Ingegneria del Software: una prospettiva storica

Fase pionieristica (anni ’40)

- Prime applicazioni = automazione di procedimenti di calcolo → calcolatore = strumento per l’esecuzione di operazioni numeriche che difficilmente potrebbero essere eseguite manualmente con la precisione richiesta
- Linguaggi / strumenti di basso livello
- Spesso sviluppatore del sw = utente del sw
- Applicazioni di vita breve

Dal calcolo alla elaborazione dati

- Avvento di applicazioni di natura gestionale →
  calcolatore = macchina per creare, mantenere, modificare, distribuire informazione, strumento che accumula e gestisce un patrimonio di conoscenze di natura strategica per l'organizzazione di cui sta al servizio
- Distinzione tra sviluppatori e utenti
- Esigenza di una professionalità specifica → nascono EDP e sw house
- Dall'arte (lavoro individuale creativo) all'artigianato (lavoro di piccoli gruppi specializzati)
- Aumento vertiginoso della quantità e della qualità del sw
- Aumento della criticità e della complessità delle applicazioni
- Automazione di settori non tradizionali

Fase industriale (seconda metà anni ’60)

- Produttività
- Qualità (deve essere certificata)
- L’attività di sviluppo e manutenzione del sw coinvolge gruppi di lavoro, anche di grosse dimensioni, il cui lavoro deve essere pianificato e coordinato
- L’attività di progettazione del sw deve essere sempre meno sviluppata manualmente e informalmente e sempre più essere supportata da strumenti automatici
- Lo sviluppo deve seguire metodologie efficaci e deve aderire a *standard* di produzione che rendano la stessa insensibile al ricambio di personale
- Nuove applicazioni devono potere essere ottenute anche assemblando componenti standardizzati già sviluppati

La crisi del sw

- Immaturità rispetto alle altre discipline ingegneristiche
  Es. ingegneria civile: metodologie assestate che consentono una descrizione precisa e rigorosa dell'artefatto da progettare, modelli a diverso livello di astrazione, produzione di disegni dettagliati in base ai quali il sistema sarà realizzato, collaudo finale allo scopo di fornire una certificazione

- Domanda di sw largamente insoddisfatta, ritardi fra il momento in cui sorge la richiesta di un'applicazione e il momento in cui la richiesta inizia a essere soddisfatta; cause:
  - ✓Carenza numerica di personale specializzato
  - ✓Grande quantità di addetti impiegati nella manutenzione

Spirale: carenza di personale specializzato → uso di personale impreparato/inesperto → produzione di sw di cattiva qualità → grande sforzo di manutenzione richiesto → impiego di una grande quantità di personale → carenza di personale

La crisi del sw (cont.)

- Rapporto

$$\frac{\text{costo sw}}{\text{costo hw}}$$

crescente; cause:

✓la produzione di sw è scarsamente automatizzabile (attività brain-intensive)
✓complessità

Si parla spesso di quella che viene definita 'crisi del sw', che però è solo apparentemente una crisi. Si tratta invece del fatto ben noto che numerosi prodotti sw in uso, che non sono stati progettati da ingegneri qualificati, siano di qualità più o meno scadente. Certo, questo sw è difficile da modificare ed è tecnicamente mal documentato. La dipendenza del funzionamento del sw dalla presenza di chi lo ha sviluppato è talmente grande che in molti casi lo si può considerare come il prodotto di un'attività amatoriale, certamente non come un lavoro professionale e tanto meno come esempio di tecnica e capacità ingegneristica (W. Zuser, S. Biffl, T. Grechenig, M. Köhle, 2001**)**

Quando un programma sw ha successo, ovvero quando risponde alle esigenze delle persone che lo usano, si comporta senza problemi in un lungo arco di tempo, è facile da modificare e ancora più facile da utilizzare, cambia in meglio la nostra vita.
Quando il sw fallisce gli obiettivi, ovvero quando gli utenti sono insoddisfatti, quando il sw è soggetto a errori, quando è difficile da modificare e ancora più difficile da utilizzare, si verificano varie situazioni negative.
Tutti noi vogliamo realizzare del sw che cambi in meglio il mondo, evitando tutto ciò che accade quando non si riesce a ottenere un buon risultato. Per ottenere ciò è necessario introdurre disciplina nella progettazione e nella realizzazione del sw. Questo è il motivo per cui è necessario un approccio ingegneristico (Pressman, 2004)

