INGEGNERIA DEL SOFTWARE
2020/21

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 risultati prove, etc

- Accessibili
  - Da didawiki
  - Dalle pagine web personali: www.di.unipi.it/~gori
    www.di.unipi.it/~semini

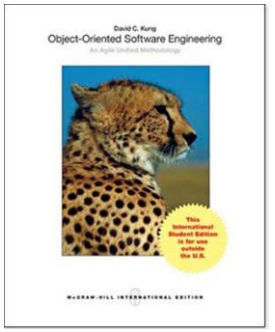
Materiale didattico: capitoli di altri libri (in inglese)

Per la prima parte del corso: 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

Cap 1,3 e 10

- **Object-Oriented Software Engineering**,



David C. Kung

Cap 2

Materiale didattico: libro

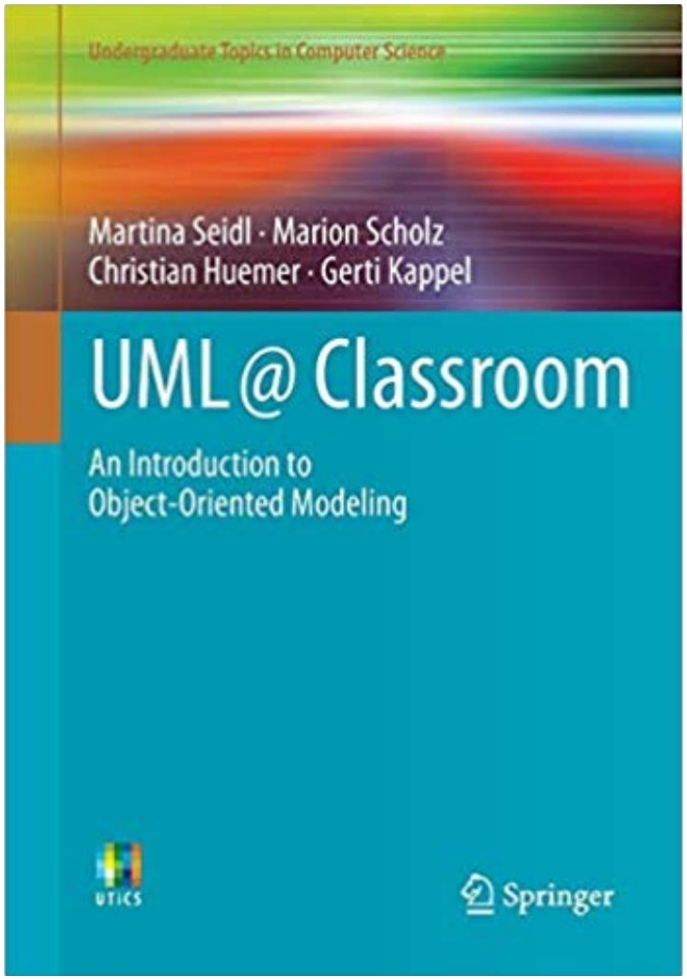
UML @ Classroom: An Introduction to Object-Oriented Modeling

M. Seidl, M. Scholz C. Huemer, G. Kappel

Springer Verlag, 2015.

