Ingegneria del Software

Introduzione

Antonino Staiano
e-mail: **antonino.staiano@uniparthenope.it**

Materiale Didattico

- Libro di Testo
  - I. Sommerville, "Ingegneria del Software", 8ª Ed., Pearson (Addison-Wesley)
- Altri riferimenti consultabili
  - B. Bruegge, A.H. Dutoit, "Object-Oriented Software Engineering – Using UML, Patterns, and Java", 2ª Ed., Pearson (Prentice Hall)
  - A. Binato, A. Fuggetta, L. Sfardini, "Ingegneria del Software- Creatività e Metodo", 2ª Ed., Pearson (Addison-Wesley)
  - R. Pressman, "Principi di Ingegneria del Software", McGraw-Hill
- Altro materiale
  - Slide del corso e materiale didattico
    - **Piattaforma e-learning**

Orario Lezioni

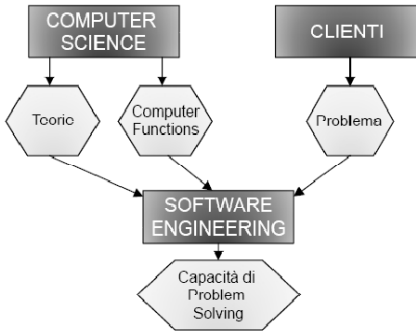
- Martedì 12:30 – 14:30
- Giovedì 12:30 – 14:30

Aula 11, II Piano

Perché un corso di Ingegneria del Software?

- Produrre software non è (solo) un'arte e neppure (solo) una scienza: è un'industria
  - Lavoro inserito in un contesto di gruppo e azienda
  - Vincoli economici e requisiti di qualità
- Come in ogni industria, per produrre software sono state sviluppate metodologie di progetto, di sviluppo e di verifica
- Un informatico deve necessariamente conoscerle
  - Non basta essere i migliori programmatori
  - Bisogna essere in grado di **analizzare**, **progettare** e **gestire** un progetto software nella sua interezza

Ruolo dell'Ingegneria del Software



Obiettivi del corso (I)

- Conoscenze che si intendono trasmettere:
  - Concetti di base dell'ingegneria del software, dei processi di ingegneria del software e delle relative fasi, attività e distribuzione
  - Definizione, proprietà e analisi di modelli
  - Metodi di analisi e progettazione e importanza dei linguaggi di modellazione del software per la comunicazione tra diversi attori coinvolti in un processo di ingegneria del software
  - Concetti e tecniche di analisi, testing e debugging del software
  - Concetti di manutenzione del software, e principali problematiche della gestione dei progetti software

Obiettivi del corso (II)

- Capacità che si intendono sviluppare:
  - Saper costruire modelli di sistemi con un procedimento passo passo.
  - Saper produrre documenti software durante le varie fasi del processo di sviluppo e modificarli per produrre versioni successive nell'ambito di processi software iterativi ed incrementali.
  - Sapere usare la notazione UML per modellare il software.
  - Saper usare un approccio ingegneristico all'analisi, design, testing e manutenzione del software.

