

# Lo standard IEEE12207



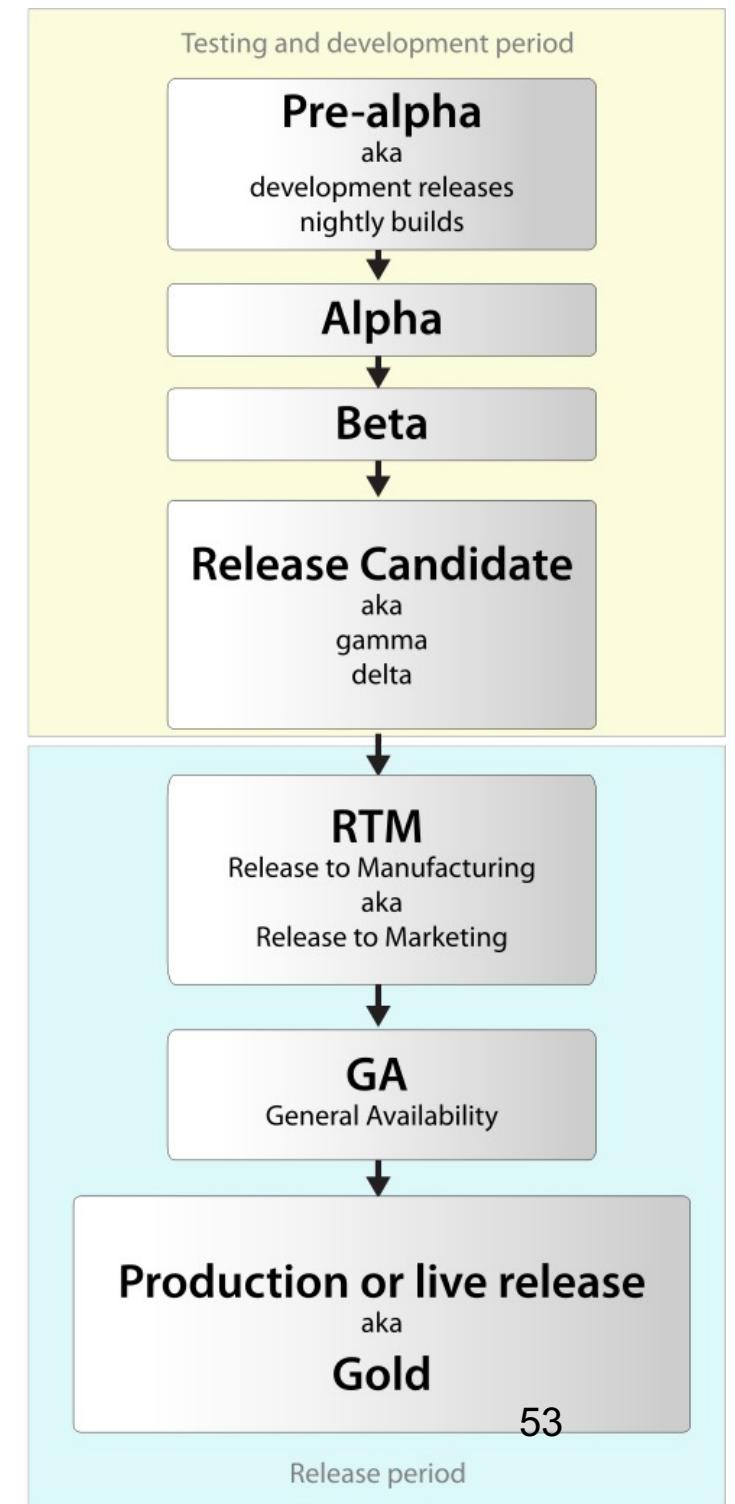
# II SWEBOK: sw engineering Body of Knowledge

## Knowledge areas SWEBOk 3.0

<b>Table I.1. The 15 SWEBOK KAs</b>
Software Requirements
Software Design
Software Construction
Software Testing
Software Maintenance
Software Configuration Management
Software Engineering Management
Software Engineering Process
Software Engineering Models and Methods
Software Quality
Software Engineering Professional Practice
Software Engineering Economics
Computing Foundations
Mathematical Foundations
Engineering Foundations

