# Ingegneria del software

**Paola Barra**
**a.a. 2022/2023**

# Modalità d'esame

- Progetto obbligatorio (durante il corso)
  - Gruppi costituiti da 2-3 persone
  - Analisi e progettazione di un sistema software
- Prova orale

# Materiale didattico

Oltre a queste slide...

- Handbook of software engineering, Springer
- Introduzione all'ingegneria del software moderna, Ian Sommerville

## Canale Teams : 1tjk254

Ricevimento a richiesta su Teams.

## Email:

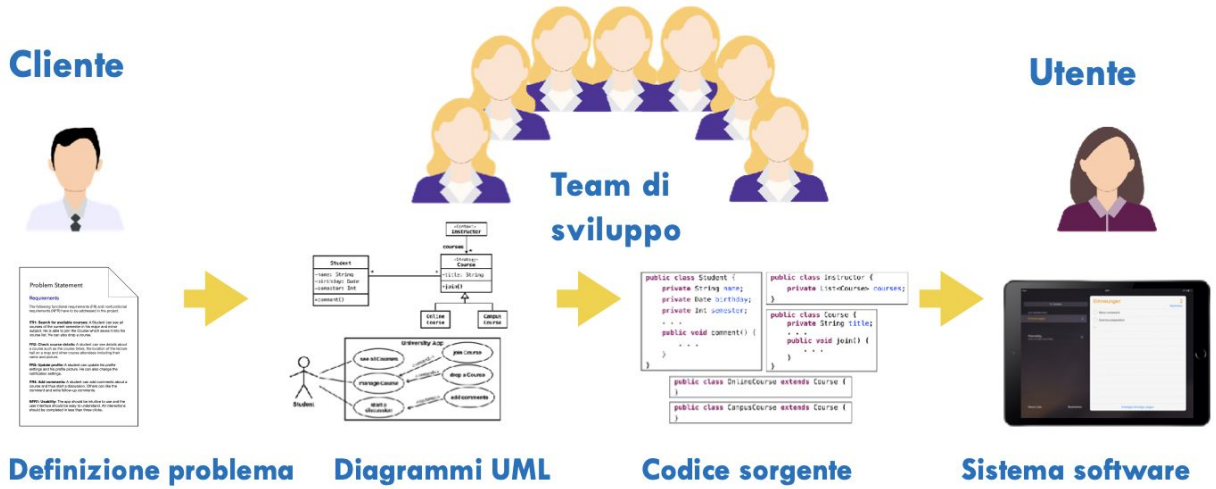
paola.barra@uniparthenope.it

## Orario lezioni

| | | |
|---|---|---|
| Martedì | 14:00 - 16:00 | Aula 18 |
| Giovedì | 11:00 - 13:00 | Aula 18 |

# Obiettivi del corso

- Apprendere ed applicare metodologie, tecniche, flussi di lavoro e strumenti per trasformare la definizione di un problema fornita da un cliente in un sistema software usato dagli utenti finali



# Obiettivi del corso

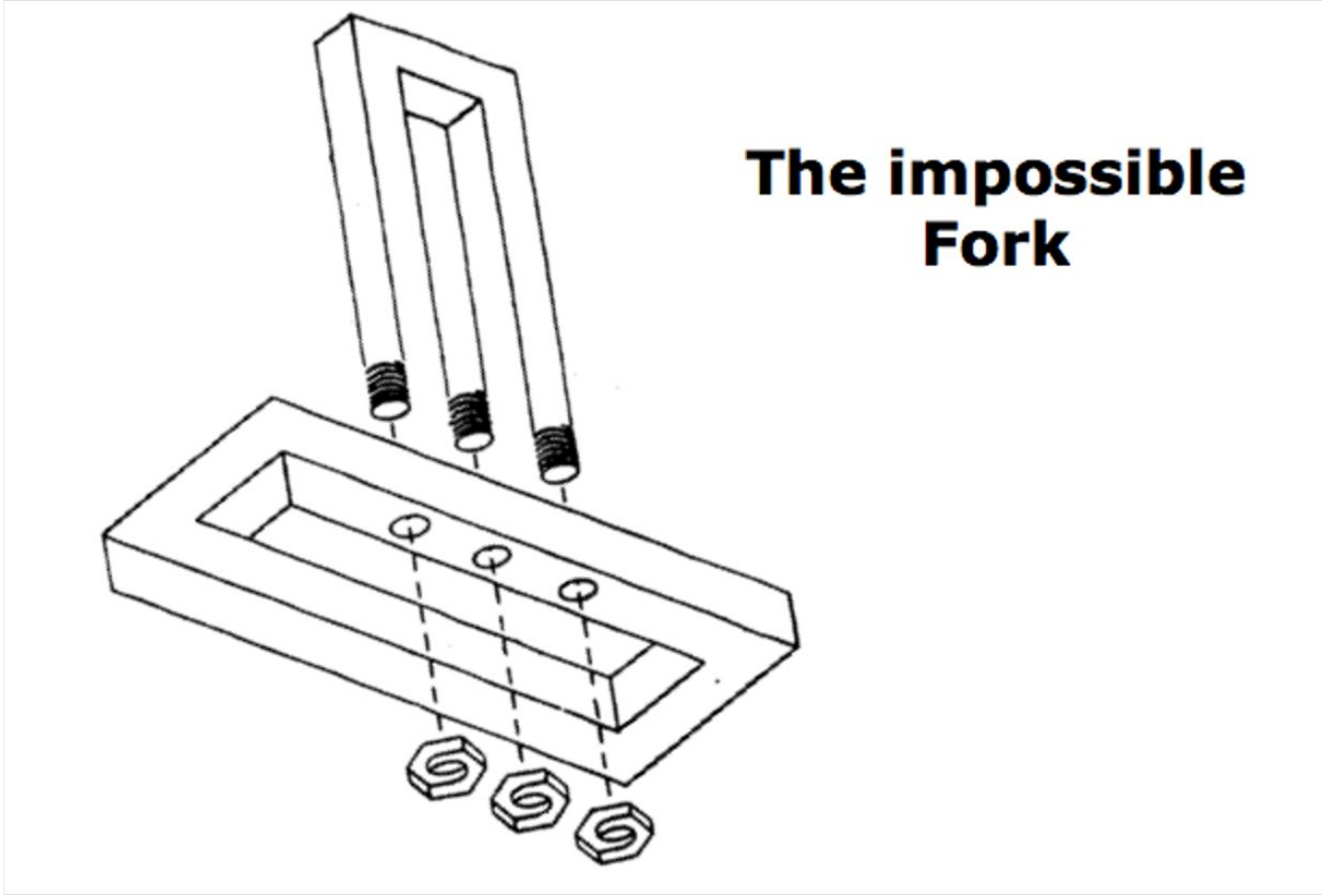
- il corso di Ingegneria del software presenta i metodi, le tecniche e gli strumenti fondamentali, di documentazione della specifica, analisi e progetto di sistemi software complessi da un punto di vista architetturale.

- Affronteremo l'analisi e la specifica dei requisiti e progettazione con UML, Testing, Deployment, Manutenzione e smaltimento.

- il corso richiederà la collaborazione attiva in un progetto da sviluppare in gruppo

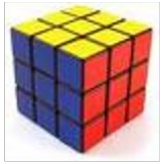
- Questo corso vuole "preparare alla professione"

Siete in grado di sviluppare il sistema?