Di riferimento per la parte di UML

Per una convenzione con l'universita'
il PDF è disponibile su didawiki



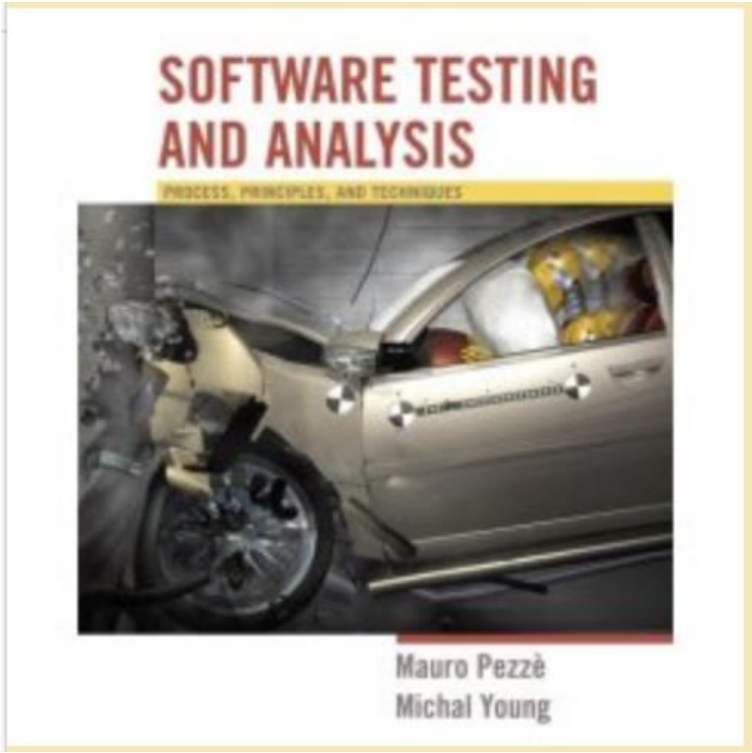
Materiale didattico: libro (pdf su didawiki)

Software Testing and Analysis: Process, Principles, and Techniques

M. Pezze' M.Young

Di riferimento per la parte di testing

Per concessione dell'autore,
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

Fino a che perdureranno le attuali restrizioni causa COVID19 che impediscono di fare esami in presenza, la prova intermedia sarà di tipo autovalutativo.

Modalità di esame:

Fino a che perdureranno le attuali restrizioni causa COVID19 che impediscono di fare esami in presenza, le modalità di esame saranno quelle dell'a.a. 2019-2020:

progetto + orale

Appena sarà possibile invece si tornerà alla modalità tradizionale:

scritto e orale.

Obiettivi di apprendimento

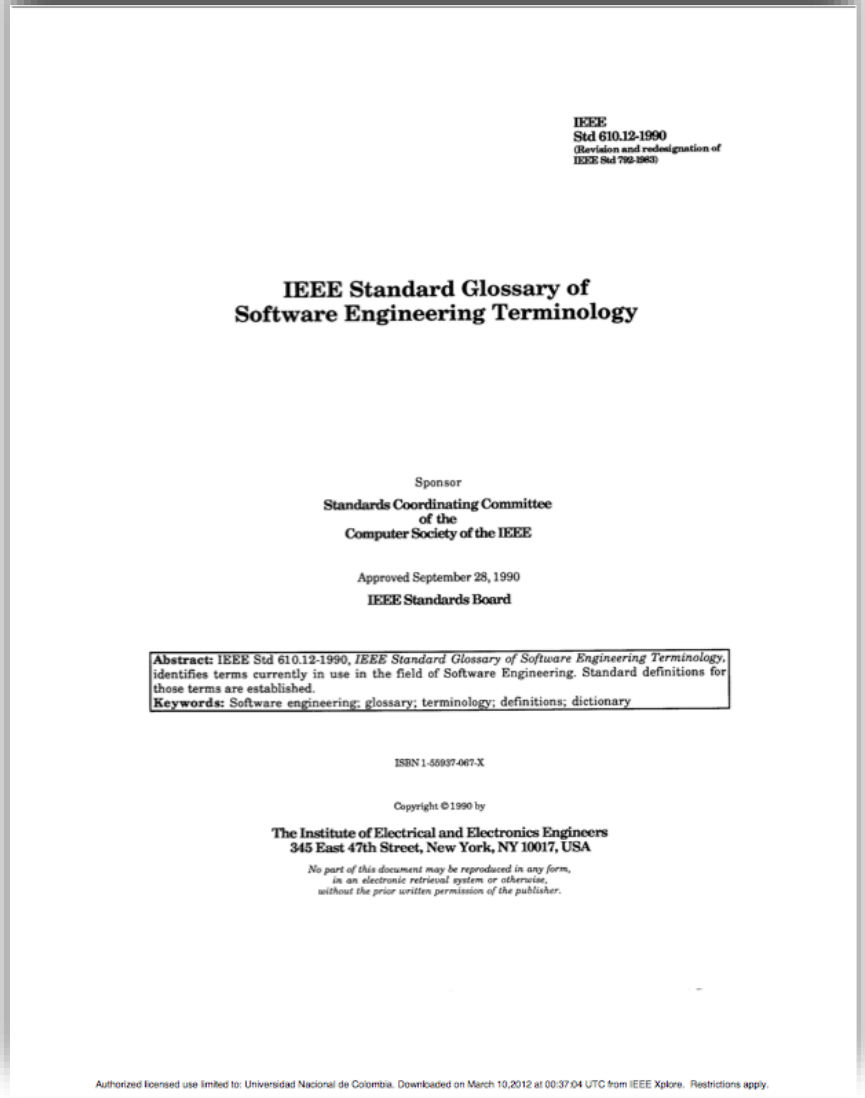
- Introduzione alle tecniche di modellazione dell' ingegneria del software.
- ***Conoscenze***. Lo studente conoscerà i principali modelli di processi di sviluppo software, e le tecniche di modellazione proprie delle varie fasi.
- ***Capacità***. Lo studente saprà utilizzare notazioni di modellazione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