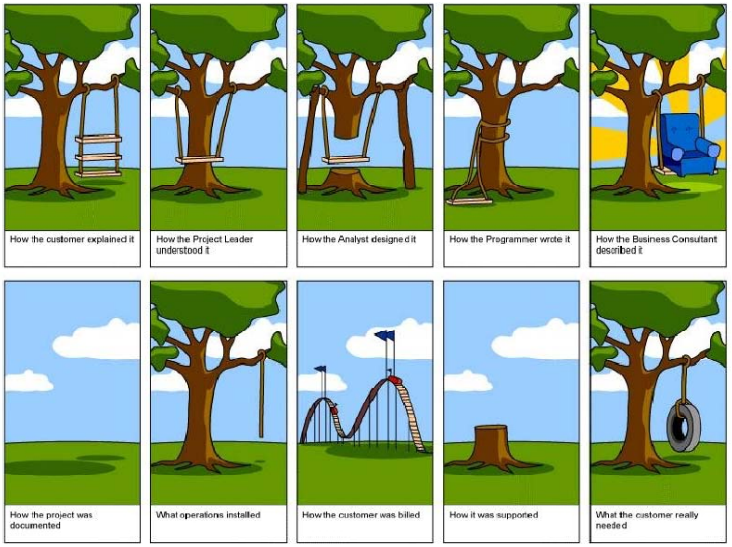
Programma di massima

- Concetti di base, definizioni e problematiche dell'Ingegneria del Software.
  - Modelli di ciclo di vita del software.
  - Analisi e specifica dei requisiti.
  - Progettazione e architetture software.
- Modellazione orientata agli oggetti; Unified Modeling Language (UML), introduzione al Rational Unified Process.
- Software testing
  - processo e documenti di testing;
  - principali tecniche di testing black box e white box;
- Cenni su software project management, qualità del software e manutenzione del software.

Modalità d'esame

- Progetto obbligatorio
  - Gruppi costituiti da 2-3 persone
  - Analisi, progettazione e documento di testing per un sistema software
- Prova orale

L'Ingegneria del Software: Introduzione



La crisi del software

- Il concetto di crisi del software è emerso alla fine del 1960. Dijkstra affermava:

  "*La causa principale della crisi del software sta nell'accresciuta potenza dei computer di diversi ordini di grandezza! In soldoni: fin quando non c'erano computer, la programmazione non rappresentava alcun problema; quando sono arrivati i primi (non molto potenti) computer, programmare è divenuto un problema moderato, e ora che abbiamo computer enormi (in termini di potenza), programmare è divenuto un altrettanto enorme problema*"
- Le cause della crisi del software erano collegate alla complessità dei processi software ed alla relativa immaturità dell'ingegneria del software. La crisi si manifestava in diversi modi:
  - Progetti oltre il budget
  - Progetti oltre i limiti di tempo
  - Software di scarsa qualità
  - Software che spesso non rispettava i requisiti
  - Progetti ingestibili e codice difficile da manutenere

Software non ingegnerizzato

- Alcuni fatti (2000-2002):
  - Mercato mondiale dell'ICT: 2.153.000.000.000 di Euro
  - Costo dei difetti nel software (considerando solo l'economia USA): 60.000.000.000 di $
  - Percentuale dei progetti software completati rispettando preventivi di spesa e tempo: 16%
  - La metà dei progetti software commerciali sforano il preventivo di spesa del: 90%
  - Percentuale dei costi di manutenzione su quelli totali di un software: 70%

Qualche (mis)fatto concreto

- **Therac-25** (1985-1987)
  - Diversi pazienti furono irradiati con dosi massicce di raggi X da un'apparecchiatura per terapie contro il cancro. Sistema controllato da sistema di input software contenente un baco. Almeno 5 persone furono uccise.
- **London Ambulance Service** (1992)
  - Il LAS introdusse un sistema automatico di selezione e invio delle unità mediche sostituendo le procedure manuali fino a quel punto in uso, introducendo meccanismi di selezione più avanzati. L'appalto fu bandito a giugno 1991 ed assegnato ad un costo di £1,1 milioni. Furono fissate precise scadenze non negoziabili. Il sistema mal progettato e realizzato causò, quando messo in opera, altissimi ritardi nell'arrivo delle unità (anche 11 ore) e circa 30 persone morirono a causa dei ritardi. Il software fu poi dismesso.

Qualche (mis)fatto concreto

- **Denver Airport Baggage System** (1995)
  - L'aeroporto di Denver doveva essere inaugurato nel 1995 e doveva basarsi su di un sistema di smistamento dei bagagli avveneristico. Il sistema software avrebbe dovuto controllare 26 miglia di nastri trasportatori indirizzando correttamente i bagagli. Il sistema però non riuscì mai a fornire le garanzie necessarie e ritardò l'apertura dello scalo per mesi al costo di un milione di dollari al giorno. Alla fine il progetto fu abbandonato.
- **Il vettore spaziale Ariane 5** (1996)
  - Il vettore Ariane 5 dell'agenzia spaziale europea esplose durante il volo inaugurale il 4 giugno 1996 dopo 39 secondi dal decollo. L'esplosione fu causata da un segnale di autodistruzione emesso dal sistema di controllo che erroneamente, a causa di buffer overflow, pensava di essere fuori rotta. La progettazione e costruzione del vettore era costata 500 milioni di dollari.

