A cura di Karina Chichifoi - Ultima revisione: 29 giugno 2020

Ingegneria del Software Q&A

https://github.com/TryKatChup/IngegneriaSoftware_QA

Premessa

Ho scritto questo file in modo da facilitare lo studio e il superamento dell'esame di Ingegneria del Software. Tuttavia è consigliato integrare questo materiale con le slide del professore Marco Patella, disponibili sulla piattaforma *Insegnamenti Online*.

Contribuire alla guida

Se ritieni di poter migliorare la guida, oppure se sono state aggiunte altre domande al di fuori di questo file, o se hai trovato un errore, visita la repository GitHub ed apri una *issue*, oppure inviami un messaggio. Ogni contributo è ben accetto :)

Link Repository: https://github.com/TryKatChup/IngegneriaSoftware_QA

Figura 1: QR Code alla repository di GitHub

1 Modulo 1

Domanda 1.1
Come viene implementata l'ereditarietà multipla?

Risposta:

Domanda 1.2
Si esegua una classificazione del polimorfismo secondo Cardelli-Wegner e si mostri l'implementazione del polimorfismo per inclusione.

Risposta:

Domanda 1.3
Procedimento di compilazione ed esecuzione del codice all'interno del framework .NET tramite il CLR.

Risposta:

Domanda 1.4
Tipi di dati in .NET

Risposta:

Domanda 1.5
Differenza tra tipi valore e tipi riferimento in .NET

Risposta:

Domanda 1.6
Garbage Collector in C#

Risposta:

Domanda 1.7
Passaggio dei parametri in C#

Risposta:

Domanda 1.8
Concetto di delegato in C#

Risposta:

Domanda 1.9
Metaprogrammazione e riflessione in C#

Risposta:

Domanda 1.10
Spiegare i quattro bad design (fragilità, immobilità, rigidità, viscosità)

Risposta:

Domanda 1.11
Principio di singola responsabilità con almeno un esempio

Risposta:

Domanda 1.12
Principio di inversione delle dipendenze con almeno un esempio

Risposta:

Domanda 1.13
Principio di segregazione delle interfacce con almeno un esempio

Risposta:

Domanda 1.14
Principio aperto/chiuso con almeno un esempio

Risposta:

Domanda 1.15
Principio di sostituibilità di Liskov con almeno un esempio

Risposta:

Domanda 1.16
Principi per l'architettura dei package

Risposta:

Domanda 1.17
Pattern Singleton con esempi

Risposta:

Domanda 1.18
Pattern Observer con esempi

Risposta:

Domanda 1.19
Pattern Strategy con esempi

Risposta:

Domanda 1.20
Pattern Adapter con esempi

Risposta:

Domanda 1.21
Pattern Decorator con esempi

Risposta:

Domanda 1.22
Pattern Composite con esempi

Risposta:

Domanda 1.23
Modello LMU nei VCS con vantaggi e svantaggi

Risposta:

2 Modulo 2

Domanda 2.1
Spiegare il modello a cascata e le sue criticità.

Risposta: Il modello a cascata (waterfall model) è un modello di processo di sviluppo software che prevede fasi sequenziali distinte tra loro:

- studio di fattibilità
- analisi dei requisiti
- analisi del problema
- progettazione
- implementazione
- collaudo
- manutenzione

Ciascuna fase di sviluppo deve essere svolta in maniera esaustiva, prima di passare alla successiva, in modo da non tornare più indietro. Per questo modello è importante definire:

- **semilavorati**: consistono in documentazione di tipo cartaceo, codice dei singoli moduli, sistema nel suo complesso.
  Vengono prodotti alla fine di una fase, e utilizzati dalla fase successiva; in questo modo viene garantito un controllo della qualità del lavoro eseguito in ogni fase.
- **date**: stabiliscono una scadenza entro la quale devono essere prodotti i semilavorati, in modo da tracciare il progresso del lavoro (workflow).

L'efficacia del modello a cascata è determinata dai seguenti fattori:

- **immutabilità dell'analisi**: i clienti sono in grado di esprimere tutte le loro richieste sin da subito, pertanto nella fase iniziale del progetto si possono definire tutte le funzionalità che il software deve eseguire
- **immutabilità del progetto**: progettare l'intero sistema prima di avere scritto codice risulta possibile

Un importante vantaggio di questo approccio risulta essere un maggiore controllo dell'andamento del progetto; tuttavia la rigidità di questo modello rappresenta un grosso svantaggio, in quanto:

- Man mano che il sistema prende forma le sue specifiche cambiano in continuazione, così come la visione che i clienti hanno del sistema
- Spesso, per avere prestazioni migliori, occorre revisionare il progetto.

Per risolvere parzialmente i problemi sopra citati si è introdotto un modello a cascata con forme limitate di retroazione a un livello. Una possibile soluzione al problema consiste nel realizzare un prototipo che, una volta terminato il compito, viene abbandonato (*throw-away prototyping*); successivamente viene costruito il sistema reale rispettando il modello a cascata.
Quest'ultimo approccio risulta talmente dispensioso da eliminare i vantaggi economici del modello a cascata.

