

Ingegneria del Software

Introduzione

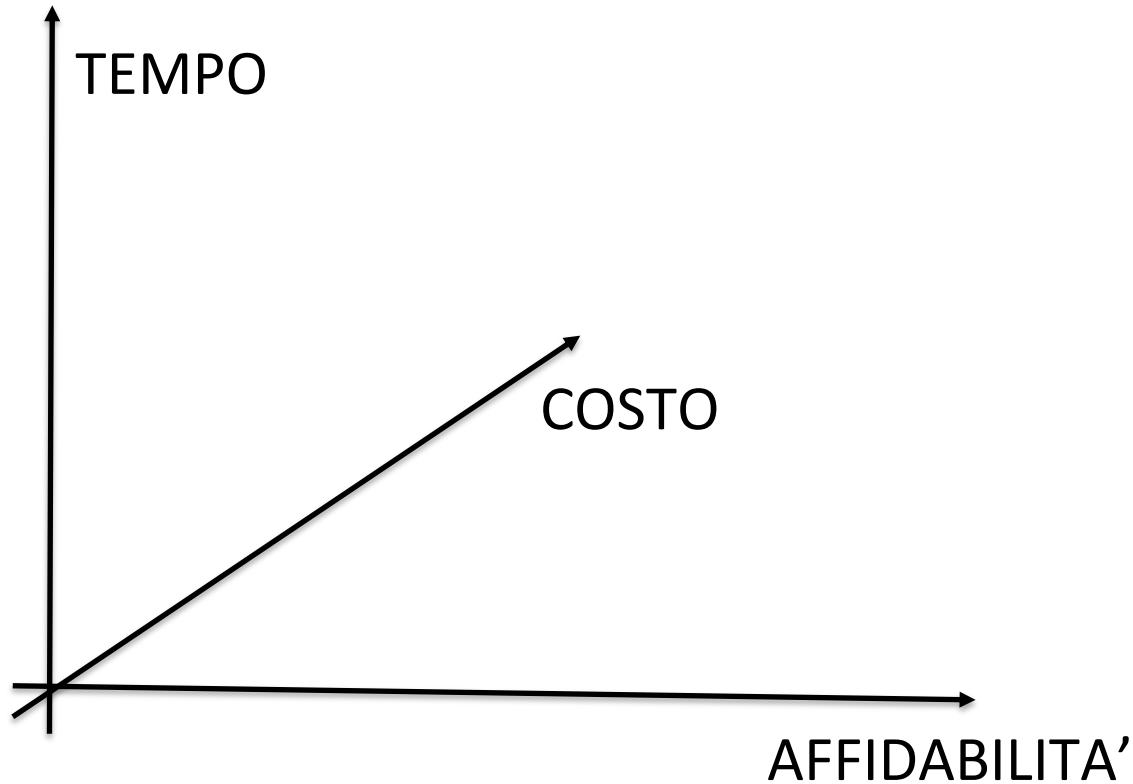
Prof. Paolo Giorgini

A.A. 2024-2025

Software

- Software sempre più indispensabile
 - servizi, sanità, PA, commercio, finanza, industria, intrattenimento
 - Pensate all'impatto dell'IA su tutti i settori produttivi
- Disponibilità del software
 - Venduto con un click
 - Fenomeno delle App
- Incorporato in ogni genere di sistema
 - trasporti, macchinari medici, sistemi telecom, sistemi militari, apparecchi industriali, intrattenimento, ufficio, ...

Dimensioni dello sviluppo software

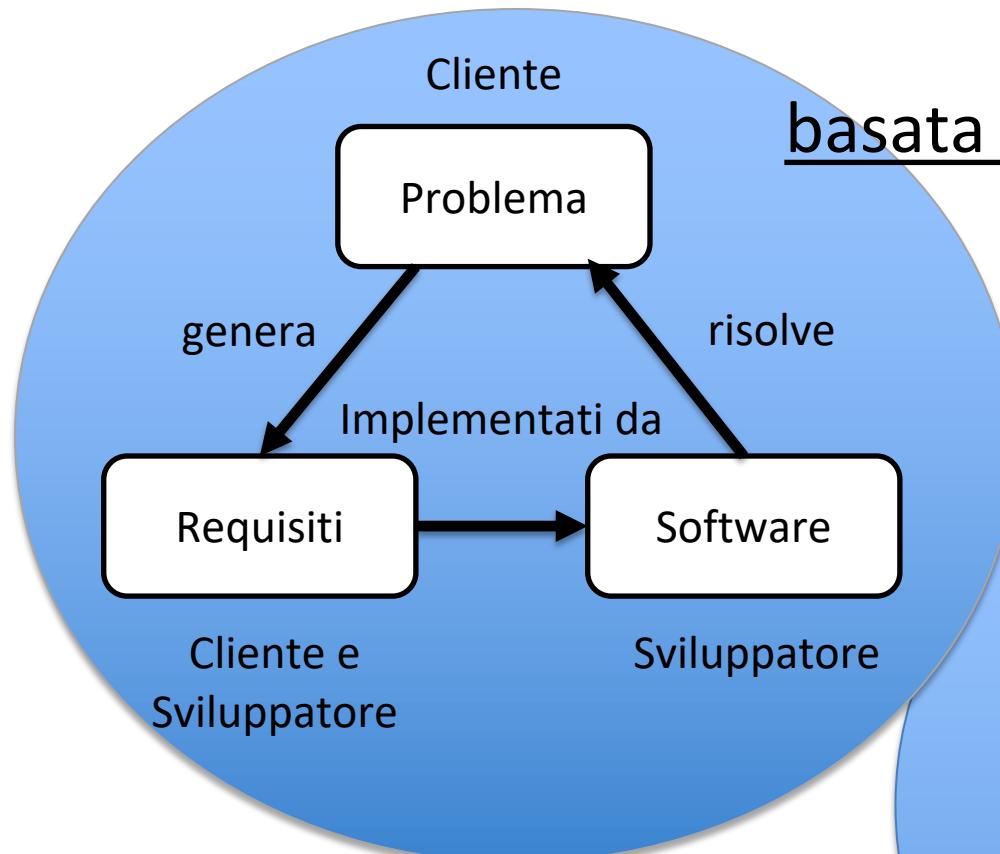


Manutenibilità, estensibilità, integrabilità, portabilità, usabilità, documentazione, ...

