Ingegneria del Software (matr. pari-dispari): Esempi di domande a risposta multipla

1. Cosa è un metodo ?
   - ❑ Un sistema per fare qualcosa in modo migliore
   - ☑ Un procedimento generale per risolvere classi di problemi
   - ❑ Un particolare approccio o filosofia per fare qualcosa

2. Quale di queste affermazioni non è appropriata per il modello di processo a cascata ?
   - ❑ Le fasi del processo sono in progressione sequenziale
   - ❑ I semilavorati all'uscita di una fase sono congelati e non possono essere più modificati
   - ☑ Le funzionalità del sistema sono sviluppate in maniera incrementale

3. Un prototipo viene sviluppato per:
   - ❑ Ridurre i costi di sviluppo
   - ❑ Rilasciare rapidamente al cliente una prima versione del sistema
   - ☑ Interagire con il committente per convalidare i requisiti

4. Quale tipo di prototipazione parte con i requisiti meglio compresi ?
   - ☑ Prototipazione esplorativa
   - ❑ Prototipazione throw-away
   - ❑ Prototipazione mock-ups

5. Quale tipo di prototipazione tende a realizzare l'interfaccia utente?
   - ❑ Prototipazione esplorativa
   - ❑ Prototipazione breadboards
   - ☑ Prototipazione mock-ups

6. Quale di queste affermazioni è appropriata per il modello di processo incrementale ?
   - ❑ Ogni versione produce funzionalità/sottosistemi più affidabili
   - ☑ Ogni versione aggiunge nuove funzionalità/sottosistemi al sistema
   - ❑ Ogni versione raffina le funzionalità/sottosistemi che sono presenti fin dall'inizio

7. Quale di queste affermazioni è appropriata per il modello di processo iterativo ?
   - ☑ Ogni versione raffina le funzionalità/sottosistemi che sono presenti fin dall'inizio
   - ❑ Ogni versione aggiunge nuove funzionalità/sottosistemi al sistema
   - ❑ I requisiti a più alta priorità vengono rilasciati per primi

8. Cosa è un team ?
- [x] Un insieme di persone a cui è assegnato un task comune, ma che lavorano individualmente e senza necessità di interazione
- [ ] Un insieme di persone che rivedono un work product e che propongono azioni
- [ ] Un piccolo insieme di persone che lavorano in stretta interazione sulla stessa attività o task

9. Il tempo durante il quale un certo lavoro in un progetto deve essere fatto, viene denominato:
- [ ] Work-product
- [x] Schedule
- [ ] Task

10. Il lavoro che deve essere eseguito da un partecipante al progetto, viene denominato:
- [ ] Work-product
- [ ] Schedule
- [x] Task

11. Quale di queste affermazioni non è valida per una struttura organizzativa di reporting gerarchica ?
- [x] Lo stato è riportato dall'alto verso il basso
- [ ] Le decisioni vengono comunicate dall'alto verso il basso
- [ ] Le informazioni relative a stato e decisioni sono unidirezionali

12. Cosa è un liason all'interno di un team ?
- [ ] Il leader del team
- [x] Il responsabile della comunicazione con un altro team
- [ ] Il consulente amministrativo del team

13. Che tipo di ruolo è un configuration manager ?
- [ ] Uno sviluppatore
- [ ] Un manager
- [x] Un liason

14. Che cosa è un GANNT ?
- [x] Un grafico a barre che descrive lo schedule e la durata dei vari task
- [ ] Un grafo che descrive le dipendenze tra i task del progetto
- [ ] Un grafo che descrive le dipendenze tra i work-product del progetto

15. Che cosa è un PERT?
- [ ] Un grafico a barre che descrive lo schedule e la durata dei vari task
- [x] Un grafo che descrive le dipendenze tra i task del progetto
- [ ] Un grafo che descrive le dipendenze tra i work-product del progetto

16. Quale dei seguenti non è una comunicaizione pianificata?
- ❑ Definizione del problema
- ❑ Review di progetto
- ☑ Richiesta di modifica

17. Quale dei seguenti è una comunicazione non pianificata?
- ☑ Risoluzione di un problema
- ❑ Ispezione
- ❑ Rilascio

18. Quale dei seguenti meccanismi di comunicazione è asincrono?
- ❑ Intervista strutturata
- ❑ Riunione
- ☑ Posta elettronica

19. Cosa è un modello ?
- ☑ Un'astrazione che descrive il sistema o un sottoinsieme di un sistema
- ❑ Un insieme di regole grafiche o testuali per rappresentare viste
- ❑ Una vista di aspetti del sistema

20. Cosa è una vista ?
- ❑ Un'astrazione che descrive il sistema o un sottoinsieme di un sistema
- ☑ Una visualizzazione di particolari aspetti di un modello
- ❑ Un insieme di regole grafiche o testuali

21. Cosa è una notazione ?
- ❑ Un'astrazione che descrive il sistema o un sottoinsieme di un sistema
- ❑ Una visualizzazione di particolari aspetti di un modello
- ☑ Un insieme di regole grafiche o testuali per rappresentare viste

22. Cosa è UML ?
- ❑ Una notazione grafica per progettare sistemi software
- ☑ Un insieme di linguaggi per modellare software
- ❑ Un modello astratto per descrivere sistemi software

23. Quale di questi diagrammi non è usato per descrivere il comportamento dinamico di un sistema software ?
- ❑ Sequence diagram
- ❑ Activity diagram
- ☑ Use case diagram

24. I diagrammi dei casi d'uso:
- [x] Descrivono il comportamento funzionale del sistema così come visto dagli utenti
- [ ] Descrivono il comportamento dinamico tra gli attori e il sistema e tra gli oggetti del sistema
- [ ] Descrivono il comportamento dinamico di un sistema, in particolare il workflow

25. I diagrammi delle attività:
- [ ] Descrivono il comportamento funzionale del sistema così come visto dagli utenti
- [ ] Descrivono il comportamento dinamico tra gli attori e il sistema e tra gli oggetti del sistema
- [x] Descrivono il comportamento dinamico di un sistema, in particolare il workflow

26. I diagrammi di sequenza:
- [ ] Descrivono il comportamento funzionale del sistema così come visto dagli utenti
- [x] Descrivono il comportamento dinamico tra gli attori e il sistema e tra gli oggetti del sistema
- [ ] Descrivono il comportamento dinamico di un sistema, in particolare il workflow

27. I diagrammi di stato:
- [ ] Descrivono la struttura statica del sistema: oggetti, attributi e relazioni
- [ ] Descrivono il comportamento dinamico tra gli attori e il sistema e tra gli oggetti del sistema
- [x] Descrivono il comportamento dinamico di un singolo oggetto come una macchina a stati finiti

28. I diagrammi delle classi:
- [x] Descrivono la struttura statica del sistema: oggetti, attributi e relazioni
- [ ] Descrivono il comportamento dinamico tra gli attori e il sistema e tra gli oggetti del sistema
- [ ] Descrivono il comportamento dinamico di un singolo oggetto come una macchina a stati finiti

29. In UML le classi e gli oggetti (istanze) sono rappresentate con:
- [x] Rettangoli
- [ ] Ovali
- [ ] Rettangoli con angoli arrotondati

30. In UML i casi d'uso sono rappresentati con:
- [ ] Rettangoli
- [x] Ovali
- [ ] Rettangoli con angoli arrotondati

31. In UML gli stati sono rappresentati con:
- [ ] Rettangoli
- [ ] Ovali
- [x] Rettangoli con angoli arrotondati

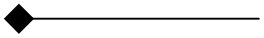
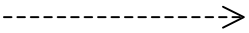
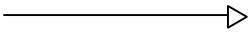
32. Quale di questi elementi non è parte di un diagramma dei casi d'uso:
- ❑ Attore
- ❑ Caso d'uso
- ☑ Classe

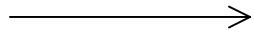
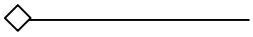
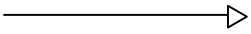
33. Quale di questi elementi non è parte di un diagramma di sequenza:
- ❑ Attore
- ☑ Caso d'uso
- ❑ Oggetti

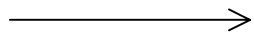
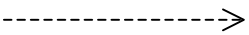
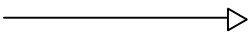
34. Che tipo di relazione può esistere tra un attore e un caso d'uso:
- ☑ Associazione
- ❑ Dipendenza
- ❑ Generalizzazione

35. Che tipo di relazione può esistere tra due attori:
- ❑ Associazione
- ❑ Dipendenza
- ☑ Generalizzazione

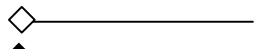
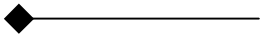
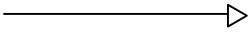
36. Quale di queste relazioni non esiste tra due casi d'uso:
- ☑ Associazione
- ❑ Dipendenza
- ❑ Generalizzazione

37. Quali di queste relazioni rappresenta una dipendenza:
- ❑ 
- ☑ 
- ❑ 

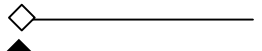
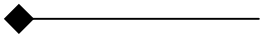
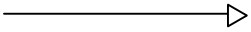
38. Quali di queste relazioni rappresenta una generalizzazione:
- ❑ 
- ❑ 
- ☑ 

39. Quali di queste relazioni rappresenta una associazione:
- ☑ 
- ❑ 
- ❑ 

40. Quali di queste relazioni rappresenta una aggregazione:

- [x] 
- [ ] 
- [ ] 

41. Quali di queste relazioni rappresenta una composizione:

- [ ] 
- [x] 
- [ ] 

42. Quali di questi è un requisito funzionale:

- [x] Il sistema deve visualizzare l'ora in base alla sua locazione
- [ ] Il tempo di risposta deve essere meno di un secondo
- [ ] Il linguaggio di implementazione deve essere Java

43. Quali di questi è un requisito non funzionale:

- [ ] Il sistema deve visualizzare l'ora in base alla sua locazione
- [x] Il tempo di risposta deve essere meno di un secondo
- [ ] Il linguaggio di implementazione deve essere Java

44. Cosa si intende per correttezza dei requisiti ?

- [x] I requisiti rappresentano la vista dell'utente
- [ ] Sono descritti tutti i possibili scenari del sistema
- [ ] Non ci sono requisiti funzionali e non funzionali che si contraddicono

45. Cosa si intende per completezza dei requisiti ?

- [ ] I requisiti rappresentano la vista dell'utente
- [x] Sono descritti tutti i possibili scenari del sistema
- [ ] Non ci sono requisiti funzionali e non funzionali che si contraddicono

46. Cosa si intende per consistenza dei requisiti ?

- [ ] I requisiti rappresentano la vista dell'utente
- [ ] Sono descritti tutti i possibili scenari del sistema
- [x] Non ci sono requisiti funzionali e non funzionali che si contraddicono

47. Cosa si intende per chiarezza dei requisiti ?

- [x] Non ci sono ambiguità nei requisiti
- [ ] I requisiti possono essere implementati e rilasciati
- [ ] Non ci sono requisiti funzionali e non funzionali che si contraddicono

48. Cosa si intende per realismo dei requisiti ?
- ❑ Non ci sono ambiguità nei requisiti
- ☑ I requisiti possono essere implementati e rilasciati
- ❑ Per ogni funzione del sistema è possibile individuare un insieme di requisiti funzionali

49. Cosa si intende per tracciabilità dei requisiti ?
- ❑ Non ci sono ambiguità nei requisiti
- ❑ I requisiti possono essere implementati e rilasciati
- ☑ Per ogni funzione del sistema è possibile individuare un insieme di requisiti funzionali

50. Quale di questi requisiti vanno affrontati durante il design ma non durante l'implementazione ?
- ❑ Requisiti ad alta priorità
- ☑ Requisiti a media priorità
- ❑ Requisiti a bassa priorità

51. Cosa si intende per interface engineering ?
- ❑ Lo sviluppo parte da zero, non esiste un sistema precedente
- ❑ Riprogettazione o reimplementazione di un sistema esistente con nuove tecnologie
- ☑ Fornire i servizi di un sistema esistente in un nuovo ambiente operativo

52. Che tipo di scenari sono usati per descrivere un sistema futuro?
- ❑ as-is scenarios
- ☑ visionary scenarios
- ❑ training scenarios

53. Che tipo di scenari sono usati per descrivere un sistema esistente ?
- ☑ as-is scenarios
- ❑ visionary scenarios
- ❑ training scenarios

54. Che tipo di scenari sono usati per guidare un utente nuovo nell'utilizzo del sistema ?
- ❑ as-is scenarios
- ❑ visionary scenarios
- ☑ training scenarios

55. Perché si usano gli scenari ?
- ☑ Per individuare i requisiti del sistema
- ❑ Per analizzare le associazioni tra gli oggetti del sistema
- ❑ Per analizzare il flusso di eventi di un caso d'uso

56. Come si descrive un caso d'uso ?
- ❑ Mediante uno scenario
- ☑ Mediante un flusso di eventi
- ❑ Mediante un insieme di requisiti funzionali