Qualche (mis)fatto concreto

- **Mars Climate Orbiter** (1999)
  - La sonda Mars Climate Orbiter fu "smarrita" dopo il decollo con una perdita economica di circa 125 milioni di dollari. Successive investigazioni sulle responsabilità appurarono che in fase di progettazione non fu specificato il sistema di misura da utilizzare. Differenti team di sviluppo, nello sviluppo dei vari moduli fecero differenti assunzioni sul sistema di misurazione da applicare (metrico (Kg) ed imperiale (pounds)).
- **Welfare Management System** (2004)
  - Un sistema di gestione della previdenza pubblica canadese costato diverse centinaia di milioni di dollari fu incapace di adattarsi a modifiche nei tassi di calcolo. Il sistema fu sottoposto a sole sei settimane di test di accettazione.

Qualche (mis)fatto concreto

- **Toyota Prius** (2005)
  - Errori nella programmazione delle schede di controllo dell'auto causarono il richiamo di oltre 160.000 unità. L'auto passava all'alimentazione elettrica e proseguiva per pochi chilometri prima di arrestarsi quando lanciata ad alta velocità.

Nascita dell'Ingegneria del Software

- Anni '50 appaiono i primi linguaggi di alto livello (COBOL)
  - I programmi cominciano a diventare via via più complessi
  - Programmare comincia a diventare un lavoro
- Anni '60-'70...la crisi del software
  - Le tecniche di sviluppo adottate fino a quel punto risultano non scalare
  - Gli sviluppatori sembrano incapaci di sviluppare software capace di utilizzare a fondo i miglioramenti nello hardware.
- Conferenze NATO (1968 Garmisch-D, 1969 Roma-I)
  - Vengono poste le basi per una nuova disciplina
  - sono necessarie nuove metodologie, strumenti formali, nuovi processi, nuovi strumenti di sviluppo, nuovi modi di organizzare il lavoro, etc..
  - la produzione del software si deve trasformare da un'attività artigianale ad un'attività "ingegneristica"

E oggi?

- Molti passi sono stati fatti ma da allora....
  - La **complessità** dei sistemi **continua a crescere** vertiginosamente
  - il software viene continuamente applicato in **nuovi domini**
  - moltissimi sistemi software falliscono o vengono cancellati dopo esser partiti e trovandosi spesso anche in una fase avanzata dello sviluppo
- spessissimo il rilascio del software avviene in **ritardo rispetto alle scadenze stabilite**
- altrettanto spesso la **qualità** del software rilasciato è "discutibile"

Ingegneria del software: definizioni

- IEEE:
  - Applicazione di un approccio sistematico, disciplinato e quantificabile allo sviluppo, supporto e manutenzione del software
- Sommerville:
  - L'Ingegneria del Software è una disciplina ingegneristica che riguarda tutti gli aspetti della produzione del software. L'ingegnere del software deve adottare un approccio sistematico ed organizzato al suo lavoro ed utilizzare gli strumenti e le tecniche più appropriate a seconda del problema da risolvere, dei vincoli di sviluppo e delle risorse disponibili.
- Ghezzi, Jazayeri, Mandrioli:
  - L'Ingegneria del Software è la branca dell'informatica che riguarda lo sviluppo di sistemi software le cui dimensioni richiedono l'intervento di uno o più team di sviluppo … programmare è principalmente un'attività personale, mentre l'ingegneria del software è essenzialmente un'**attività di team**.

Ingegneria del software: definizioni

- Emmerich:
  - L'Ingegneria del Software è una branca dell'ingegneria dei sistemi che riguarda lo sviluppo di sistemi software complessi e di grandi dimensioni. Essa si focalizza su: obiettivi e limiti reali per i servizi forniti dai sistemi software; la precisa specifica della struttura di questi sistemi, del loro comportamento e l'implementazione di tali specifiche; le attività richieste al fine di garantire che le specifiche e gli obiettivi siano raggiunti; l'evoluzione di tali sistemi nel tempo. Infine essa riguarda anche i processi, i metodi e gli strumenti per lo sviluppo **economicamente vantaggioso e pianificato** del software.

Cosa si evince?