# L’ingegneria del software non è semplicemente la stesura di codice





- Problem solving
  - Capire il problema
  - Proporre una soluzione e un piano di lavoro
  - Ingegnerizzazione di un Sistema basato sulla soluzione usando una buon progetto
- Modellizzazione (trattare con la complessità)
  - Creare astrazioni e modelli
  - Notazioni per le astrazioni
- Acquisizione di conoscenza
  - Scoperta, analisi, progettazione, validazione del sistema e del processo della soluzione
- Gestione (della logica)
  - Rendere le decisioni di progetto e sviluppo esplicite a tutti gli stakeholder coinvolti

# Ingegneria del software: Una definizione operativa

- L'ingegneria del software è una collezione di tecniche, metodologie e strumenti che aiutano la produzione di

***Un Sistema software di alta qualità sviluppato con un budget fissato entro una scadenza fissata***
***Mentre sono in atto continui cambiamenti***

**Sfida: trattare con la complessità ed il cambiamento**

# Tecniche, metodologie e strumenti

- **Tecniche**
  - Procedure formali per la produzione di risultati usando una qualche notazione ben definita
- **Metodologie:**
  - Raccolta di tecniche applicate durante lo sviluppo del software e unificate da un approccio *filosofico*
- **Tool:**
  - Strumento o sistemi automatizzati per realizzare una tecnica
  - CASE = Computer Aided Software Engineering

# Perché un corso di Ingegneria del Software?

- Produrre software non è (solo) un'arte e neppure (solo) una scienza: è un'industria
  - Lavoro inserito in un contesto di gruppo e azienda
  - Vincoli economici e requisiti di qualità
- Come in ogni industria, per produrre software sono state sviluppate metodologie di progetto, di sviluppo e di verifica
- Un informatico deve necessariamente conoscerle
  - Non basta essere i migliori programmatori
  - Bisogna essere in grado di **analizzare**, **progettare** e **gestire** un progetto software nella sua interezza

# Perchè ingegneria del software?

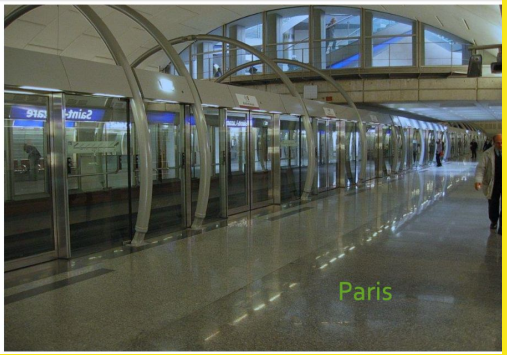
1 artigiano, 10 gg., 5.000 euro

1000 persone
330 imprese
2 anni - 7/24
202 milioni per progettazione e costruzione

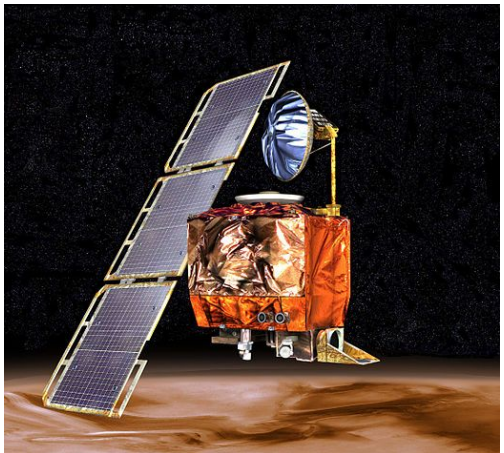
La differenza che c'è tra costruire un ponticello e un grande ponte.

La differenza che c'è tra un piccolo software "artigianale" e un grande programma con migliaia di utenti

# Esempi di gestione del software.

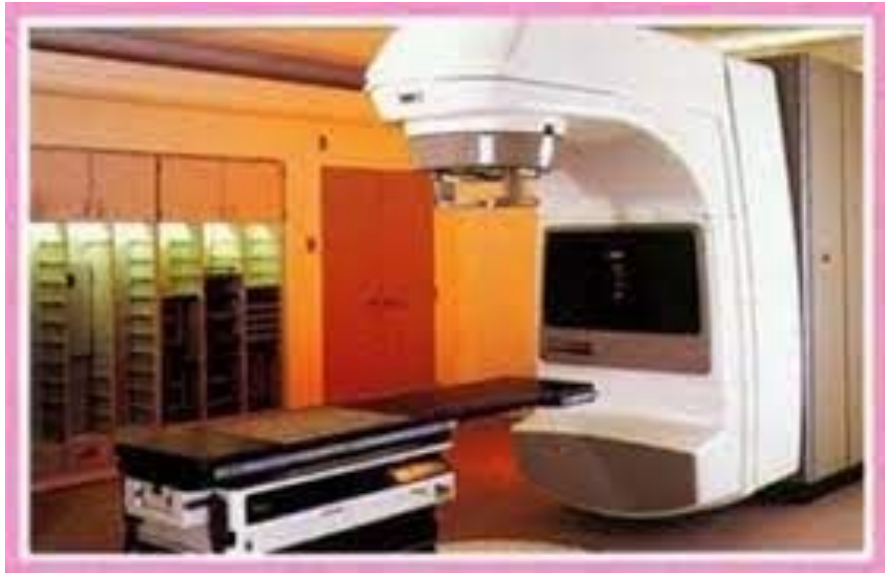




# Terac 25

Uno dei primi sistemi che gestiva in automatico la radioterapia in USA e Canada.



Si inserivano i dati dell'utente, ma l'interfaccia non era progettata in base al reale utilizzo.

Risultato:

6 persone si sono bruciate, 3 sono morte.

Cosa non funzionava?  Errore nel software, non c'era un meccanismo di sincronizzazione software/hardware

# Patriot, 1991

Sistema antimissile intelligente usato durante la guerra del Golfo.

- Una caserma a Dhahran (Arabia Saudita) colpita per un difetto nel sistema di guida: 28 soldati americani morti.
- Concepito per funzionare ininterrottamente per un massimo di 14 ore. Fu usato per 100 ore: errori nell' orologio interno del sistema accumulati al punto da rendere inservibile il sistema di tracciamento dei missili da abbattere
- Scarsa robustezza

# Patriot : il problema sottovalutato

Orologio interno è  in decimi di secondo.

Un secondo = un decimo di secondo x 10

1/10 in binario è


che corrisponde a 0.09 in decimale.
Quindi si porta dietro un errore di 0.01 secondo: imprecisione tollerata per 14 ore.

In 100 ore ci sono 360.000 secondi.

Dopo 100 ore abbiamo un errore di 3600 secondi = quindi 1 ora

# London Ambulance Service, 1992

Sistema per gestire il servizio ambulanze di Londra: 1500-2000 chiamate e solo 200 ambulanze.

Fino a quel momento era gestito a mano su quale ambulanza deve andare dove.

Si pensa di informatizzare il servizio di gestione delle partenze delle ambulanze.

## Errori

- l'addestramento al sistema è stato fatto 10 mesi il rilascio del software.
- l'interfaccia utente era poco utilizzabile
- pensato per un numero di chiamate inferiore rispetto a quelle reali.
- non c'era nessuna soluzione di backup in caso di crash.

A causa di questo sistema sono morte circa 30 persone.

