Non è consentito consultare nessun materiale e usare dispositivi mobili.
Il file .odt deve essere consegnato nell'apposita cartella sul desktop.
Non fare logout e non spegnere.

Ingegneria del Software — Esame di teoria – fila A

A.A. 2022-2023 — 19 gennaio 2022

NOME: COGNOME: MATRICOLA:

Fornire una definizione di Ingegneria del software:

L'ingegneria del software è l'applicazione del processo dell'ingegneria al processo software. Ciò include le 5 fasi di: specifica, progettazione, implementazione, collaudo e manutenzione.

Indicare gli scopi principali di ogni fase del processo software:

Specifica: deduzione ed analisi dei requisiti (ovvero ciò che bisogna implementare/gestire durante il processo); Progettazione: definizione dell'architettura, del controllo e del comportamento dei componenti; Implementazione: scrittura del codice; Collaudo: testing del sistema e correzione errori; Manutenzione: modifiche del sistema dopo la consegna

Per ogni fase del processo software indicare un esempio di strumento CASE:

Specifica: editor dei requsiti (UML) (Visual Paradigm)
Progettazione: editor di diagrammi di progetazione (UML) (Visual Paradigm)
Implementazione: IDE, Git
Collaudo: strumenti per ispezione del codice automatico (SonarCube, ECLEmma)
Manutenzione: strumenti per la gestione di versione di codice e documentazione (Git, JavaDoc)

Supponendo di dover sviluppare un sistema software per gestire le prenotazioni alle lezioni in presenza, 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: prenotazione posto, creazione utente, modifica orario lezione
RNF prodotto: implementare interfaccia chiara ed intuitiva
RNF organizzativo: utilizzo di Java per implementazione, SQLite come db
RNF esterno: opportuno trattamento di dati degli utenti

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

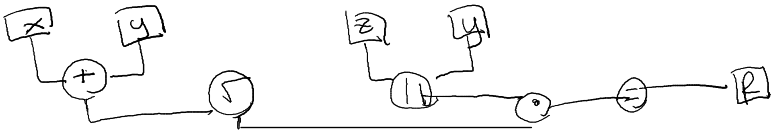
Studio di fattibilità: documento di fattibilità
Deduzione dei requisiti: requisiti utente
Analisi dei requisiti: requisiti di sistema
Validazione dei requisiti: documento dei requisiti validato

Indicare due tipi di tracciabilità dei requisiti e spiegare a cosa servono:

Tracciabilità della sorgente: reperire la fonte di informazione relativa al requisito
Tracciabilità del progetto: individuare i componenti del sistema che realizzano il requisito
Tracciabilità dei test: individuare i test-case usati per collaudare il requisito
Tracciabilità dei requisiti: individuare i requisiti dipendenti

Rappresentare la seguente formula con un modello data-flow in cui *x, y, z* sono i dati di input iniziali, i nodi filter rappresentano gli operatori matematici, e *f* è l'ouput finale:

$\sqrt{x+y}\cdot|z-y|=f$



Indicare due stili di controllo e spiegare come funzionano:

Controllo centralizzato: un componente centrale, detto controllore, guida il funzionamento del sistema
Controllo basato su eventi: nessun controllore, si basa su eventi esterni, ogni componente gestisce determinati eventi

Indicare quattro possibili difetti che l'analisi statica del codice può individuare:

Codice non raggiungibile
Salti incondizionati
Variabili dichiarate mai usate
Errori nell'aritmetica dei puntatori

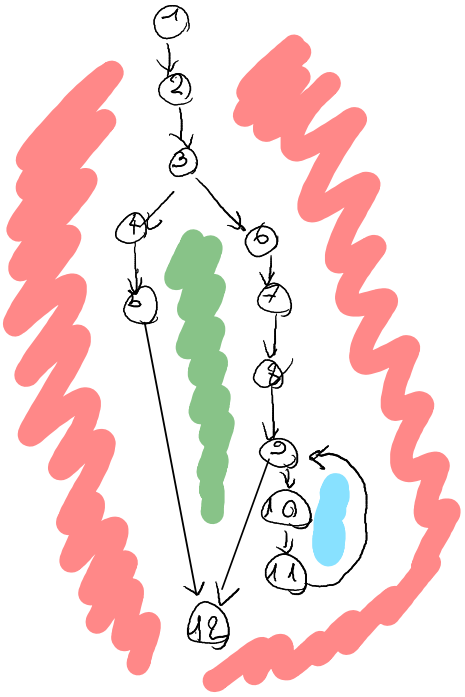
Supponendo che l'esecuzione di un test-case abbia riscontrato la presenza di un difetto, cosa si intende per

