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

La molteplicità di una associazione denota quanti oggetti il sorgente può referenziare legittimamente, cioè descrive il numero min e max di istanze di associazioni a cui un'istanza dell'entità può partecipare. La molteciplità degli attributi descrive il num min e max dei valori dell'attributo associati ad ogni istanza dell'entità.

2) Descrivere le categorie dei requisiti non funzionali secondo il modello FURPS+

• Requisiti di qualità

- Usabilità : riguarda caratteristiche come l'estetica e la consistenza dell'interfaccia utente, la facilità ad imparare il sistema ed interpretarne gli output.
- Affidabilità : capacità del sistema o di una componente di fornire la funzione richiesta sotto certe condizioni per un periodo di tempo. Esempio:tempo medio prima di fallire accettabile, abilità nello scoprire specifici difetti, capacità di supportare attacchi alla sicurezza.
    - Robustezza: il grado in cui un sistema o una componente può funzionare correttamente in presenza di input non validi o condizioni di stress.
    - Safety: una misura di conseguenze catastrofiche sull'ambiente.
- Performance : riguardano attributi quantificabili del sistema 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 : riguarda caratteristiche come la testabilità, adattabilità, manutenibilità,
  compatibilità, configurabilità, installabilità, scalabilità e localizzazione.

3) Descrivere la differenza tra oggetti Entità, Boundary e Control

- Gli oggetti Entity rappresentano l'informazione persistente tracciata dal Sistema
- Gli oggetti Boundary rappresentano le interazioni tra gli attori e il Sistema quindi un'interfaccia tra attori e sistema
- Gli oggetti Control rappresentano i compiti di controllo svolti dal Sistema , e sono responsabili del coordinamento degli oggetti boundary e entità.

4) Descrivere i criteri per la convalida dei requisititi

• Correttezza: il sistema è corretto se rappresenta il punto di vista del cliente

• Completezza: tutti i possibili scenari del sistema sono descritti, inclusi quelli di comportamenti non comuni.

• Consistente: la specifica del sistema non contraddice se stessa.

• Disambigua: è descritto esattamente uno e un solo sistema (non è possibile interpretare la
specifica in modi differenti)

• Realistica: se il sistema può essere implementato tenendo conto dei vincoli.

Altre due caratteristiche desiderabili in una specifica di sistema sono:

• Verificabilità: la specifica dei requisiti è verificabile se una volta che il sistema è stato costruito, test ripetuti possono essere delineati per 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

• Extend: presenta casi eccezionali o rari. Il flusso degli eventi eccezionali è portato fuori dal flusso degli eventi principali per rendere più chiaro il caso d'uso.

• Include: consente di rappresentare un comportamento che si vuole portare al di fuori del caso d'uso per motivi di riuso oppure perché una funzione del problema originale troppo complessa da risolvere immediatamente, quindi viene decomposta.
. Include è utile per ridurre ridondanze e di conseguenza possibili inconsistenze.

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

Il modello incrementale è utilizzato per la progettazione di grandi software che richiedono tempi ristretti. Vengono rilasciate delle release funzionanti (***deliverables***) anche se non soddisfano pienamente i requisiti del cliente.

Vi sono due tipi di modelli incrementali:

Modello ad implementazione incrementale

Le fasi alte del modello a cascata vengono realizzate e portate a termine, il software viene finito, testato e rilasciato ma non soddisfa tutte le aspettative richieste dall'utente(mancano alcuni sottosistemi che implementano funzionalità meno rilevanti). Le funzionalità non incluse vengono comunque implementate e aggiunte in tempi diversi in base alla loro priorità. Con questo tipo di modello diventa fondamentale la parte di integrazione tra sottosistemi.

Modelli Iterativi (a sviluppo e consegna incrementale)

E' un particolare modello a cascata in cui ad ogni fase viene applicato il modello ad implementazione incrementale e successivamente le singole fasi vengono sviluppate e integrate con il sistema esistente. Il sistema viene sviluppato seguendo le normali fasi e ad ogni fase l'output viene consegnato al cliente.