Domanda 2.2
Spiegare il modello a cascata e il modello iterativo

Risposta: Per il modello a cascata si veda la domanda 2.1.
Il modello iterativo prevede un numero elevato di passi nel ciclo di sviluppo che iteramente aumentano il livello di dettaglio del sistema. Uno svantaggio di questo modello è che non può essere utilizzato nella realizzazione dei progetti significativi. Un esempio di processo di sviluppo che utilizza il modello iterativo è *Rational Unified Process* (RUP).

Domanda 2.3
Illustrare RUP

Risposta: Il *Rational Unified Process* (RUP) rappresenta un modello di processo software **iterativo** (si veda domanda 2.2) e **ibrido** (contiene elementi di tutti i modelli di processo generici) pensato per software di grandi dimensioni.
Esistono tre aspetti importanti del processo di sviluppo:

- **Prospettiva dinamica**: mostra l'evoluzione del modello nel tempo. È composta da 4 fasi:

  1. **Avvio**: lo scopo di questa fase è di delineare il *business case*, ovvero comprendere il tipo di mercato a cui si rivolge, le entità esterne (persone e sistemi) che interagiscono con il sistema.
     Durante la fase di avvio si utilizzano modelli di caso d'uso e si effettua una valutazione dei rischi.
  2. **Elaborazione**: questa fase definisce la struttura complessiva del sistema; comprende l'analisi del dominio e una prima fase di progettazione dell'architettura.
     Occorre soddisfase alcuni criteri, tra i quali:
     - Modello dei casi d'uso completo all' 80%
     - Descrizione dell'architettura del sistema
     - Sviluppo dell'architettura del sistema
     - Sviluppo di un'architettura eseguibile adatta agli use case significativi
     - Revisione dei business case e dei rischi
     - Pianificazione del progetto complessivo

     Nota bene: al termine di questa fase modificare il progetto risulterà più difficile e dannoso.
  3. **Costruzione**: durante questa fase avviene la progettazione, la programmazione e il collaudo del sistema. Lo sviluppo delle diverse parti del sistema avviene in parallelo; successivamente vengono integrate.
     Al termine di questa fase il sistema software dovrebbe essere funzionante e la relativa documentazione dovrebbe risultare pronta.
  4. **Transizione**: il sistema passa dall'ambiente di sviluppo a quello dell'utente finale. Quest'ultimo viene istruito nell'utilizzo del sistema, e si effettua *beta testing* del sistema a scopo di verifica e validazione.

- **Prospettiva statica**: si focalizza sulle attività di produzione del software, note come *workflow*; la descrizione di questi ultimi è orientata ai modelli associati a UML. Esistono sei workflow principali:

  1. **Modellazione delle attività aziendali**: i processi aziendali vengono modellati, sfruttando il *business case*
  2. **Requisiti**: vengono sviluppati i casi d'uso per la stesura dei requisiti; avviene l'identificazione degli attori che interagiscono con il sistema
  3. **Analisi e progetto**: attraverso l'utilizzo dei modelli architetturali e sequenziali degli oggetti e delle componenti viene creato e documentato un *modello di progetto*
  4. **Implementazione**: i componenti vengono implementati; grazie alla generazione automatica del codice a partire dai modelli precedentemente definiti
  5. **Test**: vengono testati i sottocomponenti e il sistema finale
  6. **Rilascio**: il prodotto viene distribuito agli utenti

  Oltre ai 6 workflow principali vengono definiti 3 workflow di supporto:

  1. **Gestione della configurazione e delle modifiche**: gestisce i cambiamenti del sistema
  2. **Gestione del progetto**: gestisce lo sviluppo del sistema
  3. **Ambiente**: fornisce agli sviluppatori degli strumenti adeguati

- **Prospettiva pratica**: suggerisce le *buone prassi* da seguire nello sviluppo dei sistemi.
  Esistono sei fasi fondamentali:

  1. **Sviluppare ciclicamente il software**: pianificare (??) e consegnare le funzioni aventi la priorità più alta
  2. **Gestire i requisiti**: documentare ogni richiesta esplicita del cliente e ogni cambiamento effettuato, analizzandone l'impatto
  3. **Usare architetture basate sui componenti**: strutturare l'architettura del sistema in più componenti
  4. **Usare modelli visivi del software**: utilizzare grafici UML per la rappresentazione statica e dinamica del software
  5. **Verificare la qualità del software**: assicurarsi che vengano raggiunti gli standard di qualità previsti dall'organizzazione
  6. **Controllare le modifiche del software**: utilizzare strumenti e pratiche che permettono di gestire modifiche al software

Domanda 2.4
Tipologie di analisi dei requisiti (in particolare quelli della sicurezza)

Risposta:

Domanda 2.5
Si illustri brevemente il ciclo di vita della valutazione del rischio

Risposta:

Domanda 2.6
Principali categorie di requisiti per la sicurezza

Risposta:

Domanda 2.7
Commentare eventuali errori di un diagramma UML

Risposta:

Domanda 2.8
Linee guida di progettazione nella sicurezza

Risposta:

Domanda 2.9
White box e black box testing

Risposta:

Domanda 2.10
Capacità di sopravvivenza del sistema

Risposta: