INGEGNERIA DEL SOFTWARE
AA. 2018/2019

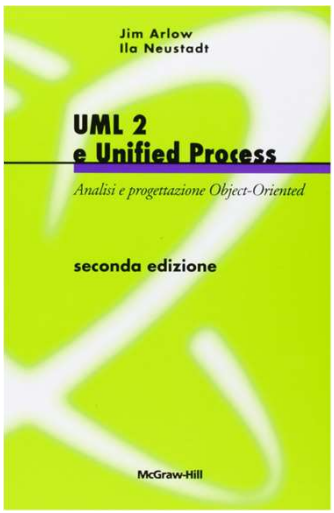
Roberta Gori, Laura Semini
Dipartimento di Informatica
Università di Pisa

Pagine Web del corso

- Pagina comune a IS A e IS B
  - Per il materiale didattico
  - Sottopagine dei singoli corsi A/B
    - Per comunicazioni, risultati prove, ecc.
- Accessibili
  - Da didawiki
  - Dalle pagine web personali: www.di.unipi.it/~gori
    www.di.unipi.it/~semini

Materiale didattico: libro

- **Libro**
  - J. Arlow, I. Neustadt, *UML 2 e Unified Process, Seconda Edizione italiana*, McGraw-Hill, 2006.



Materiale didattico: capitoli di altri libri (in inglese)

- Disponibili in copisteria (la prima che trovate sulla sinistra in via san Lorenzo) alcune pagine da:
  - Object Oriented and Classical Software Engineering , Stephen R.Schach, Fifth edition
  - Object-Oriented Software Engineering, David C. Kung

Materiale didattico: libro (pdf su didawiki)

A. Binato, A. Fuggetta, L. Sfardini
Ingegneria del Software Addison Wesley (2006)

Di riferimento per alcuni argomenti

Essendo fuori stampa, il PDF è disponibile su didawiki

Materiale didattico: lucidi e dispense

- **Su didawiki verranno resi disponibili i lucidi delle lezioni**
- **Dispense (scaricabili da didawiki)**
  - C. Montangero, L. Semini, *Dispensa di architettura e progettazione di dettaglio*.
    - Utile quando si comincerà a parlare di progettazione (circa metà corso).
  - C. Montangero, L. Semini (a cura di), *Il controllo del software - verifica e validazione*.
    - Utile nelle ultime lezioni del corso
- Esercizi:
  - Compiti degli anni passati.
  - Esercizio più **datati**: V. Ambriola, C. Montangero, L. Semini, *Esercizi di Ingegneria del Software*.
- + … altro materiale che verrà reso disponibile quando necessario.

Modalità d'esame

- Scritto: appello o prove in itinere
  - Il voto delle scritto vale solo per l'appello, le prove in itinere esonerano dallo scritto per l'appello estivo.
- Ammissione all'orale
  - Prove itinere: voto minimo 16, media minima 16
  - Appello: voto minimo 16
- Problemi disponibili in anticipo

Obiettivi di apprendimento

- Introduzione alle tecniche di modellazione in ingegneria del software.
- ***Conoscenze***. Lo studente conoscerà i principali modelli di processi di sviluppo software, e le tecniche di modellazione proprie delle varie fasi.
- ***Capacità***. Lo studente saprà utilizzare notazioni di modellazione come UML2 per l'analisi dei requisiti e la progettazione sia architettonica sia di dettaglio di un sistema software.

Software Engineering secondo IEEE 610

- A systematic, disciplined, quantifiable approach [...]
- Engineering è una disciplina che ha lo scopo di produrre fault-free software,
  - consegnato nei tempi previsti,
  - che rispetti il budget iniziale,
  - che soddisfi le necessità del committente,
  - facile da modificare.
- Disciplina sia tecnologica che gestionale

