FACOLTÀ DI INGEGNERIA
UNIVERSITÀ DEGLI
STUDI DI PALERMO

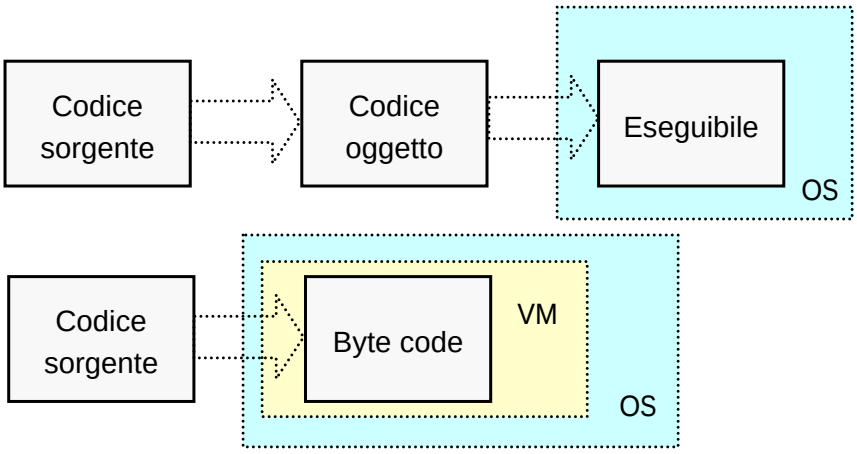
Corso di Laurea Specialistica
Ingegneria Gestionale

Sistemi ICT per il Business Networking

Il Software

Docente: **Vito Morreale** (vito.morreale@eng.it)

Introduzione



Il software: da arte ad artigianato ad industria

- Fase pionieristica:
  - **Scopo**: “macinare” numeri
  - **Strumenti** di programmazione molto limitati (es. Linguaggi di programmazione di basso livello)
  - **Programmatore** = **utente** (es. Scienziati, ricercatori, tecnici)
  - Utilizzo limitato delle “applicazioni”

ARTE

Il software: da arte ad artigianato ad industria

- Avvento delle **applicazioni gestionali**
  - **Scopo**:
    - Gestione di informazioni di diversa natura (operative, strategiche, ecc.)
    - Supporto alle decisioni (es. gestione del personale, produzione, gestione delle scorte, ecc.)
  - **Programmatore** ≠ **utente** (es. impiegati)
    - Sviluppo di una professionalità specifica in grado di produrre e tenere aggiornate le applicazioni che altri usano
    - Centri EDP nelle aziende
    - Nascita delle software house

ARTIGIANATO

Il software: da arte ad artigianato ad industria

- Produzione di software come **industria**
  - **Scopo**:
    - Aumentare la produttività e la qualità del software (prodotto) attraverso l'introduzione o il miglioramento dei processi
  - **Gruppi di lavoro** di grandi dimensioni: il lavoro deve essere pianificato e coordinato
  - Utilizzo di **strumenti** automatici (IDE, CASE)
  - Certificazione della **qualità** dei prodotti
  - Gestire i **cambi di personale**
  - Linee guida di provata efficacia ➔ **METODOLOGIE**
  - **Assemblaggio** e **riuso** di componenti già esistenti

INDUSTRIA

Il software: peculiarità

- A parità di dimensione, la **COMPLESSITA'** di capire, descrivere, progettare un frammento di software è di gran lunga superiore a quella corrispondente per qualunque altro prodotto ➔ difficoltà nello sviluppo
- Il software è **INVISIBILE** e **INTANGIBILE**: non esistono astrazioni o analogie figurative che possono aiutare nella comprensione, descrizioni e comunicazione
- Alcune applicazioni sono intrinsecamente **CRITICHE** (es. transazioni bancarie, monitoraggio della salute dei pazienti, sicurezza, militare, ecc.) ➔ impossibile tollerare errori

Ingegneria del software

- **Definizione informale**: disciplina basata su solide basi teoriche e metodologiche che permettano la progettazione, produzione e manutenzione di applicazioni che forniscano le caratteristiche di qualità previste mediante l'**uso delle risorse previste**
- **Definizione IEEE** (Glossario): l'approccio sistematico allo sviluppo, all'operatività, alla manutenzione ed al ritiro del software
- **Altra definizione**: l'ingegneria del software è la disciplina (teorie, metodi e strumenti) tecnologica e manageriale che riguarda la produzione sistematica e la manutenzione dei prodotti software che vengono sviluppati e modificati entro i tempi ed i costi preventivati e con le qualità richieste

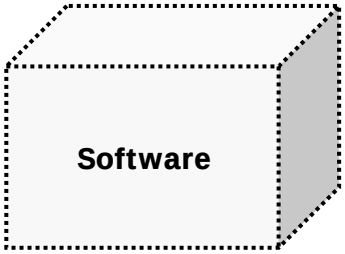
TEORIA + METODI + TECNOLOGIE + PRATICITA'

Fattori di qualità del Software (1)

- **Esterne** (percepibili da un osservatore esterno per cui il sistema è una black-box):
  - Affidabilità
  - Correttezza
  - Robustezza
  - Sicurezza & Innocuità
  - Efficienza
  - Usabilità
  - Interoperabilità

