Ingegneria del Software — Esame di teoria – fila A

A.A. 2020-2021 — 28 gennaio 2021

NOME: COGNOME: MATRICOLA:

Indicare gli scopi principali di ogni fase del processo software:

Specifica: si individuano i requisiti funzionali e non, si stila il documento dei requisiti; Progettazione: si progetta l'architettura del sistema, il suo deployment e la sua interfaccia; Implementazione: si scrive il codice attenendosi ai requisiti; Collaudo: si testa il codice per valutarne l'assenza di bachi prima del rilascio; Manutenzione: si modifica il sistema per eliminare eventuali difetti post-rilascio, o aggiungere funzionalità.

Supponendo di dover sviluppare un sistema software per la vendita on-line di prodotti, fornire tre esempi di requisito funzionale, un esempio di requisito non funzionale di prodotto, un esempio di requisito non funzionale organizzativo, un esempio di requisito non funzionale esterno:

RF: creazione nuovo utente, aggiunta nuovo prodotto, generazione di notifiche
RNF prodotto: stesura di una ricca documentazione
RNF organizzativo: interfaccia utente realizzata tramite Java Swing
RNF esterno: integrazione con circuiti di pagamento

Indicare cosa è prodotto in termini di documentazione, da ogni fase del processo di Specifica:

Studio di fattibilità: rapporto di fattibilità; Deduzione dei requisiti: requisiti utente; Analisi dei requisiti: requisiti di sistema; Validazione dei requisiti: documento dei requisiti.

Descrivere due modi per ottenere informazioni sui requisiti:

Dialogo con stakeholder: vengono svolte interviste con gli utenti finali per dedurre i requisiti che dovranno essere implementati nel prodotto

Analisi del dominio applicativo: si studia l'insieme di entità reali su cui il sistema software ha effetto (area economica, sociale, tecnica …)

Rappresentare la seguente formula con un modello data-flow in cui *a*, *b*, *c*, *d* sono i dati di input iniziali, i nodi filter rappresentano gli operatori matematici, e *f* è l'ouput finale: $(a + |b| \; c) / \sqrt{d} = f$

Si consideri l'architettura stratificata su 3 livelli. Spiegare lo scopo di ogni livello:

Nell'architettura a tre livelli, ogni livello è realizzato su una macchina diversa (client e due server). Il livello presentazione si occupa di mostrare un'interfaccia all'utente e raccogliere input; il livello elaborazione si occupa di elaborare i dati ottenuti in input e produrre dati in output; il livello gestione dati si occupa della modifica, aggiunta, rimozione di dati da un database.

Nella fase di collaudo, cosa si intende per verifica e per validazione?

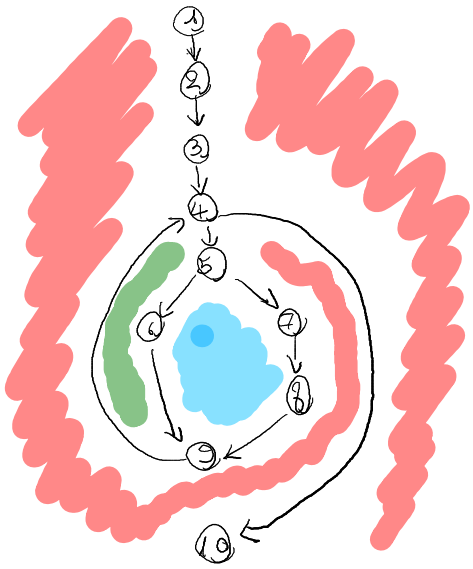
Verifica: si controlla che il prodotto implementato realizzi ogni servizio senza malfunzionamenti, si scoprono errori di programmazione
Validazione: si controlla che il prodotto implementato soddisfi i requisiti del committente, si scoprono anomalie o omissioni

Fare un esempio di caratteristica che deve essere collaudata con l'ispezione, e un esempio di caratteristica che deve essere collaudata con il testing, spiegandone i motivi.

Ispezione: struttura del codice, in quanto solamente guardando il codice è possibile verificarne la struttura (es. metodi dallo scopo poco chiaro, mancanza di commenti …)
Testing: velocità di risposta del sistema, in quanto risulta evidente solo quando il sistema viene effettivamente testato in situazioni simili a quelle a cui andrà incontro una volta rilasciato (stress testing)

// Segue il codice della funzione calcola_potenza:

```
1 float calcola_potenza(float base, int esp)
  {
2   int i=0;
3   float potenza=1;
4   while ( i < abs(esp) )
    {
5     if (esp > 0)
6           potenza = potenza * base;
7     else
8           potenza = potenza / base;
9     i++;
    }
10  return potenza;
  }
```



Disegnare il flow-graph corrispondente al programma.
1) Indicare un metodo per calcolare la complessità ciclomatica ed applicarlo al flow-graph ottenuto.
2) Cosa indica il valore ottenuto?
3) Individuare i cammini indipendenti all'interno del flow-graph.
4) Per ogni cammino ottenuto definire un test-case che determini tale cammino.

