Lezione 1
Ingegneria del Software II- Introduzione e Motivazione

Riferimenti bibliografici

- I. Sommerville – Ingegneria del Software – 8a edizione – Cap.1
- R. Pressman- Principi di Ingegneria del Software- 4 edizione- Cap. 1
- F.P. Brooks, “No Silver Bullet: Essence and Accidents of Software Engineering”, IEEE Computer, 1987.
  - Scaricabile all’indirizzo: http://www.virtualschool.edu/mon/SoftwareEngineering/BrooksNoSilverBullet.html

Obiettivi della lezione

1. Richiamare concetti basilari di Ingegneria del Software

2. Illustrare le motivazioni dell'IS e le sue attuali Sfide

Software- Definizioni

- Programmi per computer con la relativa documentazione, quale ad esempio requisiti, modelli di progetto, e manuali utente.
  - non solo *programmi*, ma l'insieme degli 'artifatti' che lo compongono, prodotti durante il suo sviluppo
    - un programma verrà usato dal suo autore, che lo ha sviluppato senza preoccuparsi di altri utenti, di portabilità, affidabilità, …
    - un sistema software, essendo rivolto ad altri utenti, dovrà essere usabile, portabile, affidabile, etc...
- La definizione IEEE (*Institute of Electrical and Electronic Engineers*)
  - insieme di programmi, procedure, regole, e ogni altra documentazione relativa al funzionamento di un sistema di elaborazione dati

Software- la dimensione economica

- Le economie di tutte le nazioni industrializzate dipendono dal software.
- Sempre più sistemi sono controllati dal software.
- Gli investimenti per il software rappresentano una parte significativa del PIL di tutte le nazioni industrializzate.

Software- Classificazioni

- Il Software può essere:
  - Standalone/ Embedded/ a supporto di processi (di produzione industriale o aziendale)
  - Generico/ Personalizzato
  - Prodotto ex-novo/ Configurato/ Riusato/ …
  - Software di sistema/ Applicativo/ Embedded/ di Produttività/ Applicazioni Web/ Open-source…

Software- La rilevanza

- Sempre più critico:
  - Security Critical (militare/ medico/ spaziale)
  - Mission Critical (bancario/ produzione industriale/ …)
- Sempre più complesso:
  - Software composto da Milioni di linee di codice (anche per semplici applicazioni)
- Sempre più diffuso:
  - Sistemi embedded, Telefonia, Applicazioni Web …

Software: Il problema dei Costi

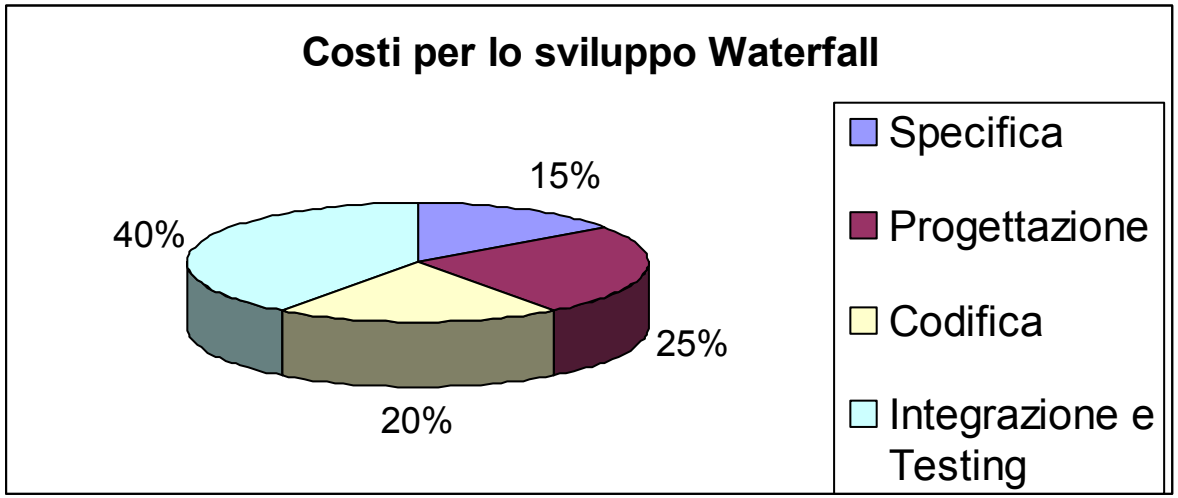
- I costi del Software spesso dominano i costi complessivi dei sistemi informatici. I costi del software per PC sono spesso maggiori dei costi dell'hardware stesso.