Nasce una nuova disciplina

In una conferenza NATO tenuta a Garmisch nel 1968 viene coniato il termine “Ingegneria del Software” a testimoniare l’esigenza di una disciplina ingegneristica, basata su solide basi teoriche e metodologiche che permettano la progettazione, produzione e manutenzione di applicazioni che forniscano caratteristiche di qualità prefissate mediante l’uso delle risorse previste

IEEE Standard Glossary of Software Engineering

Software = i programmi, le procedure, le regole e l'eventuale documentazione associata e i dati relativi all'operatività di un sistema di elaborazione

Ingegneria del Software = approccio sistematico allo sviluppo, all'operatività, alla manutenzione e al ritiro del sw

IEEE Standard 610.12-1990 - Definizione di Ingegneria del Software

(1) Applicazione di una strategia sistematica, disciplinata e misurabile allo sviluppo, esercizio e manutenzione del software; cioè applicazione dell'ingegneria al sw;
(2) Studio delle strategie di cui al punto precedente.

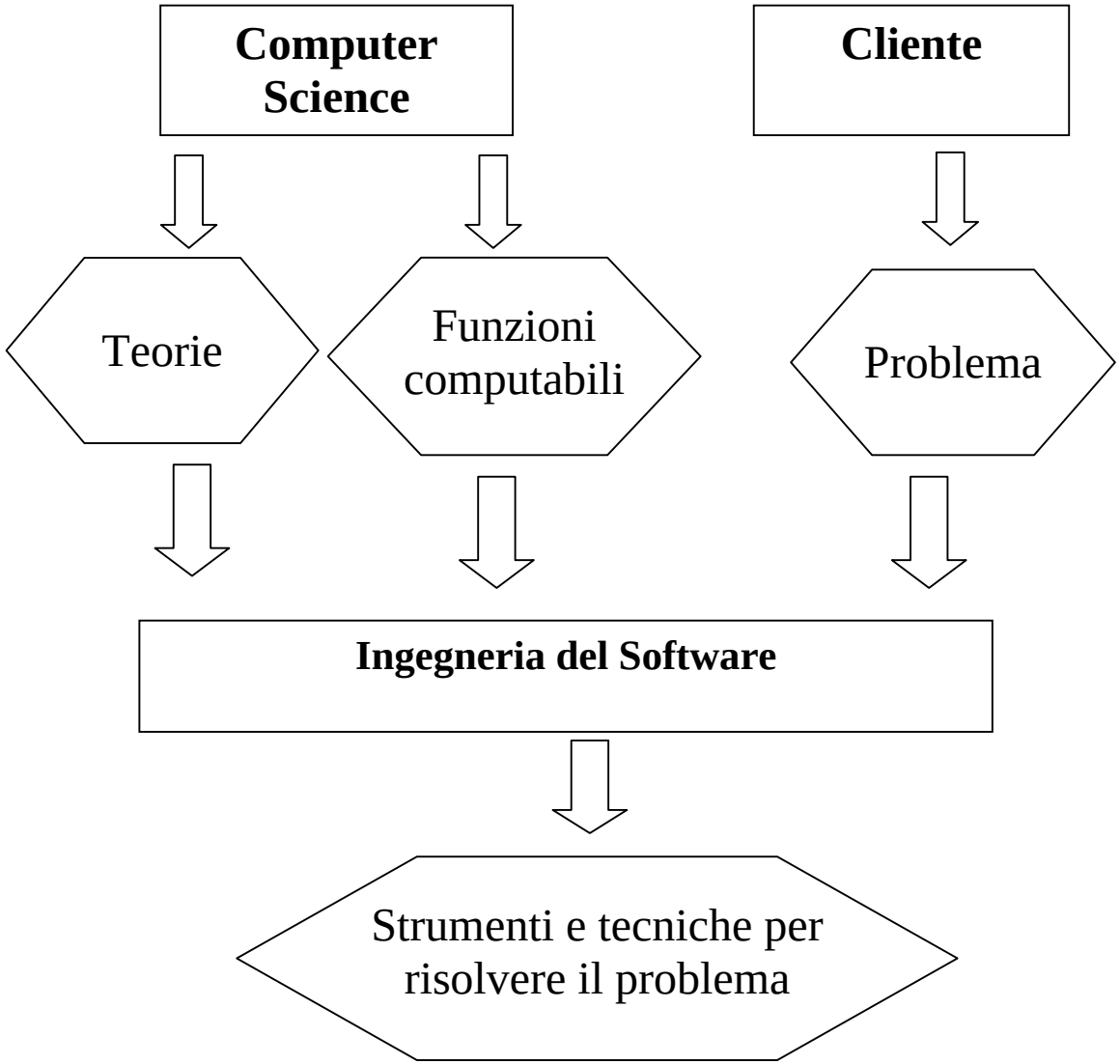
L’Ingegneria del Software

Definizione: disciplina che si occupa di studiare teorie, metodi e strumenti per migliorare la qualità dei prodotti sw

Obiettivo: mettere in grado lo specialista di sw di affrontare la complessità di grossi progetti gestendo in modo produttivo le (limitate) risorse disponibili, specie quelle umane, e sviluppando prodotti con le caratteristiche di **qualità** desiderate entro prefissati vincoli economici e temporali

Contenuti: teorici di base + metodi rigorosi + tecnologie + sensibilità pratica di tipo empirico
L’oggetto non è la computazione e la teoria sottostante ma piuttosto l’uso dell’informatica per risolvere dei problemi (chimico vs. ing. chimico)

Relazione tra Computer Science e Software Engineering



Caratteristiche esclusive del sw

- Malleabilità apparente (sembra facile modificare il sw ma non è facile operare modifiche al fine di ottenere un certo comportamento desiderato)
- Assenza del processo di assemblaggio
- Immaterialità $\Rightarrow$ solo progettazione

