Ingegneria e sviluppo del software I

Panoramica sull'Ingegneria del Software:

Che influsso ha avuto l'avvento di Internet sull'ingegneria del software:

Ha portato allo sviluppo di sistemi basati su servizi, altamente distribuiti e massivi. Inoltre ha supportato lo sviluppo delle "app" nell'ambito dei dispositivi mobili.

Da quali due grossi tipi di attività derivano i costi del software e in quali percentuali:

60% sviluppo. 40% testing.

Ian Sommerville definisce il "software" come…:

...i programmi e la documentazione a loro associata.

Ian Sommerville definisce l'ingegneria del software come quella disciplina ingegneristica che…:

...si occupa di tutti gli aspetti della produzione del software

Il software può essere sviluppato per due grandi ambiti di utilizzo. Quali:

Un ambito generale (grande pubblico) e quello customizzato (cliente specifico).

Qual è la differenza tra "Informatica" e "Ingegneria del Software":

Informatica → teoria e aspetti fondamentali della scienza / Ing. → produzione e fornitura.

Quali attributi/caratteristiche deve avere il software per essere definito "buon software":

Fornire le funzionalità e la performance richieste dall'utente. Mantenibile, affidabile e usabile.

Quali sono le migliori tecniche e i migliori metodi in ingegneria del software:

Non ci sono tecniche e metodi che sono validi sempre in assoluto. Dipende dal contesto.

Quali sono le quattro attività fondamentali dell'ingegneria del software:

Specifica, sviluppo, validazione e evoluzione del software.

Quali sono le sfide chiave alle quali l'ingegneria del software deve fare fronte:

Sempre maggiore diversità, tempi di consegna sempre più ridotti, software affidale.

Rispetto alle altre l'ingegneria del software sia tutto sommato una disciplina ingegneristica:

Giovane

Si dice che un prodotto software debba essere "accettabile" per l'utenza per cui è stato progettato:

Significa che il software deve essere: comprensibile, usabile e compatibile con gli altri già in uso.

Si dice che un prodotto software debba essere "affidabile". Cosa significa:

Funzionare come atteso, senza errori, disponibile quando richiesto.

Si dice che un prodotto software debba essere "efficiente". Cosa significa:

Che deve evitare lo spreco (di risorse di sistema quali memoria, CPU, ecc.).

Si dice che un prodotto software debba essere "mantenibile". Cosa significa:

Deve essere scritto in modo da potere essere evoluto per soddisfare i mutevoli requisiti dei clienti.

Si dice che un prodotto software debba essere "sicuro". Cosa significa:

Significa che si deve impedire l'uso malevolo o il danneggiamento del sistema.

Si dice che il costo dell'evoluzione del software personaliz. superi spesso quello di sviluppo. Perché?:

Perché i requisiti degli utenti cambiano/evolvono continuamente e così dovrà fare il software.

Version Management:

Da un punto di vista architetturale un VCS moderno appartiene a uno di due tipi. Quali?

Centralizzato o distribuito

Devo interrompere la sviluppo di una funzionalità e non ho terminato il lavoro. Voglio mettermi il più possibile al riparo dal rischio di perdere quanto realizzato finora. Cosa faccio?

Un push del branch di lavoro sul remoto (in modo da potere fare affidamento sul sistema di backup).

Due ingegneri software usano le seguenti due procedure diverse per l'implementazione di una funzionalità. Quale dei due utilizza l'approccio migliore e perché?

DEV1. Isola correttamente le modifiche in un branch separato.

Git cosa realizza il comando: git clone <URI>?

Crea una copia locale (clona) del repository che si trova all'URI dato.

In che modo un VCS distingue una versione di uno stesso componente software da un'altra?

Ad ogni versione viene associata una chiave univoca (il che permette di gestire diverse versioni delle stesso componente senza dovergli cambiare nome ogni volta).

In che modo un VCS minimizza l'uso dello storage?

Evitando di persistere copie identiche degli stessi file e persistendo eventualmente i delta (che verranno applicati in sequenza a partire da una copia master per ottenere la copia attuale).

In che modo un VCS può fungere da sistema di backup per i componenti software in fase di sviluppo?

Permettendo la segregazione dei componenti in branch separati e la persistenza upstream (e.g. su un server remoto gestito adeguatamente) degli stessi.

In che modo un Version Control System (VCS) supporta sviluppatori diversi che lavorano contemporaneamente allo stesso componente software:

Garantendo che i cambiamenti fatti dai vari sviluppatori non interferiscano, fondendo automaticamente le modifiche quando possibile ed eventualmente chiedendo di risolvere i conflitti.