57. Cosa esprime una relazione di inclusione tra casi d'uso ?
- ❑ Una variante del normale flusso di eventi di un caso d'uso
- ❑ Una specializzazione di un caso d'uso
- ☑ Una decomposizione funzionale di un caso d'uso

58. Cosa esprime una relazione di estensione tra casi d'uso ?
- ☑ Una variante del normale flusso di eventi di un caso d'uso
- ❑ Una specializzazione di un caso d'uso
- ❑ Una decomposizione funzionale di un caso d'uso

59. Cosa esprime una relazione di generalizzazione tra casi d'uso ?
- ❑ Una variante del normale flusso di eventi di un caso d'uso
- ☑ Una specializzazione di un caso d'uso
- ❑ Una decomposizione funzionale di un caso d'uso

60. Un oggetto che interagisce con un attore in un sequence diagram è:
- ❑ Un entity object
- ☑ Un boundary object
- ❑ Un control object

61. Quale di queste categorie di requisiti non funzionali indica la facilità di cambiamenti al sistema dopo il rilascio ?
- ❑ Reliability
- ❑ Performance
- ☑ Supportability

62. Quale di queste categorie di preudo-requirements indica vincoli legati alle modalità di rilascio del sistema?
- ❑ Interface requirements
- ☑ Packaging requirements
- ❑ Operations requirements

63. Una trasformazione applicata agli oggetti di una classe e definita in fase di analisi è chiamata:
- ☑ Operazione
- ❑ Signature
- ❑ Metodo

64. Una connessione tra due istanze di oggetti è chiamata:
- ❑ Associazione
- ☑ Link
- ❑ Relazione

65. Una associazione è:
- ❑ Una connessione tra due istanze di oggetti
- ❑ Una dipendenza tra classi
- ☑ Un mapping bidirezionale tra classi

66. Quale di questi tipi di oggetti cambia meno frequentemente in un sistema software ?
- ❑ Boundary object
- ❑ Control object
- ☑ Entity object

67. Cosa indica un ruolo ?
- ☑ Una estremità dell'associazione
- ❑ La navigabilità dell'associazione
- ❑ La direzione del nome dell'associazione

68. Cosa indica la direzione di una associazione ?
- ❑ Una estremità dell'associazione
- ☑ La navigabilità dell'associazione
- ❑ La direzione del nome dell'associazione

69. Nel diagramma seguente, "filename"

- ❑ E' un attributo della classe directory
- ☑ E' un attributo della classe file
- ❑ E' il nome della associazione

70. Cosa indicano le frecce in un diagramma di sequenza ?
- ☑ Eventi inviati da un oggetto ad un altro
- ❑ Relazioni tra oggetti
- ❑ Dipendenze tra oggetti

71. Cosa indica la linea tratteggiata verticale sotto un oggetto in un diagramma di sequenza ?
- ❑ L'invio di un messaggio all'oggetto
- ❑ Il periodo di attivazione di una operazione dell'oggetto
- ☑ Il periodo durante il quale è possibile inviare un messaggio all'oggetto

72. Cosa indica un rettangolo verticale sotto un oggetto in un diagramma di sequenza ?
- ❑ L'invio di un messaggio all'oggetto
- ☑ Il periodo di attivazione di una operazione dell'oggetto
- ❑ Il periodo durante il quale è possibile inviare un messaggio all'oggetto

73. Cosa dovrebbe contenere la prima colonna di un diagramma di sequenza ?
- ☑ L'attore che inizia il caso d'uso
- ❑ Un boundary object
- ❑ Il control object che gestisce il caso d'uso

74. Cosa dovrebbe contenere la seconda colonna di un diagramma di sequenza ?
- ❑ L'attore che inizia il caso d'uso
- ☑ Un boundary object
- ❑ Il control object che gestisce il caso d'uso

75. Cosa dovrebbe contenere la terza colonna di un diagramma di sequenza ?
- ❑ L'attore che inizia il caso d'uso
- ❑ Un boundary object
- ☑ Il control object che gestisce il caso d'uso

76. Quali di queste affermazioni non è corretta ?
- ☑ Un boundary object accede ad un entity object
- ❑ Un entity object accede ad un control object
- ❑ Un control object crea un boundary object

77. In un diagramma di stato, quando viene valutata la condizione di guardia su una transizione ?
- ❑ Quando il sistema si trova nello stato sorgente della transizione
- ☑ Quando si verifica l'azione indicata sulla transizione
- ❑ Quando si verifica l'evento indicato sulla transizione

78. Quale di queste affermazioni non si riferisce ad un'attività in un diagramma di stato ?
- ☑ Operazione che prende tempo per essere completata
- ❑ E' associata ad un evento
- ❑ E' associata ad uno stato

79. Quale di queste affermazioni non si riferisce ad un'azione in un diagramma di stato ?
- ❑ Operazione che prende tempo per essere completata
- ☑ E' associata ad uno stato
- ❑ E' associata ad un evento

80. Quale di queste affermazioni si riferisce ad un diagramma di stato ?
- ❑ Serve ad individuare le relazioni temporali tra oggetti nel tempo
- ❑ Serve ad individuare la sequenza di operazioni come risposta ad uno o più eventi
- ☑ Serve ad individuare i cambiamenti degli oggetti nel tempo

81. Quale di queste affermazioni non si riferisce ad un diagramma di sequenza ?
- ☑ Serve ad individuare i cambiamenti degli oggetti nel tempo
- ❑ Serve ad individuare le relazioni temporali tra oggetti nel tempo
- ❑ Serve ad individuare la sequenza di operazioni come risposta ad uno o più eventi

82. Quale di questi prodotti dell'analisi dei requisiti vanno usati durante la fase di definizione degli obiettivi di design ?
- ❑ Modello dei casi d'uso
- ❑ Modello a oggetti
- ☑ Requisiti non funzionali

83. Quale di questi prodotti dell'analisi dei requisiti vanno usati durante la fase di decomposizione del sistema ?
- ☑ Modello dei casi d'uso
- ❑ Modello a oggetti
- ❑ Requisiti non funzionali

84. Quale di questi prodotti dell'analisi dei requisiti vanno usati durante la fase di design relativa a mapping hardware/software e a gestione dei dati persistenti ?
- ❑ Modello dei casi d'uso
- ☑ Modello a oggetti
- ❑ Requisiti non funzionali

85. Come viene rappresentato un sottosistema in UML ?
- ❑ Con una classe
- ❑ Con un caso d'uso
- ☑ Con un package

86. Cosa è un servizio di un sottosistema ?
- ❑ Un insieme di operazioni con signature completamente specificata
- ☑ Un gruppo di operazioni che condividono uno scopo comune
- ❑ Un insieme di associazioni, eventi e vincoli legati tra di loro

87. Cosa è l'interfaccia di un sottosistema ?
- ☑ Un insieme di operazioni con signature completamente specificata
- ❑ Un gruppo di operazioni che condividono uno scopo comune
- ❑ Un insieme di associazioni, eventi e vincoli legati tra di loro

88. Quali di queste affermazioni è vera ?

- ❑ L'accoppiamento misura le dipendenze tra le classi di un sottosistema
- ☑ In un sistema con elevato accoppiamento le modifiche ad un sottosistema hanno forte impatto sugli altri sottosistemi
- ❑ L'obiettivo del system design è massimizzare l'accoppiamento

89. Quali di queste affermazioni è vera ?

- ❑ La coesione misura le dipendenze tra i sottosistemi di un sistema
- ❑ In sottosistemi con elevata coesione le modifiche ad un sottosistema hanno forte impatto sugli altri sottosistemi
- ☑ Le classi di un sottosistema con elevata coesione eseguono task simili

90. A cosa servono le partizioni di un sistema ?

- ☑ A dividere verticalmente un sistema in sottosistemi debolmente accoppiati
- ❑ A dividere orizzontalmente un sistema in sottosistemi a diversi livelli di astrazione
- ❑ A dividere un sistema sia orizzontalmente che verticalmente in sottosistemi indipendenti

91. A cosa servono i layer di un sistema ?

- ❑ A dividere verticalmente un sistema in sottosistemi debolmente accoppiati
- ☑ A dividere orizzontalmente un sistema in sottosistemi a diversi livelli di astrazione
- ❑ A dividere un sistema sia orizzontalmente che verticalmente in sottosistemi indipendenti

92. A chi fornisce servizi un layer di un sistema ?

- ❑ Ai layer di livello più basso
- ❑ Ai layer dello stesso livello
- ☑ Ai layer di livello più alto

93. In un'architettura software aperta:

- ❑ I sottosistemi di un livello possono accedere solo ai sottosistemi del livello immediatamente inferiore
- ☑ I sottosistemi di un livello possono accedere ai sottosistemi di qualunque livello inferiore
- ❑ I sottosistemi di un livello possono accedere sia ai sottosistemi dei livelli inferiori che a quelli dei livelli superiori

94. In un'architettura software chiusa:

- ☑ I sottosistemi di un livello possono accedere solo ai sottosistemi del livello immediatamente inferiore
- ❑ I sottosistemi di un livello possono accedere solo ai sottosistemi dello stesso livello
- ❑ I sottosistemi di un livello possono accedere ai sottosistemi di qualunque livello inferiore

95. Quali sono gli obiettivi di design di un'architettura software aperta ?
- ❑ Manutenibilità
- ☑ Efficienza
- ❑ Affidabilità

96. Quali sono gli obiettivi di design di un'architettura software chiusa ?
- ❑ Usabilità
- ❑ Efficienza
- ☑ Manutenibilità

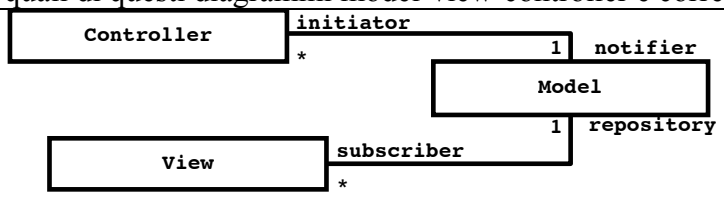
97. Quale di queste affermazioni è falsa?
- ❑ In un'architettura client-server, il server fornisce i servizi al client
- ❑ In un'architettura client-server, il server conosce l'interfaccia del client
- ☑ Una repository architecture è un caso particolare di architettura client-server

98. Quali di queste affermazioni è vera ?
- ☑ In un'architettura peer-to-peer non c'è differenza tra client e server
- ❑ Un'architettura peer-to-peer favorisce una gestione centralizzata dei dati
- ❑ Un'architettura peer-to-peer consente di evitare problemi di deadlock

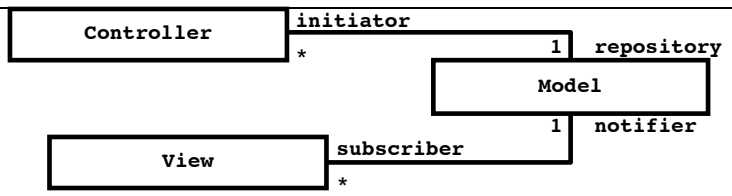
99. Un'architettura model-view-controller, il sottosistema "model":
- ☑ E' responsabile della conoscenza del dominio applicativo
- ❑ E' responsabile della visualizzazione degli oggetti del dominio applicativo
- ❑ E' responsabile della sequenza dei interazioni con l'utente

100. Dire quali di questi diagrammi model-view-controller è corretto:

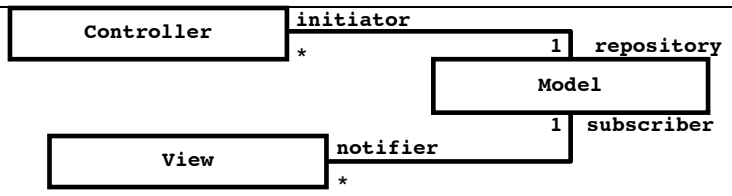
❑



☑



❑



101. Le relazioni tra i componenti in un component diagram sono
- [x] associazioni
- [ ] dipendenze
- [ ] generalizzazioni

102. Quale di queste affermazioni relative ad un component diagram è falsa ?
- [x] Mostra il mapping hardware/software
- [ ] Mostra la struttura del sistema a compilation time
- [ ] Mostra la struttura a design time

103. Quale di queste affermazioni relative ad un deployment diagram è falsa ?
- [x] Mostra la struttura del sistema a run-time
- [ ] Mostra il mapping hardware/software
- [ ] Mostra le dipendenze tra componenti e interfacce dei sottosistemi

104. Le relazioni tra i nodi di un deployment diagram sono:
- [ ] associazioni
- [x] dipendenze
- [ ] generalizzazioni

105. In un thin client model
- [x] Il sistema è two thier e la logica applicativa è eseguita sul server
- [ ] Il sistema è two thier e la logica applicativa è eseguita sul client
- [ ] Il sistema è three thier e la logica applicativa è eseguita sul client

106. Se il mio obiettivo è fornire implementazioni diverse per uno stesso sottosistema
- [x] Uso un bridge pattern
- [ ] Uso un adapter pattern
- [ ] Uso un façade pattern