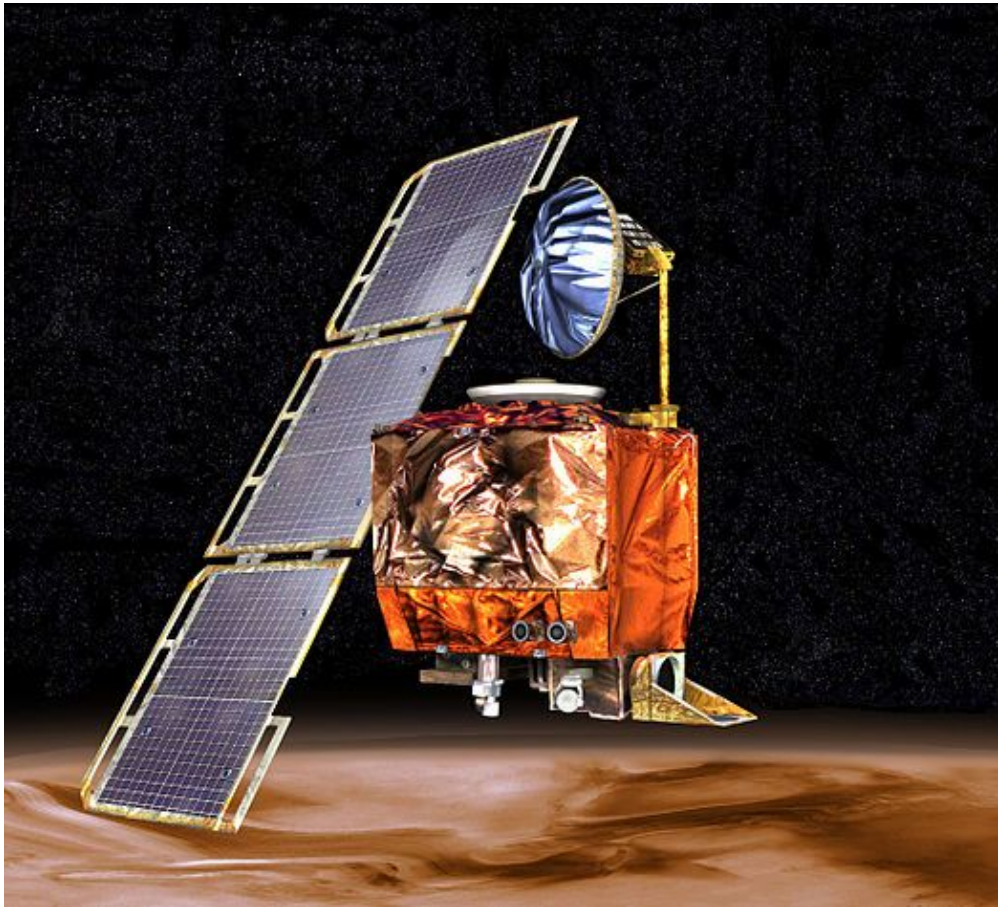
Dopo 3 giorni il sistema è stato abbandonato, costato 11.000.000 Euro.

# Il vettore spaziale, Ariane 5 (1996)

Il vettore Ariane 5 dell'agenzia spaziale europea esplose durante il volo inaugurale il 4 giugno 1996 dopo 39 secondi dal decollo. L'esplosione fu causata da un segnale di autodistruzione emesso dal sistema di controllo che erroneamente, a causa di un buffer overflow, pensava di essere fuori rotta. La progettazione e costruzione del vettore era costata 500 milioni di dollari

# Mars Climate Orbiter (1999)

La sonda Mars Climate Orbiter fu “smarrita” dopo il decollo con una perdita economica di circa 125 milioni di dollari. Successive investigazioni sulle responsabilità appurarono che in fase di progettazione non fu specificato il sistema di misura da utilizzare. Differenti team di sviluppo, nello sviluppo dei vari moduli fecero differenti assunzioni sul sistema di misurazione da applicare (metrico (Kg) ed imperiale (pounds)).



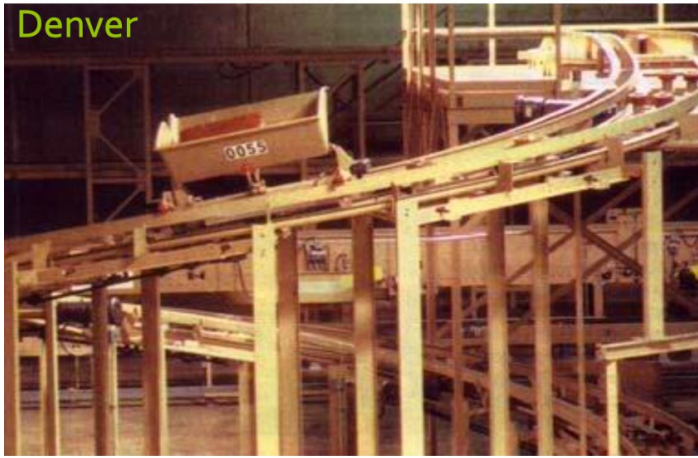
# Aeroporto di Denver, 1995

L’aeroporto di Denver doveva essere inaugurato nel 1995 e doveva basarsi su di un sistema di smistamento dei bagagli avveniristico. Il sistema software avrebbe dovuto indirizzare correttamente i bagagli, ma non riuscì mai a fornire le garanzie necessarie. Alla fine il progetto fu abbandonato

Sistema di smistamento dei bagagli:

6 anni di lavoro e 193.000.000$

- 35 km di rete
- 4000 carrelli
- 5000 motori
- 2700 fotocellule
- 400 ricevitori radio
- 59 raggi laser
- 100 PC




Errore di gestione nella sincronizzazione.

Quando il sistema andava in stallo, non c’era un backup sullo stato precedente quindi i bagagli non si trovavano più.

La manutenzione è costata 1.000.000$ al giorno.

# UN ESEMPIO POSITIVO!!!

## Metro Est Ovest Rapide, 1998

Linea 14 della metropolitana di Parigi.

Prima linea integralmente automatizzata

8 km, 7 stazioni, 19 treni. Intervallo tra 2 treni: 85 secondi.

Esempio positivo:

Siemens ha pensato a un sistema di vetri che oscura l'accesso ai binari e si apre direttamente sul treno, eliminando la possibilità di suicidi.

**ANCORA FUNZIONANTE!**

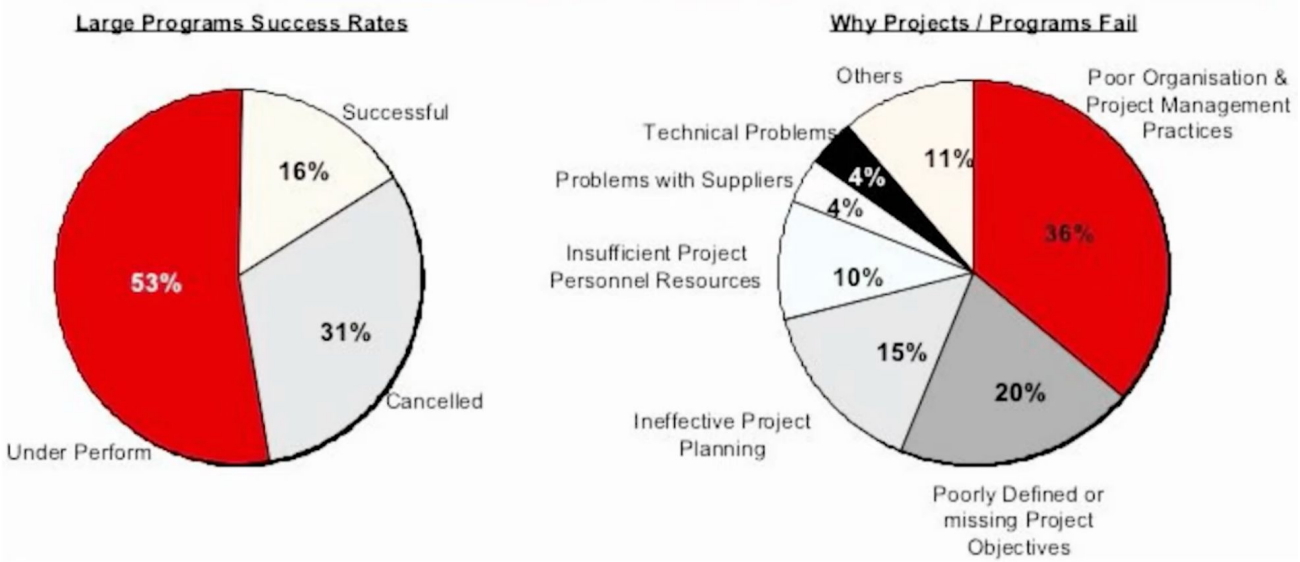
# Quindi?

