



UNIVERSITÀ DEGLI STUDI
DI MILANO

Ingegneria del software

Laurea Triennale in Informatica

Lezione 1: Informazioni logistiche

Informazioni pratiche

Orario lezioni

Lun	14:30-17:00	aula 403
Mer	14:30-17:00	aula 403

Orario Laboratorio

Gio	13:30-17:30	due turni equivalenti
-----	-------------	-----------------------

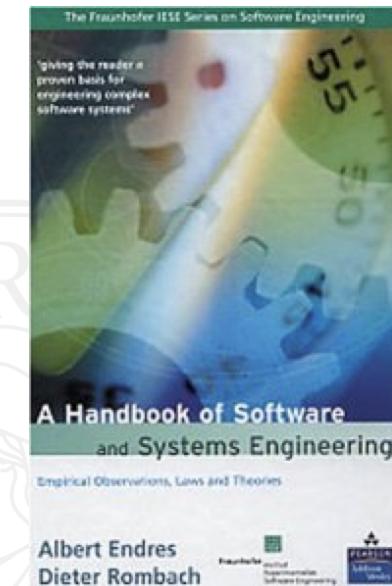
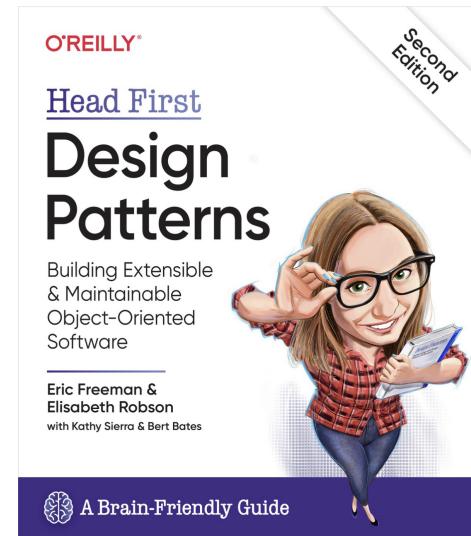
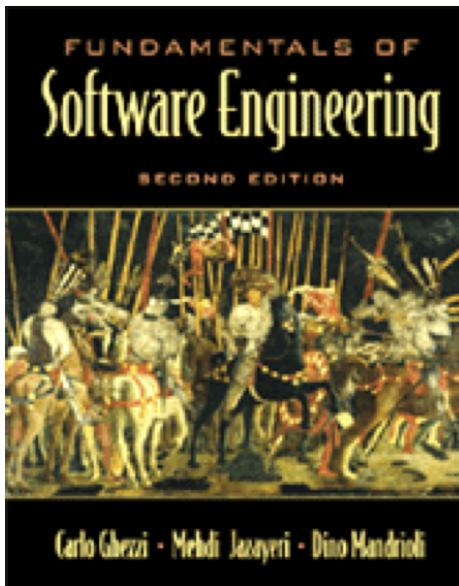
Ricevimento su appuntamento

email: carlo.bellettini@unimi.it



Libro del corso

- Non c'è un vero "libro di testo"



- ma anche altri... o risorse di rete

<https://bellettini.di.unimi.it>

Programma corso

- Processi di gestione del Software
- Progettazione del Software
- Verifica e convalida
- Specifiche del Software



Modalità d'esame

Esame di laboratorio

- prova pratica in laboratorio di 4 ore
- oppure durante l'anno ci saranno 2 laboratori valutati

Esame orale per la teoria





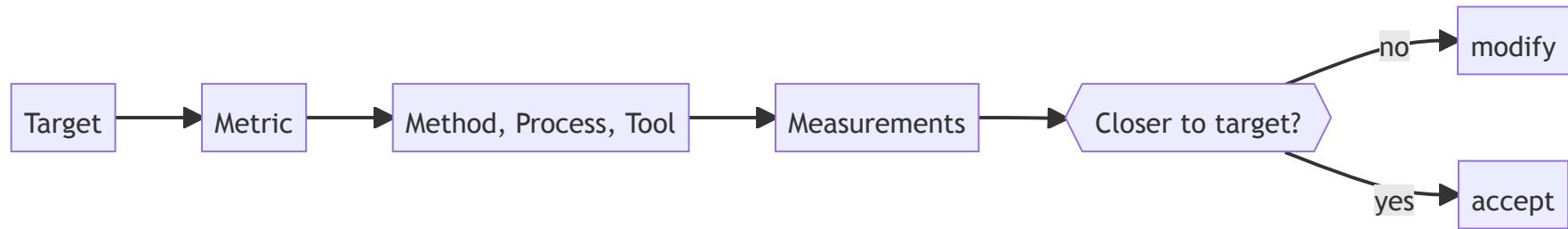
UNIVERSITÀ DEGLI STUDI
DI MILANO

Introduzione al corso

Processi di produzione del sw

- Negli anni '50 e '60 si è colta la necessità di superare metodi di produzione artigianale
- Studio quindi di tecniche e metodologie che potessero migliorare e "*assicurare*" software di qualità

Approccio ingegneristico



Ma qual è il nostro Target?