107. Se il mio obiettivo è quello di realizzare una architettura chiusa
- [ ] Uso un adapter pattern
- [ ] Uso un bridge pattern
- [x] Uso un façade pattern

108. Un invariante è
- [ ] Un predicato che deve essere vero prima dell'invocazione di un metodo di una classe
- [ ] Un predicato che è vero dopo l'invocazione di un metodo di una classe
- [x] Un predicato che è vero prima e dopo l'invocazione di un metodo di una classe

109. Quale di queste affermazioni relative a JavaDoc è corretta ?
- [x] Il commento che precede un metodo consente di specificare la precondizione del metodo
- [ ] Il commento che precede un metodo consente di specificare la postcondizione del metodo
- [ ] Il commento che precede un metodo consente di specificare l'invariante della classe

110. Se devo realizzare una associazione qualificata uso come struttura dati
- [x] Una tabella
- [ ] Un insieme
- [ ] Una lista

111. Se devo estendere una classe mi interessa conoscere
- [ ] Tutti i membri privati, protetti e pubblici della classe
- [x] Solo i membri protetti e pubblici della classe
- [ ] Solo i membri pubblici della classe

112. La realizzazione delle associazioni del modello a oggetti
- [x] E' una trasformazione del modello a oggetti
- [ ] E' una trasformazione di tipo forward engineering
- [ ] E' una trasformazione di refactoring

113. Se devo memorizzare dati voluminosi e necessari per un periodo breve di tempo
- [ ] uso una struttura dati in memoria
- [x] uso un file
- [ ] uso un database

114. Se devo effettuare una trasformazione prima sul codice sorgente e poi sul modello a oggetti faccio
- [ ] Prima operazioni di forward engineering e poi di refactoring
- [ ] Prima operazioni di object model transformation e poi di forward engineering
- [x] Prima operazioni di refactoring e poi di reverse engineering

115. Se eredito un contratto in UML quale di queste affermazioni non è corretta ?
- [x] La precondizione del metodo nella sottoclasse consente al metodo di gestire meno casi del corrispondente metodo nella superclasse
- [ ] Il metodo nella sottoclasse deve assicurare la stessa postcondizione del corrispondente metodo nella superclasse
- [ ] L'invariante della sottoclasse può essere più restrittivo dell'invariante della superclasse

116. Quali di questi operazioni non è opportuno implementare durante la realizzazione di contratti ?
- [ ] Controllare che la precondizione all'inizio di un metodo sia soddisfatta
- [ ] Controllare che la postcondizione all'uscita di un metodo sia soddisfatta
- [x] Incapsulare il codice di controllo in metodi separati ai fini del riuso

117. L'evento percepito dall'utente come differenza tra comportamento atteso di un sistema software e comportamento esibito dal sistema software viene denominato

- [x] Failure
- [ ] Fault
- [ ] Error

118. Nel testing di integrazione bottom-up ho bisogno di realizzare

- [x] Test Driver
- [ ] Test Stub
- [ ] Sia test driver che test stub

119. Quale di queste affermazioni è falsa ?

- [ ] Nel testing white box la derivazione dei casi di test mira a coprire la maggior parte di una classe di elementi della struttura del codice
- [ ] Nel testing white box il numero di casi di test da eseguire dipende dal criterio di copertura adottato
- [x] Nel testing white-box non si controlla se l'output prodotto in corrispondenza di un input è uguale all'output atteso

120. Quando si effettua il testing di regressione ?

- [ ] Dopo aver effettuato il test di integrazione e prima di effettuare il test di sistema
- [ ] Dopo aver effettuato il testing di sistema e prima di effettuare il testing di accettazione
- [x] Dopo aver effettuato delle modifiche al codice per correggere un fault

1. **Che cosa è l'ingegneria del software secondo Bruegge, si può definire mediante 4 attività fondamentali. Elencarne almeno 3.**
   L'ingegneria del software è una collezione di attività, metodologie e strumenti che aiutano nella produzione del software nei vincoli di tempo, costo e qualità imposti mentre occorre il cambiamento. Le attività fondamentali sono: modellazione, problem solving, acquisizione della conoscenza, fondamenti logici come guida.

2. **le seguenti categorie di requisiti estendono il modello FURPS di base:**
   interfaccia, packaging, legali,operazione

3. **un work product è:**
   un artefatto prodotto durante l'attività di sviluppo

4. **in un diagramma di stato (state chart) un'attività è:**
   un comportamento che è eseguito quando un oggetto risiede in uno stato

5. **un caso d'uso specifica**
   tutti i possibili scenari per una data funzionalità del sistema

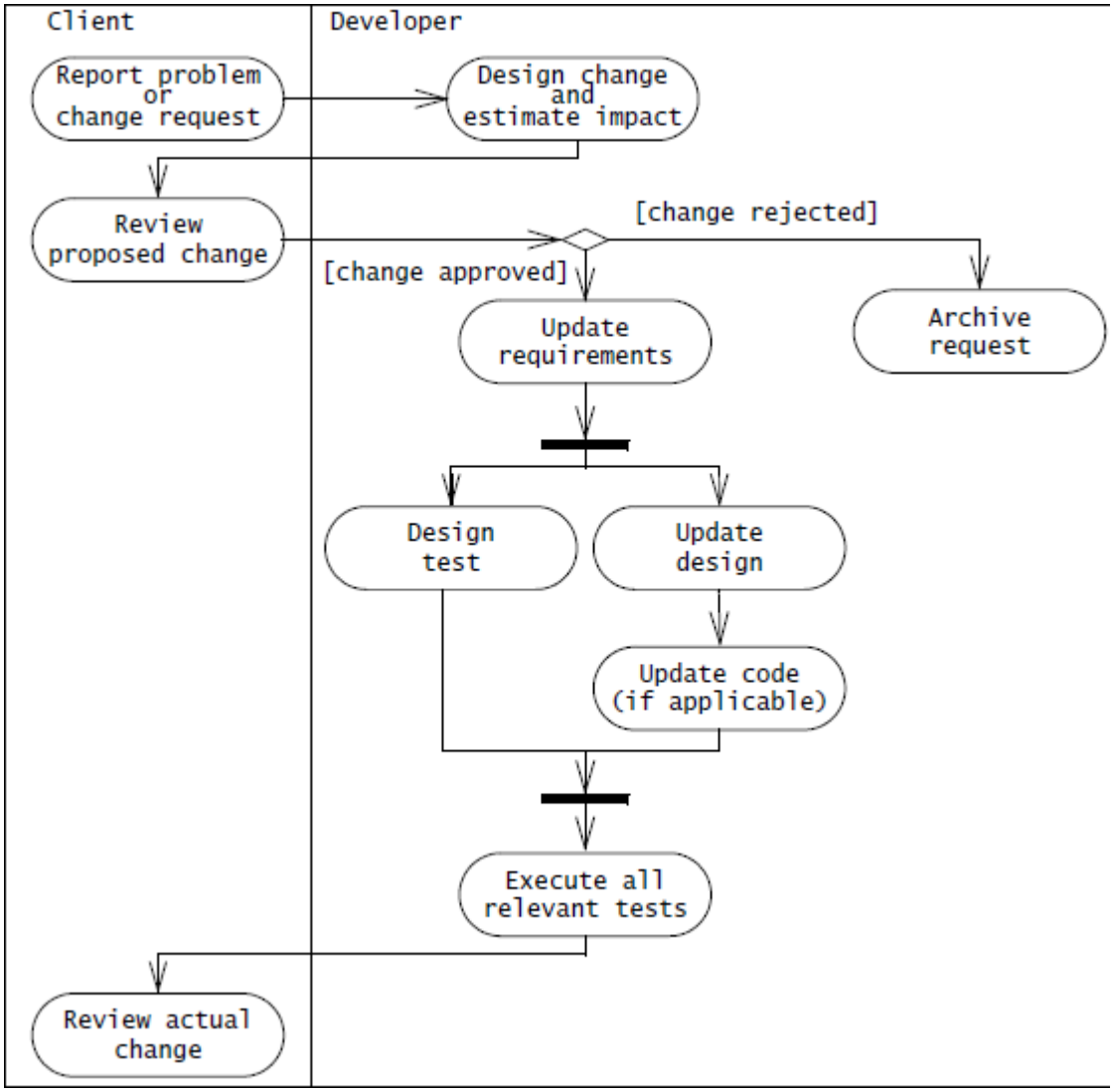
6. **indicare i tre tipi di comunicazione pianificata e esprimere brevemente lo scopo di ognuna:**
   presentazione del problema: viene spiegato il problem statement
   riesame tra pari: vi sono due tipi: informale (walkthrough) dove il codice viene sottoposto al giudizio di altri sviluppatori e formale (ispezione) dove il codice viene sottoposto ad una serie di criteri.
   Brainstorming: si cercano tutte le soluzioni possibili ad un problema, anche quelle sbagliate perché potrebbero generare nuove soluzioni.

7. **Indicare cosa rappresentano i seguenti oggetti:**
   entity object: rappresenta l'informazione persistente all'interno del sistema
   boundary object: rappresenta l'oggetto di comunicazione tra l'utente e il sistema
   control object: rappresenta l'oggetto che si occupa del controllo del flusso degli eventi (la logica del sistema)

8. **Le schede CRC servono per definire:**
   struttura e relazioni delle classi del sistema

9. **L'attività che identifica concetti più specifici a partire da concetti di più alto livello viene detta:**
   specializzazione

10. Disegnare il diagramma delle attività UML che definisce un processo di revisione



1. **Fornire la definizione di coesione ed accoppiamento**
   L'accoppiamento è l'insieme delle dipendenze tra sottosistemi. I sistemi possono essere debolmente o fortemente accoppiati.
   La coesione è l'insieme delle dipendenze interne ai sottosistemi, in particolare tra le classi.
   L'accoppiamento deve essere debole, mentre la coesione alta.

2. **Fanno parte della requirement elicitation le seguenti attività:**
   identificare requisiti non funzionali, identificare gli scenari

3. **Quale tra le seguenti affermazioni è falsa? Un modello è utile per:**
   considerare i dettagli del sistema

4. **La specifica delle seguenti informazioni "nome task, ruolo assegnato, descrizione task, input, output" qualifica:**
   un task

5. **la domanda "quanti task può eseguire il sistema in un certo periodo di tempo?" fa riferimento al criterio di disegno di:**
   throughput

6. **fanno parte della fase di "start" del processo di project management le seguenti attività:**
   team formation, project kickoff

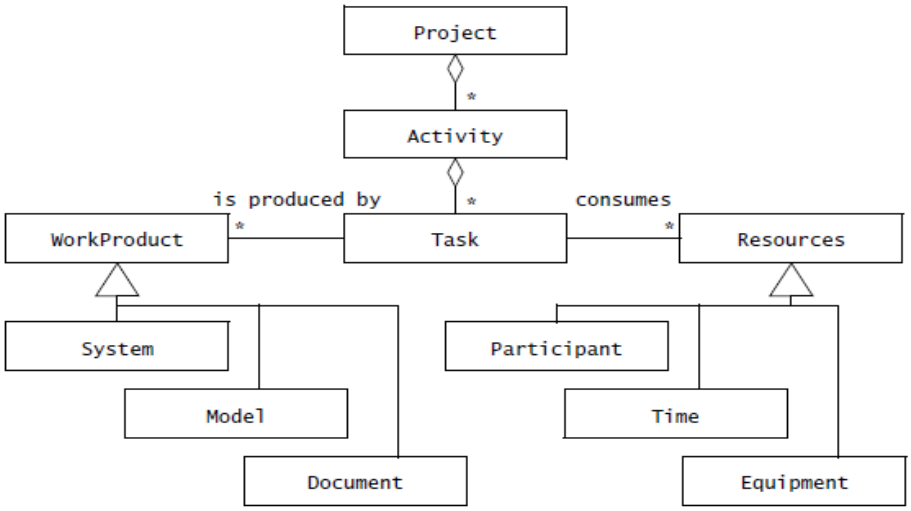
7. **fanno parte del "client sign-off":**
   schedule,RAD,un processo di revisione

8. **un test stub:**
   simula un componente chiamato da un altro sottotest

9. **una versione che viene resa disponibile a altri sviluppatori durante un progetto viene detta:**
   promotion

10. **elencare le attività del system design**
    decomposizione del sistema in sottosistemi
    mapping hw/sw
    polizza di controllo degli accessi
    scelta dell'architettura software
    flusso di controllo globale
    gestione dei dati persistenti
    gestione delle condizioni limite

11. riprodurre il diagramma delle classi UML che rappresenta i concetti di ingegneria del software



1. **fornire una definizione di use case**
   un caso d'uso rappresenta la descrizione di una funzione del sistema dal punto di vista dell'utente.