Il modello evolutivo si basa su due tipologie di sviluppo basate sui prototipi:

Prototipazione evolutiva (Sviluppo Esplorativo)

Si iniziano a sviluppare le parti del sistema che sono già ben specificate aggiungendo nuove caratteristiche secondo le necessità fornite dal cliente man mano. Consente di scrivere il software in maniera incrementale, Il prodotto software evolve continuamente.

Prototipo usa e getta (Throw-Away)

Lo scopo di questo tipo di prototipazione è quello di identificare meglio le specifiche richieste dall'utente sviluppando dei prototipi che sono funzionanti. Il prototipo sperimenta le parti del sistema che non sono ancora ben comprese oppure serve per valutare la fattibilità di un approccio.

Il prototipo è un mezzo attraverso il quale si interagisce con il committente per accertarsi di aver ben compreso le sue richieste, per specificare meglio tali richieste, per valutare la fattibilità del prodotto.

Non appena il prototipo è stato verificato da parte del cliente o da parte degli sviluppatori può essere buttato via.

7) Descrivere le categorie di requisiti non funzionali (requisiti di qualità e pseudo – requisiti)

- **Requisiti di qualità**
  - Usabilità : riguarda caratteristiche come l'estetica e la consistenza dell'interfaccia utente, la facilità ad imparare il sistema ed interpretarne gli output.

- Affidabilità : capacità del sistema o di una componente di fornire la funzione richiesta sotto certe condizioni per un periodo di tempo. Esempio:tempo medio prima di fallire accettabile, abilità nello scoprire specifici difetti, capacità di supportare attacchi alla sicurezza.
    - Robustezza: il grado in cui un sistema o una componente può funzionare correttamente in presenza di input non validi o condizioni di stress.
    - Safety: una misura di conseguenze catastrofiche sull'ambiente.
- Performance : riguardano attributi quantificabili del sistema 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 : riguarda caratteristiche come la testabilità, adattabilità, manutenibilità,
  compatibilità, configurabilità, installabilità, scalabilità e localizzazione.

Vincoli o pseudoRequisiti:

- Implementazione : vincoli sull'implementazione del sistema, incluso l'uso di tool specifici, linguaggi di programmazione, o piattaforme hardware
- Interfacce : vincoli imposti da sistemi esterni, incluso sistemi legacy e formati di scambio
- Operazioni : vincoli sull'amministrazione e sulla gestione del sistema
- Packaging : vincoli sulla consegna del sistema (vincoli sui mezzi di installazione)
- Legali : riguardano licenze, regolamentazioni, e certificazioni.

8) Descrivere le caratteristiche dei modelli di processo iterativi e incrementali

Entrambi i modelli prevedono più versioni successive del sistema. Ad ogni istante, dopo il primo rilascio, esiste una versione del sistema N in esercizio e una versione N+1 in sviluppo.

- **Modello ad implementazione incrementale**
  Le fasi alte del modello a cascata vengono realizzate e portate a termine, il software viene finito, testato e rilasciato ma non soddisfa tutte le aspettative richieste dall'utente(mancano alcuni sottosistemi che implementano funzionalità meno rilevanti). Le funzionalità non incluse vengono comunque implementate e aggiunte in tempi diversi in base alla loro priorità. Con questo tipo di modello diventa fondamentale la parte di integrazione tra sottosistemi.
- **Modelli Iterativi (a sviluppo e consegna incrementale)**
  E' un particolare modello a cascata in cui ad ogni fase viene applicato il modello ad implementazione incrementale e successivamente le singole fasi vengono sviluppate e integrate con il sistema esistente. Il sistema viene sviluppato seguendo le normali fasi e ad ogni fase l'output viene consegnato al cliente.

9) Descrivere i messaggi, attivazione e lifelines in un diagramma di sequenza

• Messaggi: sono rappresentati da frecce, vengono inviati da un oggetto all'altro. La ricezione di un messaggio determina l'attivazione di un operazione. Possono essere sincroni, asincroni, di risposta e flusso di controllo.
Attivazione: è rappresentata da un rettangolo da cui altri messaggi possono prendere origine.
Lifelines: sono rappresentate da dei box con delle linee tratteggiate scendenti dal centro. Rappresentano ruoli oppure instanze di oggetti che partecipano alla sequenza che viene modellata.