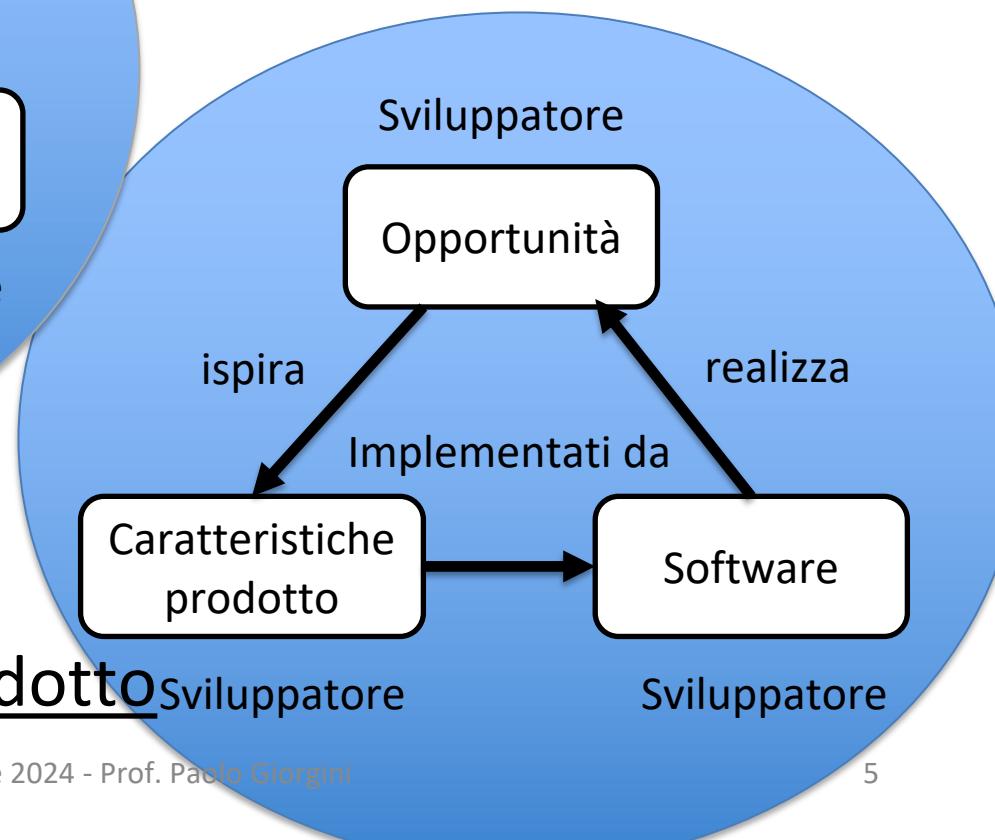
Qualità del software

- Software di qualità
 - Parametri di qualità e misure
 - Interni (es. verificabilità, manutenibilità, riusabilità, portabilità ...)
 - Esterni (es. correttezza, affidabilità, robustezza, efficienza, usabilità, scalabilità ...)
- Necessità di strumenti e tecniche per
 - aumentare la qualità e allo stesso tempo semplificare, accelerare e rendere più economico lo sviluppo
- Ingegneria del software
 - Principi, metodi, strumenti, processi

Ingegneria del Software



basata sul progetto



basata sul prodotto

Who are we?



Tipica interazione cliente-provider



Vorrei un software che mi permetta di vendere i miei prodotti online

Deve “collegarsi” al sistema di gestione del magazzino e delle vendite

Mi raccomando, la sicurezza!!

Versione web e mobile

Lo vorrei pronto a fine mese



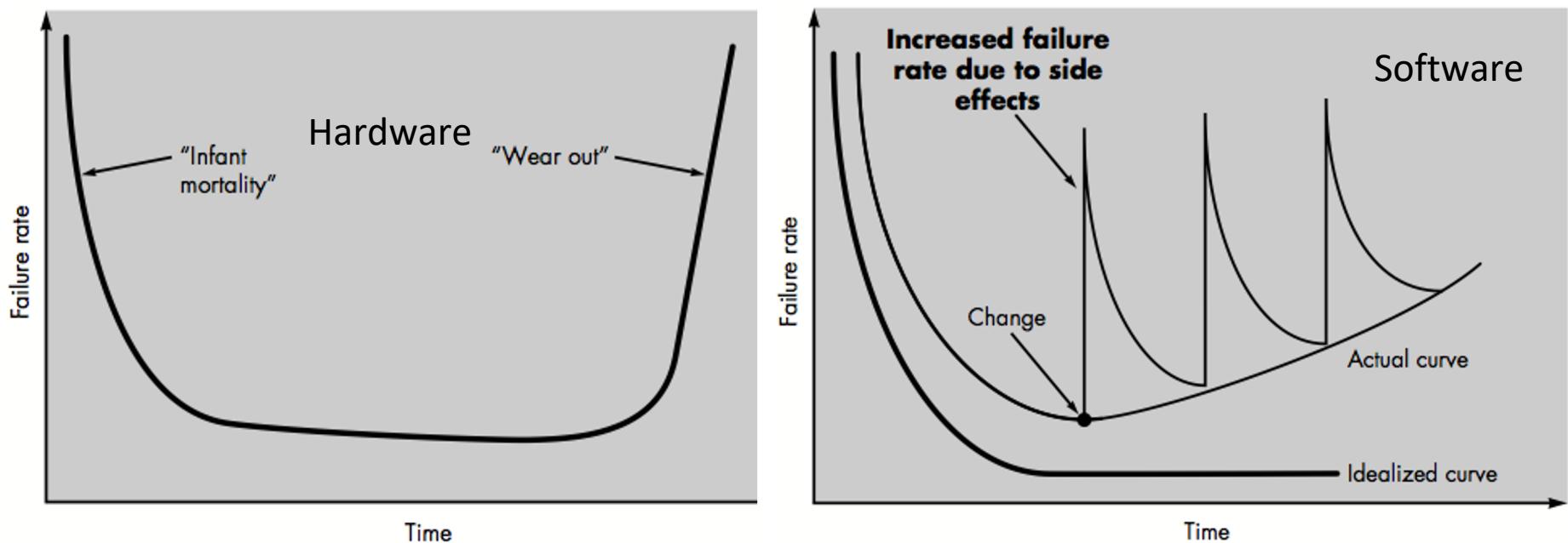
Prodotto intangibile

- Perché tempi così lunghi?
- Perché costi così elevati?
- Perché non sono stati individuati tutti gli errori prima della consegna?
- Perché così tanto tempo e impegno nel mantenimento di un software esistente?
- Perché è difficile misurare i progressi dello sviluppo del software?