2. **Quale tra le seguenti affermazioni, è falsa?**
   È possibile dimostrare che un modello rappresenta la realtà

3. **La comunicazione non pianificata include:**
   richiesta di chiarimenti, risoluzione di un problema, richiesta di cambio

4. **Elencare tre domande relative al requisito non funzionale di "prestazioni del sistema"**
   tempo di risposta, quanto deve essere veloce il sistema nel rispondere alle richieste dell'utente?
   Throughput, quanti task si riescono a svolgere in un tempo fissato?
   precisione, i task sono coerenti con i requisiti richiesti?

5. **Quale tra le seguenti attività è sotto la responsabilità del project manager?**
   Comunicare con i clienti

6. **Sia W l'insieme dei work product e D l'insieme dei deliverable di un progetto software. Qual è la relazione tra W e D?**
   Deliverable è sottoinsieme di work product

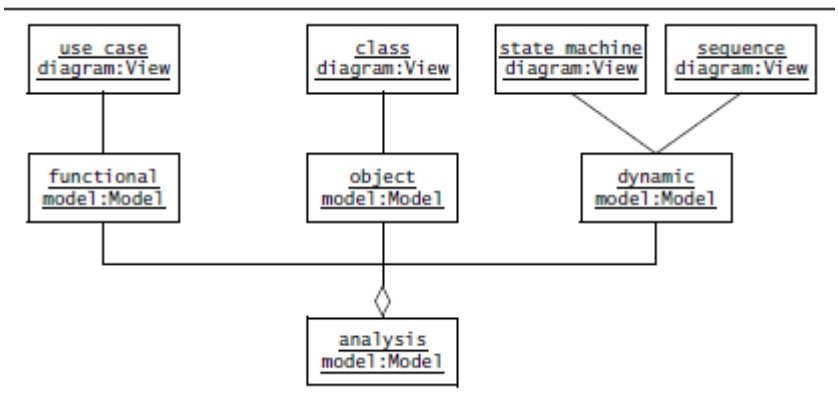
7. **Gli scopi del client review sono:**
   confermare i cambi ai requisiti di sistema

8. **Durante la scrittura dei diagrammi delle classi la tecnica di qualificazione serve a:**
   ridurre la molteplicità delle associazioni

9. **I seguenti concetti riguardano l'organizzazione di un progetto:**
   individuare le associazioni tra classi

10. **Elencare le responsabilità dell'architetto del sistema software**
    È un ruolo di integrazione, unifica casi d'uso e modello a oggetti da un punto di vista del sistema.

11. **Elencare le attività dell'analisi e riprodurre il corrispondente diagramma UML**



Identificare oggetti entità, boundary e control.
Mappare i casi d’uso in oggetti con i sequence diagram
Modellare interazioni tra oggetti con schede CRC
Identificare associazioni, aggregazioni e attributi.
Modellare comportamenti dipendenti dallo stato di oggetti individuali

1. **Definire la nozione di "work product"**
   Un work product è un artefatto prodotto durante la realizzazione di un task o di un'attività. Il work product può essere interno (promotion) se è rivolto agli sviluppatori, deliverable se è da consegnare all'utente.

2. **Durante l'analisi gli sviluppatori mirano a produrre un modello del sistema che è:**
   completo, consistente,non ambiguo

3. **Elencare i nomi delle sezioni che compongono un caso d'uso in forma testuale**
   Nome del caso d'uso
   Attori partecipanti
   Condizioni d'entrata
   Flusso di eventi
   Condizioni di uscita
   Requisiti di qualità

4. **Il processo teso a ricercare un certo numero di soluzioni a un problema per poi valutarle viene detto:**
   brainstorming

5. **Estendono il modello FURPS al modello FURPS+:**
   requisiti di implementazione, requisiti legali

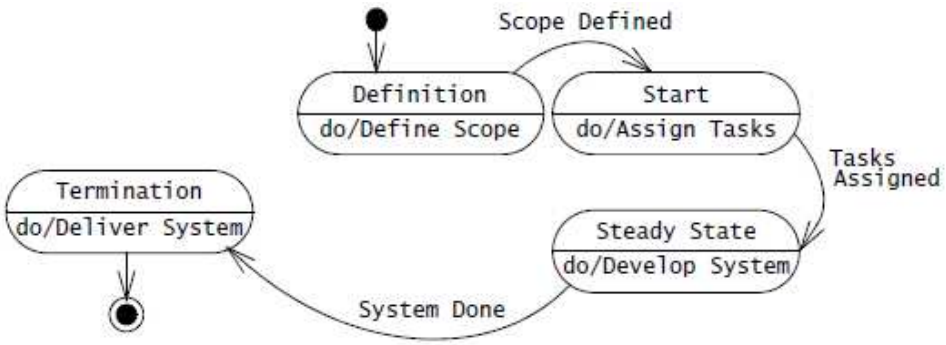
6. **Un diagramma di sequenza contiene le stesse informazioni di un:**
   diagramma di attività

7. **Elencare almeno cinque attività che vengono svolte durante l'analisi dei requisiti**
   Identificare oggetti boundary, entity, control, attributi, aggregazioni, associazioni

8. **Nei diagrammi di attività le swimlane servono a:**
   raggruppare le azioni svolte dai ruoli

9. **Possono estendere i diagrammi gia presenti in UML:**
   stereotipi, vincoli

10. **Disegnare il diagramma degli stati in un progetto software contenente i seguenti stati:"definition","start","steady state","termination"**



1. **L’acquisizione della conoscenza è:**
   un processo non lineare

2. **I diagrammi delle classi rappresentano:**
   la struttura del sistema

3. **Tra una classe e una sua istanza esiste una relazione di:**
   generalizzazione

4. **Uno schedule costituisce:**
   il mapping di un task sull’asse dei tempi

5. **Quale tra i seguenti requisiti è verificabile?**
   Il sistema fornirà risposte in meno di un secondo

6. **In UML un’associazione tra due classi è:**
   la rappresentazione di un insieme di collegamenti tra oggetti

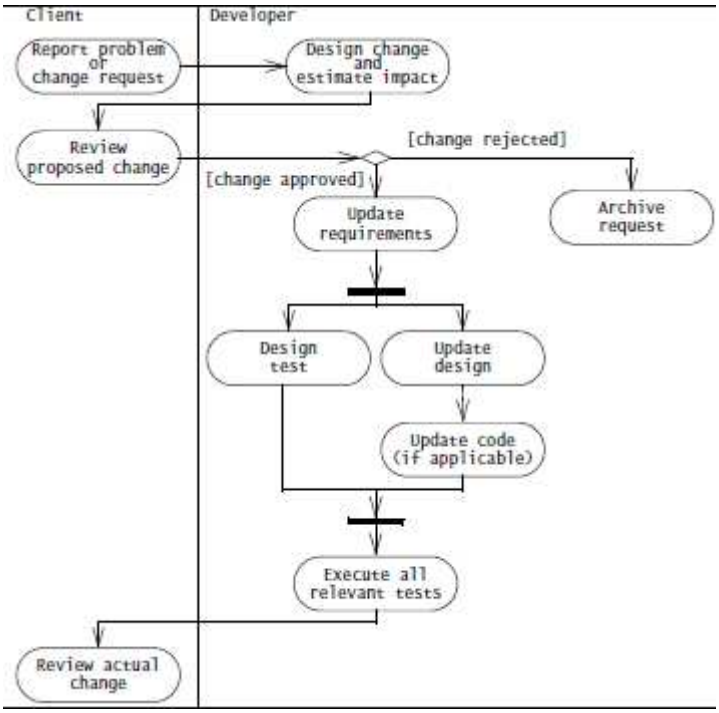
7. **Una richiesta di chiarimenti costituisce un evento di comunicazione:**
   sincrono e non pianificato

8. **In UML un diagramma state chart è una notazione per:**
   descrivere la sequenza di stati di un oggetto

9. **Nei diagrammi di sequenza quale delle seguenti euristiche è inadatta?**
   Oggetti entità accedono a oggetti di controllo

10. **Lo scopo dell’analisi è quello di:**
    strutturare e formalizzare i requisiti utente

11. **Riprodurre il diagramma delle attività che rappresenta un processo di revisione che si attiva a seguito di una richiesta di cambiamento oppure di un problema segnalato.**



1. **L'identificazione degli obiettivi di design (design goal)**
   Permette di stabilire le necessità hw/sw

2. **Una classe C1 nel package P1 usa una classe C2 nel package P2. La relazione tra P1 e P2 è:**
   di dipendenza

3. **Nei diagrammi di stato un'azione è definita come:**
   un comportamento atomico eseguito nella macchina a stati

4. **La comunicazione pianificata include:**
   status meeting, peer review

5. **Definire la nozione di metodo**
   Un metodo è un procedimento generale per risolvere uno specifico problema.

6. **In java i contratti per la specifica delle interfacce possono essere implementati con il meccanismo di:**
   meccanismo delle eccezioni

7. **Il pilot testing**
   È effettuato da un selezionato numero di utenti

8. **Quali di queste attività fanno tutte parte dell'ispezione dei componenti?**
   Overview, preparation, follow-up

9. **Nel contesto del configuration management, una versione di un configuration item che è stata formalmente riesaminata e poi approvata che può cambiare solo mediante una change request viene detta:**
   baseline

10. **Definire il concetto di baseline nel contesto del configuration management**
    È una versione di un configuration item che è stata formalmente riesaminata e successivamente approvata, che può essere cambiata solo dopo una richiesta di cambiamento.

1. **Un work product è un artefatto prodotto durante lo sviluppo software e si distingue in "interno" destinato a usi propri del team di sviluppo e "deliverable" da consegnare invece al cliente. Associare a ogni artefatto il tipo appropriato:**
    a. manuale d'uso: deliverable
    b. manuale dei test: interno
    c. diagramma delle classi: interno
    d. specifiche software: deliverable

2. **fornire la definizione di metodo**
   un metodo è un procedimento generale per risolvere uno specifico problema.

3. **il dominio applicativo riguarda tutti gli aspetti del problema dell'utente. Il dominio applicativo include:**
   ambiente fisico, utenti, processi

4. **i diagrammi dei casi d'uso possono includere i seguenti tipi di relazione:**
   comunicazione, estensione, generalizzazione

5. **un task è definito come l'unità di lavoro assegnabile a:**
   un ruolo

6. **lo scopo del project review è fornire informazioni:**
   al project manager per stimare stato del progetto; ai team per esaminare interfacce

7. **l'ingegneria delle interfacce riguarda:**
   il ridisegno di interfacce utente in un sistema pre-esistente

8. **elencare i tre modelli che compongono il modello d'analisi**
   modello funzionale, modello a oggetti, modello dinamico

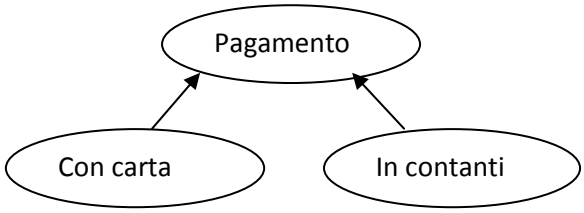
9. **fanno parte del "client sign-off":**
   schedule, RAD, un processo di revisione

10. **quando è possibile progettare test ripetibili per dimostrare che il sistema soddisfa la specifica, s dice che:**
    la specifica dei requisiti è verificabile

1. **“system”, “model”, “document” sono sottoclassi dirette della classe:**
   work product

2. **Un “ruolo” è definito come:**
   un insieme di task di tipo manageriale e tecnico che sono assegnati ad un singolo o ad un team.

3. **Illustrare con un esempio di modello grafico minimale la relazione di ereditarietà nei casi d’uso**



4. **Il dominio delle soluzioni rappresenta:**
   lo spazio di modellazione di tutti i possibili sistemi.

5. **In una diagramma delle classi un’associazione molti a molti rappresenta:**
   una molteplicità 0..n oppure 1..n su entrambi i lati.

6. **Una specifica dei requisiti che non contraddice se stessa viene detta:**
   consistente

7. **Descrivere (scopi e partecipanti) la comunicazione pianificata “problem inspection”**
   L’ispezione è un tipo di comunicazione pianificata detta “riesame tra pari” che avviene tra sviluppatori. Durante questo tipo di comunicazione viene sottoposto il codice sorgente ad una serie di criteri per verificare se li rispetta.