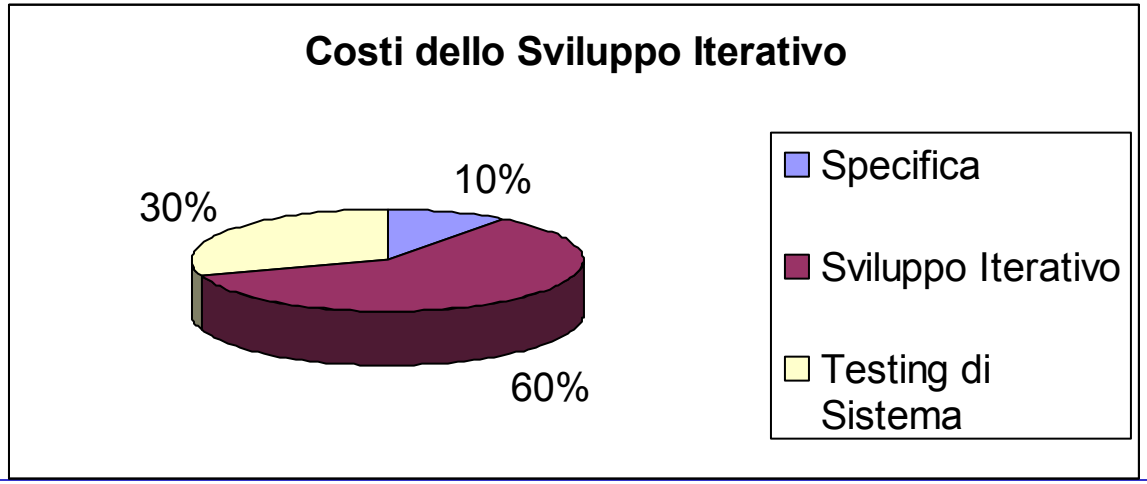
- É più costoso manutenere il software piuttosto che svilupparlo, soprattutto per sistemi legacy.

Software – distribuzione dei costi

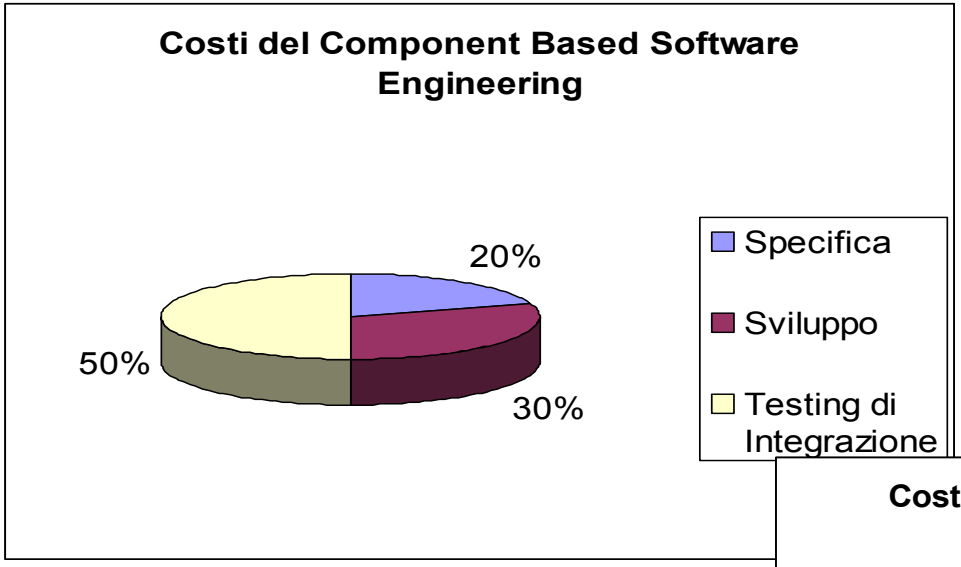
- Circa il 60% dei costi è speso per le attività di sviluppo, il 40% per il testing.
- Per software personalizzato, I costi per l'evoluzione spesso superano quelli di sviluppo.
- I costi variano in base al tipo di sistema sviluppato e ai requisiti di qualità richiesti quali le prestazioni o l'affidabilità.
- La distribuzione dei costi dipende anche dal tipo di modello di sviluppo adottato.