Mando le mie modifiche "upstream" con Git. Quale comando uso?

git push

Voglio integrare le ultime modifiche del mio team, nel "dev" branch, prima di iniziare la mia sessione di lavoro. Con Git che comandi uso?

git checkout dev; git pull

Software Dependencies:

Aumentando il numero di dipendenze aumenta il rischio di incorrere nel fenomeno detto "dependency hell". Di cosa si tratta?

Del fenomeno in cui le dipendenze non possono essere risolte (conflitto di versioni, riferimenti circolari, ...).

Che cosa è una "dipendenza transitiva"?

È una dipendenza di una dipendenza.

In ambiente Java qual è il nome di uno strumento di gestione delle dipendenze?

Maven

L'utilizzo di dipendenze software introduce un "problema" nella gestione. Quale?

Fiducia. Le dipendenze vanno testate (essendo codice terze parti)

Per il prodotto di un cliente definisco una dipendenza senza stabilire/fissare il numero di versione. Il sistema scaricherà quindi sempre l'ultima versione disponibile per la build. È una buona/cattiva scelta. Perché?

È una buona scelta solo se posso testare sistematicamente la dipedenza (funzionalità, sicurezza, affidabilità, performance, ...). Altrimenti è una cattiva scelta.

Realizzando un prodotto software voglio riutilizzare quanto più codice esistente possibile. Indica almeno due motivi per cui questa affermazione è vera o falsa.

Vero. Voglio per esempio minimizzare il tempo di sviluppo e ridurre il rischio di bug.

Voglio affidarmi ad una libreria matematica terze parti. Decido di scaricare la libreria e di copiarla nella gerarchia di progetto. Faccio una buona scelta? Perché?

Salvo casi particolare è una cattiva scelta. Così facendo riduco l'aggiornamento della libreria ad un processo manuale. Meglio utilizzare un sistema di gestione di dipendenze.

Volendo cifrare delle password decido di implementare un algoritmo mio. Prendo una buona decisione? Si/No? Perché?

No. Il problema è già stato risolto e una mia implementazione rischia solo di essere debole.

Un sistema di gestione delle dipendenze fornisce tipicamente gli artifatti attraverso un repository in rete (Internet). Quale rischio comporta questo e come si può mitigare?

Un rischio di disponibilità del servizio. Si può quindi creare un proprio repository mirror.

Una "dipendenza software" è in generale definibile come...

Qualsiasi codice terze parti su cui il sistema fa affidamente per funzionare.

System build:

È sempre consigliabile automatizzare il processo di build di un sistema. Perché?

Perché questo processo implica assemblare molte informazioni riguardo al software da costruire e al suo ambiente operativo. Operare manualmente è dispendioso, rischioso, lento e non ripetibile.

È sempre consigliabile eseguire il processo di build in un ambiente "vergine". Perché?

Per evitare il rischio che dipendenze locali influenzino (negativamente o positivamente) il risultato del processo.

È sempre consigliabile eseguire il processo di build prima di tutto in locale sul proprio sistema di sviluppo prima di eseguire una push upstream delle proprie modifiche. Perché?

Per ridurre al minimo il rischio di condividere un sistema non costruibile.

Nel sistema di build Maven qual è la sezione relativa alla costruzione degli artifatti?

La sezione "build".

Il meccanismo di build "a cascata" è utile a minimizzare cosa?

Il numero di componenti da ricostruire e quindi le risorse e il tempo necessario per il completamente del processo di build.

Il processo di build coinvolge tipicamente tre sistemi/piattaforme. Quali?

I sistemi/piattaforme di: sviluppo, build, destinazione/target.

In ambiente Java, e utilizzando Maven come sistema di build, cosa devo configurare nel file pom.xml volendo costruire un .jar eseguibile (quindi con tutte le dipendenze al suo interno)?

Un plugin adatto (eventualmente associato alla fase "package").

In ambito C qual è il nome di uno strumento di build?

make

Nel sistema di build Maven la fase che costruisce l'artifatto in un formato distribuibile (jar, war, ...) è...

Package

Un programma eseguibile può essere costruito in generale in maniera statica oppure dinamica. Qual è la differenza tra le due?

Statica: le dipedenze sono incluse nell'artifatto eseguibile. Dinamica: nell'artifatto eseguibile sono inclusi solo i riferimenti alle dipendenze.