# Ciclo di vita di un rilascio sw

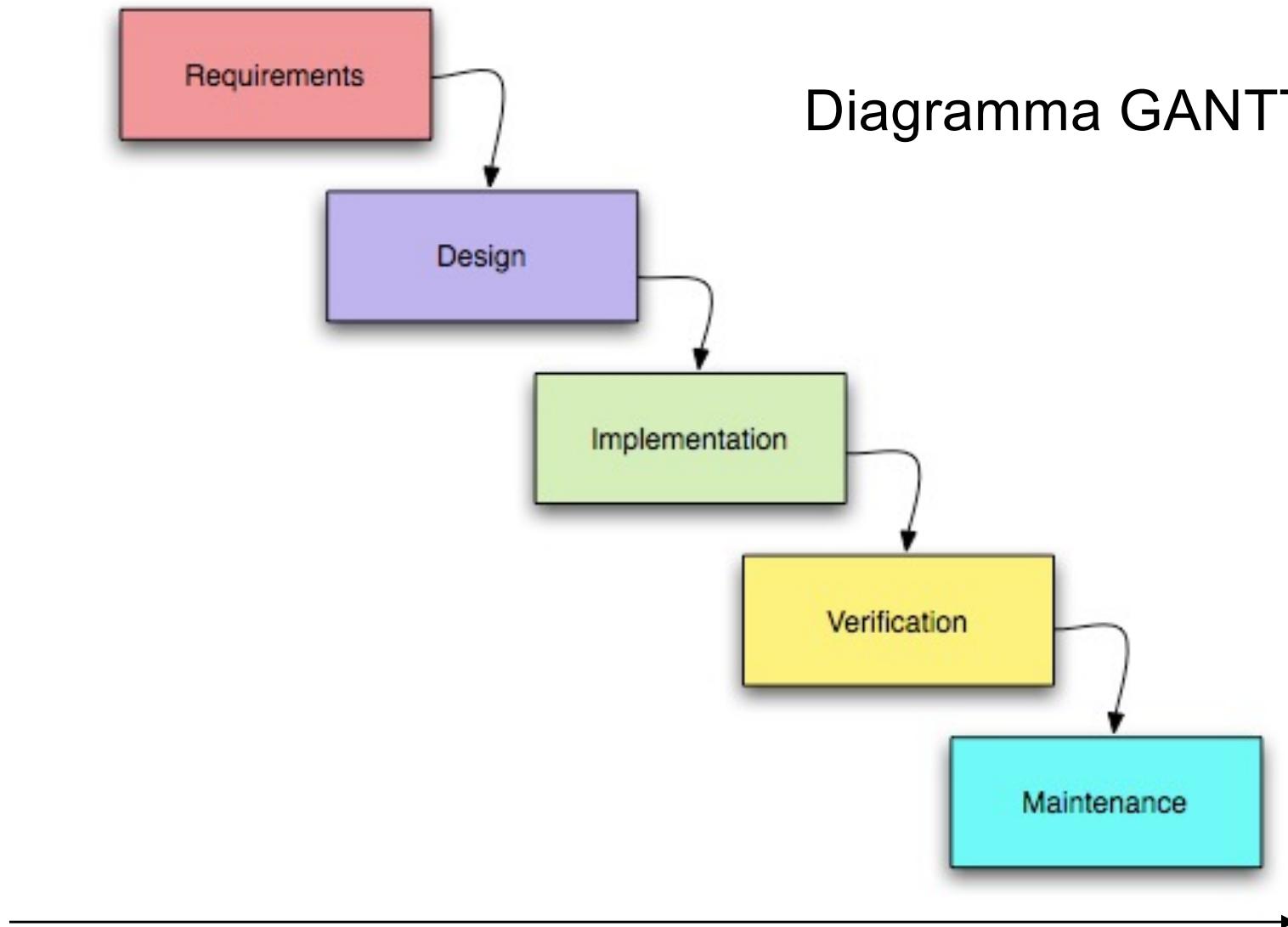
- Un **rilascio software** (*software release*) è una versione di un prodotto che viene immessa sul mercato
- Il *ciclo di vita della release* è l'insieme delle fasi del suo sviluppo e della sua vita operativa