- Introduzione di metodologie per lo sviluppo di sistemi di **medio/grandi dimensioni**
- Si raccomanda **disciplina e sistematicità**
- Si raccomanda l'introduzione di **metodi quantificabili al fine di poter paragonare differenti soluzioni possibili**
- Comporta lo sviluppo di sistemi che richiedono l'intervento **di team di sviluppo**. Dunque la **comunicazione** diventa uno degli aspetti più importanti.

Cosa si evince?

- Non riguarda soltanto la programmazione (per certi versi aspetto marginale)
- **Costi di sviluppo e tempi** sono un aspetto fondamentale dello sviluppo
- Si richiedono **capacità di gestione e di pianificazione**
- Obiettivo focale è la spinta verso la produzione di **qualità**
- Necessità di applicare **strumenti di supporto**

Il Software nell'economia moderna

- Le economie di tutte le nazioni più evolute dipendono dal software e la maggior parte dei sistemi sono controllati da software
- L'Ingegneria del Software ha a che fare con teorie, metodi e strumenti per progettare,costruire e mantenere software di grandi dimensioni
- Il software costa più dell'hardware e il mantenimento costa più dello sviluppo
- Obiettivo: sviluppo cost-effective del software

Il Software nell'economia moderna

Informatica	val	Telecom	val	Media	val
Microsoft	356	Vodafone	157	Viacom	72
Intel	224	Verizon	132	Liberty Media	37
IBM	197	SBC	123	ComCast	34
Cisco	138	NTT DoCoMo	103	Clear Channel	28
AOL	126	Vodafone ag	98	Cox Comm.	21
Oracle	91	Bellsouth	71	British Sky	20
Dell	73	Deutsche Telekom	66	The Thomson	19
Vivendi	51	AT&T	65	Gannet	18
Texas Instr.	48	Telefonica	58	TheNews Group	15
HP	44	Telecom Italia	54	McGrawHill	12
SAP	43	China Mobile	54	Tribune	11
Taiwan semic.	43	NTT	48	Grupo televisa	10
Fujitsu	40	TIM	44	Reed	10
Sun	39	France Telecom	40	USA Network	9

Valorizzazione in borsa in G$ delle principali aziende ICT

Cosa è un prodotto software?

- Qualcosa di più di un insieme di linee di codice…
- Un insieme di:
  - Programmi per computer
  - Tutta la documentazione che descrive la struttura del sistema
  - I dati di configurazione, che permettono di installarlo
  - Il manuale utente

Dimensione dei SW attuali

- Di quante linee di codice sono costituiti i software attuali?
  - The Gimp **650.000**
  - Kernel Linux nel 2002 **4.141.432**
  - Open Office **10.000.000**
  - Suite Mozilla (Firefox + Thunderbird) **30.000.000**
  - Windows Vista **50.000.000**
  - OS X 10.4 **86.000.000**

Evoluzione della Produzione di Software

- **Arte:** applicazioni sviluppate da singole persone e utilizzate dagli stessi sviluppatori
- **Artigianato:** applicazioni sviluppate da piccoli gruppi specializzati per un cliente
- **Industria:** diffusione del software in diversi settori; crescita di dimensioni, complessità e criticità delle applicazioni; mercato e concorrenza; necessità di migliorare la produttività e la qualità; gestione dei progetti; evoluzione del software

Programmi vs Prodotti

- **Programma**: l'autore è anche l'utente (e.g., non è documentato, quasi mai è testato, non c'è progetto)
  - non serve approccio formale
- **Prodotto software**: usato da persone diverse da chi lo ha sviluppato
  - è software industriale il cui costo è circa 10 volte il costo del corrispondente programma
  - è necessario un approccio formale allo sviluppo

Tipologie di Prodotti Software

- **Prodotti generici** (general purpose)
  - sistemi stand-alone prodotti da una organizzazione e venduti a un mercato di massa
- **Prodotti specifici** (specific purpose)
  - sistemi commissionati da uno specifico utente e sviluppati specificatamente per questo da un qualche contraente

- La fetta maggiore della spesa è nei prodotti generici ma il maggior sforzo di sviluppo è nei prodotti specifici
- La differenza principale?
  - Chi fornisce la specifica del prodotto (il produttore o il consumatore).