Configuration Management:

Il "Change Management" è quel processo che ha a che fare con ...

Il tenere traccia delle richieste di cambiamento ai sistemi (consegnati) da parte dei clienti e degli sviluppatori.

Il "Configuration Management" si preoccupa della gestione di sistemi software "in evoluzione". Con quali 3 risorse/aspetti in particolare?

Linee di condotta (policies), processi e strumenti

Il "Release Management" è quel processo che ha a che fare con ...

Il preparare il software per il rilascio all'esterno (per l'uso da parte dei clienti) e il tenere traccia delle versioni.

Il "System Building" è quel processo che si occupa ...

Dell'assemblaggio dei componenti, dei dati e delle librerie, compilandoli e lincandoli, per ottenere un sistema eseguibile.

Il "Version Control" è un'attività che ha a che fare con ...

Il tenere traccia delle molteplici versioni dei componenti di sistema e far sì che i cambiamenti dei vari sviluppatori non interferiscano gli uni con gli altri.

Software Design:

Il costrutto di un linguaggio di programmazione che permette di raggruppare in un'unica struttura i dati e i metodi che operano su di essi si definisce ... (1 parola)

Encapsulation

Il grado di interconnessione (interrelazione, interdipendenza, ...) tra i componenti di una struttura software (e.g. tra due classi) è detto? (1 parola)

coupling

Il principio di "information hiding" si riferisce alla necessità di nascondere cosa esattamente?

Le decisioni di design che hanno più probabilità di dovere essere modificate.

Il processo che definisce i componenti strutturali di un sistema e le loro relazioni è un processo di design (i.e. progettazione) di cosa? (1 parola)

Architettura

Il software è sottoposto a continuo cambiamento. Per questo motivo vogliamo progettare componenti che se modificati permettano di ridurre al minimo ... cosa?

Le conseguenze delle modifiche sugli altri componenti del sistema.

In object-orientation "ereditarietà" e la ricerca di un basso accoppiamento (i.e. loose coupling) possono dirsi in conflitto. Perché?

Perché in una gerarchia le modifiche ad una superclasse hanno conseguenze dirette e inevitabili sulle sottoclassi. Il che contrasta con la ricerca di un design che limiti questo effetto.

La misura della forza della relazione tra i componenti di una struttura software (e.g. i metodi di una classe, le classi di un package, ...) è detta? (1 parola)

Coesione

Nel modello OSI è evidente il principio di progettazione detto "Separation of Concerns". Con quale tipo di architettura?

A layer.

Osserva il codice ..static void sort(List<T> lis, Comparator<? Super T> comp) {...} approccio utilizzato?

Inversion of control

Pensando ad alcune delle metriche di qualità del design (...)... codice con Database

La soluzione di DEV2 è più generica perché disaccoppia il DBMS.

Pensando ad alcune delle metriche di qualità del design (come sopra).. codice con funzioni matematiche.

La classe di DEV1 è coesa (sono tutte funzioni matematiche). La classe di DEV2 non lo è.

Pensando ad alcune delle metriche di qualità del design (come sopra)... codice con liste.

La soluzione di DEV2 rispetta il principio di primitivity. È più generica/flessibile.

Programmare in funzione di "interfacce" invece che in funzione di "implementazioni" permette di ridurre l'accoppiamento di un sistema. Spiega brevemente.

Si occoppiano i componenti sulla base di un contratto (interfaccia) in modo che possa essere sostituita qualsiasi implementazione lo rispetti (rendendo così il design flessibile, evolvibile, ...).

Tra i principi di design SOLID la "S" indica il principio detto: "Single Responsibility". Definiscilo.

Il principio per il quale i componenti software devono essere progettati in maniera tale che il loro cambiamento deve far capo ad un singolo gruppo di interessi.

Tra i principi di design SOLID la "O" indica il principio detto: "Open-Closed". Definiscilo brevemente.

Open (aperto) alle estensioni. Closed (chiuso) alle modifiche. Si deve permettere di estendere le funzionalità di un sistema senza dovere modificare direttamente i suoi componenti.

Tra i principi di design SOLID la "L" indica il principio detto: "Liskov Substitution". Definiscilo.

Un principio per il quale un'istanza di un sottotipo deve potere essere sostituita ad una di un supertipo senza alterare il funzionamento desiderato del programma.

Tra i principi di design SOLID la "I" indica il principio detto: "Interface Segregation". Definiscilo.