8. **Modellare, con un diagramma degli stati, stati e transizioni dello svolgimento dell’esame di IS dalla prenotazione all’esito (superato non superato)**

9. **la specifica di un lavoro da eseguire a completamento di un task o attività viene chiamata:**
   work package

10. **elencare i nomi dei modelli che compongono il “modello d’analisi”**
    modello funzionale, modello a oggetti, modello dinamico

1. **sia W l'insieme dei work product e D l'insieme dei deliverable di un progetto software. Qual è la relazione tra W e D?**
   deliverable è sottoinsieme di work product

2. **quale tra le seguenti affermazioni è falsa? Un modello è utile per:**
   studiare la proprietà emergente del sistema

3. **un tipo di dato astratto è specificato in:**
   un linguaggio a oggetti

4. **i diagrammi dei casi d'uso descrivono il comportamento di un sistema come percepito da:**
   utenti del sistema

5. **in UML un link rappresenta:**
   una connessione tra oggetti

6. **la specifica delle seguenti informazioni "nome task, ruolo assegnato, descrizione task, input, output" qualifica:**
   un task

7. **il V-Model si occupa della pianificazione di attività di testing ed esecuzione dei test case relativamente a:**
   tutte le fasi del ciclo di sviluppo

8. **sono esclusivamente di pianificazione i seguenti insiemi di documenti:**
   test design, test case

9. **le seguenti attività gestionali "configuration item identification, release management, variant management" fanno parte del:**
   change management

10. **quale tra le seguenti caratteristiche è estranea alla definizione di progetto?**
    Serve a creare un prodotto o un servizio

1. **i seguenti elementi fanno parte della specifica di un work package:**
   nome task, descrizione task, dipendenze sugli input

2. **un vincolo è una regola collegata a un elemento di modellazione UML che:**
   restringe la semantica dell'elemento modellato

3. **gli scopi del client review:**
   confermare i cambi ai requisiti del sistema

4. **quale tra le seguenti attività non fa parte della requirement elicitation?**
   Identificare i dati persistenti

5. **una baseline è definita come una versione di un work product che è:**
   una versione di work item formalmente riesaminata e approvata

6. **il testing di un insieme comune di funzionalità presso un ristretto e selezionato gruppo di utenti viene detto:**
   pilot testing

7. **nell'ambito della specifica delle interfacce un contratto è interpretabile come un accordo tra:**
   l'utente della classe e l'implementatore della classe

8. **quale delle seguenti tecniche è rivolta esclusivamente a scoprire staticamente gli errori (senza l'esecuzione del programma o dei modelli che lo rappresentano)?**
   Fault avoidance

9. **un processo aziendale di change management può essere modellato mediante:**
   activity diagram

10. **quale tra le seguenti attività è estranea al project management?**
    Gestione qualità del software

La struttura organizzativa di un ente pubblico è suddivisa in 4 livelli

1. **fornire una definizione di metodologia**
   una metodologia è un insieme di metodi che servono per risolvere classi di problemi

2. **quale domanda non è attinente al requisito di supportability?**
   Quanti utenti concorrenti deve supportare il sistema?

3. **il modello dell'analisi deve essere**
   completo, consistente, non ambiguo

4. **qual è lo scopo dei diagrammi di deployment?**
   Mostrare le relazioni fisiche tra i componenti hw e sw

5. **un constraint è una regola associata a un elemento di modellazione UML che:**
6. ne restringe la semantica

7. **la politica delle protezioni in un sistema sw può essere definita tramite la matrice degli accessi. Elencare i tre possibili approcci per la rappresentazione della matrice degli accessi riportandone le caratteristiche essenziali.**

   La matrice si può rappresentare attraverso tre approcci:
   tabella d'accesso globale: rappresenta esplicitamente ogni cella nella matrice come una tripla (attore, classe, operazione). Se tale tripla non è presente l'accesso è negato.
   Lista di controllo degli accessi: associa una coppia (attore, operazione) a ciascuna classe.
   Capability: associa una coppia (classe, operazione) a ciascun attore.

8. **Gli strumenti software definite "same time, different place groupware" consentono di:**
   collaborare tra utenti in rete in modalità sincrona

9. **il documento che contiene : inputs, drivers, stubs, e risultati attesi dei test viene chiamato:**
   test case specification

10. **giustificare l'uso di uno strumento software per la gestione delle configurazioni**

1. **fornire una definizione di work package**
   è la specifica di come deve essere realizzato il work product. Al suo interno sono indicati nome del task, input, output, schedule.

2. **quale domanda tra le seguenti è inadatta a individuare gli attori di un sistema?**
   Chi paga per la realizzazione del sistema?

3. **Un diagramma state chart è una notazione per:**
   descrivere la sequenza di stati di un oggetto

4. **Quali concetti non sono propri della fase di system design?**
   Generalizzazione e specializzazione

5. **La "differenza tra quanto specificato e comportamento osservato" fa riferimento al criterio di:**
   affidabilità

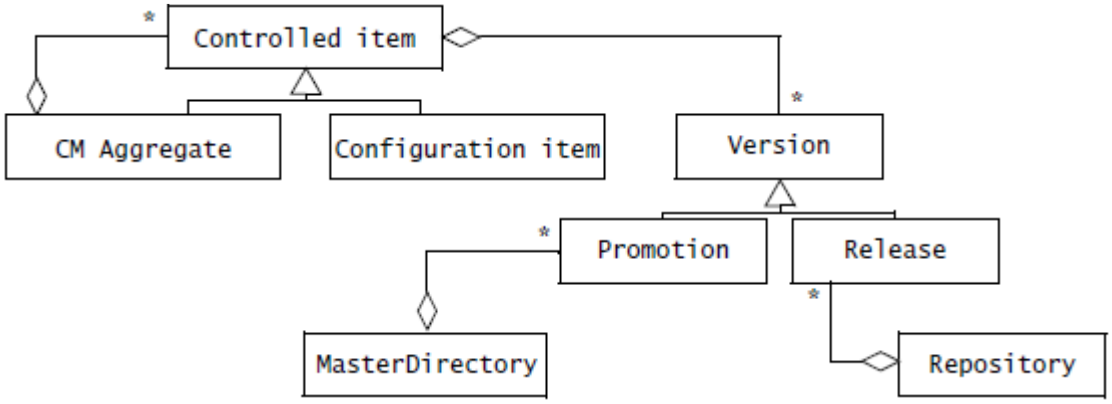
6. **Il testing funzionale, di usabilità e di prestazioni eseguito dal cliente nell'ambiente di sviluppo viene definito:**
   acceptance testing

7. **La comunicazione pianificata include:**
   problem presentation, project review, peer review

8. **Una versione di un configuration item che è stata formalmente riesaminata e che può essere cambiata solamente tramite una richiesta di cambiamento viene chiamata:**
   baseline

9. **Giustificare l'uso del linguaggio OCL**
   L'OCL è un linguaggio che consente l'uso dei vincoli per specificare formalmente gli elementi in un singolo modello. Un vincolo è espresso con un'espressione booleana che restituisce true/false.

10. **Riprodurre il diagramma delle classi UML che rappresenta i concetti di configuration management**



1. **Fornire una definizione di modello e giustificarne l'uso in ingegneria del software**
   Un modello è un'astrazione della realtà. Un'astrazione è utile perché si focalizza sui dettagli rilevanti ignorando quelli non rilevanti. Un modello è utilizzato quando abbiamo a che fare sistemi troppo piccoli, troppo grandi, troppo complessi o anche troppo costosi poiché un modello è una rappresentazione economica della realtà. In ingegneria del software usiamo i modelli per dominare la complessità.

2. **Un dominio applicativo:**
   rappresenta tutti gli aspetti del problema utente

3. **Un link rappresenta:**
   una connessione tra oggetti

4. **Uno schedule costituisce:**
   il mapping di un task sull'asse dei tempi

5. **Nei diagrammi statechart uno stato rappresenta:**
   una condizione soddisfatta da un insieme di attributi di un oggetto

6. **In UML un'associazione tra due classi è:**
   la rappresentazione di un insieme di collegamenti tra oggetti

7. **La comunicazione non pianificata include:**
   request for change, request for clarification, issue resolution

8. **Il modello FURPS è stato esteso includendo i seguenti requisiti non funzionali:**
   operation requirements, legal requirements

9. **La seguente domanda "i casi d'uso sono stati tutti nominati usando frasi verbali?" fa riferimento a:**
   non ambiguità della specifica

10. **Giustificare l'uso delle schede CRC durante l'analisi dei requisiti**
    Le schede CRC si usano per individuare le classi, le responsabilità e le collaborazioni tra le classi. Vengono usate perché sono uno strumento facile da utilizzare e modellare.

11. **Riprodurre l'indice del RAD commentando brevemente le finalità di ogni paragrafo**
    - Introduzione: introduce il sistema
    - Requisiti funzionali: si indicano le funzionalità che deve supportare il sistema
    - Requisiti non funzionali: si indicano i vincoli che non sono relati alle funzioni del sistema
    - Modelli del sistema
      - Scenari: istanze di caso d'uso
      - Casi d'uso: rappresentazione funzionale del sistema. Descrive il sistema come flusso di eventi.
      - Modello a oggetti: descrizione statica del sistema. Comprende diagramma delle classi.

    - Modello dinamico: descrizione dinamica del sistema. Comprende i diagrammi di attività, i sequence diagram e gli state-chart diagram.
    - User interface: comprende i navigational path e i mock-up
- Glossario: definizione dei termini utilizzati all'interno del documento. Serve per evitare le incomprensioni e le ambiguità.

1. **Giustificare la seguente affermazione: "l'acquisizione della conoscenza è un processo non lineare"**
   Con l'aggiunta di un singolo dato si può invalidare l'intero modello

2. **Nei diagrammi state chart un'azione esprime sempre:**
   un comportamento atomico eseguito in punti specifici nella macchina a stati

3. **I tre tipi principali di interazione tra i partecipanti a un progetto sono:**
   reporting, decision, communication

4. **quale attività non fa parte delle attività di requirements elicitation?**
   identificare gli oggetti boundary

5. **qual è l'esempio corretto di bilanciamento (trade-off) tra obiettivi di progetto?**
   Tempo di consegna vs qualità

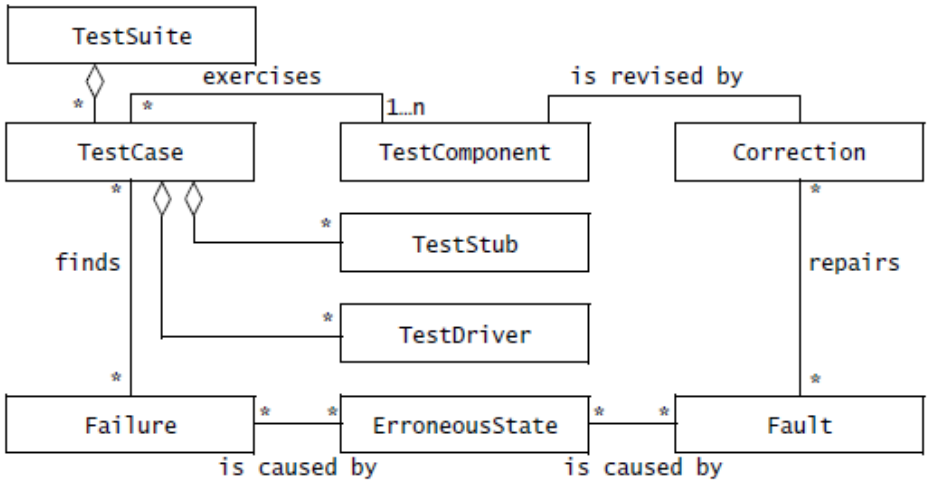
6. **l'attività di specifica delle interfacce include:**
   specificare i contratti, identificare le boundary conditions

7. **fornire una definizione di test case**
   è un insieme di input e risultati aspettati che esercita un componente di test con lo scopo di causare fallimenti e rilevare errori.

8. **Il path testing costituisce una tecnica di:**
   whitebox testing

9. **I cambi a un configuration item associati con una singola revisione di una configurazione vengono chiamati:**
   change set

10. **Riprodurre il diagramma delle classi che rappresenta gli elementi usati durante il testing e le relazioni tra di essi**



1. **Il lavoro di un team cross-funzionale:**
   ha impatto sullo sviluppo di molti sottosistemi

2. **Quali delle seguenti affermazioni è falsa. Il riesame del cliente (client review) è usato per:**
   ispezionare il codice sorgente

3. **Quale di queste affermazioni è falsa? Un project review è condotto**
   Durante il test di accettazione del cliente

4. **Quale tra i seguenti concetti non è collegato al pattern Facade?**
   Pattern proxy

5. **Le boundary condition:**
   definiscono come il sistema viene inizializzato

6. **Il re factoring:**
   migliora la leggibilità del codice sorgente

7. **cosa contiene il test report summary:**
   la lista di tutte le failure riscontrate nel testing

8. **l'interfaccia di un sistema include:**
   nome operazioni, tipi input e output

9. **una capability list risponde alla domanda:**
   a quali oggetti questo attore può accedere

10. **quali dei seguenti concetti fanno parte del Configuration Management:**
    version, repository, software library

1. **in UML uno stato è:**
   una condizione soddisfatta dagli attributi di un oggetto

2. **una specifica software si dice realistica quando:**
   il sistema software può essere implementato entro i vincoli

3. **in UML un'associazione tra due classi è:**
   la rappresentazione di un insieme di collegamenti tra oggetti

4. **uno stereotipo è un:**
   meccanismo di estensione per classificare elementi UML

5. **descrivere l'attività di acquisizione della conoscenza in ingegneria del software**
   (requirement elicitation) è la prima attività che si realizza durante lo sviluppo software. Vengono raccolti tutti i dati relativi al problema e successivamente vengono formalizzati in vere e proprie informazioni. L'acquisizione della conoscenza è un'attività non lineare in quanto un'introduzione di una nuova conoscenza può invalidare il modello.