Processo Software

- Il modo in cui produciamo il software
- Inizia quando iniziamo a esplorare il problema e finisce quando il prodotto viene ritirato dal mercato
- Fasi:
  - analisi dei requisiti
  - specifica
  - progettazione
  - implementazione
  - integrazione
  - mantenimento
  - ritiro
- Riguarda anche tutti i tools e le tecniche per lo sviluppo e il mantenimento e tutti i professionisti coinvolti

Software Engineering secondo IEEE 610

- L'approccio sistematico allo sviluppo, all'operatività, alla manutenzione e al ritiro del software [glossario IEEE]
  - 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
(Revision and redesignation of
IEEE Std 792-1983)

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

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 telex

- 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

Il caso Toyota

Toyota "Unintended Acceleration" Has Killed 89

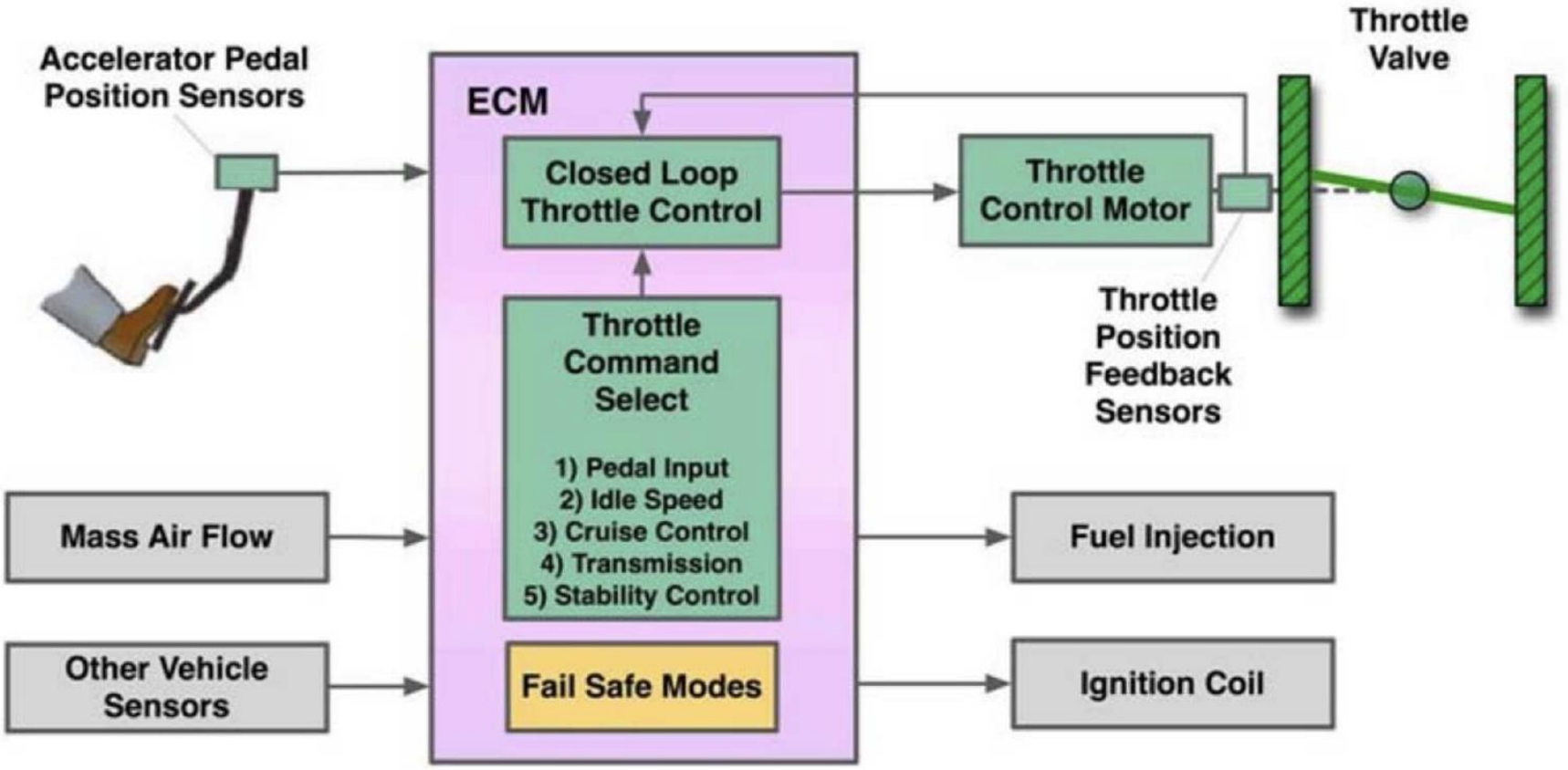
A 2005 Toyota Prius, which was in an accident, is seen at a police station in Harrison, New York, Wednesday, March 10, 2010. The driver of the Toyota Prius told police that the car accelerated on its own, then lurched down a driveway, across a road and into a stone wall. (AP Photo/Seth Wenig) / AP PHOTO/SETH WENIG

Definition of UA by NHTSA (February 2011)

"Unintended Acceleration" (UA) refers to the occurrence of any degree of acceleration that the vehicle driver did not purposely cause to occur

- Not a problem of Toyota alone of course
- There have been announcements about UA issues caused by software bugs with, e.g., some GM, Honda (and, in 2017, even Tesla models)