Principali problemi

- Numero e tipo **persone** coinvolte
 - il programmatore non è il cliente e questo crea problemi di comunicazione
- **Dimensioni** del software
 - milioni di linee di codice
 - migliaia di anni uomo
- **SOFT**ware
 - malleabilità porta a moltiplicarsi di versioni e evoluzioni

Perché studiare un processo?

- Convinzione che un buon **processo** produca un **prodotto** di qualità
- Quali sono le *qualità* a cui miriamo nel software?
 - **che funzioni**
 - **che sia bello**
 - **che mi faccia diventare ricco**

Cosa vuol dire *che funziona?*

- che fa quello che è stato chiesto
 - **CORRETTEZZA**

... ma se mi è stato chiesto qualcosa di sbagliato o incompleto?

R.Glass' Law (L1):
Requirements deficiencies are the *prime source of projects failures*

- che mi posso *fidare*
 - **AFFIDABILITÀ**
- che non fa male
 - **INNOCUITÀ (SAFETY) e/o ROBUSTEZZA**

Cosa vuol dire *bello*?

- facile da usare
 - **USABILITÀ**

Nielsen-Norman's Law (L26):
Usability is quantifiable

- veloce
 - **EFFICIENZA** nell'uso delle risorse
- pulito
 - **VERIFICABILITÀ**

Come fa a *farmi diventare ricco*?

- non rifare qualcosa di già fatto
 - **RIUSABILITÀ** di componenti

McIlroy's Law (L15):

Software reuse reduces cycle time and increases productivity and quality

- semplificare gli interventi post consegna
 - **MANUTENIBILITÀ**
 - correzione errori (**RIPARABILITÀ**)
 - estensione dei requisiti, nuove situazioni (**EVOLVIBILITÀ**)

M. Lehman's Laws (L27 e L28):

A system that is used will be changed

An evolving system increases its complexity unless work is done to reduce it

Come deve essere un processo?

... anche lui deve: funzionare, essere bello, farmi diventare ricco ...

- resistere agli imprevisti
 - **ROBUSTEZZA**
- essere veloce
 - **PRODUTTIVITÀ**
- cogliere l'attimo
 - **TEMPISMO**

Volatilità dei *requirements*



Copyright © 2002 United Feature Syndicate, Inc.

<https://dilbert.com>

<https://xkcd.com>

<https://phdcomics.com>

<https://geek-and-poke.com>

Processo di produzione del software

- Riconoscimento che:
 - produrre software non è *solo* scrivere codice
 - bisogna risolvere problema **comunicazione**
 - bisogna essere **rigorosi**

Bauer-Zemanek's Hypothesis (H3):
Formal methods significantly reduce design errors, or eliminate them early

- ci sono tanti **aspetti** da considerare (uno alla volta)

Modellare ciclo di vita del sw

- Identificare vari passi e attività necessarie
 - precedenze temporali
 - soggetti diversi
- Porsi due domande
 - Cosa devo fare adesso?
 - Fino a quando?