Ingegneria del software - risposte a queste domande attraverso metodi sistematici e tecniche consolidate nel tempo

Caratteristiche del software

- Si sviluppa o si struttura, non si fabbrica nel senso tradizionale
 - Sviluppo software \neq fabbricazione hardware



Il software non si consuma

Software Evolution

Cambiamento continuo (1974). gli E-system devono essere costantemente incrementati (1974)

Crescita continua (1980). il contenuto funzionale degli E-

Riduzione della qualità (1996): gli utenti percepiscono la qualità degli E-system come in **costante declino**, a meno che i sistemi non vengano gestiti rigorosamente e adattati ai cambiamenti dell'ambiente operativo

adattatura una decrescita. Pertanto la crescita incrementale media rimane di più costante durante l'evoluzione del sistema

- La legge della crescita continua (1980): il **contenuto funzionale degli E-system deve essere costantemente incrementato** in modo da mantenere la soddisfazione degli utenti nel corso della vita del sistema
- La legge della riduzione della qualità (1996): gli utenti percepiscono la qualità degli E-system come in **costante declino**, a meno che i sistemi non vengano gestiti rigorosamente e adattati ai cambiamenti dell'ambiente operativo
- La legge di feedback del sistema (1996): I processi evolutivi di un E-system costituiscono sistemi di feedback multilivello, multiciclo, multiagente e devono essere trattati come tali per ottenere un miglioramento significativo

Fonte: Lehman, M., et al, "Metrics and Laws of Software Evolution—The Nineties View," *Proceedings of the 4th International Software Metrics Symposium (METRICS '97)*, IEEE, 1997, reperibile sul sito

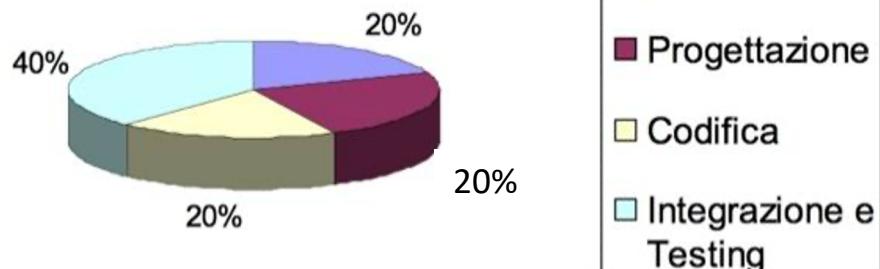
Ma siamo proprio sicuri?

- Management
 - Se siamo in ritardo, possiamo sempre recuperare aumentando il numero di programmatori
 - Se decido di far realizzare un progetto software a una terza parte posso restare tranquillo perché tutto il lavoro sarà svolto esternamente
- Programmatori
 - Scritto il software e messo in opera il nostro lavoro è finito
 - Il solo prodotto di un progetto concluso è il programma funzionante

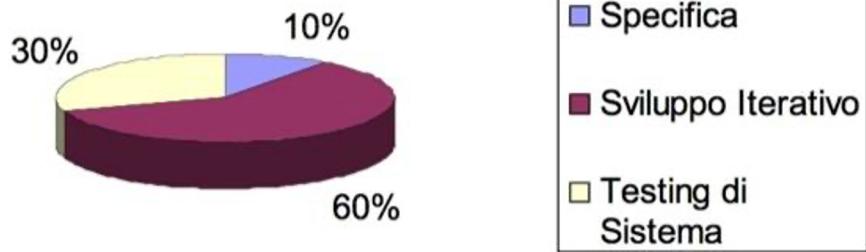
Costi del software

- I costi del Software > Hardware
 - mantenimento/evoluzione > sviluppo
 - variano in base al tipo, le prestazioni o l'affidabilità
- La distribuzione dei costi dipende anche dal tipo di **modello di sviluppo** adottato
 - Casi in cui il 60% dei costi è speso per le attività di sviluppo, il 40% per il testing

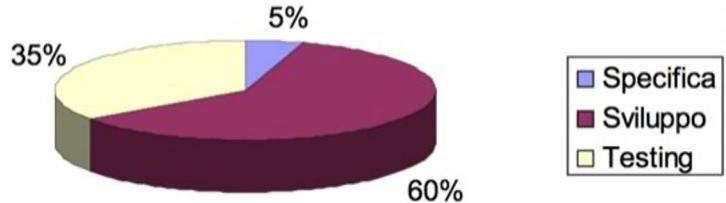
Costi per lo sviluppo Waterfall



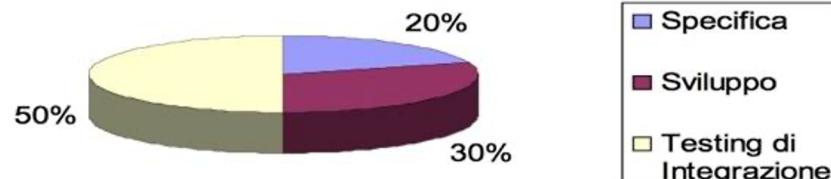
Costi dello Sviluppo Iterativo



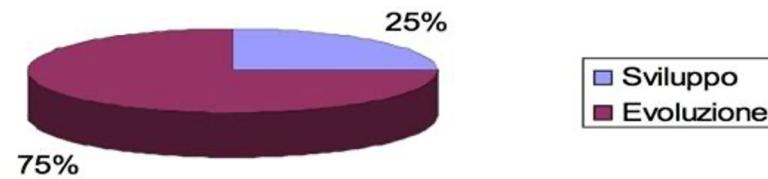
Costi di sviluppo di un prodotto generico