- UA is only one of the manifestations of software faults that endanger safety
- The level of awareness for such issues has increased dramatically after the Toyota case

The ETCS-i System



- It mainly controls the throttle valve
  - Fuel injection and ignition are adjusted, taking into account several parameters, so as to ensure proper combustion
- It was first investigated by a NASA team in 2010–2011

The ETCS-i System Is Safety Critical!!!

Pumping brakes with throttle valve fully open causes loss of brake power assist (force required: 7–20 kg $\Longrightarrow$ 80 kg)

- As a result, braking, the most instinctive driver's response, may not allow the driver to stop the car
- In several cases, drivers burned the brakes without being able to stop the car

The Oklahoma Trial Investigation (2013)

Embedded software system's expert witnesses:

Michael Barr studied the ETCS-i software in depth and found the likely cause for the UA

Philip Koopman studied the highly-confidential Toyota design documents for the ETCS-i and the reports produced by NESC, by Michael Barr and others who had had access to the code

They found many extremely serious problems, both with the system's design and with the software

In their reports there is a lot of crystal-clear evidence of inadequate engineering practice

Okla. jury: Toyota liable in acceleration crash

(By SEAN MURPHY / Associated Press / October 24, 2013)

OKLAHOMA CITY (AP) Toyota Motor Corp. is liable for a 2007 crash that left one woman dead and another seriously injured after a Camry suddenly accelerated, an Oklahoma jury decided Thursday.