Processo produttivo del sw (= ciclo di vita): dove termina lo *sviluppo* di una applicazione e dove inizia la sua *evoluzione*?

La qualità è relativa

Prospettive sulla qualità	Osservatore
Trascendentale: la qualità è qualcosa che possiamo riconoscere ma non definire (ideale verso cui tendere)	*filosofo*
Utente (qualità esterna): la qualità è adatta allo scopo	clienti, operatori commerciali
Produzione: la qualità è conforme alle specifiche (del processo costruttivo)	sviluppatori
Prodotto (qualità interna): la qualità deriva dalle caratteristiche intrinseche del prodotto (metriche del sw)	ricercatori
Valore: la qualità dipende da quanto il cliente è disposto a pagarla	

Fattori di qualità di prodotti e processi sw

Il nocciolo dell'ingegneria del software è l'attenzione alla qualità (Pressman, 2004)

Qualità esterne: percepibili da un osservatore esterno che esamina una black-box (sono quelle di interesse per SE)

Qualità interne: percepibili esaminando la struttura interna di una white-box (sono quelle che permettono di realizzare le qualità esterne)

Fattori di qualità del prodotto

1. Affidabilità: i risultati delle elaborazioni sono quelli voluti o presentano disturbi tollerabili (concetto soggettivo); deontologia professionale dell'ingegnere del sw: necessità di verificare a posteriori l'effettiva affidabilità delle applicazioni sviluppate (ad es. mediche o safety critical)
2. Correttezza: il sw è conforme ai requisiti (caratteristica oggettiva)
   N.B. sw corretto non è necessariamente sw affidabile
3. Robustezza: il sw si comporta in modo accettabile anche in corrispondenza di situazioni non specificate dai requisiti
4. Sicurezza: l'elaborazione protegge l'accesso a info private
5. Safety (innocuità): il livello di pericolosità dell'elaborazione non è mai intollerabile
6. Prestazioni: tempo di esecuzione e memoria occupata (valutazione attraverso la teoria della complessità del calcolo, oppure mediante misure durante esecuzioni-campione, oppure misure in ambienti simulati, oppure analisi di un modello del sistema – es. analisi probabilistica attraverso la teoria delle code)

Fattori di qualità del prodotto (cont.)