6. **La domanda "quale documentazione dovrebbe essere messa a disposizione dell'utente?" fa parte dei requisiti di:**
   usabilità

7. **Quali tra le seguenti specifiche non vengono definite durante la fase di scomposizione di un sistema software?**
   Casi d'uso limite, oiettivi del progetto

8. **Elencare le attività della fase "specifica delle interfacce":**
   specificare precondizioni, post condizioni, invarianti
   indicare i metodi: firma, tipo di parametri e tipo di ritorno

9. **Un diagramma PERT è?**
   Uno schedule rappresentato come grafo aciclico

   Disegnare un diagramma di attività che descrive una situazione…

1. **Quale tra i seguenti è un requisito funzionale?**
   Il sistema dovrà rilasciare titoli di viaggio

2. **in UML una vista è:**
   un sottoinsieme del modello del sistema

3. **quale dei seguenti aspetti non attiene alla validazione dei requisiti?**
   Decomponibilità

4. **un oggetto partecipante è un oggetto introdotto da:**
   diagramma di sequenza

5. **quale tra i seguenti tipi di work product non è interno?**
   Modello funzionale

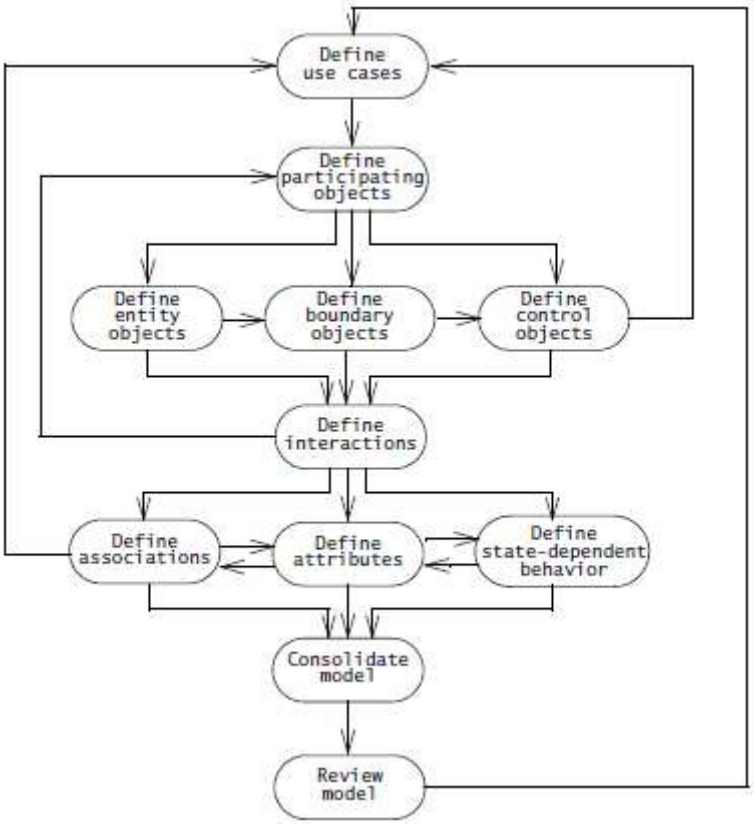
6. **in UML le classi di analisi rappresentano:**
   un'astrazione ben definita del dominio di applicazione

7. **in un diagramma delle attività una ricongiunzione (join) sincronizza due o più flussi concorrenti quando:**
   ha molte transizioni in ingresso e una in uscita

8. **i diagrammi di collaborazione rappresentano la sequenza di messaggi:**
   dall'alto verso il basso

9. **quali tra le seguenti domande è estranea alla determinazione del requisito non funzionale di affidabilità?**
   Quanti utenti concorrenti dovrà supportare il sistema?

10. Riprodurre il diagramma UML delle attività di analisi a partire dalla definizione dei casi d'uso



1. **Quale tra i seguenti è un requisito non funzionale?**
   Il sistema dovrà essere facile da utilizzare

2. **In quale dei seguenti casi è inopportuno usare il procedimento di falsificazione?**
   Durante la raccolta dei requisiti

3. **Quale delle seguenti affermazioni è sbagliata? La struttura di UML è costituita da:**
   linguaggi orientati agli oggetti

4. **Quale domanda, tra le seguenti, è inadatta a individuare gli attori di un sistema?**
   Chi paga per la realizzazione di un sistema?

5. **Quale tra i seguenti requisiti è verificabile?**
   Il sistema fornirà risposte in meno di un secondo

6. **In UML un'associazione tra due classi è:**
   la rappresentazione di un insieme di collegamenti tra oggetti

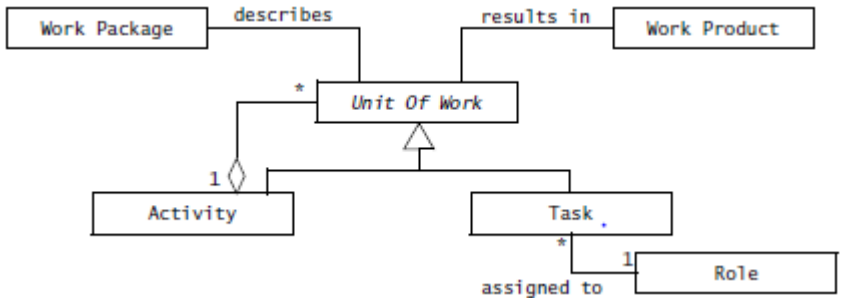
7. **Nei diagrammi delle classi ogni classe può essere composta da:**
   nome, attributi, operazioni

8. **Un diagramma state chart è una notazione per:**
   descrivere la sequenza di stati di un oggetto

9. **Nei diagrammi di sequenza quale delle seguenti euristiche è inadatta?**
   Oggetti entità accedono a oggetti di controllo

10. **Quali dei seguenti elenchi di concetti vengono tutti gestiti dal project manager per organizzare un progetto?**
    Ruoli, work product, tasks, schedule

11. **La specifica dell'unità di lavoro che deve essere svolto per completare un task o un'attività è descritta da un work package.**



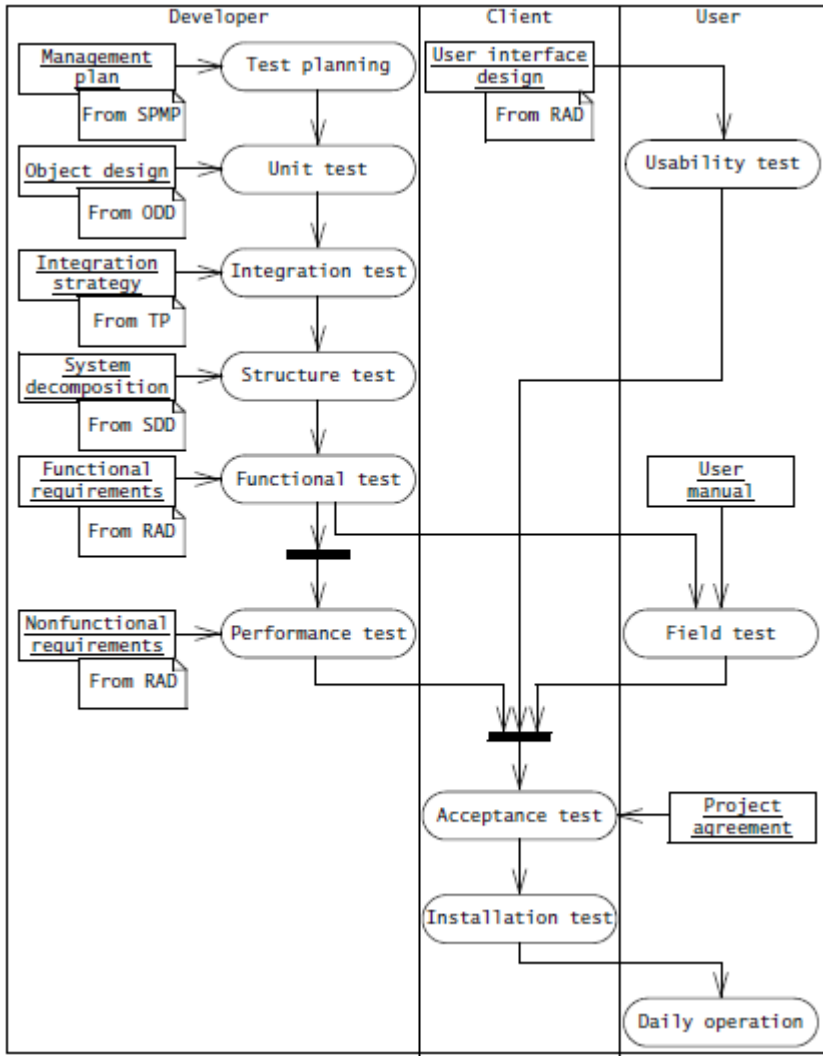
1. **Fornire una definizione di: entity, boundary e control objects**
   Entity: oggetti che memorizzano dati persistenti del sistema
   Boundary: oggetti responsabili dell'interazione tra utente e sistema
   Control: oggetti responsabili della gestione delle funzionalità del sistema

2. **Un difetto di disegno o di codifica che può causare il comportamento anomalo di un componente software viene detto:**
   fault

3. **I seguenti criteri vengono usati per determinare le classi di equivalenza nel testing blackbox**
   Copertura, disgiunzione, rappresentatività

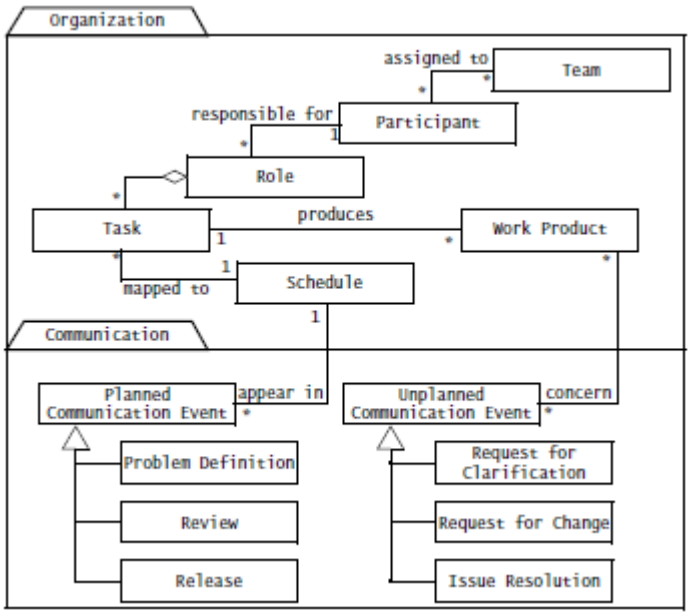
4. **Elencare i quattro tipi di trasformazione utilizzabili nello spazio dei modelli e nello spazio del codice**
   Trasformazioni di modello: opera sui modelli a oggetto
   Re factoring: opera sul codice sorgente (non cambia la funzionalità)
   Forward engineering: produce un template di codice sorgente che corrisponde ad un modello a oggetti
   Reverse Engineering: parte dal codice sorgente per risalire al modello.

5. **Riprodurre il diagramma di attività UML che rappresenta le attività di testing ed i corrispondenti work product**



1. **Descrivere l’attività di problem solving in ingegneria del software**
   1. Formulare il problema.
   2. Analizzare il problema.
   3. Ricerca di soluzioni.
   4. Decidere la soluzione appropriata.
   5. Specificare la soluzione.

2. **Diagramma delle classi che mette in relazione i concetti di comunicazione e organizzazione**



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.

Descrivere le categorie dei requisiti non funzionali secondo il modello FURPS+
Usabilità: quanto il sistema è user-friendly.
Affidabilità: è affidabile se fa quello per cui è stato progettato.
Performance: comprende attributi quali tempo di risposta, disponibilità, precisione, throughput.
Supportabilità: capacità del sistema di supportare i cambiamenti dopo essere stato sviluppato.

Descrivere la differenza tra oggetti Entity, Boundary, Control
Entity Object: dati persistenti del sistema
Boundary Object: oggetti che permettono l'interazione tra utente e sistema
Control Object: oggetti che si occupano di gestire la logica del sistema

Descrivere i criteri per la convalida dei requisiti
Correttezza: si rappresenta accuratamente il sistema
Consistenza: non ci sono contraddizioni tra i requisiti funzionali e non
Non ambiguità: è descritto esattamente un sistema
Completezza: è rappresentato tutto il sistema (tutti gli scenari)

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 o 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. Il flusso degli eventi eccezionali è portato fuori dal flusso degli eventi principali per rendere più chiaro il caso d'uso.