The jury awarded $1.5 million in monetary damages to Jean Bookout, the driver of the car who was injured in the crash, and $1.5 million to the family of Barbara Schwarz, 70, who died.

It also decided Toyota acted with "reckless disregard" for the rights of others, a determination that sets up a second phase of the trial on punitive damages that is scheduled to begin Friday.

… e un esempio di successo

Un grosso sistema realizzato correttamente 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 i produttori di software possono aver successo ma devono imparare dagli errori

- Anche le opere degli ingegneri qualche volta crollano ma molto più raramente del software (es. sistema operativo)

- Anche i produttori di software devono diventare ingegneri (del software)

Aspetti storici: nascita

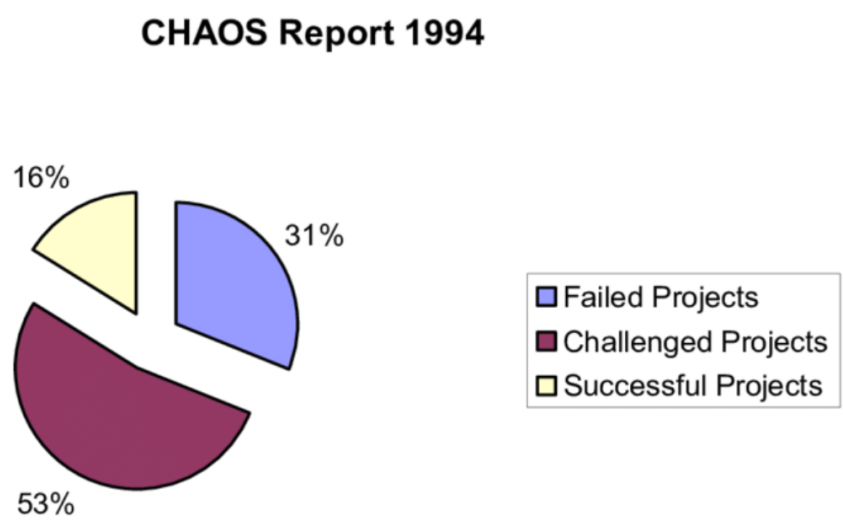
- Contesto degli anni '60
  - DA: software sviluppato informalmente
    - ad es., per risolvere sistemi di equazioni
  - A: grandi sistemi commerciali
    - OS 360 per IBM 360 (milioni di righe di codice)
    - sistemi informativi aziendali, per gestire tutte le informazioni delle funzioni aziendali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

  - L'idea: la produzione di software deve usare tecniche e paradigmi di consolidate discipline ingegneristiche

Analisi dello Standish Group 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. Scarso coinvolgimento degli utenti
2. Requisiti e specifiche incompleti
3. Modifiche a specifiche e requisiti
4. Mancanza di supporto esecutivo
5. Ignoranza tecnologica
6. Mancanza di risorse
7. Attese irrealistiche
8. Obiettivi non chiari
9. Tempi di sviluppo non realistici
10. Nuove tecnologie

Project Challenged Factors	% of Responses
1. Lack of User Input	12.8%
2. Incomplete Requirements & Specifications	12.3%
3. Changing Requirements & Specifications	11.8%
4. Lack of Executive Support	7.5%
5. Technology Incompetence	7.0%
6. Lack of Resources	6.4%
7. Unrealistic Expectations	5.9%
8. Unclear Objectives	5.3%
9. Unrealistic Time Frames	4.3%
10. New Technology	3.7%
Other	23.0%

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 manifatturieri (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 edificio crolla parzialmente, non si aggiusta come fosse un'araba fenice.
- Quando un sistema operativo "crasha" lo facciamo ripartire.
- Questo perché è stato progettato per minimizzare l'effetto del fallimento: non si perdono i documenti su cui stava lavorando
- La fault tolerance è una qualità del sw

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.