- Anche i produttori di software possono avere successo ma devono imparare dagli errori
- Anche le opere degli ingegneri qualche volta crollano (es. ponte Morandi) molto più raramente dei software (es. crash del sistema operativo)
- Anche i produttori di software devono diventare ingegneri (del software)

# Analisi dello Standish Group 1994

Per ogni 100 sistemi software:

- 16.2 % completati in tempo.
- 52,7% in ritardo (difficoltà nelle fasi iniziali, cambi di tecnolgia, difetti nel prodotto finale)
- 31,1% progetti abbandonati



# Standish Group report: le cause di abbandono

1. Scarso coinvolgimento degli utenti
2. Requisiti e specifiche incompleti
3. Modifiche a specifiche e requisiti
4. Mancanza di supporto esecutivo
5. Ignoranza tecnologica
6. Mancanza di risorse
7. Attese irrealistiche
8. Obiettivi non chiari
9. Tempi di sviluppo non realistici
10. Nuove tecnologie

Approfondimento: 1. Dispensa requisiti

# Nascita dell'Ingegneria del Software

- Anni '50 appaiono i primi linguaggi di alto livello (COBOL)
  - I programmi cominciano a diventare via via più complessi
  - Programmare comincia a diventare un lavoro
- Anni '60-'70...la crisi del software
  - Le tecniche di sviluppo adottate fino a quel punto risultano non scalabili
  - Gli sviluppatori sembrano incapaci di sviluppare software in grado di utilizzare a fondo i miglioramenti nello hardware
- Conferenze NATO (1968 Garmisch-D, 1969 Roma-I)
  - Vengono poste le basi per una nuova disciplina
  - sono necessarie nuove metodologie, strumenti formali, nuovi processi, nuovi strumenti di sviluppo, nuovi modi di organizzare il lavoro, etc..
  - la produzione del software si deve trasformare da un'attività artigianale ad un'attività "ingegneristica"

# E oggi?

- Da allora sono stati fatti molti passi avanti tuttavia ...
  - La **complessità** dei sistemi **continua a crescere** vertiginosamente
  - il software viene continuamente applicato in **nuovi domini**
  - moltissimi sistemi software falliscono o vengono cancellati dopo esser partiti e trovandosi spesso anche in una fase avanzata dello sviluppo
- Spessissimo il rilascio del software avviene in **ritardo rispetto alle scadenze stabilite**
- Altrettanto spesso la **qualità** del software rilasciato è "discutibile"

# Ingegneria del software: definizioni

- IEEE:
  - Applicazione di un approccio sistematico, disciplinato e quantificabile allo sviluppo, supporto e manutenzione del software
- Sommerville:
  - L'Ingegneria del Software è una disciplina ingegneristica che riguarda tutti gli aspetti della produzione del software. L'ingegnere del software deve adottare un approccio sistematico ed organizzato al suo lavoro ed utilizzare gli strumenti e le tecniche più appropriate a seconda del problema da risolvere, dei vincoli di sviluppo e delle risorse disponibili.
- Ghezzi, Jazayeri, Mandrioli:
  - L'Ingegneria del Software è la branca dell'informatica che riguarda lo sviluppo di sistemi software le cui dimensioni richiedono l'intervento di uno o più team di sviluppo … programmare è principalmente un'attività personale, mentre l'ingegneria del software è essenzialmente un'**attività di team**.

# Ingegneria del software: definizioni

- Emmerich:
  - L'Ingegneria del Software è una branca dell'ingegneria dei sistemi che riguarda lo sviluppo di sistemi software complessi e di grandi dimensioni. Essa si focalizza su: obiettivi e limiti reali per i servizi forniti dai sistemi software; la precisa specifica della struttura di questi sistemi, del loro comportamento e l'implementazione di tali specifiche; le attività richieste al fine di garantire che le specifiche e gli obiettivi siano raggiunti; l'evoluzione di tali sistemi nel tempo. Infine, essa riguarda anche i processi, i metodi e gli strumenti per lo sviluppo **economicamente vantaggioso e pianificato** del software.

# Cosa si evince?

- Introduzione di metodologie per lo sviluppo di sistemi di **medio/grandi dimensioni**
- Si raccomanda **disciplina e sistematicità**
- Si raccomanda l'introduzione di **metodi quantificabili al fine di poter paragonare differenti soluzioni possibili**
- Comporta lo sviluppo di sistemi che richiedono l'intervento **di team di sviluppo**. Dunque la **comunicazione** diventa uno degli aspetti più importanti.

# Cosa si evince? (cont.)

- Non riguarda soltanto la programmazione (per certi versi aspetto marginale)
- **Costi di sviluppo e tempi** sono un aspetto fondamentale dello sviluppo
- Si richiedono **capacità di gestione e di pianificazione**
- Obiettivo focale è la spinta verso la produzione di **qualità**
- Necessità di applicare **strumenti di supporto**

# Il Software nell'economia moderna

- Le economie di tutte le nazioni più evolute dipendono dal software e la maggior parte dei sistemi sono controllati da software
- L'Ingegneria del Software ha a che fare con teorie, metodi e strumenti per progettare, costruire e mantenere software di grandi dimensioni
- Il software costa più dell'hardware e il mantenimento costa più dello sviluppo
- Obiettivo: sviluppo cost-effective del software

# Il Software nell'economia moderna

| Informatica | val | Telecom | val | Media | val |
|---|---|---|---|---|---|
| Microsoft | 356 | Vodafone | 157 | Viacom | 72 |
| Intel | 224 | Verizon | 132 | Liberty Media | 37 |
| IBM | 197 | SBC | 123 | ComCast | 34 |
| Cisco | 138 | NTT DoCoMo | 103 | Clear Channel | 28 |
| AOL | 126 | Vodafone ag | 98 | Cox Comm. | 21 |
| Oracle | 91 | Bellsouth | 71 | British Sky | 20 |
| Dell | 73 | Deutsche Telekom | 66 | The Thomson | 19 |
| Vivendi | 51 | AT&T | 65 | Gannet | 18 |
| Texas Instr. | 48 | Telefonica | 58 | TheNews Group | 15 |
| HP | 44 | Telecom Italia | 54 | McGrawHill | 12 |
| SAP | 43 | China Mobile | 54 | Tribune | 11 |
| Taiwan semic. | 43 | NTT | 48 | Grupo televisa | 10 |
| Fujitsu | 40 | TIM | 44 | Reed | 10 |
| Sun | 39 | France Telecom | 40 | USA Network | 9 |

**Valorizzazione in borsa in G$ delle principali aziende ICT**

# Cosa è un prodotto software?

- Qualcosa di più di un insieme di linee di codice…
- Un insieme di:
  - Programmi per computer
  - Tutta la documentazione che descrive la struttura del sistema
  - I dati di configurazione, che permettono di installarlo
  - Il manuale utente

# Dimensione dei SW attuali

- Di quante linee di codice sono costituiti i software attuali?
  - The Gimp **650.000**
  - Kernel Linux nel 2002 **4.141.432**
  - Open Office **10.000.000**
  - Suite Mozilla (Firefox + Thunderbird) **30.000.000**
  - Windows Vista **50.000.000**
  - OS X 10.4 **86.000.000**

# Programmi vs Prodotti

- **Programma**: l'autore è anche l'utente (e.g., non è documentato, quasi mai è testato, non c'è progetto)
  - non serve approccio formale
- **Prodotto software**: usato da persone diverse da chi lo ha sviluppato
  - è software industriale il cui costo è circa 10 volte il costo del corrispondente programma
  - è necessario un approccio formale allo sviluppo

# Tipologie di Prodotti Software

- **Prodotti generici** (general purpose)
  - sistemi stand-alone prodotti da una organizzazione e venduti a un mercato di massa
- **Prodotti specifici** (specific purpose)
  - sistemi commissionati da uno specifico utente e sviluppati specificatamente per questo da un qualche contraente

- La fetta maggiore della spesa è nei prodotti generici ma il maggior sforzo di sviluppo è nei prodotti specifici
- La differenza principale?
  - Chi fornisce la specifica del prodotto (il produttore o il consumatore)

# Problemi della produzione software: costi

- Il software ha costi elevati
  - Sono i costi delle risorse usate: ore lavoro (manpower), hardware, software e risorse di supporto. Il manpower è dominante !
    - Il costo è espresso in mesi/uomo
  - La manutenzione costa più dello sviluppo
    - Per sistemi che rimangono a lungo in esercizio i costi di manutenzione possono essere svariate volte il costo di produzione
- Produttività media
  - da 300 a 1000 linee di codice rilasciate per mese/uomo