7. Usabilità: l'utente si trova a proprio agio nell'uso del sw grazie a interfacce autoesplicative e modalità di interazione intuitive
8. Interoperabilità: l'applicazione può facilmente integrarsi con altre
9. Flessibilità / Configurabilità
10. Controllabilità delle versioni

Fattori di qualità sia del processo che del prodotto

11. Verificabilità: le proprietà del prodotto sono verificabili, lo stato di avanzamento di un progetto deve essere controllabile, unitamente al grado di soddisfacimento dei vincoli
12. Manutenibilità: il sw è stato progettato per essere modificabile in modo semplice e affidabile, affinché possa facilmente evolvere
13. Riusabilità
14. Comprensibilità (di tutti i prodotti del processo e del processo stesso → processo visibile, o trasparente, e prodotto possibilmente modulare)

Manutenzione del sw

Attività post-rilascio (50-70% dei costi complessivi); si articola in:

- Correttiva (≅20% dei costi di manutenzione): correzione degli errori residui
- Adattativa(20-25%): adattamento a cambiamenti nell'ambiente
- Perfettiva (50%, è conseguenza della malleabilità del sw): aggiunta/rimozione di funzionalità, miglioramento di alcune caratteristiche di qualità → differenza del concetto di manutenzione rispetto all'ingegneria tradizionale

Situazione di retroazione: l'organizzazione del lavoro fa nascere esigenze di automazione mediante sw → introduzione del sw e conseguente modifica dell'organizzazione del lavoro → necessità di modifiche del sw

Manutenzione del processo produttivo: modifiche ai piani inizialmente previsti

Fattori di qualità del processo

15. Produttività (difficile da caratterizzare)

Aree applicative del sw

Classificazione per insiemi omogenei di caratteristiche di qualità da quantificare all'avvio di un progetto

- Sistemi informativi
- Sistemi in tempo reale
- Sistemi distribuiti
- Sistemi embedded

Molti sistemi hanno caratteristiche comuni a più aree (es. sistema informativo distribuito e in tempo reale, sistema embedded in tempo reale)

Sistemi informativi

- Si interfacciano a una base di dati
- Sono orientati ai dati
- Molti forniscono una GUI web
- Molti consentono personalizzazioni (ad es. definizione e generazione di nuovi rapporti)

Es.: sistemi bancari, bibliotecari, di gestione del personale

Sistemi in tempo reale

- Devono rispondere a determinati eventi in ingresso entro un intervallo di tempo predefinito e spesso molto limitato (quindi sussistono vincoli quantitativi relativi ai tempi di risposta)
- Spesso fanno parte di sistemi complessi (di automazione di fabbrica, sorveglianza, ecc.)
- Sono orientati al controllo

Es.: sistemi di monitoraggio di pazienti/impianti, sistemi di controllo del volo di un aero, sistemi di difesa, sistema di gestione del mouse

N.B. Definizione sbagliata di sistema in tempo reale: sistema che richiede tempi di risposta veloci. Infatti una risposta troppo veloce può essere scorretta quanto una risposta tardiva

Sistemi in tempo reale: una duplice classificazione

In base all'algoritmo di schedulazione (che stabilisce l'ordine di esecuzione delle azioni che devono essere compiute dal sistema):

- basati su priorità (a ogni azione è associata una priorità)
- basati su deadline (a ogni azione è associato il tempo entro cui deve essere iniziata/completata)

In base al momento di esecuzione delle azioni:

- basati su eventi (le azioni devono essere eseguite in risposta a eventi)
- basati sul tempo (le azioni devono essere eseguite a istanti di tempo predeterminati; tutti i componenti esecutori di azioni sono sincronizzati)