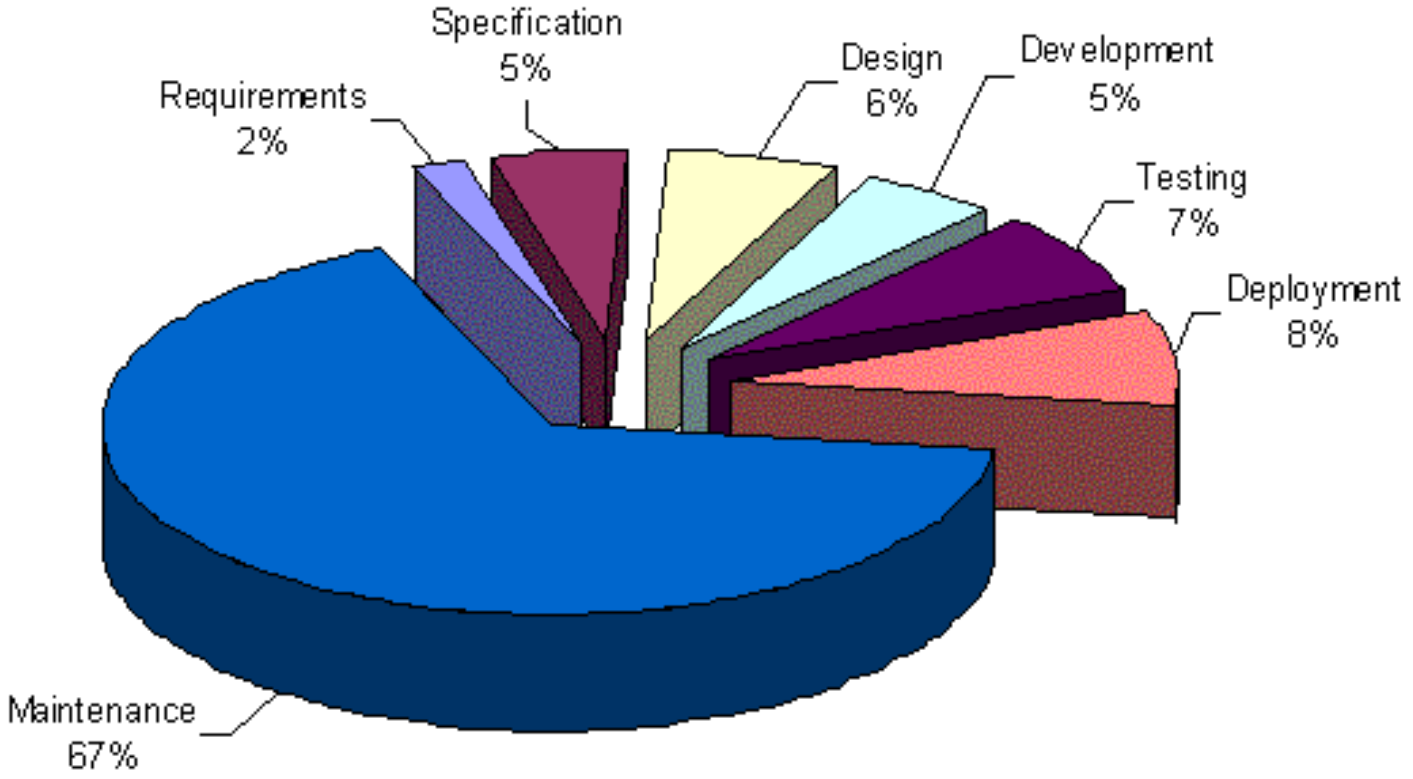
- Esempio: Una ditta di software che utilizzi una tecnica CTold scopre una nuova tecnica CTnew che permetterebbe di velocizzare la scrittura del codice di un fattore 10 rispetto a CTold.

