Università di Roma TorVergata
Laurea Magistrale in Informatica

Corso: Modelli e Qualità del Software
Federica Magliocca, 0318517,
federica.magliocca@students.uniroma2.eu
9 CFU

Nome della prova: Prit1 MQS

Data della Prova: 17/04/2023

Domande

Quesito n.1L
Elencare gli indici di qualità del software più rilevanti per le attività di manutenzione perfettiva e adattativa.

Quesito n.2L
Definire l'indice di qualità flessibilità, concettualmente e attraverso i suoi attributi, e dare il metodo di calcolo del
suo valore.

Quesito n.3L
Descrivere la checklist dell'attributo complesssità e quella dell'attributo modularità, ed il metodo di calcolo del loro
valore. Nel descrivere la checklist limitarsi a dire quali aspetti del progetto software si propone di verificare.

Quesito n.4L
Elencare i metodi di progetto preliminare e dire cosa si propongono di ottenere.

Quesito n.5L
Descrivere il metodo TA di progetto preliminare e dire se permette di raggiungere un dato livello di coesione e
quale.

Quesito n.6L
Descrivere i concetti di coesione e coupling, precisando se trattasi di indici, attributi o sotto-attributi di qualità. Ove
trattasi di attributi o sotto-attributi specificare rispettivamente a quale indice o attributo si riferiscono. Darne inoltre
le scale di misura col relativo significato.

Quesito n.7L
Dare il flow-graph del modulo descritto dalla formula: F=D1((D0(D1);P1), D3(D0))
ed esprimerne pseudo-codice, depth of nesting , D-structuredness, complessità ciclomatica CC e CC essenziale..

Quesito n.8L
Specificare quali indici di qualità, e relativi attributi, si verificano con static verification testing (SVT) a livello di
progetto preliminare e quali a livello di progetto dettagliato. Descrivere cosa si verifica con Code SVT.

Quesito n.9L
Descrivere lo scopo del dynamic validation testing (DVT), e descrivere il processo DVT e le strategie DVT.

Quesito n.10L
Descrivere lo scopo del dynamic validation testing (DVT) e la pianificazione DVT. Descrivere i vari tipi di DVT e
dire quale tipo di DVT si applica al test di unità e moduli, quale al test di sottosistema e quale al test di sistema .

Quesito n.11L
Dire in quale tipo di DT testing si applica l' equivalence partitioning, descriverne lo scopo e il procedimento di
applicazione. Dire quali sono le classi di equivalenza da creare per un modulo che effettui una ricerca binaria.

Risposte

Risposta n.1
Gli indici di qualità del software più rilevanti per le attività di manutenzione perfettiva e adattativa sono:

- **portabilità**: sforzo richiesto per trasferire un prodotto da un ambiente hardware e/o software ad un altro
- **riutilizzabilità**:misura in cui un prodotto (o parti di esso) può essere riutilizzato in altre applicazioni
- **interoperabilità**: sforzo necessario per accoppiare un prodotto con un altro
- **evolubilità**:sforzo necessario per aggiornare il prodotto al fine di soddfisfare nuovi requisiti

Risposta n.2
L'indice di qualità flessibilità è lo sforzo necessario per modificare un prodotto operativo. I suoi attributi sono:

- consistenza +
- tracciabilità +
- modularità +
- generalità +
- auto documentazione +
- complessità -

Il valore dell'indice di qualità di flessibilità viene calcolato, come quello degli altri indici, utilizzando i risultati del calcolo del punteggio dei suoi attributi (calcolato attraverso la checklist) e l'impatto di ogni attributo sull'indice (che può essere positivo o negativo).
Ovvero attraverso la seguente fomula,

$$V_{index} = \frac{\sum V_{attribute(+)} + \sum \left(1 - V_{attribute(-)}\right)}{number\ of\ attr.\ associated\ to\ index}$$

dove $V_{attribute(+)}$ e $V_{attribute(-)}$ sono i punteggi degli attributi che incidono positivamente e negativamente sull'indice, rispettivamente.
V_{index} è un valore compreso tra 0 e 1.
La tabella seguente identifica i valori di soglia per l'accettazione della qualità dell'indice.