Sistemi distribuiti

Distribuzione su computer diversi, collegati da una rete di TLC, di

- dati e/o
- componenti sw eseguibili

Java definisce un linguaggio intermedio (*bytecode*) che può essere interpretato su ogni computer $\rightarrow$ i componenti possono essere caricati in rete in maniera dinamica quando ciò risulta necessario (mobilità del codice)

Requisiti da definire:

- Livello di distribuzione
- Possibilità di tollerare il partizionamento (della rete in sottoreti disgiunte)
- Tolleranza per l'indisponibilità di uno o più computer

Sistemi distribuiti: nuove strade per ottenere la qualità

L’eventuale replica dei dati su più macchine aumenta:

- affidabilità
- prestazioni

L’eventuale trasferimento dinamico del codice al nodo che memorizza i dati sui quali il codice deve operare (es. applet Java) migliora le prestazioni

Sistemi embedded

- Spesso privi di interfacce rivolte all'utente finale
- Il sw è solo uno dei componenti di questi sistemi, quello che controlla gli altri, interfacciandosi con essi

Es. sw di controllo di aerei, robot, elettrodomestici, cruscotto dell'automobile, telefoni cellulari, macchine distributrici

Approccio ingegneristico alla produzione del sw

- Mira a garantire alti livelli di controllo sulla qualità grazie a un processo formale che descrive le varie fasi che vanno seguite nello sviluppo del sw
- Molte persone lavorano allo stesso progetto, quindi la documentazione è importante (vedi progetti dell'ingegneria civile): "disegno" di alto livello e di dettaglio
- La fase iniziale di raccolta dei requisiti del cliente prevede che siano esibiti diagrammi, prototipi e documenti tali da garantire che ciò che si svilupperà è effettivamente ciò che vuole il cliente
- Centralità della progettazione, fondamentale la componente intellettuale creativa
- Il testing del prodotto segue un processo ben formalizzato, in cui i requisiti iniziali del cliente sono riesaminati per verificarne la corretta implementazione

L’ingegnere del sw

Alle diverse fasi corrispondono figure professionali diverse (suddivisione di competenze e ruoli); nei piccoli progetti 2 o 3 persone coprono tutti i ruoli:

- analista
- progettista
- programmatore
- ingegnere del testing
- istruttore (mostra agli utenti come usare il sistema)
- gruppo manutenzione (include analisti, progettisti, …)
- gruppo controllo qualità
- librarian (redattore di documenti usati durante la vita del sistema, ad es. specifiche dei requisiti, descrizioni di progetto, documentazione di programma, manuali di addestramento, ecc.)
- squadra di gestione delle configurazioni (documenta le corrispondenze fra requisiti, progetto, implementazione e test, ad es. per poter dire ai manutentori quale funzione modificare se è richiesto un cambiamento nei requisiti, ecc.; inoltre coordina le diverse versioni di un sistema)

I fattori che hanno cambiato l'ingegneria del sw

1. Criticità del time-to-market: prodotti e servizi devono essere pronti prima di quelli della concorrenza →
   - ✓ le tecniche tradizionali di revisione e testing non possono essere usate se richiedono un tempo troppo lungo che non sarà recuperato in termini di riduzione dei malfunzionamenti
   - ✓ il tempo speso nell'ottimizzazione del codice per aumentare la velocità di esecuzione o ridurre lo spazio di occupazione può essere un investimento poco saggio, è meglio invece richiedere più memoria (vedi anche punto successivo)
2. Diminuzione dei costi dell'hw

I fattori che hanno cambiato l'ingegneria del sw (cont.)

3. Disponibilità di notevole potenza di calcolo su personal computer: gli utenti sviluppano applicazioni spreadsheet, sistemi informativi, piccoli programmi, interfacce specializzate, simulazioni, ecc. → agli ingegneri del sw è richiesto lo sviluppo di sistemi progressivamente più complessi
4. Diffusione di reti locali e geografiche → a utenti e sviluppatori è sempre più facile trovare info e non è necessario sviluppare applicazioni per tale ricerca
5. Tecnologia orientata agli oggetti → disponibilità di vaste collezioni di moduli rapidamente riusabili
6. Interfacce grafiche
7. Limiti dei modelli di processo rigidi sviluppati inizialmente dall'ingegneria del sw → sviluppo parallelo di sottosistemi