# 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:
  - **Specifiche** 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**

## Diagramma GANTT



# Il processo di sviluppo del sw

- **Processo software**: insieme dei ruoli, delle attività e dei documenti necessari per creare un sistema software

# Esempi: ruoli attività documenti

- Esempi di **ruoli**: stakeholder, progettista, sviluppatore, tester, manutentore, ecc.
- Esempi di **attività**: programmare, testare, fare una riunione, fare una demo, documentare
- 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

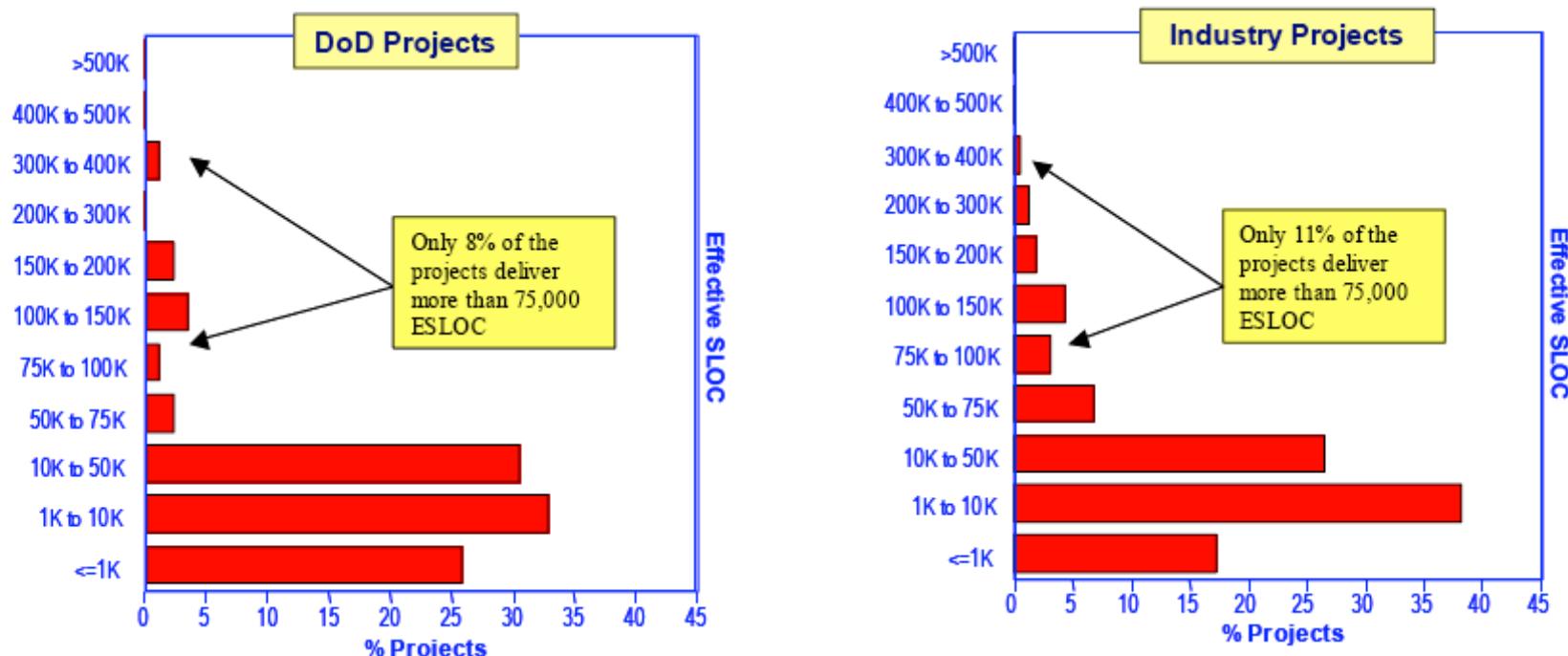
Practice / No. of Projects	India	Japan	US	Europe	Total
Architectural Specification	24	27	31	22	104
Functional Specification	83.3%	70.4%	54.8%	72.7%	69.2%
Detailed Design	95.8%	92.6%	74.2%	81.8%	85.6%
Code Generation	100.0%	85.2%	32.3%	68.2%	69.2%
Design Review	62.5%	40.7%	51.6%	54.5%	51.9%
Code Review	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
<b>No. of Projects</b>	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