Descrivere la differenza tra il modello di processo incrementale ed il modello di processo evolutivo.
Sviluppo incrementale**:** ogni versione aggiunge nuove funzionalità/sottosistemi
Sviluppo iterativo (evolutivo)**:** da subito sono presenti tutte le funzionalità/sottosistemi che vengono successivamente raffinate, migliorate

Descrivere le categorie di requisiti non funzionali (requisiti di qualità e pseudo-requisiti)
Requisiti di qualità:
Usability: è la facilità con cui l'utente può utilizzare il sistema. Include, per esempio, convenzioni adottate per l'interfaccia utente e un manuale di utilizzo.
Reliability: è un requisito di affidabilità. È l'abilità di un sistema o di una componente di compiere le sue funzioni richieste sotto determinate condizioni per un periodo di tempo specificato.

Performance: requisiti che riguardano le caratteristiche del sistema, come: tempo di risposta, throughput, disponibilità e precisione.
Supportability: requisiti che riguardano la facilità di modifiche al sistema dopo la sua distribuzione. Esempi sono l'adattabilità, la manutenibilità, e l'internazionalizzazione.

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

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 incrementale:
Le fasi alte del processo sono completamente realizzate. Il sistema così progettato 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, o insieme di attività, di integrazione di nuovi sottosistemi prodotti con quelli già in esercizio.
Modello iterativo:
da subito sono presenti tutte le funzionalità/sottosistemi che vengono successivamente raffinate, migliorate.

Descrivere i concetti di messaggi, attivazione e lifelines in un diagramma di sequenza 
Messaggio: scambio di eventi tra oggetti
Attivazione: tempo in cui un oggetto è impegnato in una determinata operazione
Lifelines: tempo di vita dell'oggetto

Descrivere i concetti di ruolo e qualificatori nelle associazioni tra classi in UML 
Un ruolo è una stringa che etichetta un'associazione. Indica la funzione di ogni classe rispetto all'associazione.
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).

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.

Ddata la seguente tabella delle attività disegnare il relativo diagramma di pert ed individuare il percorso critico

Attività	Durata (gg)	Dipendenze
T1	3	
T2	7	T1
T3	3	T1
T4	4	T2
T5	3	T2,T3
T6	4	T3
T7	5	T4,T5

Critical path: T1->T2->T4->T7

Descrivere la differenza tra aggregazione e composizione in un diagramma delle classi UML
L'aggregazione indica che le parti coinvolte in questa relazione possono esistere in maniera indipendente dall'intero.
La composizione indica che le parti coinvolte nella relazione non possono essere indipendenti dall'intero.

Descrivere le categorie di pseudo-requirements secondo il modello FURPS+
Requisiti di implementazione: vincoli per la realizzazione del sistema, compreso l'uso di attrezzi specifici, linguaggi di programmazione o piattaforme hardware.
Requisiti di interfaccia: vincoli imposti da sistemi esterni, inclusi sistemi legacy e scambi di formati.
Requisiti di operazione: sono vincoli sulla amministrazione e gestione del sistema.
Requisiti per l'imballaggio(Packaging): vincoli sulla consegna effettiva del sistema.
Requisiti di legge: riguardano le leggi sulle licenze, regolamentazione e certificazione.

Quali sono le principali difficoltà che si incontrano durante la raccolta dei requisiti?
Le difficoltà principali sono legate alla comunicazione. Fraintendimenti e omissioni infatti possono portare spesso al fallimento dello sviluppo. Per evitare questi problemi, gli sviluppatori hanno a disposizione una serie di strumenti: convenzioni, infatti se gli sviluppatori si accordano su come rappresentare i modelli, risolvono la maggior parte dei problemi di comunicazione; strumenti CASE che servono per generare modelli e documenti; procedure, validazioni periodiche.

Descrivere il concetto di transizione in un diagramma di stato UML
Una transizione è un passaggio da uno stato ad un altro che si verifica allo scatenarsi di un evento a cui segue un'azione.

Spiegare da quali parti è composto un documento di analisi e specifica dei requisiti
È composto da tre sezioni:

- Modello funzionale (casi d'uso e scenari)
- Modello a oggetti (class diagram)

- Modello dinamico (sequence diagram, statechart diagram)

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.

Descrivere il meta-modello a spirale per il ciclo di vita del software. Perché 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).

Descrivere la differenza tra azione e attività in un diagramma di stato
Azione: comportamento atomico che può essere eseguito in uno specifico punto dello stato della macchina.
Attività: è legata ad uno stato. Può durare un lasso di tempo considerevole e può essere interrotta da un evento.

Descrivere i vari tipi di oggetti che è possibile individuare in fase di analisi dei requisiti e in che modo vengono individuati
Durante la fase di analisi è possibile individuare tre oggetti:
Entity Object: è responsabile dei dati persistenti, rappresenta quel dato che deve sopravvivere ad un'esecuzione del sistema.
Boundary Object: è responsabile dell'interazione tra l'utente e il sistema tramite interfaccia.
Control Object: è responsabile della gestione della logica del sistema.
Vengono individuati attraverso l'euristica di Abbot, che effettua un'analisi semantica della descrizione di uno scenario.

Spiegare la differenza tra requisiti funzionali, requisiti non funzionali e pseudo-requirements
Il requisito funzionale è una funzione che il sistema deve supportare.
Il requisito non funzionale indica gli aspetti che non sono legati direttamente alle funzionalità del sistema.
Gli pseudo-requirements sono specifici strumenti e tecnologie che il sistema deve utilizzare e rispettare.

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

il modello a cascata esegue i vari passi del processo di sviluppo software in maniera sequenziale.
Vantaggi: ogni fase è ben definita e non vengono mai inserite funzionalità indesiderate.
Svantaggi: i requisiti vengono indicati all'inizio e non controllati durante le varie fasi,ciò può comportare alti rischi e in alcuni casi porta al fallimento. Un altro svantaggio sta nel fatto che i gruppi non lavorano in parallelo quindi potrebbero esserci gruppi inattivi in attesa del compimento di una fase precedente.
Le varianti proposte per risolvere i problemi legati agli svantaggi si trovano nel modello iterativo.

Spiegare la differenza tra le seguenti qualità del software: robustezza, affidabilità e correttezza.

Robustezza: il grado con cui un sistema o una sua componente può funzionare correttamente in presenta di input invalidi o di condizioni stressanti dell'ambiente esterno.
Affidabilità: la probabilità che il software non causerà fallimenti per un periodo specificato di tempo.
Correttezza: il sistema viene rappresentato secondo le specifiche dell'utente.

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

La tracciabilità indica il fatto che ogni funzione all'interno del codice sorgente deve poter essere tracciata nel corrispondente insieme di requisiti e viceversa.

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.

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

Uno scenario rappresenta un flusso di eventi dal punto di vista dell'utente. Un caso d'uso identifica una funzione che il sistema deve supportare. Uno scenario è un'istanza di un caso d'uso e un caso d'uso rappresenta tutti i possibili scenari.

Descrivere i concetti di accoppiamento e coesione
L'accoppiamento indica le dipendenze tra i sottosistemi. Se due sistemi sono "fortemente accoppiati" cambiamenti in uno dei due si ripercuotono fortemente sull'altro. Viceversa, se sono "debolmente" accoppiati, cambiamenti in uno dei due hanno impatto lieve sull'altro (sono quasi indipendenti). Si vuole realizzare un sistema che minimizzi l'accoppiamento.
La coesione indica le dipendenze tra elementi dello stesso sottosistema. Si vuole progettare un sistema con alta coesione, in quanto se c'è coesione tra gli elementi si tendono a svolgere le stesse operazioni.

Descrivere i concetti di stratificazione e partizionamento

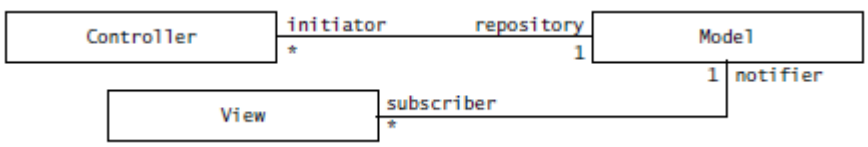
Stratificazione: suddivisione orizzontale in layer, in cui ogni layer rappresenta un'astrazione di un concetto e fornisce servizi al layer immediatamente superiore.

Partizionamento: decomposizione verticale di un sistema in sottosistemi alla pari, ognuno responsabile di una diversa classe di servizi.

Descrivere la differenza tra architetture aperte e architetture chiuse
In un'architettura aperta, ogni layer può comunicare con i layer dei livelli inferiori. Mentre in un'architettura chiusa, ogni layer può comunicare solamente con il layer immediatamente inferiore. Entrambi però forniscono servizi al layer immediatamente superiore.

Descrivere lo stile architetturale Repository
I sottosistemi fanno riferimento ad un "deposito" centralizzato che mantiene tutte le informazion.
Essi si rivolgono al repository per operare sui dati.
Vantaggi: sono adatti ad applicazioni con cambiamento costante.
Svantaggi: il repository centralizzato diventa un bottleneck in caso di un grande numero di accessi.
Ogni cambiamento sul repository ha impatto sui sottosistemi.

Descrivere lo stile architetturale Model/View/Controller
Questo stile è un caso particolare dell'architettura repository. Possiamo individuare tre tipi di sottosistemi:
Model: sottosistema che mantiene la conoscenza del dominio applicativo.
View: sottosistema che si occupa di mostrare all'utente il dominio applicativo.
Controller: sottosistema che gestisce le interazioni con l'utente.



Descrivere lo stile architetturale client/server
Il sottosistema server fornisce dei servizi al sottosistema (o ai sottosistemi) client. Questo stile è vantaggioso quando bisogna interagire con un gran numero di dati distribuiti in vari sottosistemi. Gli svantaggi riguardano il fatto che i due sottosistemi client e server hanno ruoli diversi.

Descrivere lo stile architetturale Peer-to-Peer
Questo stile è una generalizzazione del client/server, in quanto i sottosistemi possono essere sia client che server, nel senso che ognuno può richiedere o offrire servizi.

Descrivere gli stili architetturali a tre e quattro lvelli
Lo stile a tre livelli è il three tier in cui individuiamo:

- Livello dell'interfaccia: si occupa dell'interazione con l'utente e contiene tutti gli oggetti che interagiscono con esso.
- Livello delle applicazioni logiche: include gli oggetti di controllo.
- Livello di storage: mantiene i dati persistenti.

Il vantaggio sta nel fatto che vi è una debole dipendenza tra i layer.
Lo stile a quattro livelli è uguale al three tier ma decompone il livello dell'interfaccia in Presentation Client layer e Presentation Server layer. Il primo è allocato sulla macchina client, il secondo può essere allocato su uno o più server.

Descrivere i design goal relativi alle Performance
Includono i requisiti imposti sul sistema in termini di spazio e velocità:
Tempo di risposta : il tempo entro quanto una richiesta da parte di un utente deve essere soddisfatta.
Troughput : il numero di task che il sistema porta a compimento in un periodo di tempo prefissato.
Memoria : lo spazio richiesto affinché il sistema possa funzionare.

Descrivere i design goal relativi a Dependability
Determinano quanto sforzo deve essere speso per minimizzare i crash del sistema e le loro conseguenze. Includono i concetti di:
robustezza, affidabilità, disponibilità, tolleranza ai guasti, sicurezza.

Descrivere i design goal relativi a Maintenance
Determinano la difficoltà nel cambiare il sistema dopo la distribuzione. Includono i concetti di: estensibilità, modificabilità, adattabilità, portabilità, tracciabilità dei requisiti, leggibilità.

Descrivere le diverse strategie per la gestione della memorizzazione
Una volta identificati i dati persistenti, ci sono tre strategie per memorizzarli:
Flat files: astrazione di memorizzazione fornita dai sistemi operativi.
Database relazionali: strutture dati che contengono informazioni memorizzate in tabelle con uno schema predefinito. Il DBMS permette di manipolare i dati.

Database orientati agli oggetti: simile ai database relazionali ma i dati vengono memorizzati come oggetti e associazioni.

Descrivere vantaggi e svantaggi nella scelta tra flat files, database relazionali ed object-oriented
I flat files sono indicati quando si ha un gran numero di dati momentanei.
I database sia relazionali che object-oriented sono preferibili quando si hanno accessi multipli o concorrenti.
In particolare:
si preferiscono i database relazionali quando abbiamo query complesse, mentre i database object-oriented quando si ha un intenso uso di associazioni per recuperare i dati.

Descrivere i diversi approcci per la specifica del controllo degli accessi agli oggetti di un sistema software
Il controllo degli accessi si effettua attraverso la matrice degli accessi. La matrice sulle righe ha gli attori del sistema e sulle colonne le classi del sistema. Un'entry della matrice, chiamata "diritto d'accesso" specifica le operazioni che possono essere eseguite su un'istanza della classe dall'attore. La matrice si può rappresentare attraverso tre approcci:
tabella d'accesso globale: rappresenta esplicitamente ogni cella nella matrice come una tripla (attore, classe, operazione). Se tale tripla non è presente l'accesso è negato.
Lista di controllo degli accessi: associa una coppia (attore, operazione) a ciascuna classe.
Capability: associa una coppia (classe, operazione) a ciascun attore.