Costi del Component Based Software Engineering



Costi per lo Sviluppo ed evoluzione di sistemi di lunga durata



La qualità del software

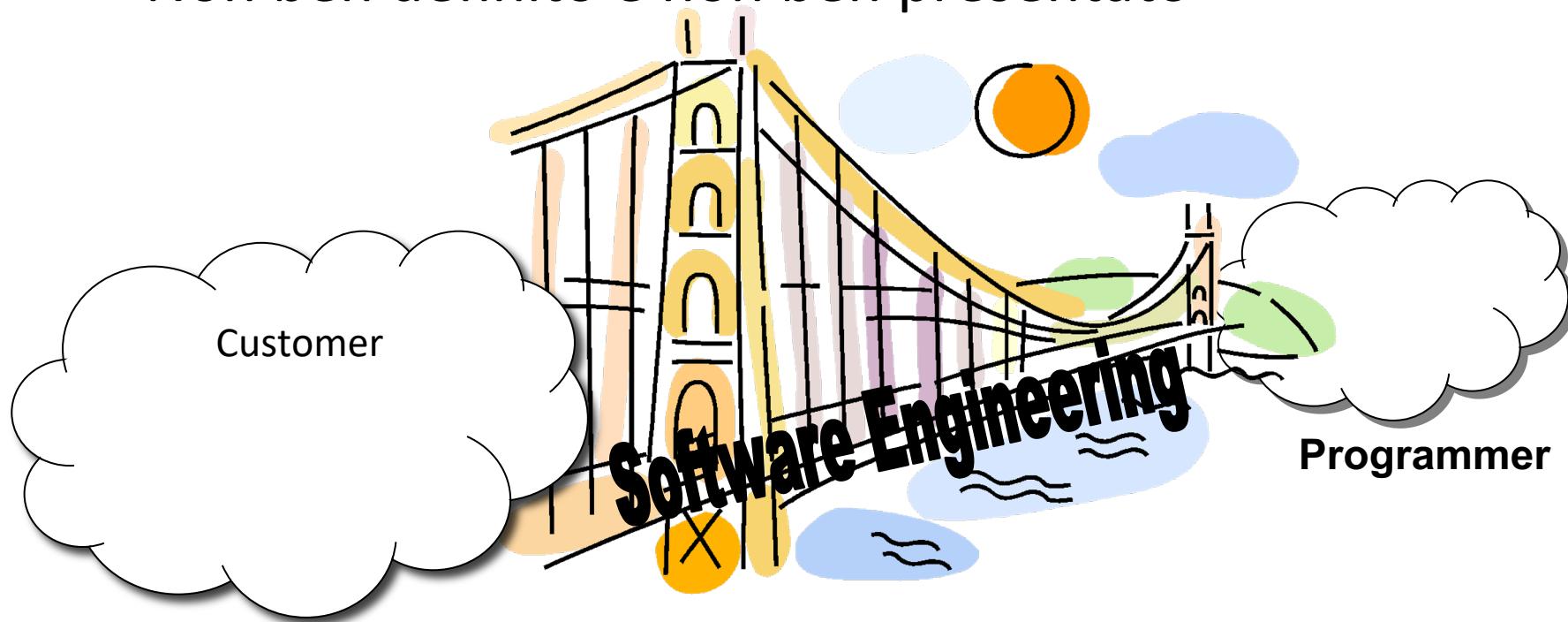
- Un software di qualità dovrebbe fornire le funzionalità e le prestazioni richieste ed essere allo stesso tempo
 - **mantenibile, fidato, efficiente, accettabile...**
- I problemi legati alla qualità del software sono difficili da rilevare:
 - molti difetti del software sono introdotti nelle fasi iniziali di specifica e design, ma verranno rilevati molto più avanti, in operatività....

L'ingegneria del software

- E' una disciplina ingegneristica che si occupa di **tutti gli aspetti** relativi allo sviluppo del software
- Gli ingegneri del software:
 - Adottano un **approccio sistematico** e organizzato per il loro lavoro;
 - Usano **strumenti e tecniche** appropriate
 - variabili a seconda del problema da risolvere, dei vincoli di sviluppo, e delle risorse disponibili

Scopo dell'Ingegneria del software

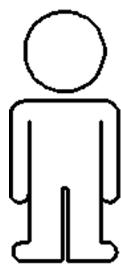
- Risolvere un problema complicato
 - Non ben definito e non ben presentato



Prima regola dell'ingegneria del software

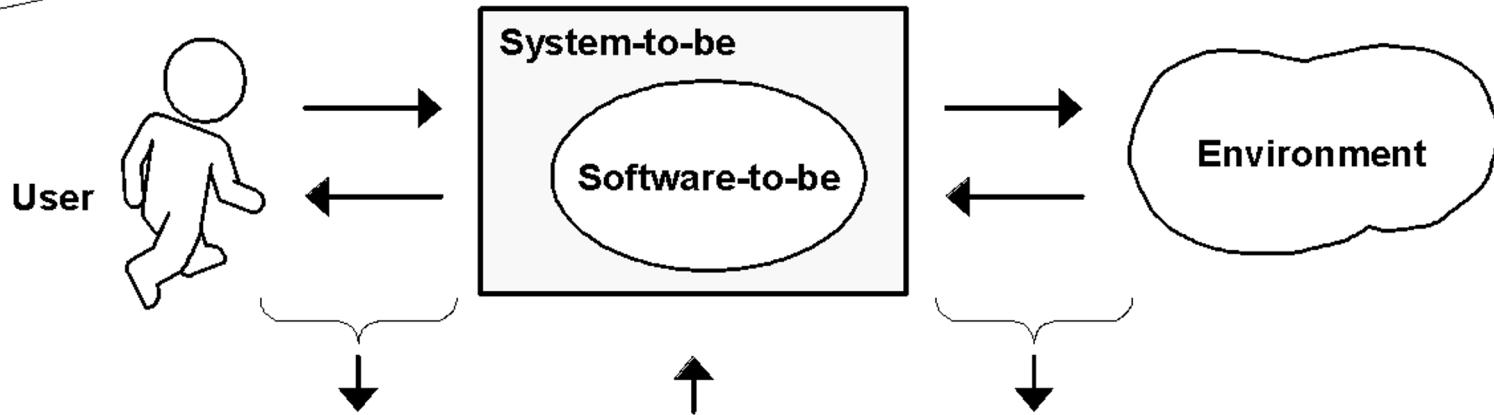
Un problema non pu' essere risolto senza averlo prima capito

Ruoli dell'Ingegneria del Software



Customer:

Requires a computer system to achieve some *business goals* by user interaction or interaction with the environment in a specified manner



Software Engineer's task:

To *understand how* the system-to-be needs to interact with the user or the environment so that customer's requirement is met and *design* the software-to-be

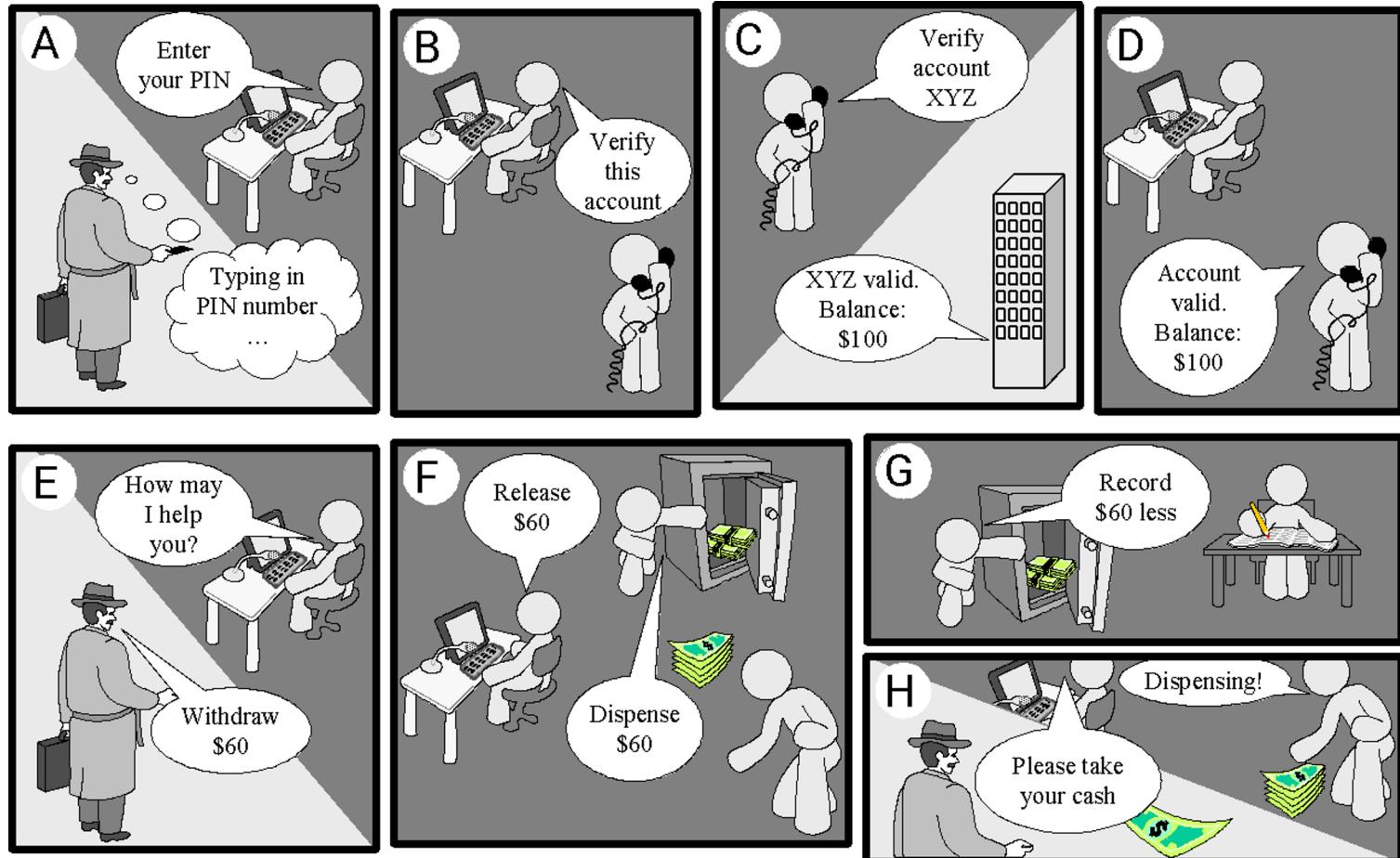
May be the same person



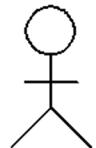
Programmer's task:

To *implement* the software-to-be designed by the software engineer

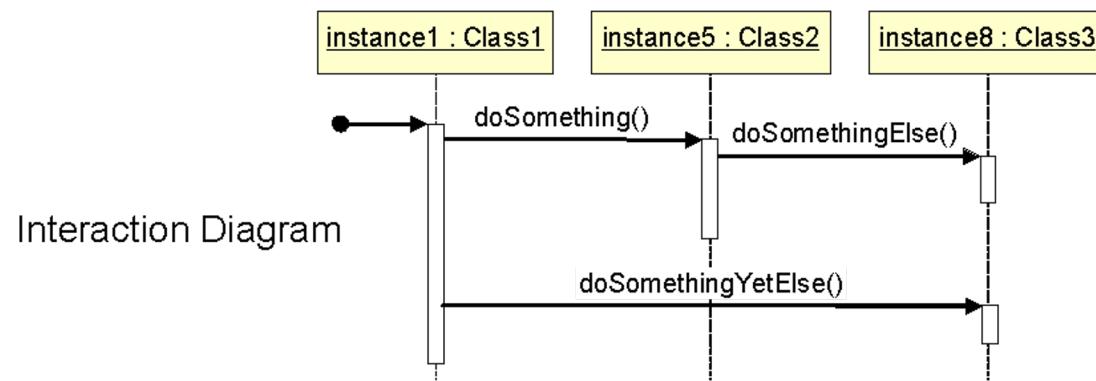
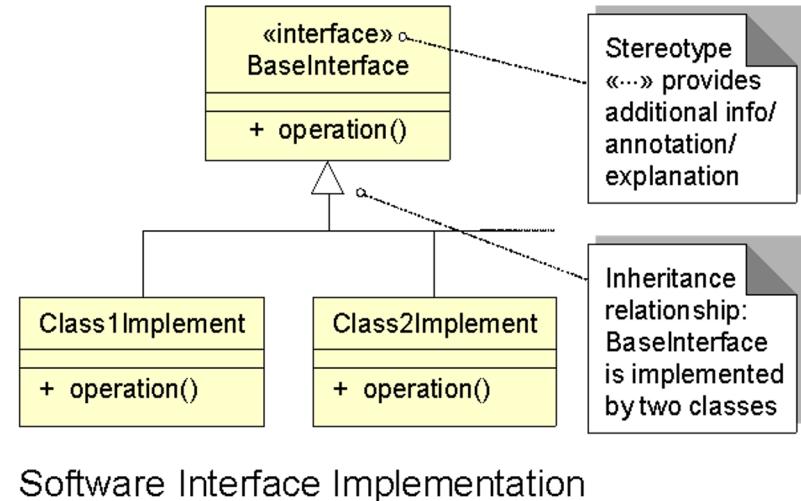
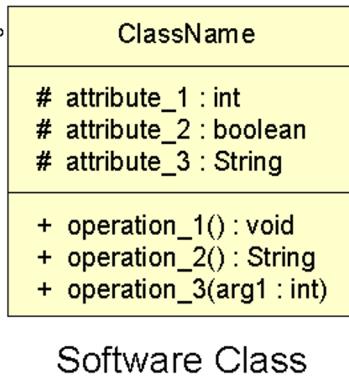
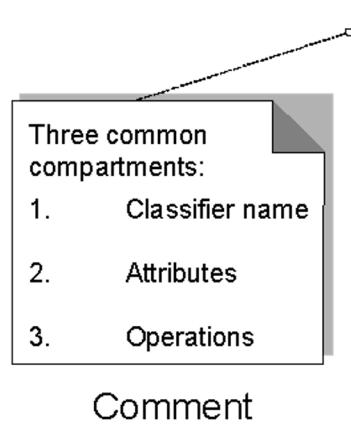
Specificare un problema