# Problemi della produzione software

- Ritardi nelle consegne e aumenti dei costi stimati
  - US Air Force - Sistema di comando e controllo (1981):
    - stima iniziale della azienda vincitrice per la fornitura: 400.000$,
    - costo successivamente rinegoziato:
      - a 700.000$,
      - poi a 2.500.000 $
      - costo finale 3.200.000$ .
    - … costo finale maggiore di circa 10 volte la stima iniziale !!
    - … e con notevole ritardo rispetto alla stima iniziale

# Problemi della produzione software

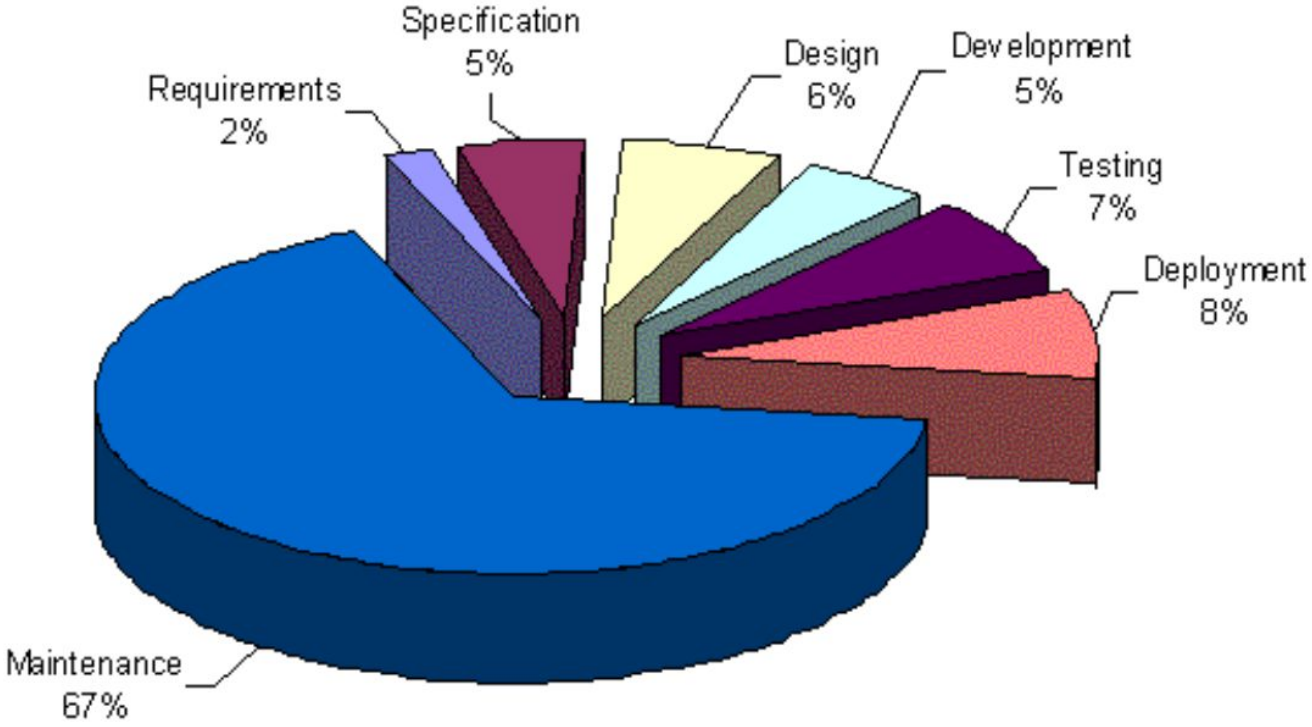
- Un' azienda nel settore della grande distribuzione aveva richiesto un sistema che, stima iniziale, sarebbe stato sviluppato in 9 mesi al prezzo di 250.000 $ (1989)
  - Due anni dopo, e dopo una spesa di 2.500.000 $, il lavoro non era stato ancora completato e fu stimato che erano necessari altri 3.600.000 $ (!!)
  - Il progetto fu abbandonato !
- Da un report USA del 1989
  - su 600 aziende contattate, più del 35% avevano progetti 'runaway' (in ritardo o fuori dal budget e la tempistica stimata)

# Uno dei più noti record negativi: Il Software dello Space Shuttle

- Costo finale: $10.000.000.000, svariati milioni di dollari in più di quanto pianificato
- Time: ritardo di 3 anni
- Qualità: Il primo lancio del Columbia fu cancellato a causa di problemi di sincronizzazione tra i 5 computer di bordo
  - Dopo lunghe (e costose) investigazioni, l'errore fu individuato in una modifica fatta 2 anni prima, quando un programmatore aveva cambiato un fattore di ritardo su un interrupt handler da 50 a 80 millisecondi
  - La possibilità che si verificasse un errore a riguardo era talmente minima che non apparve mai nelle migliaia di ore di testing
- Il software contiene ancora errori sostanziali
  - Gli astronauti hanno a disposizione un libro dei bug noti...

# I costi del software

Il prodotto software durante la sua vita diverse fasi:
analisi, specifica, progettazione, codifica, testing, manutenzione e ritiro



# La manutenzione

La manutenzione include tutti i cambiamenti al prodotto software, anche dopo che è stato consegnato al cliente.

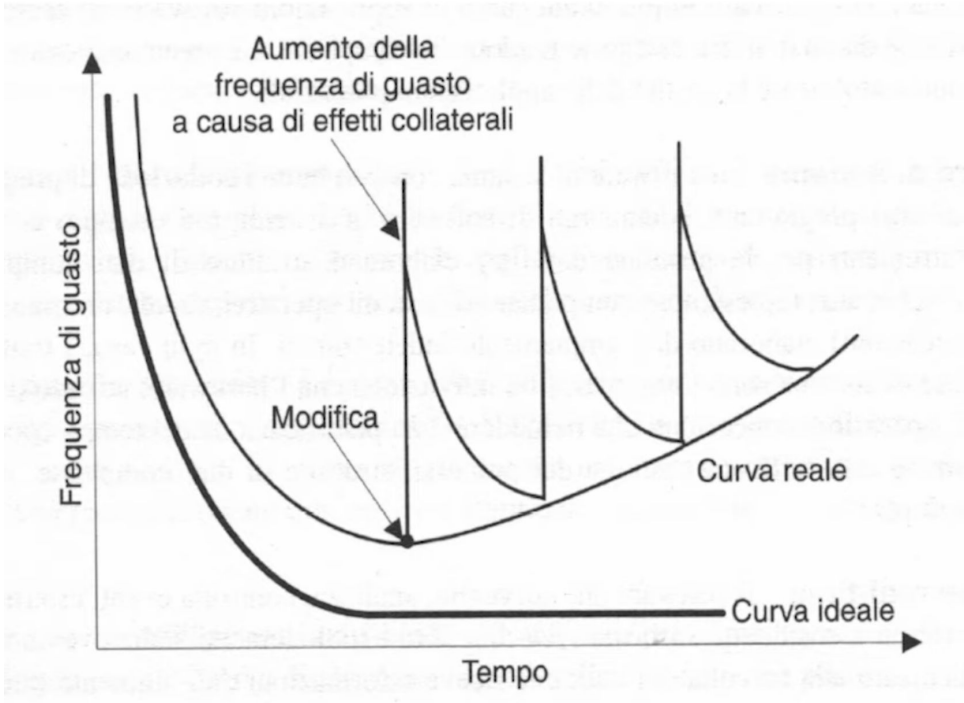
Si divide in:

- **manutenzione correttiva (20%)** rimuove gli errori lasciando invariata la specifica
- **manutenzione migliorativa,** consiste in cambiamenti alla specifica e nell'implementazione degli stessi, può essere:

  - *Perfettiva (60%)* modifiche per migliorare le qualità del software, introduzione di nuove funzionalità, miglioramento delle funzionalità esistenti.
  - *Adattiva (20%)* modifiche a seguito di cambiamenti nell'ambiente legislativo, nell'Hardware, nel sistema operativo ecc... (esempio: IVA dal 22% al 20%)