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

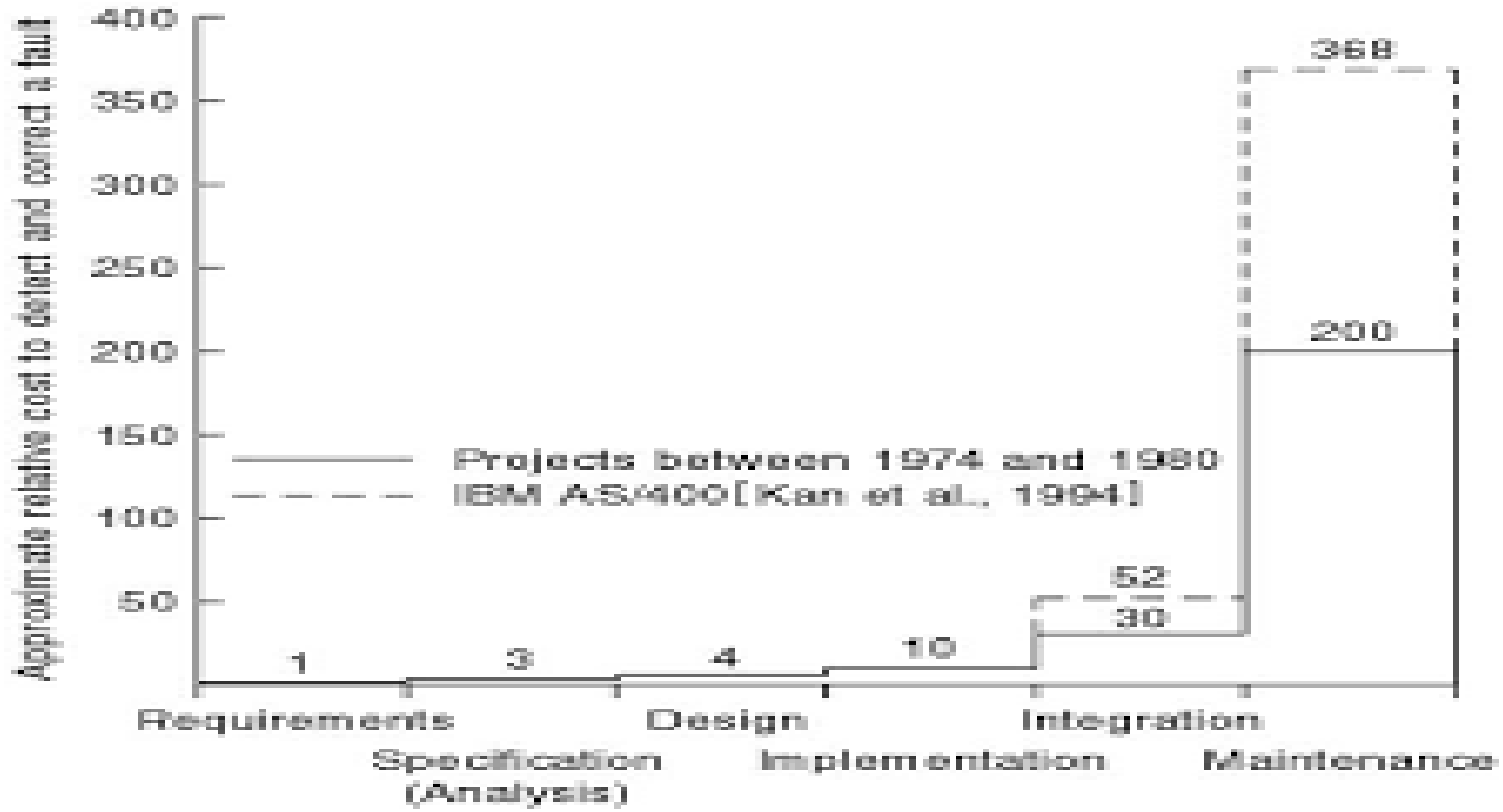


La manutenzione

- 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 oggi 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

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