Un principio per il quale è opportuno segregare comportamenti diversi in interfacce diverse per evitare di costringere un client ad implementare più comportamenti del necessario.

Tra i principi di design SOLID la "D" indica il principio detto: "Dependency Inversion". Definiscilo.

Un principio per il quale è opportuno introdurre un'astrazione tra moduli di alto e basso livello in modo da rimuovere le dipendenze tra i due livelli.

Un approccio di scomposizione orientata agli oggetti è indicato nel contesto di un sistema di che tipo?

Complesso e instabile (e.g. che evolve)

Un approccio di scomposizione funzionale è indicato nel contesto di un sistema con requisiti di che natura?

Stabili.

Processi Software:

Ci sono diversi modelli generali di processi software. Citane almeno due.

Waterfall (a cascata). Incrementale. Integrazione e configurazione di componenti riutilizzabili.

Cosa descrivono i modelli generali di processo (general process models)?

L'approccio allo sviluppo del software. l'organizzazione dei processi software..

I processi di progettazione e sviluppo (software design and development) sono quei processi che si occupano di...

...trasformare le specifiche software in sistemi software eseguibili.

I processi software devono tenere conto di un elemento che si presenta costantemente e devono includere attività per la sua gestione e integrazione. Questo elemento è il (1 parola)...

Cambiamento

I "processi software" sono quell'insieme di attività coinvolte nella ..... dei sistemi software. Mancante?

Produzione

Il cambiamento aumenta i costi dello sviluppo software. Perché?

Semplicemente, perché il lavoro fatto dovrà essere modificato/rifatto.

Il miglioramento dei processi software ha come scopo ultimo quello di migliorare la qualità del software prodotto, diminuire i costi, ridurre il tempo di sviluppo attraverso il raffinamento/cambiamento dei processi. Pensando in maniera ingegnerista, su cosa si dovrebbe basare il raffinamento/cambiamento?

Su misurazioni/dati oggettivi e la loro analisi.

Il processo di validazione del software (software validation) si occupa di...

verificare che il sistema sia conforme alle specifiche e che soddisfi le reali necessità degli utenti.

Indica le 3 fasi fondamentali del ciclo di miglioramento dei processi software.

Misurare, analizzare, modificare/cambiare.

L'evoluzione del software (software evolution) è quel processo che avviene quando...

...i sistemi software vengono cambiati per soddisfare nuovi requisiti.

L'ingegneria dei requisiti (requirements engineering) è quel processo che si occupa di definire...

...le specifiche software.

Qual è di principio l'approccio al cambiamento nei modelli iterativi? (1 parola) ...

Tollerante

Qual è di principio l'approccio al cambiamento quando si fa della prototipizzazione? (1 parola) ...

Anticipazione

Tra i modelli di processo generici troviamo il "modello a integrazione e configurazione" (integration and configuration model). Descrivilo in una frase.

Un modello che prevede la realizzazione del sistema attraverso il riutilizzo di componenti esistenti, configurandoli nel contesto e integrandoli.

Tra i modelli di processo generici troviamo il "modello incrementale" (incremental model). Descrivilo.

Un modello che organizza le attività dei processi software come fasi interlacciate e sviluppa il sistema in versioni (incrementi) successive.

Tra i modelli di processo generici troviamo il "modello a cascata" (waterfall model). Descrivilo.

Un modello che organizza le attività dei processi software come fasi separate (e sequenziali) e fornisce il sistema tutto alla fine.

Panoramica sull'ingegneria dei requisiti:

Cosa esprime un "requisito funzionale"?

Esprime un servizio che il sistema deve fornire.

Cosa esprime un "requisito non funzionale"?

Esprime i vincoli su un servizio offerto dal sistema.

Cosa esprimono i "requisiti di sistema"?

Esprimono le funzionalità, i servizi e i vincoli operativi del sistema software.

Cosa esprimono i "requisiti utente"?

Esprimono i servizi che il sistema deve fornire agli utenti del sistema e i vincoli entro i quali devono operare.

Cosa specificano i "requisiti di prodotto"?

Specificano il comportamento di run-time del software (e.g. usabilità, efficienza, affidabiltà, sicurezza).

Cosa specificano i "requisiti esterni"?

Requisiti derivanti da tutto ciò che è esterno al sistema e al suo processo di sviluppo (e.g. regolamentari, etici, legislativi).

Cosa specificano i "requisiti organizzativi"?