Specifica del problema in UML



Actor

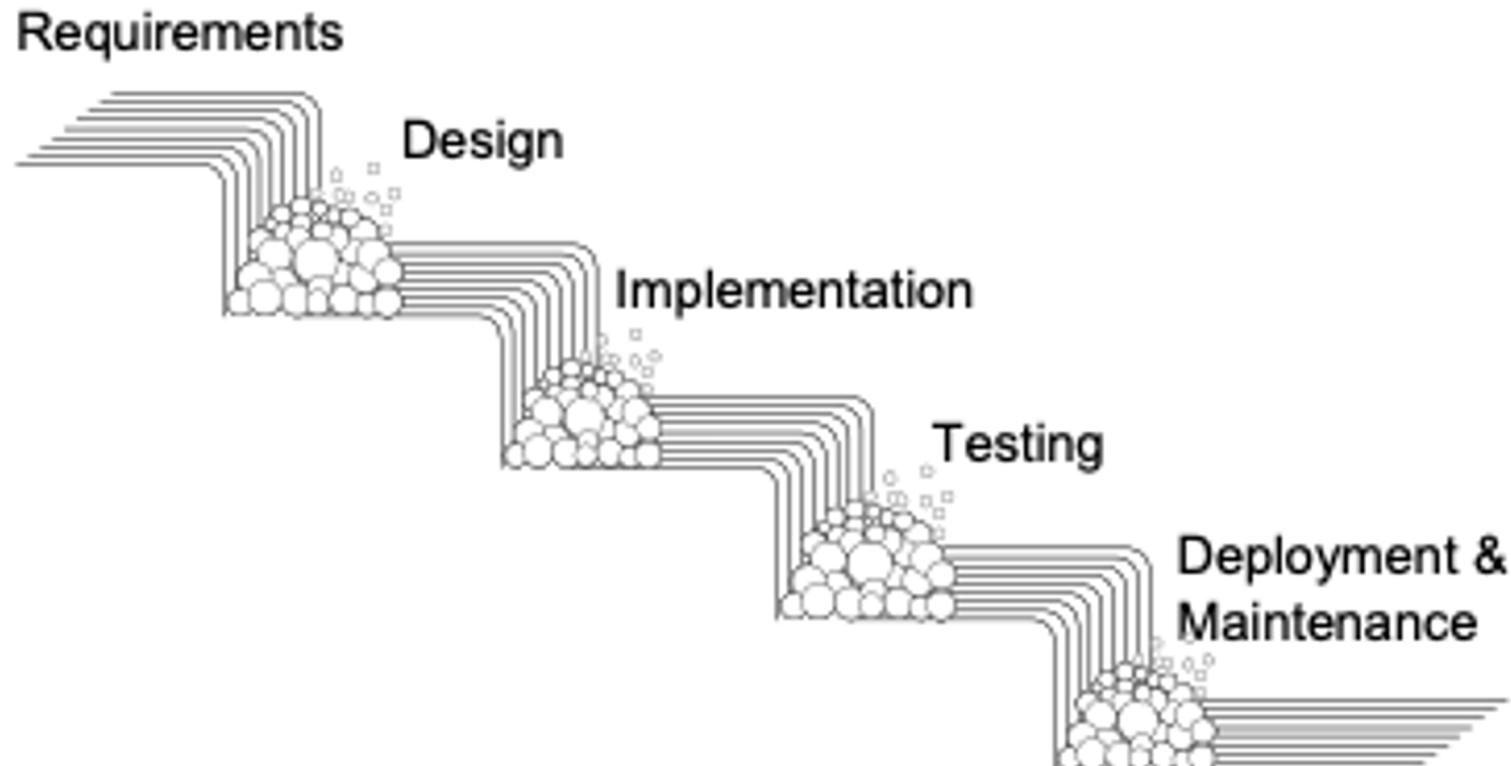


Processi Software

- Un insieme di attività aventi per obiettivo lo sviluppo o l’evoluzione di un sistema software
- Ogni processo tipicamente include:
 - **Specifico** – 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