Debugging: individuazione dell'errore nel codice

Correzione: correzione effettiva dell'errore

Test di regressione: rerun di tutti i test per vedere se la modifica non ha danneggiato altre parti del codice

```
// funzione che calcola il prodotto di due numeri tramite la somma
int prodotto(int a, int b)
{
        int p, i;
        if (b<0)
          {
                p=-1;
                printf("ERRORE\n");
          }
        else
          {
                p=0;
                i=0;
                while (i<b)
                  {
                        p=p+a;
                        i++;
                  }
          }
        return p;
}
```



Disegnare il flow-graph corrispondente al programma.
Indicare un metodo per calcolare la complessità ciclomatica ed applicarlo al flow-graph ottenuto.
Cosa indica il valore ottenuto?
Individuare i cammini indipendenti all'interno del flow-graph.

CC = n.regioni = 3
Indica il numero minimo di test case per coprire ogni cammino del codice.
1. 1,2,3,4,5,12 - 2. 1,2,3,6,7,8,9,12 - 3. 1,2,3,6,7,8,9,10,11,9 ...

Per ogni cammino ottenuto definire un test-case che determini tale cammino.

1. Input:(3,-1) – Output:-1
2. Input:(3,1) – Output:3
3. Input:(3,3) – Output:9

Considerando di nuovo il sistema software per gestire le prenotazioni alle lezioni in presenza, fornire un esempio concreto di manutenzione correttiva, migliorativa, adattativa:

Correttiva: correzione di un errore che rendeva impossibile cambiare l'orario di una lezione
Migliorativa: nuova interfaccia grafica più intuitiva
Adattiva: migrazione a database NoSql

Indicare gli scopi delle attività di pianificazione del progetto.

Scomposizione: divisione del progetto in task, identificazione delle dipendenze tra task

Tempistica (scheduling):stima dei tempi di task e progetti

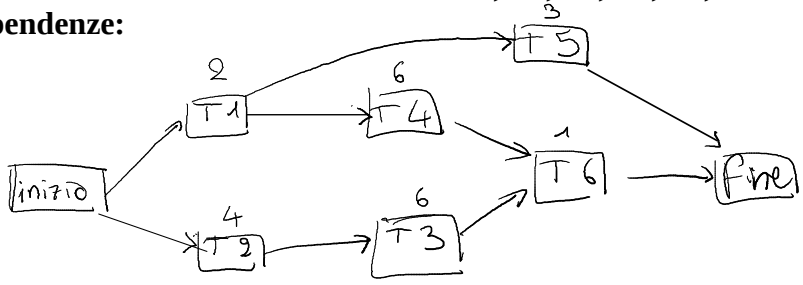
Assegnazione delle risorse: assegnazione di staff, bugdet, hw o sw ad ogni task

Considerando di nuovo il sistema software per gestire le prenotazioni alle lezioni in presenza, 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 lavoro richiesto
Progetto: no è possibile reperire sviluppatori con conoscenze adeguate
Business: viene rilasciato un prodotto concorrente sul mercato

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	2gg	-
T2	4gg	-
T3	6gg	T2
T4	6gg	T1
T5	3gg	T1
T6	1gg	T3,T4



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

In base all'activity network, indicare tutti i task da ultimare per consentire l'inizio di T6: T1,T2,T4,T3

In base alla durata dei task, individuare il cammino critico nell'activity network: T2,T3,T6

In base al cammino critico, determinare la durata del progetto: 11 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? 2 giorni

Spiegare il motivo:

dato che il cammino critico dura 11 giorni, ed il cammino T1–T4–T6 ne dura 9, T1 può tardare al massimo 2 giorni.

Indicare due differenze fondamentali tra Sviluppo Evolutivo e Sviluppo Incrementale, e spiegare come influenzano gli attributi del processo:

argomento	punteggio (su 100)	punteggio (su 31)
definizione ing. Sw	2	0.62
fasi processo sw	14	4.34
strumenti CASE	5	1.55
requisiti funz. E non funz.	6	1.86
processo specifica	8	2.48
tracciabilità	4	1.24
Data-flow	5	1.55
stili controllo	6	1.86
analisi statica	4	1.24
debugging	6	1.86
Flow-graph	11	3.41
tipi di manutenzione	3	0.93
pianificazione	7	2.17
rischi	6	1.86
activity network	9	2.79
sviluppo evolutivo/incrementale	4	1.24
Totale	100	31

Sviluppo evolutivo: non si conosce a priori quanti cicli di sviluppo ci saranno, minor supporto al cambiamento dei requisiti.
Sviluppo incrementale: si stabilisce a priori quanti cicli di sviluppo ci saranno, documentazione più accurata (visibilità più elevata), più robustezza al cambiamento di requisiti.