V_{index}	*Quality Level*
$0.66 < V \leq 1$	High
$0.33 < V \leq 0.66$	Medium
$0 < V \leq 0.33$	Low

Risposte

Risposta n.3

La checklist è un metodo per valutare la qualità di un attributo. É composto da diverse domande, e per ogni domanda sono previste quattro risposte con un dato valore.
Il team di valutazione della checklist è composto da almeno quattro persone con competenze diverse in modo che possano essere confrontati vari punti di vista e la valutazione sia più precisa.

Nel caso dell'attributo di complessità la checklist si propone di verificare che il progetto del sistema sia adeguatamente strutturato e facilmente comprensibile da parte di persone senza una conoscenza specifica del sistema, utilizzando pochi formalismi e adeguandosi a degli standard.
Il calcolo del punteggio di complessità viene eseguito attraverso la seguente formula:

$$V_{complexity} = \frac{\sum_{i=1}^{\#questions} V_{answer_i}}{\sum_{i=1}^{\#questions} \max(V_{answer_i})}$$

dove V_{answer_i} è il valore dato alla risposta scelta per la domanda *i* e $\max(V_{answer\_i})$ è il suo massimo valore.

Per l'attributo di modularità invece la checklist serve a verificare che il sistema abbia un'alta coesione e un basso coupling, e che la scomposizione del sistema sia organizzata in modo gerarchico.

Si calcola quindi la qualità della coesione e quella del coupling (V_{anwer1} e $V_{answer2}$) e infine viene calcolata la qualità della modularità tenendo conto anche delle risposte alle altre domande della checklist (nell'immagine sotto V_{anwer3} e $V_{answer4}$)

$$V_{Modularity} = \frac{V_{answer_1} + V_{answer_2} + V_{answer_3} + V_{answer4}}{\max(V_{answer_1}) + \max(V_{answer_2}) + \max(V_{answer_3}) + \max(V_{answer_4})}$$

Risposta n.4

Il progetto architetturale (o preliminare) è la fase in cui viene sviluppata l'architettura del prodotto software.

I vari metodi di progetto preliminare sono:
• Metodo FGA (Flow Graph Analysis)
• Metodo DFA (Data Flow Analysis)
• Metodo TA (Transaction Analysis)
• Metodi Data-Oriented (Jackson, Warnier, Orr)
• Metodi Object Oriented (OOD)

Questi metodi si propongono di identificare, attraverso degli step di raffinamento, i moduli (a un determinato livello di coesione) che entrano nella Structure Chart del prodotto software, cioè di ottenere un'architettura del sistema precisa e dettagliata.

Risposte

Risposta n.5

Il metodo **TA** è un metodo del progetto preliminare che prende in input un Data Flow Diagram di primo livello ottenuto in fase di analisi dei requisiti e procede in step di refinement fino identificare i moduli che entrano nella Structure Chart del prodotto software.
TA procede con il refinement fino a raggiungere un livello di coesione funzionale.

Risposta n.6

La coesione e il coupling sono sotto-attributi dell'attributo modularità, dove

- Coesione: Misura in cui il modulo espleta internamente tutte le azioni necessarie a eseguire una data funzione (cioè senza interagire con le azioni interne ad altri moduli)
- Coupling: Misura il grado di interdipendenza tra i moduli di un prodotto sw

Si calcola quindi la qualità della coesione e quella del coupling nel seguente modo (dove V_{anwer1} e $V_{answer2}$ sono il valore della coesione e dell'accoppiamento):

$$V_{answer_1} = \frac{(0 \times \%1A) + (1 \times \%1B) + (2 \times \%1C) + (3 \times \%1D)}{50}$$

$$V_{answer_2} = \frac{(0 \times \%2A) + (1 \times \%2B) + (2 \times \%2C) + (3 \times \%2D)}{50}$$

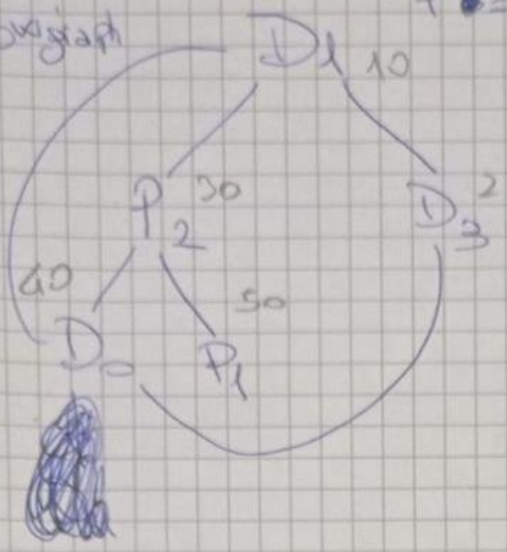
dove %X è la percentuale di moduli o coppie di moduli di tipo X rispetto a l'inteso insieme di moduli/coppie di moduli del sistema.

Risposte

Risposta n.7

$F = D_1((D_0(D_1); P_1), D_3(D_0))$

- Flowgraph



- Pseudo-codice

```
10  IF A THEN 30
20  GO TO 40
30  IF B THEN 50
40  GO TO 10
50  END
```

- Depth of Nesting

$a(F) = a(D_1((D_0(D_1); P_1), D_3(D_0))) =$

$= 1 + \max\{a(D_0(D_1); P_1), a(D_3(D_0))\} =$

$= 1 + \max\{\max\{a(D_0(D_1)), a(P_1)\}, 1 + a(D_0)\}$

$= 1 + \max\{\max\{1 + a(D_1), a(P_1)\}, 1 + a(D_0)\}$

$= 1 + \max\{\max\{2, 0\}, 2\} =$

$= 1 + \max\{2, 2\} = 1 + 2 = 3$

- D-structured

$\partial(F) = \partial(D_1((D_0(D_1); P_1), D_3(D_0))) =$

$= \min\{\partial(D_1), \partial(D_0(D_1); P_1), \partial(D_3(D_0))\} =$

$= \min\{\partial(D_1), \min\{\partial(D_0(D_1)), \partial(P_1)\}, \min\{\partial(D_3), \partial(D_0)\}\} =$

$= \min\{\partial(D_1), \min\{\min\{\partial(D_0), \partial(D_1)\}, \partial(P_1)\}, \min\{\partial(D_3), \partial($

$= \min\{1, \min\{1, 1\}, \min\{1, 1\}\} =$

$= \min\{1, 1, 1\} = 1$

- Cyclomatic Complexity e Essential Cyclomatic Complexity

$cc(F) = 6 - 5 + 2 = 3$

$Ecc(F) = 3 - 1 = 2$

Risposte

Risposta n.8

Lo Static Verification Testing (**SVT**) verifica in fase preliminare la modularità del sistema:

– **Coesione**
– **Coupling**
– **Morfologia**
– **Flusso informativo**

Nella progettazione dettagliata il **SVT** verifica la:

– **Strutturazione del modulo**:
- Depth of nesting
- D-strucuredness
- Cyclomatic complexity

– **Correttezza del modulo**:
- Prova formale di correttezza (assertion method)

Code SVT:

• **Analisi del flusso di controllo:** Verifica la presenza di loop con più punti di uscita o di ingresso, trova codice irraggiungibile, ecc.

• **Analisi dell'utilizzo dei dati:** Rileva variabili non inizializzate, variabili assegnate due volte senza un'assegnazione intermedia, variabili dichiarate ma mai utilizzate, ecc.

• **Analisi dell'interfaccia:** Controlla la consistenza delle dichiarazioni di routine e di procedura e il loro uso

• **Analisi del flusso informativo:** Identifica le dipendenze delle variabili di output. Non rileva le anomalie in sé, ma evidenzia le informazioni per l'ispezione o la revisione del codice

• **Analisi del percorso:** Identifica i percorsi attraverso il programma e stabilisce le istruzioni eseguite in quel percorso.

• **Ispezione del codice:** Particolarmente utile quando viene utilizzato un linguaggio come C che ha una tipizzazione debole e quindi molti difetti non vengono rilevati dal compilatore

Risposta n.9

Lo scopo del Dynamic Validation Testing (TVP) è dimostrare che il sistema soddisfa i requisiti utente (cioè non si discosta dai requisiti).

Un validation test riuscito richiede l'esecuzione del sistema utilizzando correttamente determinati casi di test di accettazione.