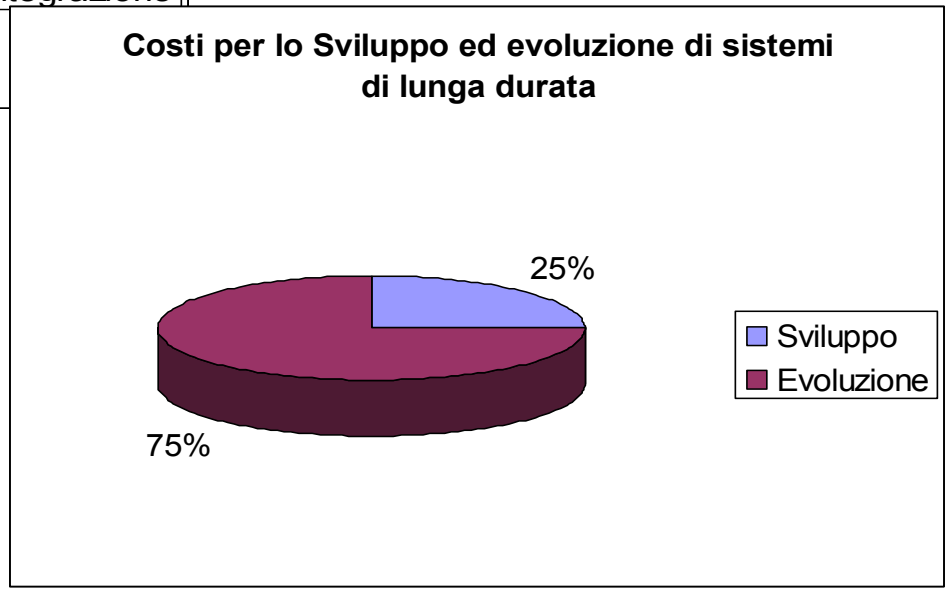
Distribuzione dei Costi per attività



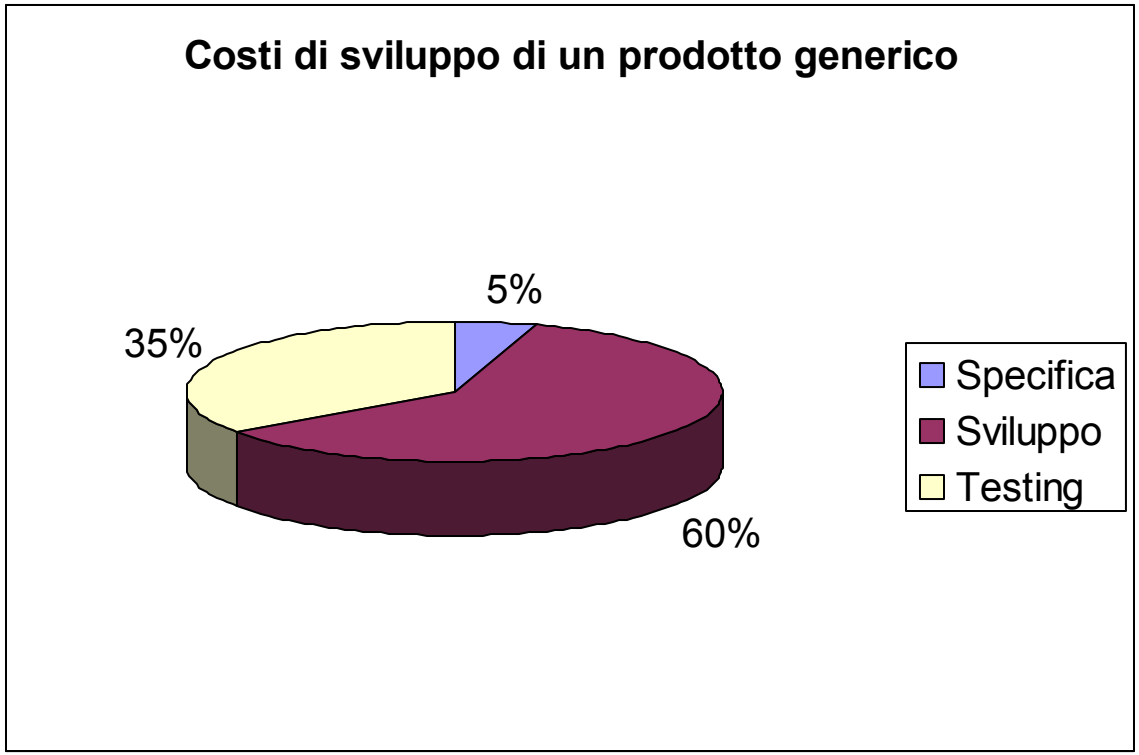


Distribuzione dei costi per attività





Costi di sviluppo di un prodotto generico



Software: Il problema della qualità

- Un software di qualità dovrebbe fornire le funzionalità e le prestazioni richieste essendo contemporaneamente **manutenibile**, **fidato**, **efficiente**, **accettabile**…

- Qualunque sviluppatore (anche privo di adeguata preparazione) è in grado di scrivere codice che '*funzionerà*' (ma fino a che punto?)
- I problemi di qualità del software sono difficili da rilevare
  - Molti difetti del software sono introdotti nelle fasi iniziali di specifica e design
  - Ma verranno rilevati molto più avanti, in operatività….

Riassumendo…

- I problemi del software:
  - Troppo costoso (fino a 10 volte più del previsto)
  - Consegnato in ritardo
  - Di qualità insoddisfacente
    - Vedi i fallimenti ‘famosi’ dei progetti:
      - ARIANE 5 (progetto aerospaziale) [vedi descrizione in: http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html]
      - Therac25 (Sistema medico per la Radioterapia) [vedi descrizione in: http://users.csc.calpoly.edu/~jdalbey/SWE/Papers/THERAC25.html]
      - …

L'Ingegneria del software