# Curva dei guasti dell'harware.

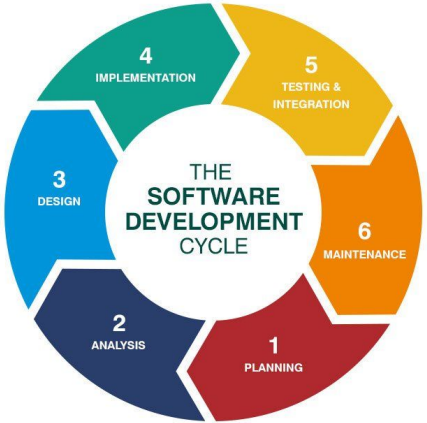

# Curva dei guasti del software.



# Processo di produzione Software : il modo in cui produciamo software

Inizia quando iniziamo ad esplorare il problema e finisce quando il prodotto viene ritirato dal mercato

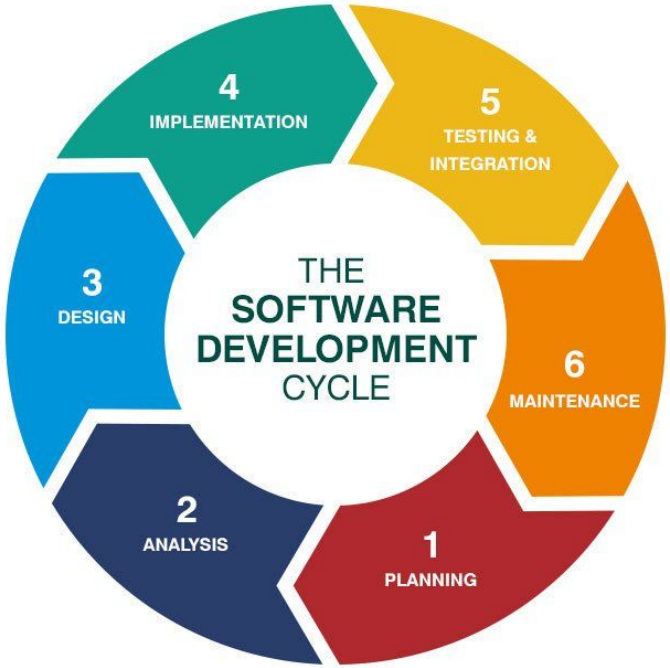
- Fasi:
  - analisi dei requisiti
  - specifica
  - progettazione
  - implementazione
  - integrazione
  - mantenimento
  - ritiro

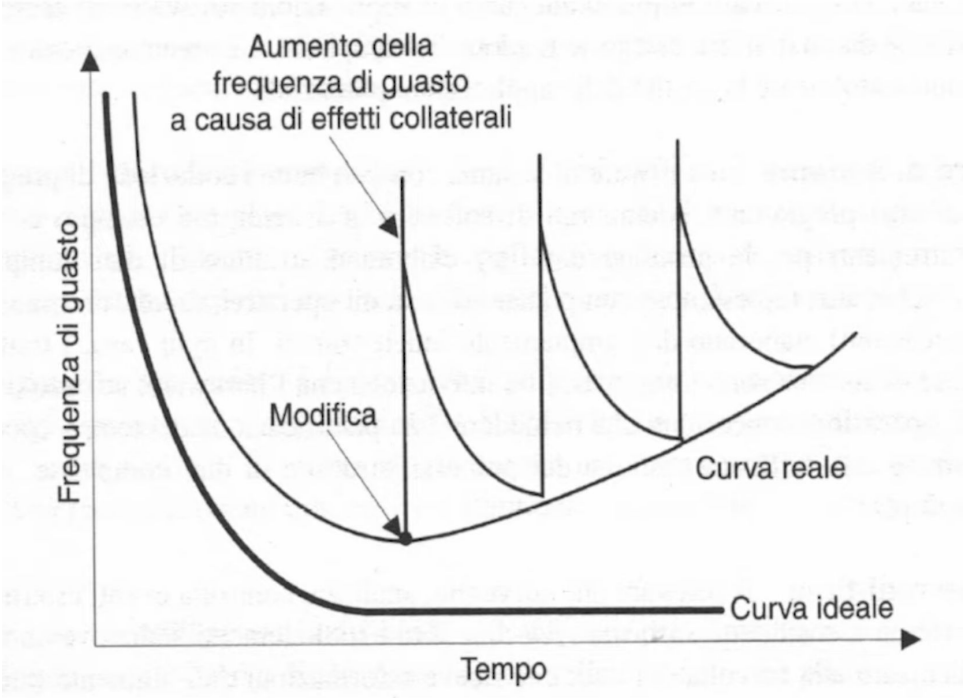



Le attività devono seguire un piano di progetto, “Project Plan”.

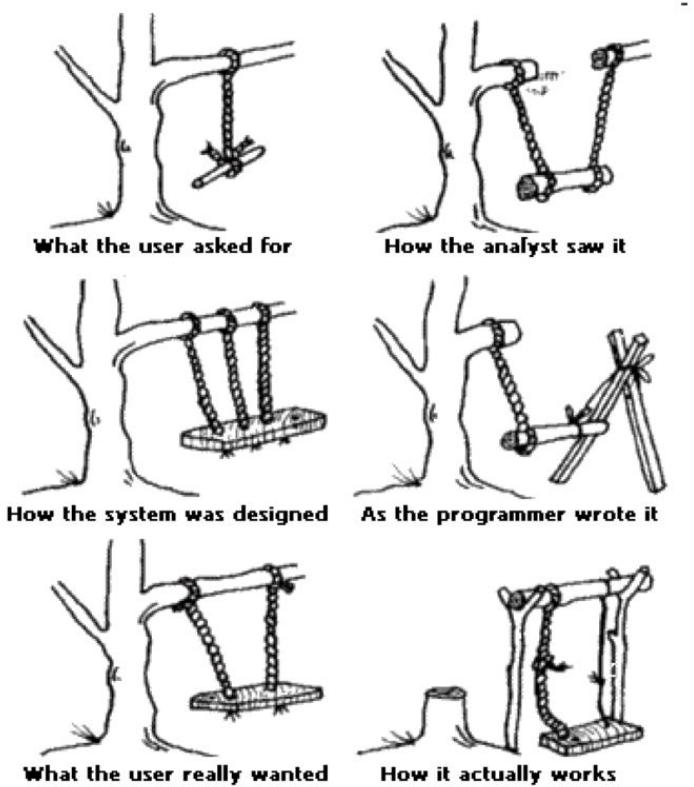
Riguarda anche tutti i tools e le tecniche per lo sviluppo e il mantenimento e tutti i professionisti coinvolti

# Ciclo di vita del software



# Ingegneria del Software: alleviare i problemi



# Necessità di un approccio ingegneristico

- Necessità di applicare principi ingegneristici alla produzione software per sviluppare:
  - il giusto prodotto
  - al giusto costo
  - nel tempo giusto
  - con la giusta qualità

# Dunque, ricapitolando

- L'**Ingegneria del Software** è una disciplina che cerca di fornire le regole per il processo di produzione del software
- Un ingegnere del software dovrebbe:
  - adottare un approccio sistematico e organizzato al proprio lavoro
  - usare strumenti e tecniche appropriate, che dipendono dal problema che deve essere risolto, dai vincoli presenti e dalle risorse disponibili

# Un prodotto software cos'è?

E' qualcosa di più di un insieme di linee di codice:

- codice strutturato in packages
- documentazione che descrive la struttura del sistema
- dati di configurazione, per l'installazione
- manuale utente.

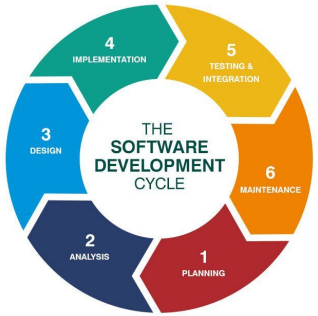
# Caratteristiche del prodotto software:

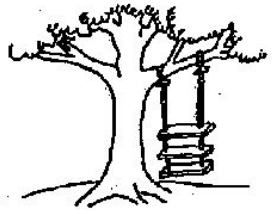
- E’ invisibile e intangibile
- Si “sviluppa” non si “fabbrica”
- E’ facilmente duplicabile e distribuibile in rete
- Non si consuma, si “deteriora” … invecchia
- In Europa non si può brevettare, si può proteggere
- Viene acquisito su licenza:
    - Proprietaria
    - Public domain
    - Open licence

# L’importanza dell’analisi dei requisiti

Se si introduce un errore durante l’analisi dei requisiti, l’errore apparirà anche nella specifica, nella progettazione e nel codice.

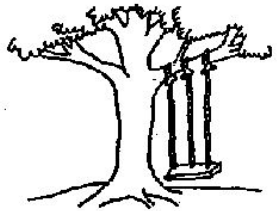
Prima individuiamo l’errore meglio è.



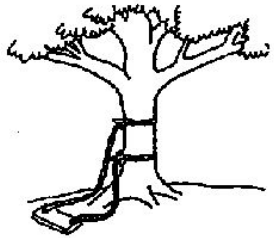
As proposed by the project sponsor

*(proposta dal committente)*



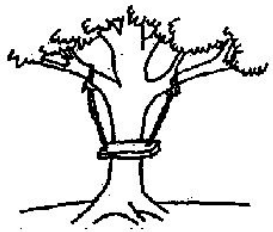
As specified in the project request

*(specificata nei requisiti)*



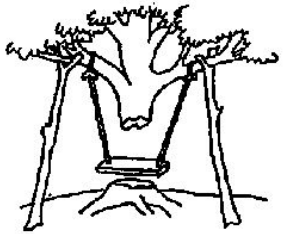
As designed by the senior analyst

*(progettata dall'analista)*



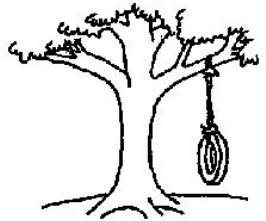
As produced by the programmers

*(prodotta dai programmatori)*



As installed at the user's site

*(installata presso l'utente)*



What the user wanted

*(ciò che l'utente voleva)*

# Lavoro in team

- La maggior parte del software è oggi prodotto da team di programmatori

- Il lavoro in team pone dei problemi:
  - Di interfaccia tra le diverse componenti del codice
  - Di comunicazione tra i membri del team
- Molto tempo deve essere dedicato alle riunioni tra i vari componenti.

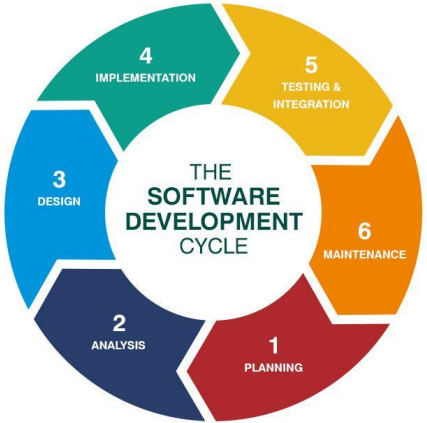
- L'ingegnere del software deve essere anche capace di:
  - gestire i rapporti umani  e organizzare un team
  - gestire gli aspetti economici e legali

# Temi dell’ingegneria del software

- Processo software
- Realizzazione di sistemi software
- Qualità del software

# Processo software

- Organizzazione e gestione dei progetti
- Metodi di composizione dei gruppi di lavoro
- Strumenti di pianificazione, analisi, controllo
- Cicli di vita del software
- Definizione e correlazione delle attività
- Modelli ideali di processo di sviluppo



# Realizzazione di sistemi software

- Strategie di analisi e progettazione
  - Tecniche per la comprensione e la soluzione di un problema
  - Top-down, bottom-up, progettazione modulare, OO
- Linguaggi di specifica e progettazione
  - Strumenti per la definizione di sistemi software
  - Reti di Petri, Z, OMT, UML
- Ambienti di sviluppo
  - Strumenti per analisi, progettazione e realizzazione
  - Strumenti tradizionali, CASE, CAST

# Qualità del software

- Modelli di qualità
  - Definizione di caratteristiche della qualità
- Metriche software
  - Unità di misura, scale di riferimento, strumenti
  - Indicatori di qualità
- Metodi di verifica e controllo
  - Metodi di verifica, criteri di progettazione delle prove
  - Controllo della qualità, valutazione del processo di sviluppo

# Gli stakeholders di un prodotto software

- Fornitore
  - chi lo sviluppa
- Committente
  - chi lo richiede (e paga)
- Utente
  - chi lo usa

# Le qualità dell'ingegneria del software

Il software è un termine che identifica tutti i programmi e le procedure che un computer utilizza per eseguire un compito. Le qualità rilevanti sono:

- affidabilità
- correttezza
- robustezza
- prestazioni
- usabilità
- verificabilità
- manutenibilità
- riusabilità
- produttività
- tempismo
- portabilità

# Ingegneria del software

E’ quella scienza che si occupa di trovare i metodi e le strategie per creare software di qualità. E’ un approccio sistematico e rigoroso allo sviluppo e alla manutenzione del software.

Affidabilità. Un software è affidabile se produce l’effetto voluto entro uno scostamento tollerabile. Non genera errori. In genere la prima versione di un programma contiene diversi errori e si considera “buggata”, non affidabile.

Correttezza. Rispetto alle specifiche dei requisiti funzionali.

Robustezza. E’ la capacità del software di essere innocuo di fronte a situazioni impreviste come può essere un guasto.

# Ingegneria del software

Prestazioni. Uso efficace delle risorse del sistema. Si può misurare analizzando la complessità degli algoritmi, monitorando l'uso dell'hardware e del software quando il programma è in esecuzione.

Usabilità. Un sistema è usabile quando gli utenti trovano facile ed intuitivo il suo utilizzo. Quando anche utenti con disabilità (ipovedenti e ipoudenti) possono usarlo.

Verificabilità. Se le sue funzionalità e prestazioni possono essere facilmente verificate.

Manutenibilità. Per manutenzione si intende la modifica del software dopo il rilascio. Queste modifiche sono correttive, adattive ed evolutive. Un software presenta manutenibilità se è facile correggere errori ed implementare nuove funzionalità

# Ingegneria del software

Riusabilità. E' la capacità di usare porzioni di codice in un nuovo software, il riuso del codice sorgente o il riuso di interi componenti abbassa i costi dello sviluppo.

Produttività. E' la capacità di rilasciare velocemente del software. Si misura come rapporto tra unità prodotte e sforzo compiuto.

Tempismo. Rilasciare un prodotto nei tempi contrattuali previsti.

Portabilità. Capacità di integrarsi con sistemi diversi.

# I principi fondmentali dell'ingegneria del software.

Abbiamo un insieme di linee guida da seguire per strutturare la nostra creatività.

Rigorosità e formalità.

Separazione dei problemi.

Modularità.

Astrazione.

Previsione dei cambiamenti.

Generalità.

Incrementalità.

# Rigorosità e formalità

Disciplina creativa ma richiede sistematicità.

Trovare la soluzione ed esprimerla in modo rigoroso e formale in modo da poterci ragionare sopra. Come la matematica.

ESEMPI:

- documentazione precisa.

## Separazione dei problemi

Separare problemi complessi in sotto-problemi, perchè noi siamo in grado di affrontare più facilmente i piccoli problemi che lo compongono.