Software vs. Altri Artefatti

- Differenze con prodotti industriali classici
  - Intangibile
  - Malleabile
  - Ad alta intensità di lavoro umano
  - Spesso costruito ad hoc invece che assemblato
  - Manutenzione = cambiamento

Problemi della produzione software: costi

- Il software ha costi elevati
  - Sono i costi delle risorse usate: ore lavoro (manpower), hardware, software e risorse di supporto. Il manpower è dominante !
    - Il costo è espresso in mesi/uomo
  - La manutenzione costa più dello sviluppo
    - Per sistemi che rimangono a lungo in esercizio i costi di manutenzione possono essere svariate volte il costo di produzione
- Produttività media
  - da 300 a 1000 linee di codice rilasciate per mese/uomo

Problemi della produzione software

- Ritardi nelle consegne e aumenti dei costi stimati
  - US Air Force - Sistema di comando e controllo (1981):
    - stima iniziale della azienda vincitrice per la fornitura: 400.000$,
    - costo successivamente rinegoziato:
      - a 700.000$,
      - poi a 2.500.000 $
      - costo finale 3.200.000$.
    - … costo finale maggiore di circa 10 volte la stima iniziale !!
    - … e con notevole ritardo rispetto alla stima iniziale

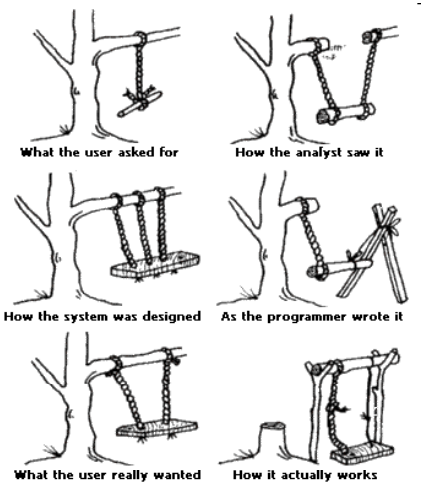
Problemi della produzione software

- Un' azienda nel settore della grande distribuzione aveva richiesto un sistema che, stima iniziale, sarebbe stato sviluppato in 9 mesi al prezzo di 250.000 $ (1989)
  - Due anni dopo, e dopo una spesa di 2.500.000 $, il lavoro non era stato ancora completato e fu stimato che erano necessari altri 3.600.000 $ (!!)
  - Il progetto fu abbandonato !
- Da un report USA del 1989
  - su 600 aziende contattate, più del 35% avevano progetti 'runaway' (in ritardo o fuori dal budget e la tempistica stimata)

Uno dei più noti record negativi: Il Software dello Space Shuttle

- Costo finale: $10.000.000.000, svariati milioni di dollari in più di quanto pianificato
- Time: ritardo di 3 (TRE) anni
- Qualità: Il primo lancio del Columbia fu cancellato a causa di problemi di sincronizzazione tra i 5 computer di bordo.
  - Dopo lunghe (e costose) investigazioni, l'errore fu individuato in una modifica fatta 2 anni prima, quando un programmatore aveva cambiato un fattore di ritardo su un interrupt handler da 50 a 80 millisecondi.
  - La possibilità che si verificasse un errore a riguardo era talmente minima che non apparve mai nelle migliaia di ore di testing.
- Il software contiene ancora errori sostanziali.
  - Gli astronauti hanno a disposizione un libro dei bug noti...

Ingegneria del Software: alleviare i problemi



Necessità di un approccio ingegneristico

- Necessità di applicare principi ingegneristici alla produzione software per sviluppare:
  - il giusto prodotto
  - al giusto costo
  - nel tempo giusto
  - con la giusta qualità

Cosa è l'ingegneria del software?

- "Software engineering" è una disciplina che cerca di fornire le regole per il processo di produzione del software
- Un ingegnere del software dovrebbe:
  - adottare un approccio sistematico e organizzato al proprio lavoro
  - usare strumenti e tecniche appropriate, che dipendono dal problema che deve essere risolto, dai vincoli presenti e dalle risorse disponibili.