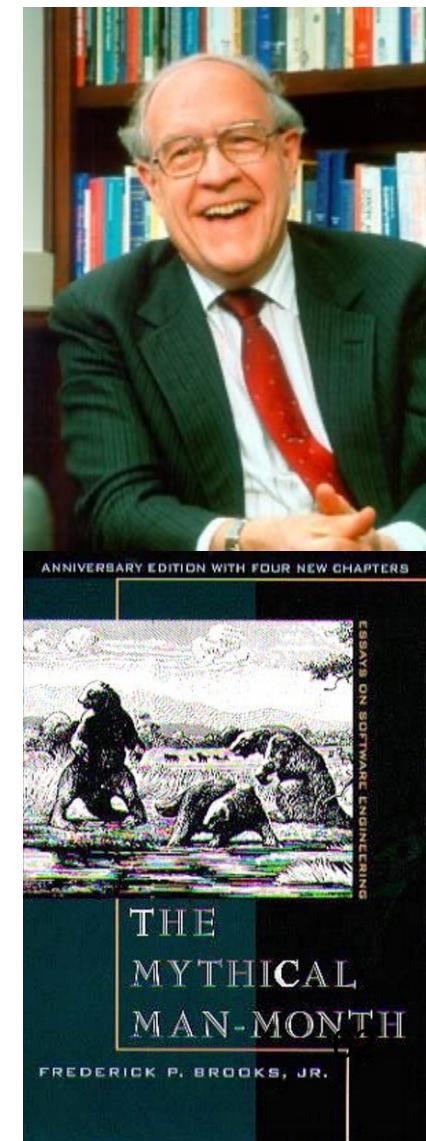
Figure 1. Frequency Distribution of new engineering software developed with a 12-month time period for DoD and private industry.

# 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”



# Miti e leggende

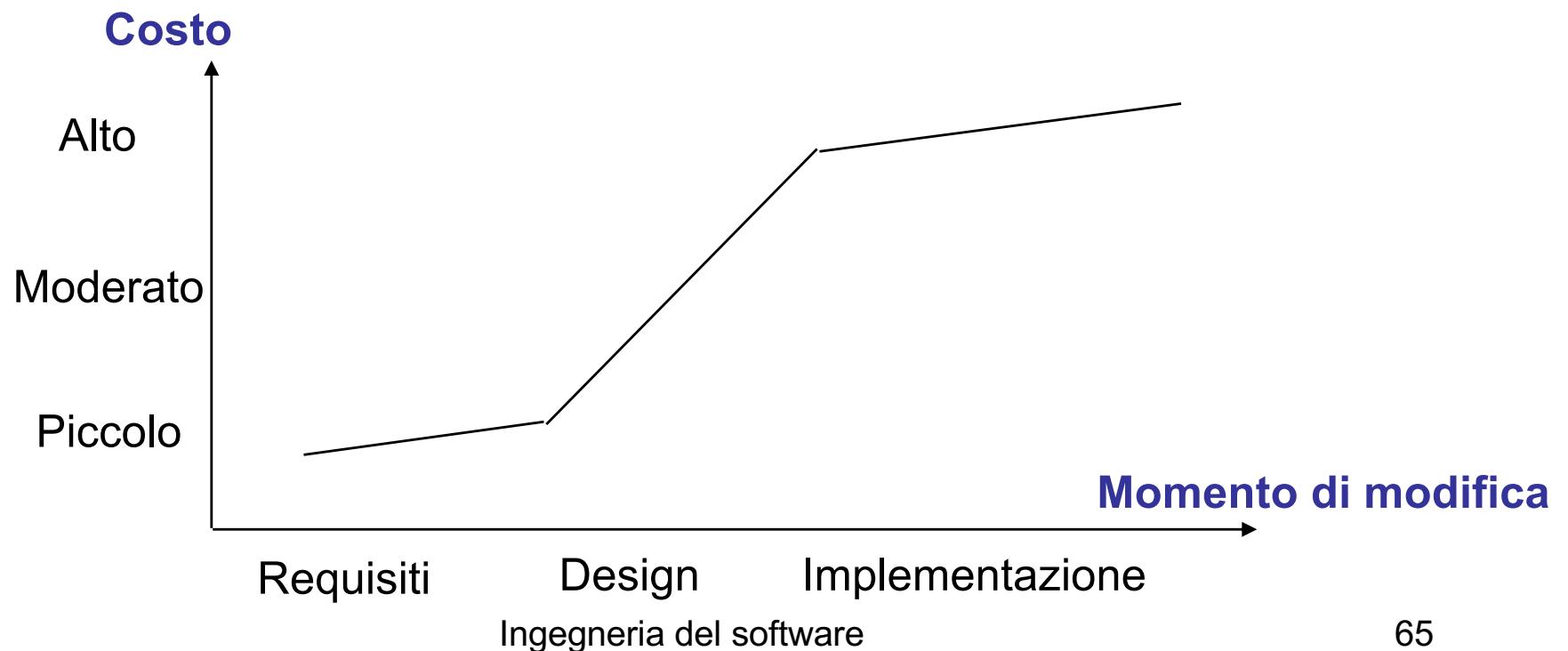
- Se il progetto ritarda, **possiamo aggiungere programmatori e rispettare la consegna**
  - **Legge di Brooks:** “Aggiungere personale ad un progetto in ritardo lo fa ritardare ancor di più”