Principi chiave dell'ingegneria del sw moderna

- Astrazione: descrizione del problema che deve essere risolto dal sistema sw e del suo contesto a un livello di generalizzazione tale che consenta di concentrarsi sugli aspetti chiave senza perdersi nei dettagli, al fine di trovare soluzioni semplici ed eleganti (N.B. l'astrazione di un problema non è il trasferimento della sua descrizione dal mondo reale a un mondo ideale, ad es. quello matematico); tipicamente comporta l'identificazione di (gerarchie di) classi di oggetti, senza passare in rassegna ogni singolo oggetto
- Metodi e notazioni di analisi e progettazione: la mancanza di metodi, notazioni e tool standardizzati che consentano ai membri di un team di comunicare fra di loro, di documentare le decisioni, di valutare la completezza e consistenza dei modelli è uno dei problemi chiave dell'Ingegneria del sw (esistono tanti metodi, notazioni e tool, ciascuno dei quali considera però solo un sottoinsieme degli aspetti e dei componenti di un progetto)

Principi chiave dell'ingegneria del sw moderna (cont.)

- Prototipazione: processo spesso iterativo che serve a identificare i requisiti di un sistema e/o per dimostrare la fattibilità di un progetto, coinvolgendo utenti e/o committenti
- Architettura sw: gli effetti della scelta dell'architettura si fanno sentire nell'intero processo di sviluppo; tanto più indipendenti sono le unità architetturali, tanto più facile è progettarle e realizzarle separatamente; esistono almeno 5 approcci non mutuamente esclusivi (si possono, ad es., applicare approcci diversi a diverse parti del sistema) per partizionare il sistema:
  - ✓Decomposizione modulare (assegnamento di funzioni a moduli)
  - ✓Decomposizione basata sui dati (cioè sulle strutture dati esterne)
  - ✓Decomposizione orientata agli eventi (che il sistema deve gestire)
  - ✓Progettazione outside-in (basata sugli ingressi del sistema)
  - ✓Progettazione OO (identificazione delle classi e delle loro relazioni reciproche)

Principi chiave dell'ingegneria del sw moderna (cont.)

- Processo sw: è ormai riconosciuto che l'organizzazione e la disciplina delle attività di progetto e dei loro prodotti contribuisce a migliorare la qualità del sw e a rendere più rapido il suo sviluppo
- Riuso (di requisiti, parti di progetti, dati, ecc.); ostacoli:
  - ✓Talvolta è più veloce costruire un piccolo componente che cercarne uno in un deposito di componenti riusabili
  - ✓Costruire un componente sufficientemente generale da poter essere usato richiede tempo aggiuntivo
  - ✓È difficile documentare le valutazioni di qualità e testing effettuate cosicché chi riusa si senta sicuro
  - ✓Non è chiaro chi sia responsabile in caso di fallimento di un componente riusabile o in caso sia necessario un aggiornamento
  - ✓È dispendioso in termini di tempo e denaro comprendere e riusare un componente scritto da qualcun altro

Principi chiave dell'ingegneria del sw moderna (cont.)

- Metriche: le misure quantitative di prodotto e di processo sono un aspetto chiave per l'avanzamento di un progetto, per supportare analisi e decisioni, per il miglioramento e per il confronto di progetti diversi
- Strumenti e ambienti integrati: i ricercatori propongono delle linee guida per confrontare ambienti diversi ma una delle maggiori difficoltà di tale confronto è costituita dal fatto che raramente gli ambienti coprono l'intero ciclo di vita
  → è l'utente che deve effettuare l'integrazione di più tool, tenendo conto di:
  - ✓Integrazione di piattaforma, cioè possibilità dei tool di collaborare su una rete eterogenea
  - ✓Integrazione di presentazione, cioè analogia d'interfaccia
  - ✓Integrazione di processo, cioè copertura delle diverse fasi
  - ✓Integrazione dei dati
  - ✓Integrazione del controllo, cioè capacità di un tool di notificare a un altro, facendolo entrare in azione