1. Conteggio dell regioni del grafo: CC=3
2. Il numero di test case minimo da realizzare per coprire ogni possibile cammino del codice
3. 1,2,3,4,5,10 – 1,2,3,4,5,6,9,4 … – 1,2,3,4,5,7,8,9,4 …
4. Input:(2,2) Output:4 – Input:(2,-1) Output:0.5 – Input:(4,0) Output:1

Supponendo di integrare i componenti del sistema in modo incrementale, indicare cosa si intende per

Testing del componente: si testa che il componente funzioni in maniera corretta, in maniera isolata

Testing di integrazione: si testa che il sistema funzioni correttamente anche dopo l'aggiunta del nuovo componente

Release testing: testing del sistema costituito da tutti i suoi componenti

Considerando di nuovo il sistema software per la vendita on line di prodotti, fornire un esempio concreto di manutenzione correttiva, migliorativa, adattativa:

Correttiva: correzione di un errore che impedisce la creazione di un nuovo utente
Migliorativa: rendere più accessibile il layout
Adattiva: aggiornamento della versione di Java utilizzata

Indicare tre fattori che influenzano il costo di manutenzione e spiegarne il motivo:

Struttura del codice: codice ben scritto e documentato è più facile da mantenere
Stabilità dello staff: i costi si riducono se lo staff che ha sviluppato il prodotto è lo steso che lo mantiene
Collaudo approfondito: una fase di collaudo approfondita riduce la probabilità di dover in futuro fare della manutenzione correttiva

Fare due esempi di reverse engineering:

dal codice si ottiene ... diagrammi di progettazione (UML)

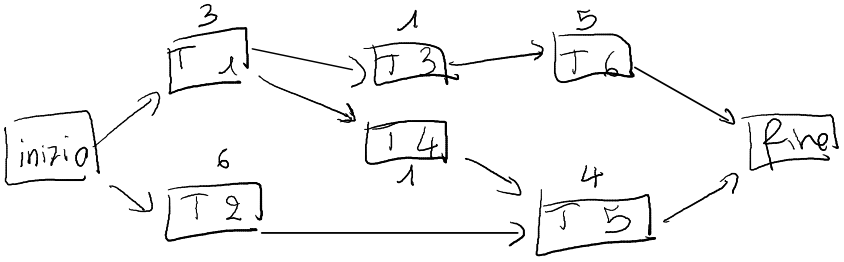
dalla base di dati si ottiene ...diagrammi di dati (E-R)

Fare un esempio di rischio di prodotto, rischio di progetto e rischio di business. Spiegare gli effetti di ciascuno:

Prodotto: gli strumenti case non sono all'altezza del compito da svolgere, intaccano la qualità del prodotto
Progetto: uno sviluppatore lascia l'azienda, può causare ritardi alla consegna del progetto
Business: viene rilasciato un prodotto concorrente

All'interno di un processo software sono stati individuati i task T1, T2, T3, T4, T5, T6. Tali task hanno le seguenti durate e dipendenze:

Task	Durata	Dipendenze
T1	3gg	-
T2	6gg	-
T3	1gg	T1
T4	1gg	T1
T5	4gg	T2, T4
T6	5gg	T3



In base alle dipendenze tra task, disegnare l'activity network.

In base all'activity network, indicare tutti i task che da ultimare per consentire l'inizio di T5: T1,T4,t2

In base alla durata dei task, individuare il cammino critico nell'activity network: T2,T5

In base al cammino critico, determinare la durata del progetto: 10 giorni

Qual è il ritardo massimo consentito a T1 in modo da non alterare la durata del progetto, assumendo che gli altri task rispettino i tempi previsti? 1 giorno

Spiegare il motivo:

T1 può tardare di un giorno, in quanto i task T1,T3 e T6 impiegano 9 giorni per essere svolti, ed il cammino critico ne dura 10.

Fornire la definizione di prototipo: versione preliminare del sistema che realizza un sottoinsieme dei requisiti

Qual è la differenza tra i due tipi di prototipo?

Per ogni tipo di prototipo, indicare un modello di processo in cui viene utilizzato:

Due tipi di prototipo: "usa e getta" ed evolutivo. Il primo realizza un insieme di requisiti poco chiari, viene mostrato al committente per validazione, ma viene poi scartato (utilizzato nei modelli a cascata); il secondo evolve nel prodotto finale , implementando man mano tutti i requisiti, utilizzato nel modello a sviluppo evolutivo.

Dato il modello di processo detto eXtreme Programming, indicare due modi per ottenere la rapidità di consegna e due modi per ottenere la qualità del prodotto, spiegandone gli effetti:

Rapidità di consegna: specifica ad alto livello (i requisiti vengono definiti in maniera approfondita durante lo sviluppo), tempo di release limitato
Qualità del prodotto: TDD (il codice è già testato), programmazione a coppie (più probabilità di notare errori)

argomento	punteggio (su 100)	punteggio (su 31)
fasi processo sw	14	4.34
requisiti funz. E non funz.	6	1.86
processo specifica	8	2.48
deduzione requisiti	4	1.24
Data-flow	5	1.55
arch. Stratificata	6	1.86
V & V	2	0.62
Ispezione / testing	4	1.24
Flow-graph	11	3.41
test incrementale	3	0.93
tipi di manutenzione	3	0.93
costo di manutenzione	6	1.86
reverse engineering	2	0.62
rischi	6	1.86
activity network	9	2.79
prototipo	7	2.17
XP	4	1.24
Totale	100	31