Fattori di qualità del Software (2)

- **Interne** (osservate esaminando la struttura interna del sistema, come se fosse una scatola trasparente o white-box)
  - Estendibilità
  - Riusabilità
  - Modularità
  - Comprensibilità
  - Manutenibilità
  - Portabilità



Qualità interne vs. esterne

- Le **qualità interne** rappresentano il modo in cui le qualità esterne possono essere raggiunte
- Alcune qualità esterne sono influenzate da alcune qualità interne
- Le qualità interne **implementano** (sono un modo per realizzare) le qualità esterne
- **Perché** definire le qualità del software?
  - Fornire un **linguaggio comune**
  - Fornire una lista di riferimento per gli **obiettivi di qualità** da raggiungere o da valutare a posteriori
- Spesso definibili soltanto in maniera intuitiva ed informale

Affidabilità

- Qualità del software del quale è possibile fidarsi, le qui **funzionalità corrispondono alle aspettative**
  - In caso di scostamenti, questi devono essere **tollerabili**
- E' un concetto largamente **soggettivo**, basato sulla valutazione di "scostamento intollerabile"

Correttezza

- Il software **fa** quello per cui è stato pensato, progettato e sviluppato: **se i requisiti sono rispettati**
- Legata alla **definizione formale** delle funzionalità del sistema (requisiti), ...
- ... **ma** purtroppo non sempre i requisiti specificano esattamente tutto ciò che si ritiene importante ➔ i requisiti sono quasi sempre incompleti
- **Importante**. Un software corretto potrebbe comportarsi in maniera ritenuta inaffidabile

Robustezza

- Comportamento corretto anche nel caso di **situazioni non specificate** o non previste espressamente nei requisiti.

Sicurezza & Innocuità

- **Sicurezza**: riservatezza nell'accesso ad informazioni private e riservate, impedendo accessi non autorizzati, sia di natura involontaria che dolosa
- **Innocuità** (safety): denota il fatto che alcuni sistemi non entrano mai i uno stato in cui il livello di pericolo può essere intollerabile

Efficienza

- Legata alle **prestazioni**:
  - Tempo di esecuzione
  - Utilizzo di memoria
- **Teoria della complessità del calcolo**
  - Limiti asintotici: caso peggiore e caso migliore
- Strumenti per la misurazione:
  - Software monitor
  - Ambienti di simulazione
- Può anche essere considerata una **qualità interna**

Usabilità

- L'utente si trova a suo **agio** nell'utilizzo dell'applicazione: interfaccia utente amichevole (**friendly**), che non obblighi l'utente a comportamenti innaturali
- Enfasi sull'**utente**
- Psicologia cognitiva
- Fattor fisico/ergonomici
- Mentalità dell'utente
- Interfacce grafiche
- ... spesso ignorata

Interoperabilità

- Facilità di **interazione con altre applicazioni** al fine di svolgere un compito combinato più complesso
- Favorisce il riuso di componenti esistenti
- Problemi tecnologici e semantici
- Interoperabilità vs. Integrazione

Estendibilità

- Facilità con cui il software può essere adattato a **modifiche delle specifiche e dei requisiti**
- **IMPORTANTE** in programmi grandi e complessi

Riusabilità

- Facilità con cui il software può essere **re-impiegato** in applicazioni diverse da quella originaria
- **Evita** di inventare nuovamente soluzioni già pensate ed implementate (eventualmente da altri)

Modularità

- Grado di organizzazione del software in **parti** ben specificate ed interagenti
- Le parti devono essere il più possibili AUTONOME e con RIDOTTA INTERAZIONE tra di loro
- Conseguenza del motto latino **divide et impera**, secondo il quale la complessità viene ridotta dalla scomposizione in parti

Comprensibilità

- Capacità del software di essere compreso e controllato anche da parte di chi **non** ha condotto o effettuato il progetto o sviluppato il codice
- Si applica sia al **software** che al **processo**
- **Facilitata** dalla modularità
- La comprensibilità del software **facilita** l'analisi della correttezza ed il riuso

Manutenibilità

- **Manutenzione**: attività che seguono il momento del rilascio
- Facilità ed economicità nell'effettuare **modifiche** al software sviluppato
- Tipi di manutenzione:
  - **Correttiva**: tende ad eliminare gli errori che sono sfuggiti al controllo prima del rilascio
  - **Adattativa**: per adattarsi a diverse condizioni al contorno (es. portabilità, cambiamenti nella configurazione hw, compilatori, applicazioni con cui si interfaccia, ecc.)
  - **Perfettiva**: aggiunta di nuove funzionalità, eliminazione di funzionalità inutili, miglioramento di alcune caratteristiche di qualità (dopo il rilascio)

Portabilità

- Esecuzione dell'applicazione sviluppata su **più piattaforme hw e sw**
- Agevolata da:
  - **Linguaggi** compatibili su più piattaforme (per esempio Java)
  - **Macchine virtuali** (come avviene per HTML e Java)

Materiale didattico

- **Il software: processo e prodotto** (capitolo 1 del libro "Ingegneria del software" di A. Fuggetta, C. Ghezzi, et al.)
  - Anche se è degli anni 90, esprime alcuni concetti di base sulle caratteristiche del software