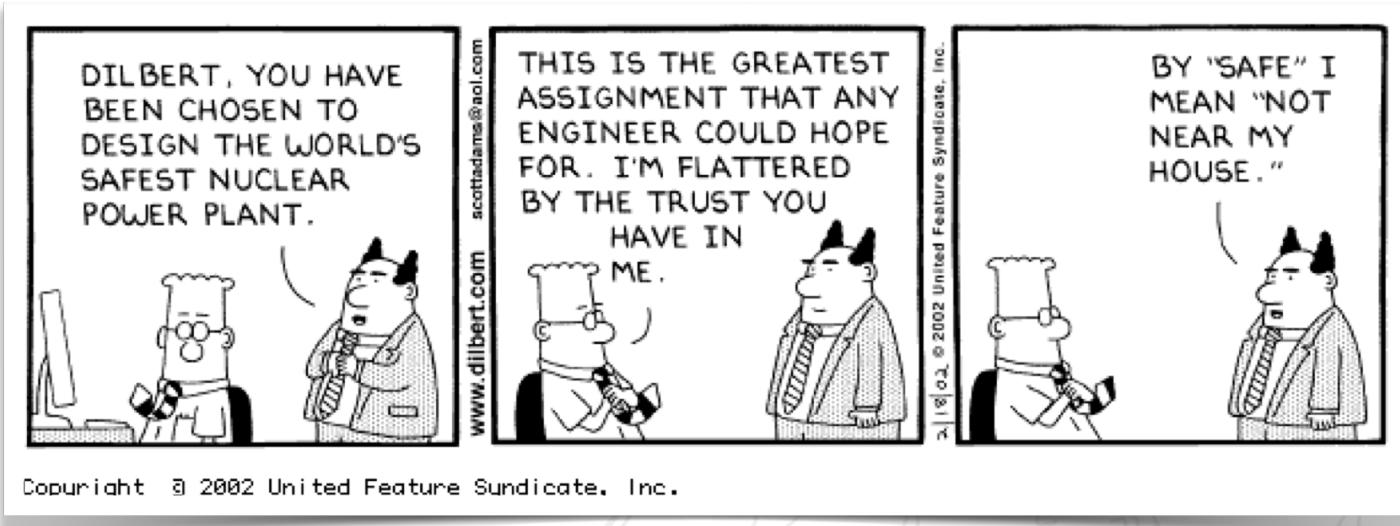
Studio di fattibilità

- Definizione preliminare del problema
 - possibile mercato e concorrenti
- Studio di diversi scenari di realizzazione
 - scelte architetturali e hardware necessario
 - sviluppo in proprio, subappalto
- Stima dei costi, tempi di sviluppo, risorse necessarie e benefici delle varie soluzioni
- Spesso difficile fare una analisi approfondita
 - spesso viene commissionata all'esterno
 - si hanno limiti di tempo stringenti
- **POSSIBILI OUTPUT:** un documento spesso in linguaggio naturale

Analisi e specifica dei requisiti

- Comprendere il dominio applicativo
- Identificare gli stakeholders
- Quali sono le funzionalità richieste?
 - **Cosa** deve fare il sistema?
 - Non interessa il **come...**
 - È il punto di vista dell'utente, non gli interessa il progetto o l'implementazione
 - Quali sono le altre qualità richieste?

Requirements e stakeholders



Requirements analysis and specifications

POSSIBILI OUTPUT:

- Documento di specifica
 - documento contrattuale approvato dal committente
 - base per il lavoro di design e verifica
 - importanza di avere documento formale
- Manuale utente o maschere di interazione
 - vista esterna per eccellenza
- Piano dei test di sistema
 - collaudi che certificano correttezza

Davis' Law (L4):

The value of models depends on the view taken, but none is best for all purposes

Progettazione (Design)

le specifiche

- Come i ~~requisiti~~ precedentemente trovati possono essere realizzati in maniera opportuna?
- Definizione dell'architettura del sistema
 - Scelta di una architettura software di riferimento
 - Scomposizione in moduli o oggetti
 - Identificazione di patterns

POSSIBILI OUTPUT:

- documento di specifica di progetto
 - diversi linguaggi



Programmazione e test di unità

- Le *scatole nere* definite al punto precedente vengono realizzate
- I singoli moduli vengono testati indipendentemente
 - framework per il test
 - moduli stub (fittizi)
 - moduli driver (guida)

Coding and module/unit testing

POSSIBILI OUTPUT

- un insieme di moduli sviluppati separatamente ...
- ... con interfaccia concordata ...
- ... singolarmente verificati

Integrazione e test di sistema

- Vengono uniti i vari componenti
- **test di integrazione:** sottoinsiemi dell'applicazione: vengono sostituiti mano a mano i moduli di testing con i moduli reali
 - top-down
 - o bottom-up?

Manutenzione

... ne abbiamo già parlato

- Correttiva
- Adattativa
- Perfettiva

OUTPUT: un prodotto migliore (per non dire corretto)

Altre attività

- Documentazione
 - Può essere vista come attività trasversale
 - Nella pratica spesso è una attività a posteriori
- Verifica e controllo qualità
 - QA: quality assurance
- Gestione del processo
 - Gestione incentivi, responsabilità, eccezioni
- Gestione configurazioni
 - Può essere vista come relazioni inter-progettuali