Software Engineering: Un'attività di Problem Solving

- Analisi: Comprendere la natura del problema e dividere il problema in pezzi più semplici
- Sintesi: Mettere insieme tutti questi pezzi in una struttura complessa ed omogenea

- Per il problem solving usiamo:
  - Tecniche e metodi:
    - Procedure formali per ottenere risultati utilizzando notazioni ben definite
  - Metodologie:
    - Collezioni di tecniche applicate nel processo di sviluppo ed unificate da un approccio coerente
  - Strumenti:
    - Applicazioni e/o sistemi (semi)automatici per applicare le tecniche

Ingegneria del software e informatica

- L'informatica è una scienza: il "cuore" sono i fondamenti teorici: linguaggi – algoritmi – complessità – formalismi ecc.
- L'ingegneria del software ha a che fare con aspetti più "pratici": come pianificare e sviluppare la produzione di software di qualità.
- Ad un ingegnere del software le conoscenze di base dell'informatica servono quanto la fisica ad un ingegnere elettrico

Scientist vs Engineer

- Computer Scientist
  - Prova teoremi su algoritmi, progetta linguaggi, definisce schemi per la rappresentazione della conoscenza…
  - Ha tempo infinito…
- Ingegnere
  - Sviluppa una soluzione per un problema di un dominio specifico, per un cliente specifico
  - Utilizza computer, applicazioni, tecniche e metodi
- Software Engineer
  - Lavora in varie aree applicative
  - Ha solo 3 mesi…
  - …in cui cambiano continuamente i requisiti e le tecnologie disponibili!

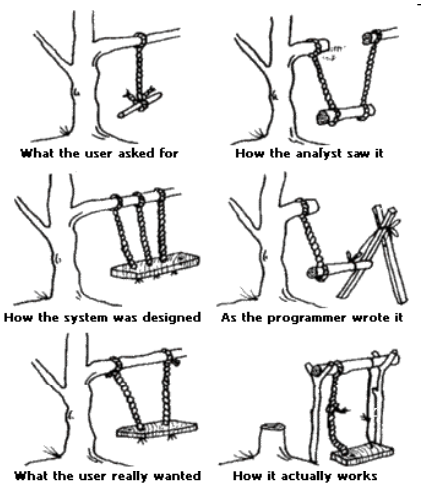
Ingegneria del Software: Scopo

- Riguarda la costruzione di software:
  - di grandi dimensioni
  - di notevole complessità
  - sviluppati tramite lavoro di gruppo
  - …mentre avvengono cambiamenti
- Progetti software di questo tipo hanno tipicamente:
  - versioni multiple
  - lunga durata
  - frequenti cambiamenti
    - eliminazione di difetti
    - adattamento a nuovi ambienti
    - miglioramenti e nuove funzionalità
- Costruzione Multi-persona di software multi-versione

Ingegneria del Software: Contesto

- **L'ingegneria di sistema** ha come oggetto tutti gli aspetti dello sviluppo di un sistema basato su computer, inclusi gli aspetti hardware, software e di processo.
- Software Engineering & System Engineering
  - La maggior parte del SW è collocata all'interno di un "sistema" misto HW/ SW
  - L'obiettivo finale di chi produce è creare tale sistema che soddisfa globalmente i requisiti dell'utente
  - Coinvolgimento nella definizione dei requisiti del sistema
- Conoscenza del dominio applicativo
  - È essenziale per un efficace sviluppo del SW
  - Il SW è utile quando riesce a condensare nei suoi algoritmi la conoscenza del dominio applicativo
- Altrimenti inutile o dannoso
- Per es. sistema di controllo di un aeroplano

Il Processo software



Processo

- Un processo è un particolare metodo per fare qualcosa costituito da una sequenza di passi che coinvolgono attività, vincoli e risorse (Pfleeger)
- Processo: una particolare metodologia operativa che nella tecnica definisce le singole operazioni fondamentali per ottenere un prodotto industriale (Zingarelli)
- Processo software: un metodo per sviluppare del software (Sommerville)

Processo software

- Insieme organizzato di attività che sovrintendono alla costruzione del prodotto da parte del team di sviluppo utilizzando metodi, tecniche, metodologie e strumenti.
- È suddiviso in varie fasi secondo uno schema di riferimento (il ***ciclo di vita del software***)
- Descritto da un modello: informale, semi- formale o formale

Processo software: standard IEEE 610.12-1990

- Processo di sviluppo del software: il processo per cui le necessità dell'utente sono tradotte in un prodotto software. Il processo prevede la traduzione delle necessità dell'utente nelle richieste del software, trasformare le richieste del software nel progetto, implementare il progetto in forma di codice, testing del codice e talvolta, installare e controllare il software per un uso operativo.
  - Nota: queste attività possono sovrapporsi o essere eseguite iterativamente.