Specificano i requisiti derivanti dalle politiche e dalle procedure nelle organizzazioni (e.g. ambientali, operazionali, di sviluppo).

Da cosa derivano i "requisiti di dominio"?

Dal dominio applicativo del sistema software.

La spirale del processo di ingegneria dei requisiti è formata da 3 attività che si ripetono iterativamente. Quali?

Elicitazione, specifica, validazione.

Nel processo di ingegneria dei requisiti capita di realizzare uno studio di fattibilità per verificare se il nuovo sistema... Indica uno dei tre motivi tipici per cui si decide in questo senso.

1) potrà essere realizzato nei tempi e nel budget previsto, 2) contribuisce agli obiettivi generali dell'organizzazione, 3) si integri nei sistemi già in uso.

Osserva la definizione di un requisito da parte di due ingegneri... Quale dei due mostra un approccio corretto? Perché?

ENG2: fornisce una definizione misurabile oggettivamente.

Miscellanea:

Dovendo descrivere in maniera dettagliata l'interazione e i messaggi scambiati dai componenti di una procedura di login di un utente è preferibile utilizzare un diag. di sequenza o un diag. delle attività?

Sequence diagram. Sono entrambi diagrammi comportamentali, ma l'activity diagram rappresenta una seq. di azioni e decisioni e non lo scambio di messaggi di classi in una sequenza temporale.

I diagrammi UML possono esprimere "comportamenti", "interazioni" e "strutture". Un diagramma delle classi in quale categoria ricade?

Strutturale

Il grado di raggiungimento di obiettivi di efficacia, efficienza e soddisfazione di utilizzo di un prodotto software da parte degli utenti in un certo contesto è detto? (1 parola)

Usabilità

La "internazionalizzazione" (i.e. i18n) di un prodotto software ha a che fare con...

il progettarlo affinché possa essere localizzato senza modifiche ingegneristiche (sovraintende alla localizzazione).

La "localizzazione" (i.e. l10n) di un prodotto software ha a che fare con...

...il processo del suo adattamento ad una certa nazione o regione (sottende all'-internazionalizzazione).

Nella progettazione della UI, a cosa si riferisce il principio del "carico sulla memoria" (i.e. memory load)?

Al fatto che si deve ridurre la quantità di informazioni che devono essere ricordate tra le azioni.

Nella progettazione della UI, a cosa si riferisce il principio della "coerenza" (i.e. consistency)?

Al fatto che i componenti comparabili dovrebbero avere lo stesso formato e operazioni comparabili dovrebbero essere attivate nello stesso modo.

Nella progettazione della User Interaction, a cosa si riferisce il principio di "efficienza" (i.e. efficiency)?

Al fatto che si deve minimizzare lo sforzo che gli utenti devono fare (pensiero, lettura, scrittura, movimento).

Nella progettazione della UI, a cosa si riferisce il principio di "guida utente" (i.e. user guidance)?

Al fatto che si deve incomporare qualche forma di guida/aiuto contestuale.

Nella progettazione della UI, a cosa si riferisce il principio della "minima sopresa" (i.e. minimal surprise)?

Al fatto che gli utenti dovrebbero essere in grado di prevedere il comportamento del prodotto.

Nella progettazione della UI, a cosa si riferisce il principio della "familiarità" (i.e. familiarity)?

Al fatto che si dovrebbero usare termini e concetti familiari agli utenti.

Nella progettazione della UI, a cosa si riferisce il principio di "rimediabilità" (i.e. recoverability)?

Al fatto che si deve permettere agli utenti di rimediare ai propri errori.

Nella progettazione della UI, a cosa si riferisce il principio del "riscontro" (i.e. feedback)?

Al fatto che si deve mantenere una comunicazione birezionale e gli utenti dovrebbero avere riscontri uditivi e visuali.

Nella Process View del framework UML, cosa rappresentano i diagrammi delle attività?

Un workflow.

Nella Use-Case View del framework UML, cosa esprimono i diagrammi Use-Case?

Le interazioni degli utenti con il sistema

Se abbiamo previsto una codifica colore (i.e. colour coding) per rappresentare stati, risultati, ecc., e vogliamo assicurarci di non incorrere in conflitti culturali, quale semplice accorgimento possiamo adottare?

Seguendo la regola di "mettere l'utente in controllo" possiamo permettere che i colori siano configurabili.

Un carattere ASCII è anche un carattere UTF-8. Vero/Falso? Perché?

Vero. I primi 128 code points Unicode sono gli stessi della tavola ASCII