10)Descrivere i concetti di ruolo e qualificatori nelle associazioni tra classi in UML

• Ruolo: I ruoli forniscono una modalità per attraversare relazioni da una classe ad un'altra
• I nomi di ruolo possono essere usati in alternativa ai nomi delle associazioni
◆ I ruoli sono spesso usati per relazioni tra oggetti della stessa classe (*associazioni riflessive*)

Qualificatore: 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*).
• Il qualificatore è attaccato alla classe *source* di una associazione e determina come gli oggetti della classe *target* sono partizionati e identificati

11) 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.

13)Descrivere la differenza tra aggregazione e composizione in un diagramma delle classi UML

La relazione di aggregazione è un'associazione speciale che aggrega gli oggetti di una classe componente come un unico oggetto della classe composta
• la si puo' leggere come "è composto da" in un verso e "è parte di" nell'altro verso

Una relazione di composizione è un'aggregazione forte
• Le parti componenti non esistono senza il contenitore
• Ciascuna parte componente ha la stessa durata di vita del contenitore

15)Quali sono le principali difficoltà che si incontrano durante la raccolta dei requisiti?

Clienti utenti e sviluppatori comunicano tra di loro per definire il sistema da realizzare. Fallimenti nella comunicazione portano a un sistema difficile da usare o che non fornisce le funzionalità richieste. Gli errori commessi in questa fase sono difficili da risolvere se scoperti nelle fasi basse del processo di sviluppo. Errori possibili:
una funzionalità che il sistema dovrebbe supportare non è specificata, funzionalità incerte o obsolete, interfaccia utente poco intuitiva e difficile da usare.

16)Descrivere il concetto di transizione in un diagramma di stato UML

I diagrammi di stato sono grafi i cui nodi sono stati e i cui archi sono transizioni, etichettati dai nomi di eventi. Le transizioni sono appunto il passaggio da uno stato all'altro il quale avviene solo se si soddisfa un evento o una condizione.

17) 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:(1.1Ambito del Sistema – 1.2Scopo del sistema – 1.3Obiettivi e scopo del sistema- 1.4Definizioni, acronimi e abbreviazioni- 1.5References -1.6Overview)
- Sistema Corrente
- Sistema proposto:(3.1Overview- 3.2Requisiti Funzionali- 3.3Requisiti non funzionali- 3.4Modelli di sistema:3.4.1Overview-3.4.2Modelli a oggetti-3.4.3Modello dinamico-3.4.4User Interface, navigational paths and screen mock-ups)
- Glossario

18)Cos'è un percorso critico in un diagramma di PERT? Qual è la sua importanza?

Un diagramma di PERT rappresenta uno schedule mediante un grafo aciclico i cui nodi sono i task da completare. L'inizio e la durata dei task sono utilizzati per calcolare il percorso critico, il quale rappresenta il più lungo percorso possibile che attraversa il grafo. La lunghezza del percorso critico corrisponde al più breve schedule possibile, assumendo di avere sufficienti risorse per completare in parallelo task che sono indipendenti. I task sul percorso critico sono i più importanti, poiché un ritardo in uno qualsiasi di essi comporta un ritardo nel progetto complessivo.

19)Descrivere il meta – modello a spirale per il Ciclo di Vita del Software. Perchè lo si definisce meta – modello?

Un modello a spirale è caratterizzato da:

• Formalizzazione del concetto di iterazione : ha il riciclo come fondamento;

• Il processo viene rappresentato come una spirale piuttosto che come una sequenza di attività: ogni giro della spirale rappresenta una fase del processo. Le fasi non sono definite ma vengono scelte in accordo al tipo di prodotto. Ogni fase prevede la scoperta, la valutazione e il trattamento esplicito dei “rischi”

• E' un meta – modello : possibilità di utilizzare uno o più modelli. Il ciclo a cascata si può vedere come caso particolare (una sola iterazione).

20) Descrivere la differenza tra azione e attività in un diagramma di stato.

• Azione: è associata alla transizione. É considerato un processo rapido non interrompibile da un evento.

• Attività: è associata allo stato. Può durare un lasso di tempo considerevole e può essere interrotta da un evento.

21) Descrivere i vari tipi di oggetti che è possibile individuare in fase di analisi dei requisiti e in che modo vengono individuati.

Il modello ad oggetti di analisi consiste di oggetti di tre tipi : Entity, Boundary e Control. Gli oggetti Entity rappresentano l'informazione persistente tracciata dal Sistema. Gli oggetti Boundary rappresentano le interazioni tra gli attori e il sistema. Gli oggetti Control rappresentano i compiti di controllo svolti dal sistema (si occupano di realizzare gli use case). UML fornisce il meccanismo degli stereotipi per consentire di aggiungere tale meta informazione agli elementi di modellazione. Tali oggetti sono individuati tramite vari tipi di euristica, tra le più diffuse è l'euristica di Abbot che mappa grammaticalmente le parole a certi tipi di entità.

22) Spiegare la differenza tra requisiti funzionali, requisiti non funzionali e pseudo – requirements.

Requisiti funzionali: descrivono le interazioni tra il sistema e il suo ambiente, indipendentemente dalla sua implementazione.
Requisiti non funzionali: descrivono gli aspetti che non sono legati direttamente alle funzionalità del sistema.es:tempo di risp deve essere inferiore a un secondo.
Pseudo-Requirements: sono specifici strumenti e tecnologie che il sistema deve utilizzare e rispettare.

23) Descrivere i vantaggi e gli svantaggi del modello di ciclo di vita a cascata e le varianti proposte per risolverli.

Vantaggi : ha definito molti concetti utili. Ha rappresentato un punto di partenza importante per lo studio dei processi SW. Facilmente comprensibile e applicabile, soprattutto per tipi di sistema ben conosciuti e sperimentati.
Svantaggi : interazione con il committente solo all'inizio e alla fine. Il nuovo sistema software diventa installabile solo quando è totalmente finito.

Variante del modello a cascata : V&V e Retroazione dove aggiungiamo una verifica per stabilire la verità della corrispondenza tra un prodotto software e la sua specifica e una convalida per stabilire l'appropriatezza di un prodotto software alla sua missione operativa.

25) Descrivere cosa si intende per mantenere la tracciabilità tra i requisiti e qual è la sua importanza.

Ogni funzione del sistema può essere individuata e ricondotta al corrispondente requisito funzionale. Include anche l'abilità di tracciare le dipendenze tra i requisiti, le funzioni del sistema, gli artefatti, incluso componenti, classi, metodi e attributi di oggetti. È cruciale per lo sviluppo di test e per valutare i cambiamenti.

26) Descrivere i vari tipi di prototipazione

- mock – ups : produzione completa dell'interfaccia utente. Consente di definire con completezza e senza ambiguità i requisiti (si può, già in questa fase, definire il manuale di utente).
- Breadboards : implementazione di sottoinsiemi di funzionalità critiche del SS, non nel senso della fattibilità ma in quello dei vincoli pesanti che sono posti nel funzionamento del SS (carichi elevati, tempo di risposta,...), senza le interfacce utente. Produce feedback su come implementare la funzionalità (in pratica si cerca di conoscere prima di garantire).
- "Throw – away" : Lo scopo è quello di identificare meglio le specifiche richieste dall'utente sviluppando dei prototipi che sono funzionanti. Il prototipo sperimenta le parti del sistema che non sono ancora ben comprese oppure serve per valutare la fattibilità di un approccio.
- "Esplorativa" : pervenire ad un prodotto finale partendo da una descrizione di massima e lavorando a stretto contatto con il committente. Lo sviluppo dovrebbe avviarsi con la parte dei requisiti meglio compresa. Consente uno sviluppo incrementale.

27) Cosa sono scenari e casi d'uso? Quale relazione esiste tra essi?

Scenario: *Uno scenario è una descrizione informale, concreta e focalizzata di una singola caratteristica di un sistema e descrive cosa le persone fanno e sperimentano mentre provano ad usare i sistemi di elaborazione e le applicazioni*.

Caso d'uso: Un caso d'uso descrive una serie di interazioni che avvengono dopo un' inizializzazione da parte di un attore e specifica tutti i possibili scenari per una determinata funzionalità (visto in altri termini uno scenario è un'istanza di un caso d'uso).

Differenze:Uno scenario è un'istanza di un caso d'uso. Un caso d'uso specifica tutti i possibili scenari per una data funzionalità.