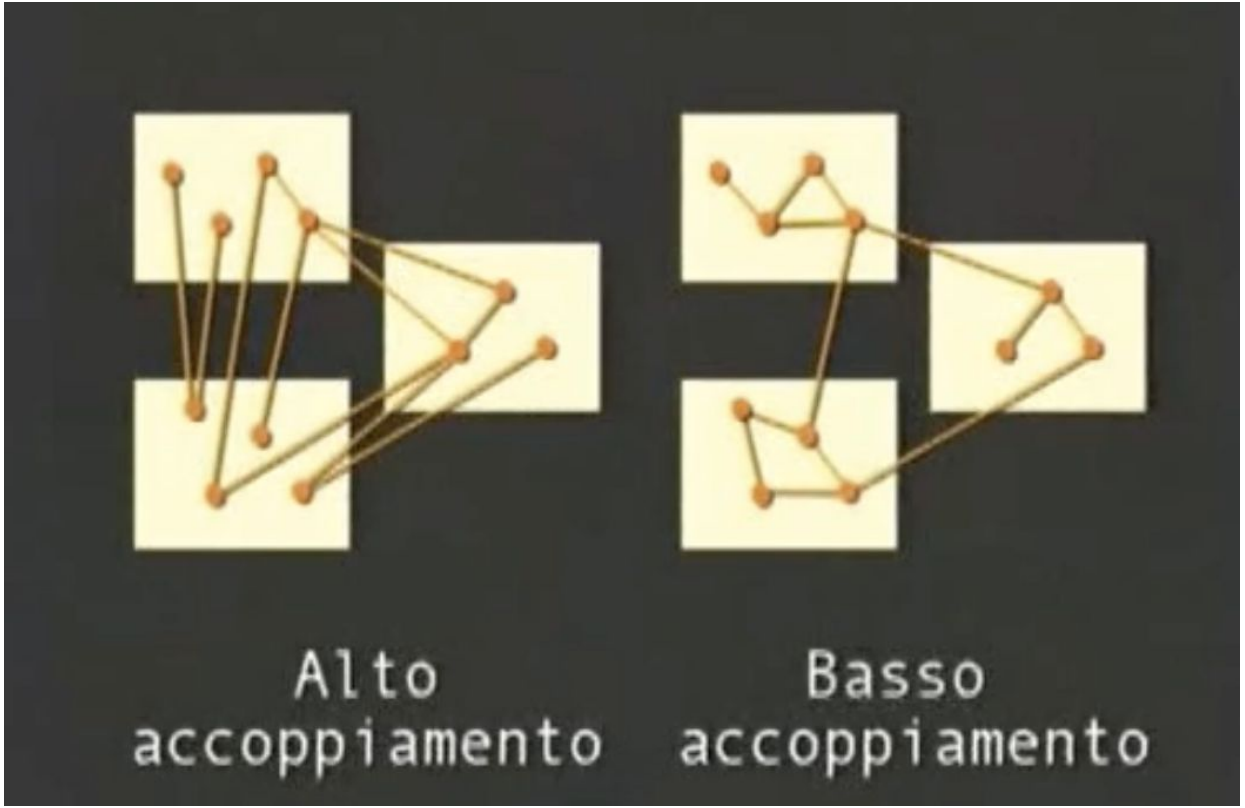
Per gestire la complessità.

## Modularità

Dividere un sistema complesso in pezzi semplici: moduli.

Caratteristiche: alta coesione nei moduli e basso accoppiamento tra i moduli.

# Sistema ad alto accomppiamento VS basso accoppiamento



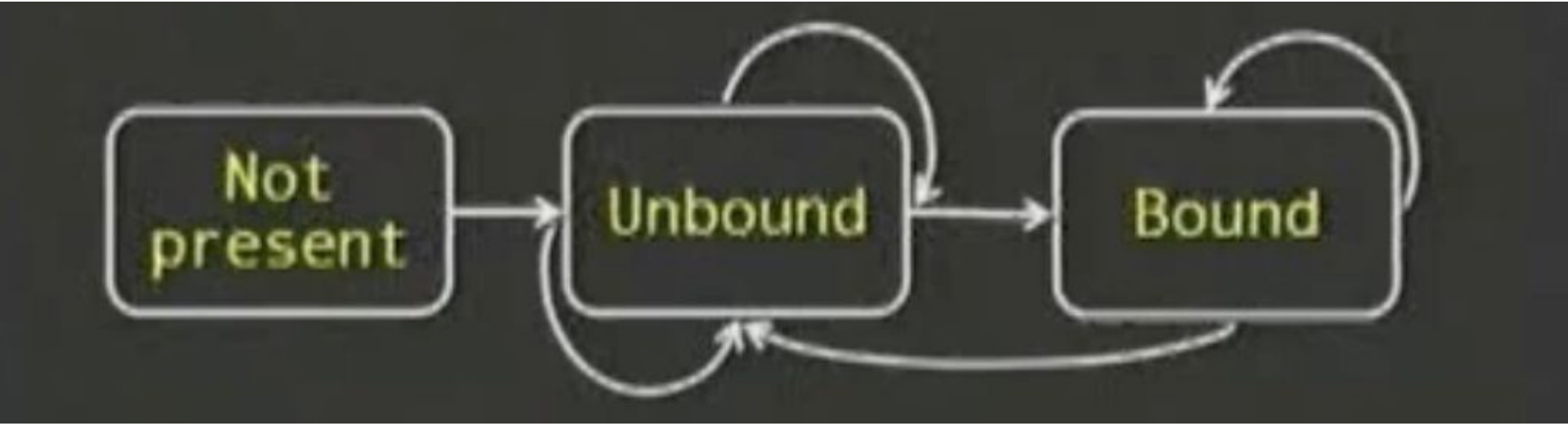

Basso accoppiamento: non sono tenuto a conoscere i dettagli di ogni modulo ma solo gli input e output che necessitano.

Ogni modulo sarà gestito indipendentemente dagli altri.

# Astrazione

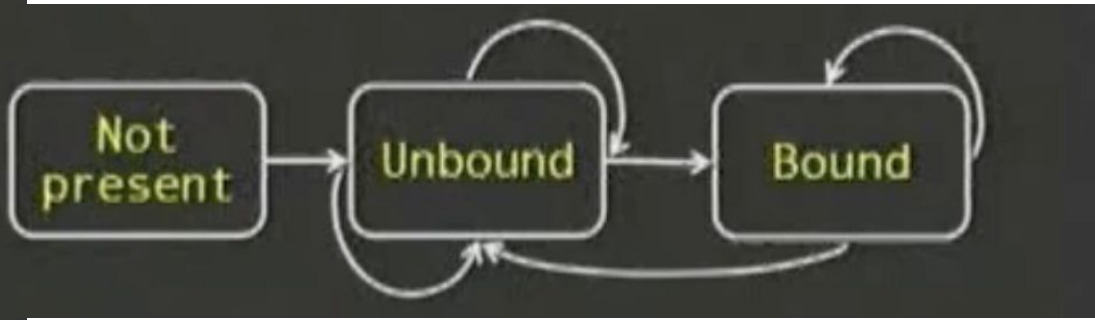
Ignorare i dettagli.

Un modello astratto.



```
package Computer;
public class Slot {
public String slotID;
private Component component = null;
    public Slot
        (String _slotID,
boolean _installed,
boolean _required,
        Component _component
        ) {
slotID = _slotID;
        installed = _installed;
        required = _required;
        component = _component;
    }
public void bind(Component c)
      component = c;
    }
      public void unbind() {
      component = null;
    }
   public boolean isBound() {
        return (component != null);
    }}
```

# Modello di astrazione del seguente codice :




Supporto al ragionamento e alla progettazione.

Quali sono i dettagli importanti e quali le caratteristiche su cui concentrarsi.

# Previsione dei cambiamenti

Capacità di prevedere i cambiamenti: cosa può accadere in futuro?

Studiare una serie di tecniche per verificare in che misura il software potrà essere utilizzato in condizioni diverse.

E’ importante perchè diverse condizioni possono condizionare il successo del software.

## Generalità

Risolvere il problema generale e non un'istanza del problema.

Più è generale una soluzione e più costa, perchè risolve più problemi.

Ogni problema ha il suo livello di generalità.

## Incrementalità

Procedere a passi successivi.