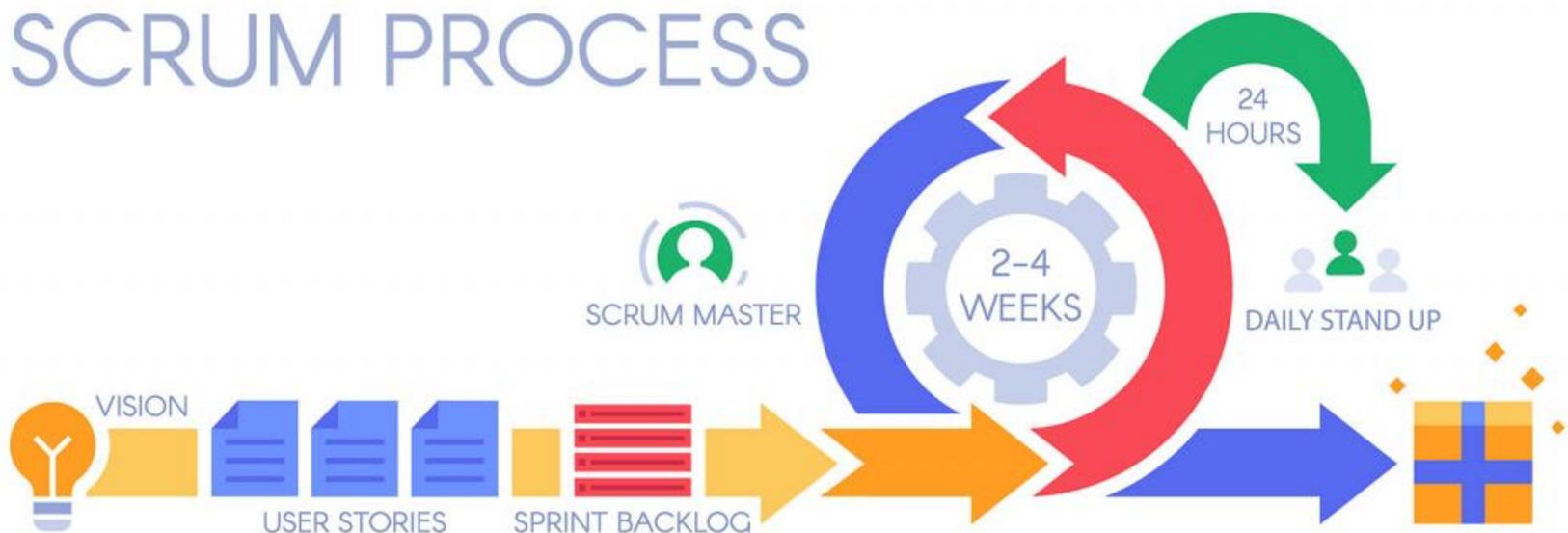
Modello di processo software

- Descrive il processo: **Waterfall**, Iterative, Incrementale, Component-based....



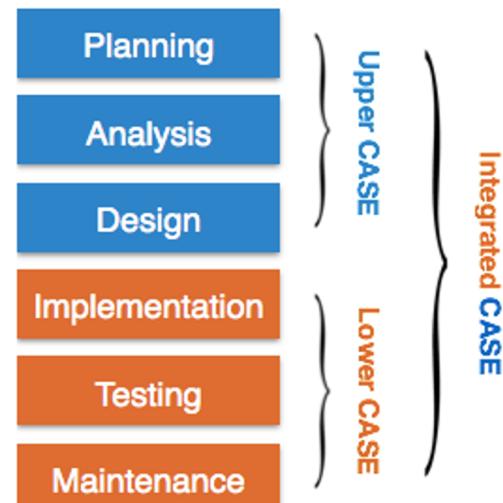
Processo Agile

SCRUM PROCESS



Metodi, strumenti, standard

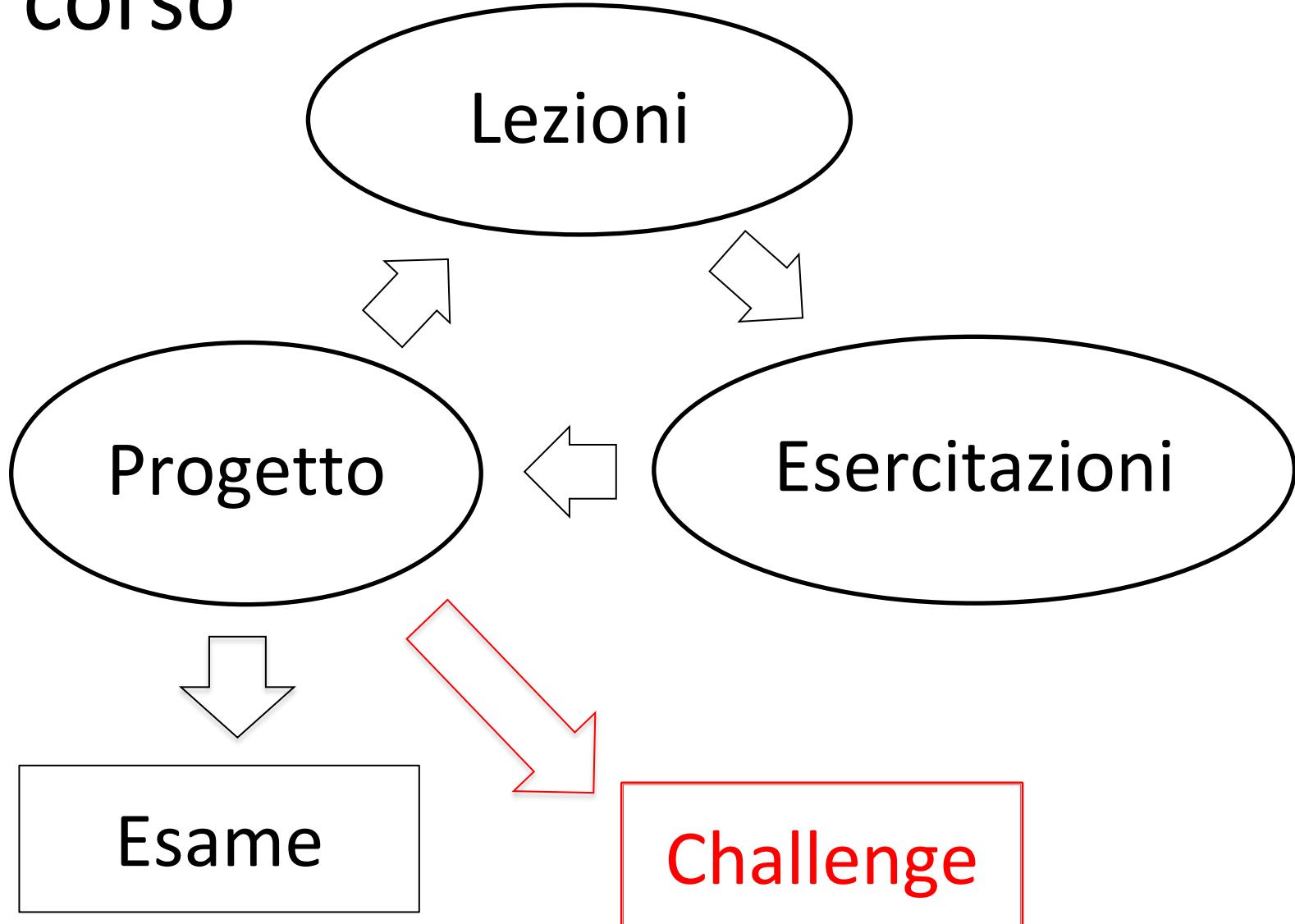
- Metodi: Approcci strutturati per sviluppare software di qualità, a costi contenuti. (Es. SADT, JSD, OOA, OOD, etc..)
 - Es. Specificano i modelli da usare, le regole a cui sottostare, forniscono una guida alle attività dei processi e alla relativa organizzazione.
- Strumenti: Sistemi software usati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



