PER ALTRI APPUNTI CONSULTARE IL SITO:

https://luigi-v.github.io/Appunti_Universita/

Per altri appunti consultare il sito: *https://luigi-v.github.io/Appunti_Universita/*

By *Vollono Luigi*

1. Descrivere i concetti di molteplicità in attributi e associazioni tra classi in UML:

La *molteplicità di associazione* indica il numero min e max di istanze che sono coinvolte nell'associazione.
La *molteplicità degli attributi* descrive il num min e max dei valori dell'attributo associati ad ogni istanza dell'entità.

2. Descrivere le categorie di requisiti non funzionali (requisiti di qualità FURPS+ e pseudo-requisiti)

Requisiti di qualità:

- *Usabilità*: estetica e consistenza dell'interfaccia utente, quanto il sistema è user-friendly.
- *Affidabilità*: capacità del sistema di fornire la funzione richiesta sotto certe condizioni per un periodo di tempo.
- *Performance*: riguardano attributi come tempo di risposta(quanto velocemente il sistema reagisce ad un input), throughput (quanto lavoro il sistema riesce a realizzare entro un tempo specificato), tempo di recupero, tempo di start-up e tempo di shutdown.
- *Supportabilità*: capacità del sistema di supportare i cambiamenti dopo essere stato sviluppato.

Pseudo-requisiti:

- *implementazione*: vincoli per la realizzazione del sistema, compreso l'uso di attrezzi specifici, linguaggi di programmazione o piattaforme hardware.
- *interfaccia*: vincoli imposti da sistemi esterni, incluso sistemi legacy e formati di scambio.
- *operazione*: sono vincoli sulla amministrazione e gestione del sistema.
- l'imballaggio(*Packaging*): vincoli sulla consegna effettiva del sistema (vincoli sui mezzi di installazione).
- *legge*: riguardano le leggi sulle licenze, regolamentazione e certificazione.

3. Descrivere la differenza tra oggetti Entity, Boundary, Control

- *Entity Object*: dati persistenti tracciati dal sistema
- *Boundary Object*: oggetti che permettono l'interazione tra utente e sistema
- *Control Object*: oggetti che si occupano di gestire la logica del sistema

4. Descrivere i criteri per la convalida dei requisiti

- *Correttezza*: il sistema è corretto se rappresenta il punto di vista del cliente.
- *Consistenza*: non ci sono contraddizioni tra i requisiti funzionali e non.
- *Non ambiguità*: è descritto esattamente un solo sistema, non interpretabile in modo diverso.
- *Completezza*: i possibili scenari del sistema sono descritti.
- *Realistica*: se il sistema può essere implementato in tempi ragionevoli.

Altre due caratteristiche:

- *Verificabilità*: la specifica dei requisiti è verificabile se dopo la sua costruzione, test ripetuti possono dimostrare che il sistema soddisfi i requisiti.
- *Tracciabilità*: ogni funzione del sistema può essere individuata e ricondotta al corrispondente insieme di requisiti.

5. Descrivere la differenza tra le relazioni "include" e "extend" in un diagramma dei casi d'uso

- *"include"* è una decomposizione funzionale di un caso d'uso in due o più casi d'uso semplificati, si utilizza per includere un flusso di eventi all'interno di un caso d'uso oppure per semplificare un caso d'uso complesso.
- *"extend"* è una variante del normale flusso di eventi, portato fuori dal flusso degli eventi principali per rendere più chiaro il caso d'uso.

6. Descrivere la differenza tra il modello di processo incrementale ed il modello di processo evolutivo.

- *Sviluppo incrementale*: utilizzato per la progettazione di grandi software che richiedono tempi ristretti, vengono rilasciate delle release funzionanti (deliverables) e ad ogni versione aggiunge nuove funzionalità/sottosistemi.
- *Sviluppo evolutivo*: Si iniziano a sviluppare le parti del sistema che sono già ben specificate aggiungendo nuove caratteristiche secondo le necessità fornite dal cliente man mano.

7. Descrivere le caratteristiche dei modelli di processo iterativi e incrementali

Entrambi i modelli prevedono più versioni successive del sistema. Dopo il primo rilascio, esiste una versione del sistema e una nuova in sviluppo.

- *Modello incrementale*:

Le fasi alte del processo sono completamente realizzate. Il sistema viene decomposto in sottosistemi(incrementi) che vengono implementati, testati, rilasciati, installati e messi in manutenzione secondo un piano di priorità in tempi diversi. Diventa fondamentale la fase di integrazione di nuovi sottosistemi con quelli già in esercizio.

- *Modello iterativo*:

da subito sono presenti tutte le funzionalità/sottosistemi che vengono successivamente raffinate e migliorate.

8. Descrivere i messaggi, attivazione e lifelines in un diagramma di sequenza

- *Messaggio*: vengono inviati da un oggetto all'altro, determina l'attivazione di un operazione.
- *Attivazione*: tempo in cui un oggetto è impegnato in una determinata operazione
- *Lifelines*: rappresentano ruoli oppure instanze di oggetti che partecipano alla sequenza che viene modellata.

9. Descrivere i concetti di ruolo e qualificatori nelle associazioni tra classi in UML

Un *ruolo* fornisce una modalità per attraversare relazioni da una classe ad un'altra. Possono essere usati in alternativa ai nomi delle associazioni. Sono spesso usati per relazioni tra oggetti della stessa classe (associazioni riflessive).
Un *qualificatore* è un attributo (o insieme di attributi) il cui valore partiziona un insieme di oggetti (detti targets) associati ad un altro oggetto (detto source).

10. Spiegare in che modo la prototipazione può supportare la raccolta e la convalida dei requisiti

La realizzazione di un prototipo del sistema o di un sottosistema è un mezzo attraverso il quale si interagisce con il committente per valutare e identificare meglio le specifiche richieste e per verificare la fattibilità del prodotto.
Il prototipo nella maggior parte dei casi va buttato via dopo essere stato valutato da parte del cliente.

11. Quali sono le principali difficoltà che si incontrano durante la raccolta dei requisiti?

Le difficoltà principali sono legate alla comunicazione. Fraintendimenti e omissioni possono portare al fallimento dello sviluppo. Oppure una funzionalità che il sistema dovrebbe supportare non è specificata, funzionalità incerte o obsolete, interfaccia utente poco intuitiva e difficile da usare.

12. Descrivere il concetto di transizione in un diagramma di stato UML

Le transizioni sono il passaggio da uno stato all'altro il quale avviene solo se si soddisfa un evento o una condizione a cui segue un'azione.

13. Spiegare da quali parti è composto un documento di analisi e specifica dei requisiti

Il RAD descrive completamente il sistema in termini di requisiti funzionali e non funzionali e serve come base del contratto tra cliente e sviluppatori.
Il RAD è composto da:

- *Introduzione*
- *Sistema* Corrente
- *Sistema* proposto
- *Glossario*

14. Cos'è un percorso critico in un diagramma di PERT e qual è la sua importanza?

Il percorso critico è l'insieme di archi che collega il task che non dipende da altri task (non ci sono archi entranti) con un task da cui nessuno dipende (non ci sono archi uscenti). Il percorso è scelto in base alla durata dei task, si scelgono i task con durata maggiore.
È importante perché il ritardo su uno dei task nel percorso, può ritardare di molto tutti gli altri task e quindi tutto il sistema.