IEEE
Std 610.12-1990

IEEE Standard Glossary of Software Engineering Terminology

Sponsor
Standards Coordinating Committee of the Computer Society of the IEEE

Approved September 28, 1990
IEEE Standards Board

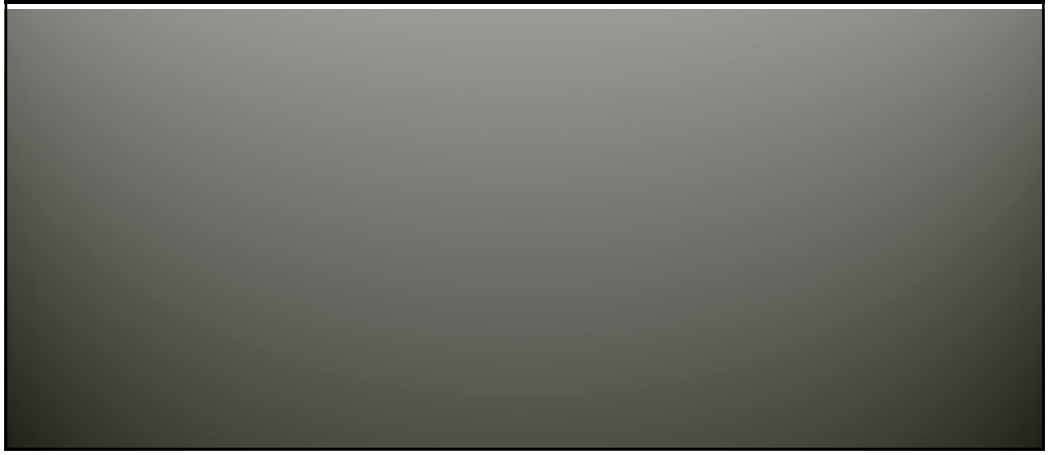
Abstract: IEEE Std 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, identifies terms currently in use in the field of Software Engineering. Standard definitions for those terms are established.
Keywords: Software engineering; glossary; terminology; definitions; dictionary.

ISBN 1-55937-067-X

The Institute of Electrical and Electronics Engineers
345 East 47th Street, New York, NY 10017, USA

Chi è l'intruso?

Alcuni esempi tristemente famosi



Denver Airport (1995 spento nel 2005)

- Sistema di smistamento dei bagagli
- 35 Km di rete, 4000 carrelli, 5000 "occhi", 56 lettori
- $ 193 000 000 di investimento
- Risultati
  - Inaugurazione dell'aereoporto ritardata 16 mesi (a 1 milione $ al giorno)
  - sforamento di 3,2 miliardi di dollari rispetto ai preventivi
- Progettazione difettosa
  - jams (mancanza di sincronizzazione)
  - No fault tolerance
  - si perdeva traccia di quali carrelli fossero pieni e quali vuoti dopo un riavvio dovuto a un jam
- Dopo anni di tentativi di "aggiustarlo"
  - Staccata la spina nel 2005

Il caso Therac- 25 (1985-1987)

- *Canada- USA:* 3 persone uccise per sovradosaggi di radiazioni
- Problema causato da editing troppo veloce dell'operatore e mancanza di controlli sui valori immessi.
- Le cause:
  - errori nel sistema SW, e di interfacciamento SW/ HW (erronea integrazione di componenti SW preesistenti nel Therac- 20).
- Poca robustezza
- Difetto latente

Il sistema antimissile Patriot (1991)

- Una caserma a Dhahran (Arabia Saudita) colpita per un difetto nel sistema di guida: 28 soldati americani morti.
- Concepito per funzionare ininterrottamente per un massimo di 14 h.
  - Fu usato per 100 h: errori nell'orologio interno del sistema accumulati al punto da rendere inservibile il sistema di tracciamento dei missili da abbattere.
- Scarsa robustezza.

London ambulance service (1992)

- Sistema per gestire il servizio ambulanze
- Ottimizzazione dei percorsi, guida vocale degli autisti
- Risultati
  - 3 versioni, costo totale: 11 000 000 Euro
  - L'ultima versione abbandonata dopo soli 3 giorni d'uso
- Analisi errata del problema:
  - interfaccia utente inadeguata
  - poco addestramento utenti
  - sovraccarico non considerato
  - nessuna procedura di backup
  - scarsa verifica del sistema

Ariane 5 (1996)

- http://it.youtube.com/watch?v=kYUrqdUyEpI
- Il sistema, progettato per l'Ariane 4, tenta di convertire la velocità laterale del missile dal formato a 64 bit al formato a 16 bit. Ma l'Ariane 5 vola molto più velocemente dell'Ariane 4, per cui il valore della velocità laterale è più elevato di quanto possa essere gestito dalla routine di conversione.
- *Overflow, spegnimento del sistema di guida, e trasferimento del controllo al 2° sistema di guida, che però essendo progettato allo stesso modo era andato in tilt nella medesima maniera pochi millisecondi prima.*
- Test con dati vecchi.
  - Fu necessario quasi un anno e mezzo per capire quale fosse stato il malfunzionamento che aveva portato alla distruzione del razzo.

Il "carattere" telugu e sistemi Apple

- 2018
- iOS , watchOS e macOS crashano quando provano a renderizzare questo simbolo
- Il simbolo è una legatura di caratteri e segni che contiene complesse istruzioni di posizionamento, visualizzarlo richiede una serie di calcoli che i sistemi operativi di Apple sbagliavano, causando il tilt.
- Apple's UIKit framework per la visualizzazione di testo

… e un esempio di successo

Un grosso sistema realizzato corretatmente nei tempi stabiliti

Un successo!!!! Linea 14 Metro Parigi (1998 con estensione nel 2003)

- La linea 14 della metropolitana di Parigi
  - Prima linea integralmente automatizzata .
- Nome di progetto, Météor: Metro Est-Ovest Rapide
  - 8 km. 7 stazioni. 19 treni. Intervallo tra 2 treni: 85 secondi.
  - Siemens Transportation Systems
  - B-method di Abrial.
  - Abstract machines
  - Generazione di codice
    - ADA, C, C++.

Quindi ???

- Anche le opere degli ingegneri qualche volta crollano ma molto più raramente del software (es. sistema operativo)
- Anche i produttori di software possono aver successo
- Anche i produttori di software devono imparare dagli errori
- Anche i produttori di software devono diventare ingegneri (del software)

Aspetti storici: nascita

- Contesto degli anni '60
  - DA: software sviluppato informalmente
    - ad es., per risolvere sistemi di equazioni
  - A: grandi sistemi commerciali
    - OS 360 per IBM 360 (milioni di righe di codice)
    - sistemi informativi aziendali, per gestire tutte le informazioni delle funzioni aziendali
  - Dalla programmazione individuale alla programmazione di squadra

- 1968, NATO Software Engineering Conference, Garmish
  - ***crisi del software – qualità del software era in generale inaccettabilmente bassa***
  - ***ingegneria del software – soluzione alla crisi del software***
  - La produzione di software deve usare tecniche e paradigmi di consolidate discipline ingegneristiche

Standish Group report 1994

- **Progetti software completati in tempo 16,2%**
- **in ritardo (il doppio del tempo): 52,7%**
  - **Difficoltà nelle fasi iniziali dei progetti**
  - **Cambi di piattaforma e tecnologia**
  - **Difetti nel prodotto finale**
- **abbandonati: 31,1%**
  - **Per obsolescenza prematura**
  - **Per incapacità di raggiungere gli obiettivi**
  - **Per esaurimento dei fondi**

Standish Group report: le cause di abbandono

1. **Requisiti incompleti**
2. **Scarso coinvolgimento degli utenti**
3. **Mancanza di risorse**
4. **Attese irrealistiche**
5. **Mancanza di supporto esecutivo**
6. **Modifiche a specifiche e requisiti**
7. **[…]**
8. **[…]**
9. **[…]**
10. **Ignoranza tecnologica**

Specificità del Software

- Il software è diverso da altri prodotti dell'ingegneria…
  - non è vincolato da materiali, né governato da leggi fisiche o da processi manufatturieri (nessun costo marginale!)
  - si "sviluppa", non si "fabbrica" nel senso tradizionale
  - non si "consuma", tuttavia si "deteriora"
  - spesso si "assembla", ma larga parte ancora si realizza *ad hoc*

Ancora differenze: fault tolerance

- Quando un elettrodomestico si rompe, si cambia un pezzo o si butta via e si compra nuovo (in ogni caso si costruisce da capo qualcosa: il pezzo o l'intero elettrodomestico)
- Quando un sistema operativo "crasha" lo facciamo ripartire. Inoltre il sistema è progettato per minimizzare l'effetto del fallimento: non si perdono i documenti su cui stava lavorando (fault tolerance)

Ancora differenze: la manutenzione

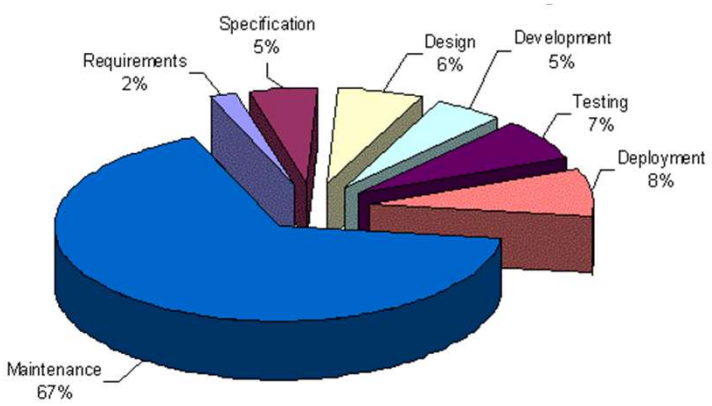
- La manutenzione di un edificio in genere si restringe a ripitturarlo, sistemare le crepe, etc…
  - Nessuno chiederebbe al costruttore di una casa di ruotarla di 90 gradi…
- Un SO, o più in generale un sistema software, può invece essere modificato per passare ad una nuova macchina con caratteristiche hardware completamente diverse…

Aspetti Economici

- Per estrarre benzina dal petrolio l'ingegnere chimico sceglie la reazione economicamente più vantaggiosa (per abbassare il costo per litro).
- Anche l'ingegnere sw è interessato a soluzioni economicamente vantaggiose.
- Esempio: Una ditta di software che utilizzi una tecnica CTold scopre una nuova tecnica CTnew che permetterebbe di velocizzare la scrittura del codice di un fattore 10 rispetto a CTold.
- Ma può comunque non adottare CTnew a causa del:
- costo dell'introduzione della tecnologia
- costo del training del personale
- costo della manutenzione

I costi del software

Il prodotto software durante la sua vita diverse fasi:
analisi, specifica, progettazione, codifica, testing, manutenzione e ritiro

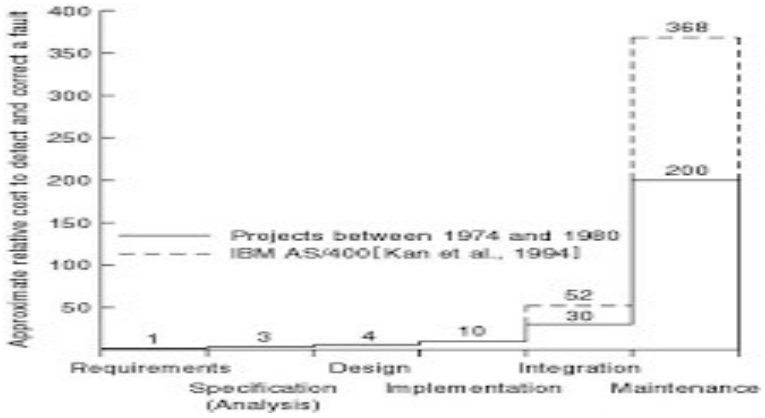


La manutenzione

- La manutenzione include tutti i cambiamenti al prodotto software, anche dopo che è stato consegnato al cliente
- Si divide in :
  - manutenzione correttiva(20%), rimuove gli errori lasciando invariata la specifica
  - manutenzione migliorativa, consiste in cambiamenti alla specifica e nell'implementazione degli stessi, può essere:
    - Perfettiva (60%): modifiche per migliorare le qualità del software, introduzione di nuove funzionalita', miglioramento delle funzionalità esistenti.
    - Adattativa (20%): modifiche a seguito di cambiamenti nell'ambiente legislativo, cambiamenti nell'Hardware, nel Sistema operativo, ecc.
      - Esempio: IVA dal 22% al 20% float aliquota=22; …; prezzotot =prezzo+(prezzo*aliquota)/100

Importanza dell'analisi dei requisiti

- Se si introduce un errore durante l'analisi dei requisiti, l'errore apparirà anche nella specifica, nella progettazione e nel codice.
- Prima individuiamo l'errore e meglio è:



Lavoro in team

- La maggior parte del software è prodotto da team di programmatori
- Il lavoro in team pone dei problemi:
  - Di interfaccia tra le diverse componenti del codice
  - Di comunicazione tra i membri del team
- Molto tempo deve essere dedicato alle riunioni tra i vari componenti.
- L'ingegnere del software deve essere anche capace di:
  - gestire i rapporti umani e organizzare un team
  - gestire gli aspetti economici e legali

Temi di ingegneria del software

- Processo software
- Realizzazione di sistemi software
- Qualità del software

Processo software

- Organizzazione e gestione dei progetti
- Metodi di composizione dei gruppi di lavoro
- Strumenti di pianificazione, analisi, controllo
- Cicli di vita del software
- Definizione e correlazione delle attività
- Modelli ideali di processo di sviluppo

Realizzazione di sistemi sw

- Strategie di analisi e progettazione
  - Tecniche per la comprensione e la soluzione di un problema
  - Top-down, bottom-up, progettazione modulare, OO
- Linguaggi di specifica e progettazione
  - Strumenti per la definizione di sistemi software
  - Reti di Petri, Z, OMT, UML
- Ambienti di sviluppo
  - Strumenti per analisi, progettazione e realizzazione
  - Strumenti tradizionali, CASE, CAST, RAD

Qualità del software

- Modelli di qualità
  - Definizione di caratteristiche della qualità
- Metriche software
  - Unità di misura, scale di riferimento, strumenti
  - Indicatori di qualità
- Metodi di verifica e controllo
  - Metodi di verifica, criteri di progettazione delle prove
  - Controllo della qualità, valutazione del processo di sviluppo

Gli stakeholders di un prodotto software

- Fornitore
  - chi lo sviluppa
- Committente
  - chi lo richiede (e paga)
- Utente
  - chi lo usa