Obiettivo del corso

- L’obiettivo del corso è di fornire **concetti, principi, approcci e tecniche** dell’ingegneria del software.
 - Affronteremo tematiche legate all’analisi e alla progettazione di un sistema software complesso esplorando modelli di processo di sviluppo, linguaggi di modellazione e tecniche di verifica e validazione.

Il corso



Lezioni

- Teoriche, Tutorials, Esercitazioni
- Seminari
 - Aziende ed esperti
- Zoom e registrazioni
 - link zoom per ciascuna lezione su moodle
 - Registrazione delle lezioni su moodle

TODO asap: iscriversi al corso su Moodle

<https://didatticaonline.unitn.it/dol/course/view.php?id=39287>

Progetto

- **Analisi, Progettazione e Sviluppo (solo in parte) di una WebApp**
- Servizio Comune di Trento
- Domani presentazione



COMUNE
DI TRENTO



COMUNE
DI TRENTO

Challenge

“100 progetti per il comune di Trento”

- I migliori 3 progetti verranno selezionati e premiati dal Comune – cerimonia in comune
- Gli informatici sfidano gli Ingegneri
- IS@LT-INF vs IS@LT-ICE
- Scelti i migliori tre progetti tra i migliori tre di IS@LT-INF e IS@LT-ICE
- Premiazione finale al comune

Progetto & Challenge

Presentazione
del Comune

Design Thinking

Pitch idea
progetto

Analisi, progettazione, sviluppo e testing
Progetto

Deliverables D1, D2, D3, D4

**VIDEO + Report
CHALLENGE**

Deliverables
ESAME

4 weeks

10 weeks

Appelli
gennaio /Febbraio

Gruppi

- Gruppi: 3 studenti
 - Form google da compilare (vedi slide successiva)
 - Ciascun gruppo lavora come un team di progetto
 - Il progetto prevede la consegna di 4 deliverable a scadenze fisse durante il corso:
 - D1. Descrizione di progetto
 - D2. Documento Analisi e Progettazione
 - D3. Implementazione
 - D4. Report di progetto

Per formare i gruppi

- Ogni gruppo (3 studenti) deve compilare
 - <https://docs.google.com/forms/d/e/1FAIpQLSdjuMRwtYjv4youI0jYXiWUaqWWcdWIcwn406beF0VXpCxSlw/viewform>
 - Scadenza: **mercoledì 11 settembre**
- Se siete <3 studenti compilate comunque la scheda (come singolo o coppia), formeremo noi poi i gruppi
- Indicate il team leader (primo membro del gruppo)

Team leader e Log

- Ogni gruppo ha un **team leader** che avra' il compito di coordinare le attivita', gestire le criticita', organizzare gli incontri del gruppo (deve ovviamente lavorare anche ai deliverable)
- Il carico di lavoro deve essere distribuito in maniera uniforme tra i membri del gruppo
 - Ricordatevi di tener traccia del lavoro svolto in maniera da poter quantificare nel report finale il lavoro svolto da ciascuno

Esame

- **Consegna** dei deliverable e documento finale
 - I deliverable hanno delle scadenze ma possono essere rivisti
- **Criteri di valutazione:** correttezza nell'uso dei linguaggi-metodi-strumenti, qualità analisi-progettazione-sviluppo, lavoro di team

4 weeks

IDEA Progetto

Design Thinking ; Requisiti ; Kanban ; Pitch

D1

Implementazione

Git / GitHub

User Stories

RESTful API

OpenAPI

Web 2.0 JavaScript

WebAPI Node.js

MongoDB

Authentication JWT + GoogleAuth

Frontend

Deployment & CI-CD

Testing Jest

D3

Analisi e Progettazione

Processi di sviluppo

Agile

Linguaggi di modellazione

Use Case Diagram

Sequence + Activity Diagram

Architetture

Component Diagram

Class Diagram

Class Diagram -> API

Testing

D2

Report finale

D4

Quando e dove

- Lezioni in presenza
 - Lunedì 11.30-13.30 (B107)
 - Martedì 13.30-15.30 (B109)
 - Mercoledì 8.30-10.30 (A101)
 - Venerdì 15.30-17.30 (A101)
 - Tipicamente i tutorial e seminari verranno fatti ogni venerdì

Materiale e libri di testo

- Slides e altro materiale disponibile su moodle
- Alcuni libri consigliati
 - **Ingegneria del software: fondamenti e principi (seconda edizione)**. Carlo Ghezzi, Mehdi Jazayeri, Dino Mandrioli, Pearson Printice Hall.
 - **Introduzione all'ingegneria del software moderna**. Ian Sommerville. Pearson Printice Hall.
 - **Ingegneria del software (ottava edizione)**. Ian Sommerville. Pearson Printice Hall.
 - **Principi di Ingegneria del Software (quinta edizione)**. Roger S. Pressman, MacGraw-Hill.

FAQ

- FAQ: Documento accessibile a questo link
<https://docs.google.com/document/d/1oYcNVvlpdjtOaUPZE6D2WBeemICtOnvTud6hvqQzApE/view>
- Form domande / richieste / commenti sempre attiva a questo link
<https://docs.google.com/forms/d/e/1FAIpQLSceamQooEln6xdqXOGOtGZXcAmptfvdO7dpDyv86kcMVFQRA/viewform>

La Squadra



Paolo Giorgini



Marco Robol



Alessandro Tomasi



Nicola Marchioro



Gabriele Padovani



Tutors