- L' ingegneria del software è una disciplina ingegneristica che si occupa di tutti gli aspetti relativi allo sviluppo del sofware.

- Gli ingegneri del software dovrebbero adottare:
  - un approccio sistematico e organizzato per il loro lavoro
  - usando strumenti e tecniche appropriate
  - variabili a seconda del problema da risolvere, dei vincoli di sviluppo, e delle risorse disponibili.

Programmazione vs. Ingegnerizzazione del software

- INGEGNERIZZAZIONE

- Sistemi di grandi dimensioni
- Elevata qualità attesa
- Lavoro in Team
- Requisiti dati dal cliente
- Molti rilasci (deliverables)
- Modifiche frequenti
- Lunga vita
- Costosità

- PROGRAMMAZIONE

- Sistemi piccoli
- Si bada alla funzionalità
- 1 programmatore
- Requisiti del programmatore
- 1 rilascio
- Poche modifiche
- Vita breve
- Free

Processi Software

- Un insieme di attività aventi per obiettivo lo sviluppo o l'evoluzione di un sistema software.
- Ogni processo software deve includere le seguenti attività fondamentali:
  - **Specifica** – definizione di ciò che il sistema dovrà fare e dei vincoli di progettazione
  - **Sviluppo** – progettazione e programmazione
  - **Convalida** – si verifica che il software sia esattamente ciò che il cliente richiede
  - **Evoluzione** – si modifica il software per adeguarlo a requisiti dell'utente e del mercato che cambiano.