Processo di produzione software

- Il processo di produzione software è un insieme di attività il cui fine complessivo è
  - lo sviluppo di un prodotto software oppure
  - la modifica di un prodotto software

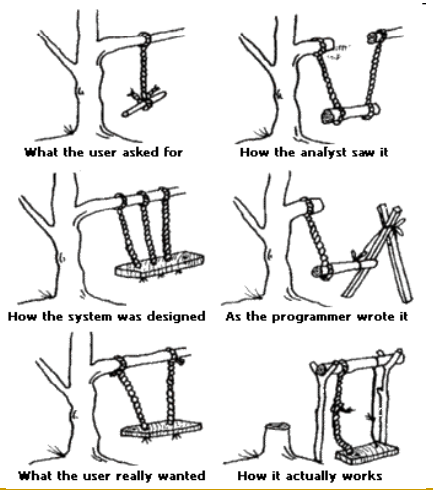
Attività richieste nel processo di sviluppo software

1. Specifica
2. Progettazione
3. Implementazione
4. Validazione
5. Installazione
6. Manutenzione

Problemi nel processo di sviluppo del software

- Specifiche incomplete/incoerenti
- Mancanza di distinzione tra specifica, progettazione e implementazione
- Assenza di un sistema di validazione
- Il software non si consuma: la manutenzione non significa riparare alcune componenti "rotte", ma modificare il prodotto rispetto a nuove esigenze

Qualità del software



Qualità del Software

- La qualità si può riferire sia al prodotto sia al processo
- Qualità del prodotto
  - I requisiti sono il fondamento su cui misurare la qualità
    - Mancanza di conformità ai requisiti => mancanza di qualità
  - Gli standard definiscono i criteri a cui attenersi nello sviluppo del software
    - Non seguire tali criteri produce una qualità insufficiente
  - Requisiti impliciti, spesso taciuti (es. facilità di manutenzione)
    - Ai fini della qualità sono da rispettare quanto quelli espliciti
- Diversi fattori (o caratteristiche) di qualità
  - Qualità interne (intrinseche alla struttura del software)
  - Qualità esterne (percepite dagli utenti)
  - Necessità di metriche per valutare le varie caratteristiche del software
    - Caratteristiche interne misurabili direttamente
    - Caratteristiche interne influenzano quelle esterne

Operatività del Prodotto

- Correttezza: in quale misura il programma rispetta i suoi requisiti.
- Affidabilità: quanto bene le funzionalità offerte rispondono ai suoi requisiti
- Efficienza: tempi di risposta, uso di memoria ...
- Usabilità: lo sforzo per imparare ad operare con il sistema
- Integrità: capacità di sopportare attacchi alla sicurezza del sistema

Revisione del Prodotto

- Manutenibilità: sforzo richiesto per localizzare e risolvere errori in un programma in esercizio.
- Flessibilità: sforzo richiesto per modificare un programma in esercizio.
- Verificabilità: sforzo richiesto per verificare il sistema, per assicurarsi che il sistema faccia effettivamente ciò per cui è stato costruito.

Transizione del Prodotto

- Portabilità: sforzo richiesto per trasferire il software da una configurazione ad altre configurazioni.
- Riusabilità: la misura della facilità con cui parti di software possono venire reimpiegate in altre applicazioni.
- Interoperabilità: sforzo richiesto per far cooperare/dialogare il sistema con altri sistemi.

Sommario

- L'ingegneria del software si occupa di teorie, metodi e strumenti per sviluppare, produrre e mantenere prodotti software
- Prodotti software consistono in programmi e relativa documentazione.
- Gli attributi di un prodotto software
- Il processo software consiste nelle attività necessarie per sviluppare software