Il processo di Dynamic Validation Test consta nell'esecuzione dei seguenti tipi di test:

- **Unit testing**: vengono testati i singoli componenti
- **Module testing**: vengono testati gli insiemi di componenti dipendenti
- **Sub-system testing**: i moduli sono integrati in sottosistemi e testati
- **System testing**: il sistema viene testato nel suo complesso
- **Acceptance testing**: test con i dati del cliente per verificare che sia accettabile

Questi test vengono generalmente eseguiti da persone diverse, in particolare:

- **Component testing**: unit e module testing eseguiti dallo sviluppatore
- **Integration testing**: sub-system e system testing eseguiti dal un team di testing indipendente
- **User testing**: acceptance testing eseguito dall'utente

Il Dynamic Validation testing può essere eseguito con 2 diverse strategie:

- **Incrementale:** vengono testati i singoli componenti e poi la loro interazione

- **Non Incrementale**

Risposte

Risposta n.10

Lo scopo del Dynamic Validation Testing (TVP) è dimostrare che il sistema soddisfa i requisiti utente (cioè non si discosta dai requisiti).
Il processo di Dynamic Validation Test consta nell'esecuzione dei seguenti tipi di test:

- **Unit testing**: vengono testati i singoli componenti
- **Module testing**: vengono testati gli insiemi di componenti dipendenti
- **Sub-system testing**: i moduli sono integrati in sottosistemi e testati
- **System testing**: il sistema viene testato nel suo complesso
- **Acceptance testing**: test con i dati del cliente per verificare che sia accettabile

Ognuna di queste fasi di test viene pianificata:

- **Acceptance test plan**: viene definita attraverso la specifica dei requisiti e la specifica del sistema
- **System integration test plan**: viene definita attraverso la specifica del sistema e la progettazione del sistema
- **Sub-system integration test plan**: viene definita attraverso la progettazione preliminare e la progettazione dettagliata del sistema
- **Unite and Module test plan**: viene definita attraverso la progettazione dettagliata del sistema

Ci sono vari tipi di DVT:

- **Top-down testing**: Inizia con un sistema di alto livello e si integra dall'alto verso il basso, sostituendo i componenti individuali mediante stub dove è necessario. Si applica al test di unità e moduli.
- **Bottom-up testing**: Integra i singoli componenti nei livelli fino alla creazione del sistema completo.Si applica al test di unità e moduli.
- **Thread testing**: Testa i vari percorsi del prodotto software. Si applica al test di sottosistema (dopo che i moduli sono stati testati individualmente).
- **Stress testing**: stressa il sistema per far emergere dei difetti, cioè aumenta il carico di lavoro finché il sistema non fallisce. Si applica al test di sistema.
- **Back-to-back testing**: esegue la vecchia versione del software e la nuova e poi confronta i risultati per vedere se ci sono delle differenze significative. Si applica al test di sistema.
-

Risposta n.11

L'**equivalence partitioning** si applica la **Black-box testing**, chiamato anche **functional testing;** è un approccio dove il programma è considerato come una scatola nera.
In altre parole non viene presa in considerazione alcuna informazione rispetto al funzionamento interno dell'applicazione.
Il tester presenta gli input al componente o al sistema ed esamina il corrispondente output. Se l'output non è quello specificato allora il test ha trovato un problema.
L'**equivalence partitioning** è una tecnica di test che divide i dati di input di un'unità software in classi di equivalenza, in cui il programma si comporta in un modo equivalente per ogni membro della classe, da cui è possibile derivare casi di test.
Per un modulo che effettui la ricerca binaria dobbiamo dare al sistema i seguenti input:

- input conformi alla pre-condizione
- input non conformi alla pre-condizione
- input dove l'elemento chiave è nell'array

- input in cui l'elemento chiave non è nell'array

dove la pre-condizione è l’array ha almeno un elemento

Le **classi di equivalenza** saranno:

- Pre-condizioni soddisfatte, elemento chiave nell’array
- Precondizioni soddisfatte, elemento chiave non nell’array
- Precondizioni insoddisfatte, elemento chiave nell’array
- Precondizioni insoddisfatte, elemento chiave non nell’array
- L'array di input ha un solo valore
- L'array di input ha un numero pari di valori
- L'array di input ha un numero dispari di valori

Concluso il 17/04/2023 alle ore 16:20