Modello di processo software

- Una descrizione semplificata del processo software, osservato da un determinato punto di vista.

- Generici modelli di processo
  - Waterfall;
  - Iterative development;
  - Incrementale;
  - Component-based software engineering
  - …

Metodi e Strumenti nell'IS

- Metodi : Approcci strutturati per sviluppare software di qualità, a costi contenuti.
  - Es. Specificano I modelli da usare, le regole a cui sottostare, forniscono una guida alle attività dei processi e alla relativa organizzazione
- Strumenti:Sistemi software usati per aiutare le attività dei processi software (es.analisi, modellazione, debugging, testing)
- E ancora …
  - Standard, Normative, Linee Guida, Principi, … (CMM, CMM-I, ISO 9000, ISO 12207, IEEE std…)

Conclusioni

- Senza adeguati sforzi, metodologie e conoscenze il software prodotto risulta di qualità scadente, destinata a peggiorare durante il suo ciclo di vita.

- La richiesta di software è elevata e sempre crescente
- Siamo in perenne ‘crisi del software’
- É necessario imparare ad ‘ingegnerizzare’il software

- Le sfide nel campo dell’Ingegneria del software sono quelle della **Produttività**, **Affidabilità**, e **Semplicità**

Sfide fondamentali per l'IS

- Eterogeneità
  - Definire tecniche in grado di produrre software operativo su piattaforme ed ambienti di esecuzione eterogenei;
- Consegna
  - Proporre tecniche in grado di consentire la consegna del software in tempi più rapidi, nel rispetto dei requisiti di qualità;
- Fiducia
  - Sviluppare tecniche che dimostrino agli utenti l'affidabilità del software.

No Silver Bullet in Software Engineering

-non ci sono soluzioni semplici per l'IS-

- **Brooks** in un articolo storico del 1987 [IEEE Computer, Aprile 87] afferma che non esiste (e non potrà mai esistere) alcun "**Silver Bullet**" (pallottola d'argento) che possa risolvere tutti i problemi dell'Ingegneria del Software.

- I problemi derivano da quelle che Brooks definisce Difficoltà **Essenziali** ed Accidentali nel software

http://www.lips.utexas.edu/ee382c-15005/Readings/Readings1/05-Broo87.pdf

Accidental Difficulties

- **Accidental difficulties**: sono legate ad aspetti della produzione del software che generano la possibilità di commettere errori:
  - linguaggi macchina complessi, lenti tempi di risposta di alcuni sistemi (es. Batch), etc..

- Miglioramenti nel processo e negli strumenti di sviluppo del software possono però abbattere gli sforzi legati a tali difficoltà:
  - Linguaggi di Alto Livello
  - Time Sharing
  - Ambienti di sviluppo unificati

Essential difficulties

• **Complexity**: il software deve modellare e controllare problemi complessi

• **Conformity**: la complessità del software deriva dalla sua necessità di sottostare ad un insieme di regole/ interfacce poco chiare, perchè dettate da persone diverse (piuttosto che da un elegante modello fisico e matematico).

• **Changeability**: i requisiti del software variano molto velocemente, già al tempo stesso di sviluppo

• **Invisibility**: il software è invisibile e non visualizzabile: ciò rende difficile mantenerne una vista complessiva

Speranze passate

- Linguaggi di programmazione di moderna concezione (es. Ada)

- Programmazione object oriented
- Intelligenza artificiale e sistemi esperti

- Generatori di codice
- Programmazione grafica

- Tecniche di testing
- Ambienti di sviluppo integrati

Soluzioni promettenti attuali (secondo Brooks)

- Buy versus build
- Processi evolutivi e prototipali
- Attenzione alla progettazione di qualità