# 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**

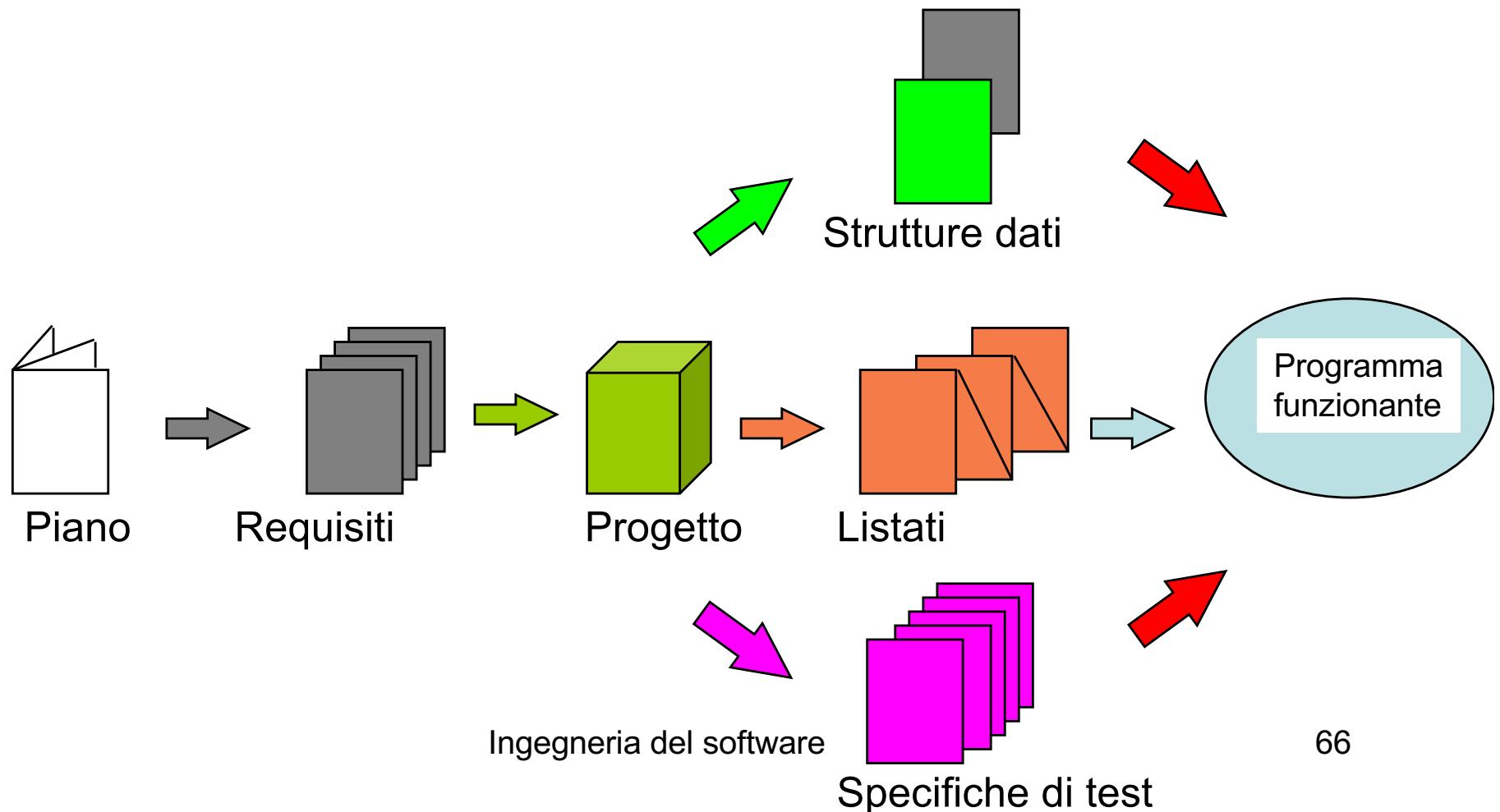
# Miti e leggende

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



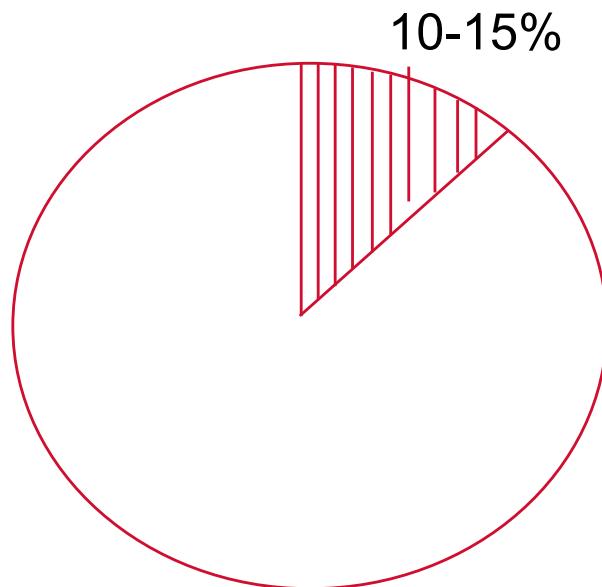
# Miti e leggende

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

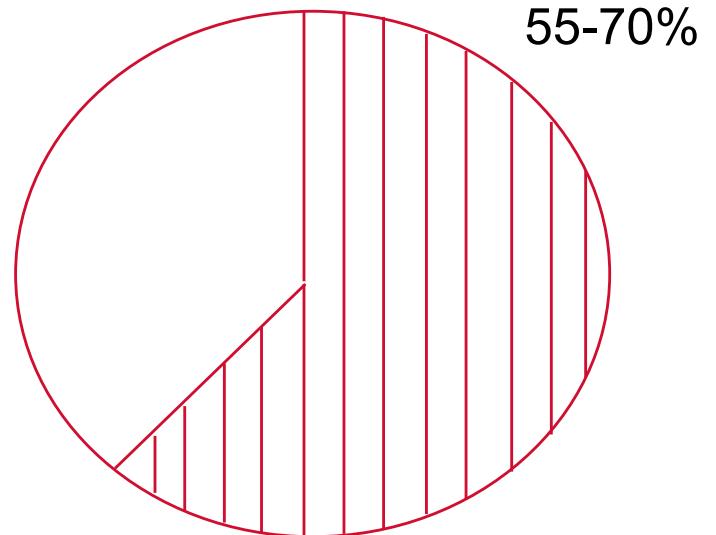


# Miti e leggende

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



Costi di manutenzione preventivi



Costi di manutenzione reali

# 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)
- <https://dokumen.tips/reader/f/guide-to-the-software-engineering-body-of-knowledge-swebok-v3>
- [swebokwiki.org/Main\\_Page](http://swebokwiki.org/Main_Page)
- [essence.ivarjacobson.com/services/what-essence](http://essence.ivarjacobson.com/services/what-essence)

# Domande?