Descrivere i tre meccanismi principali per il flusso di controllo
Procedure-driven: le operazioni attendono l'input ogni volta che hanno bisogno di dati da un attore.
Event-driven control: un ciclo principale aspetta eventi esterni. Ogniqualvolta si verifica un evento, questo viene smistato all'oggetto appropriato.
Threads: variazione concorrente di procedure-driven.

Spiegare la differenza tra ereditarietà di specifica ed ereditarietà di implementazione e fornire un esempio per ciascun tipo di relazione
L'ereditarietà di specifica (chiamata anche Interface Inheritance) eredita da una classe astratta i metodi non implementati(implementa una interfaccia) mentre l'ereditarietà di implementazione eredita tutti i metodi dalla superclasse, ma è sconsigliato usarla, si usa la delegazione, altrimenti qualunque utilizzatore della classe potrebbe chiamare un metodo della superclasse.

Spiegare l'importanza delle interfacce. Per quale motivo si dovrebbe programmare sempre riferendosi ad un' interfaccia, piuttosto che ad un'implementazione?
Le interfacce forniscono uno strumento per realizzare astrazioni di funzionalità offerte dai sottosistemi. Riferirsi ad un'interfaccia permette allo sviluppatore di non preoccuparsi

dell'implementazione dei metodi ma di riferirsi ad essi a scatola chiusa (conosce l'input, si aspetta un output ma non conosce com'è fatto il metodo).

Descrivere il Bridge Design Pattern e fornire un problema di progettazione in cui potrebbe essere adottato

È un pattern strutturale, ovvero si occupa delle modalità di composizione di classi e oggetti per formare strutture complesse.
Problema : sviluppare, testare e integrare sottosistemi realizzati da differenti sviluppatori in maniera incrementale. Per risolvere questo problema utilizziamo il design pattern bridge.
Nome : Bridge
Descrizione del problema : separare un'astrazione da un'implementazione così che una diversa implementazione possa essere sostituita eventualmente a runtime
Soluzione : una classe Abstraction definisce l'interfaccia visibile al codice client.
Implementor è una classe astratta che definisce i metodi di basso livello disponibili ad Abstraction.
Un'istanza di Abstraction mantiene un riferimento alla sua istanza corrispondente di Implementor.
Abstraction ed Implementor possono essere raffinate indipendentemente.
Conseguenze : disaccoppiamento tra interfaccia ed implementazione (un'implementazione non è più legata in modo permanente ad un'implementazione. L'implementazione di un'astrazione può essere configurata durante l'esecuzione. La parte di alto livello di un sistema dovrà conoscere soltanto le classi Abstraction ed Implementor); maggiore estendibilità (le gerarchie Abstraction ed Implementor possono essere estese indipendentemente); mascheramento dei dettagli dell'implementazione ai client (i client non devono preoccuparsi dei dettagli implementativi).

Descrivere l' Adapter Design Pattern e fornire un problema di progettazione in cui potrebbe essere adottato

È un pattern strutturale.
Nome : Adapter
Descrizione del problema : convertire l'interfaccia utente di una classe legacy in un'interfaccia diversa che il cliente si aspetta, in maniera tale che classi diverse possano operare insieme nonostante abbiano interfacce incompatibili.
Soluzione : ogni metodo dell'interfaccia verso il client è implementato in termini di richieste alla classe legacy. Ogni conversione tra strutture dati o variazioni nel comportamento sono realizzate dalla classe Adapter. Viene usata per fornire una nuova interfaccia a componenti legacy esistenti.
Conseguenze : se il Client utilizza ClientInterface allora può utilizzare qualsiasi istanza dell'Adapter in maniera trasparente senza dover essere modificato. L'Adapter lavora con la LegacyClass e con tutte le sue sottoclassi. L'adapter pattern viene utilizzato quando l'interfaccia e la sua implementazione esistono già e non possono essere modificate.

Descrivere il Abstract Factory Design Pattern e fornire un problema di progettazione in cui potrebbe essere adottato

È un pattern creazionale ovvero fornisce un'astrazione del processo di istanziazione degli oggetti e aiuta a rendere un sistema indipendente dalle modalità di creazione,

composizione e rappresentazione degli oggetti utilizzati
Problema : Consideriamo un'applicazione per un'abitazione intelligente: l'applicazione riceve eventi da sensori ed attiva comandi per dispositivi. L'interoperabilità in questo dominio è debole, di conseguenza è difficile sviluppare una singola soluzione SW per tutte le aziende.
Nome: Abstract Factory (Kit).
Descrizione del problema: Fornire un'interfaccia per la creazione di famiglie di oggetti correlati o dipendenti senza specificare quali siano le loro classi concrete.
Soluzione: Una piattaforma (es., un sistema per la gestione di finestre) è rappresentato con un insieme di AbstractProducts, ciascuno dei quali rappresenta un concetto (es., un bottone).
Una classe AbstractFactory dichiara le operazioni per creare ogni singolo prodotto. Una piattaforma specifica è poi realizzata da un ConcreteFactory ed un insieme di ConcreteProducts.
Conseguenze : Isola le classi concrete (Poiché Abstract Factory incapsula la responsabilità e il processo di creazione di oggetti prodotto, rende i client indipendenti dalle classi effettivamente utilizzate per l'implementazione degli oggetti). E' possibile sostituire facilmente famiglie di prodotti a runtime (una factory concreta compare solo quando deve essere istanziata). Promuove la coerenza nell'utilizzo dei prodotti. Aggiungere nuovi prodotti è difficile poiché nuove realizzazioni devono essere create per ogni factory.

Descrivere lo Strategy Design Pattern e fornire un problema di progettazione in cui potrebbe essere adottato
È un pattern comportamentale ovvero si occupa di algoritmi e dell'assegnamento di responsabilità tra oggetti collaboranti.
Problema : Consideriamo un'applicazione mobile che si deve connettere a diversi tipi di rete, facendo lo switch in base alla posizione ed al costo della rete disponibile. Si vuole che l'applicazione possa essere adattata a futuri protocolli di rete senza dover ricompilare l'applicazione.
Nome: Strategy (Policy)
Descrizione del problema: Definire una famiglia di algoritmi, incapsularli e renderli intercambiabili. Permette agli algoritmi di variare indipendentemente dai client che ne fanno uso.
Soluzione: Un Client accede ai servizi forniti da un Context. I servizi del Context sono realizzati utilizzando uno dei diversi meccanismi, come deciso da un oggetto Policy. La classe astratta Strategy descrive l'interfaccia che è comune a tutti i meccanismi che Context può usare. La classe Policy configura Context per usare un oggetto ConcreteStrategy.
Conseguenze : Un'alternativa all'ereditarietà (L'ereditarietà legherebbe staticamente il comportamento nel Context e mescolerebbe l'implementazione del Context con l'implementazione dell'algoritmo. Inoltre sarebbe impossibile modificare l'algoritmo dinamicamente). Policy decide quale Strategy è la migliore, in base alle circostanze correnti Scelta dell'implementazione (Attraverso Strategy è possibile fornire diverse implementazioni dello stesso comportamento). Le strategie eliminano i blocchi condizionali

Descrivere il Proxy Design Pattern e fornire un problema di progettazione in cui potrebbe essere adottato

Un proxy, nella sua forma più generale è una classe che funziona come interfaccia per qualcos'altro. L'altro potrebbe essere qualunque cosa: una connessione di rete, un grosso oggetto in memoria, un file e altre risorse che sono costose o impossibili da duplicare.
Un esempio ben conosciuto di proxy pattern è l'oggetto reference dei puntatori.
Nelle situazioni in cui molte copie di un oggetto complesso devono esistere, il proxy pattern può essere adottato per incorporare il Flyweight pattern per ridurre l'occupazione di memoria dell'oggetto. Tipicamente viene creata un'istanza di oggetto complesso, e molteplici oggetti proxy, ognuno dei quali contiene un riferimento al singolo oggetto complesso. Ogni operazione svolta sui proxy viene trasmessa all'oggetto originale. Una volta che tutte le istanze del proxy sono distrutte, l'oggetto in memoria può essere deallocato.

Descrivere il Facade Design Pattern e fornire un problema di progettazione in cui potrebbe essere adottato

Letteralmente façade significa "facciata", ed infatti nella programmazione ad oggetti indica un oggetto che permette, attraverso un'interfaccia più semplice, l'accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro, nonché a blocchi di codice complessi.

Descrivere i concetti di design pattern, frame work e libreria di classi in riferimento all'attività di riuso

I design pattern sono dei template di soluzioni a problemi comuni, raffinate nel tempo dagli sviluppatori. Un framework è una struttura di supporto su cui un software può essere organizzato e progettato. Lo scopo di un framework è di risparmiare allo sviluppatore la riscrittura di codice già steso in precedenza per compiti simili. Le librerie di classe sono il più generico possibile e non si focalizzano su un particolare dominio di applicazione fornendo solo un riuso limitato.

Descrivere i diversi tipi di contratto che possono essere specificati su una classe

Invariante: predicato riferito ad una classe che deve essere sempre vero, sia prima che dopo l'esecuzione di un'operazione.
Pre-condizione: predicato che deve risultare vero prima dell'esecuzione di un'operazione.
Post-condizione: predicato che deve risultare vero dopo l'esecuzione di un'operazione.

Descrivere i diversi tipi di trasformazione tra i modelli (modello a oggetti e codice sorgenti)

Ci sono quattro tipi d trasformazione:

- Refactoring: trasformazione che opera sul codice sorgente
- Forward engineering: produce un template di codice sorgente che corrisponde ad un modello a oggetti.
- Reverse engineering: si applica quando conosciamo il codice del sistema e dobbiamo ricavarne il design.

Descrivere in che modo è possibile mappare un'associazione uno a molti tra due oggetti e come memorizzare tale relazione in un database relazionale. Fornire un esempio.
L'associazione viene mappata come attributo delle entità con molteplicità "molti". Tale relazione viene memorizzata sottoforma di campo di una tabella.
Es : supponiamo che ad un cliente possano corrispondere più conti. Poiché i conti non hanno un ordine specifico possiamo usare un insieme di riferimenti per modellare la parte "molti" dell'associazione.

Descrivere in che modo è possibile mappare un'associazione molti a molti tra due oggetti e come memorizzare tale relazione in un database relazionale. Fornire un esempio.
L'associazione viene mappata come una nuova tabella contenenti le chiavi primarie delle entità coinvolte.
Esempio: supponiamo di avere due entità "persona" e "autobus" e la relazione "viaggio". Si crea una nuova tabella che contiene come chiavi primarie le chiavi esterne delle due entità.

Descrivere in che modo è possibile mappare una relazione di ereditarietà su di uno schema relazionale (mapping verticale e mapping orizzontale).
I database relazionali non supportano l'ereditarietà. Esistono due opzioni per mappare l'ereditarietà in uno schema di un database :
Mapping verticale : la superclasse e la sottoclasse sono mappate in tabelle distinte. Quella della superclasse mantiene una colonna per ogni attributo definito nella superclasse e una colonna che indica quale tipo di sottoclasse rappresenta quell'istanza. La tabella della sottoclasse include solo gli attributi aggiuntivi e una chiave esterna che la collega alla tabella della superclasse. Mapping orizzontale : non esiste una tabella per la superclasse ma una tabella per ciascuna sottoclasse.

Descrivere i concetti di fallimento, stato erroneo e fault(difetto)
Il fallimento si verifica quando il comportamento osservato del sistema è diverso da quello atteso.
Uno stato erroneo è uno stato in cui se si continua ad utilizzare il sistema si potrebbe andare incontro ad un fallimento.
Un fault è un errore hardware/software.

Descrivere la differenza tra fallimento meccanico ed algoritmico
Il fallimento meccanico è causato da un problema hardware. Il fallimento algoritmico dipende da problemi di implementazione o progettazione del sistema.

Descrivere le differenze tra testing black-box e white-box
il testing black-box si effettua senza conoscere l'implementazione del sistema. Fornisci un input e lo confronti con il risultato atteso (oracolo).
Il testing white-box si effettua,invece, indipendentemente dall'input, testando gli statement del codice.

Elencare i diversi tipi di test di usabilità

Scenario test :
Viene presentato a uno o più utenti un Visionary Scenario. Gli sviluppatori determinano quanto velocemente gli utenti comprendono lo scenario, la bontà del modello e come reagiscono gli utenti alla descrizione del nuovo sistema. Lo scenario selezionato dovrebbero essere il più realistico possibile.
Vantaggi: sono economici da realizzare e da ripetere
Svantaggi: gli utenti non possono interagire direttamente con il sistema.
Test di prototipo :
Agli utenti finali viene presentato una parte del software che implementa gli aspetti chiave del sistema : Prototipo verticale. Implementa completamente uno use case.
Prototipo orizzontale. Implementa un singolo layer nel sistema (per esempio un prototipo dell'interfaccia utente).
Prototipo funzionale. Usato per valutare le richieste cruciali (es: tempo di risposta)
Vantaggi: forniscono una vista realistica del sistema all'utente ed il prototipo può essere concepito per collezionare informazioni dettagliate
Svantaggi: richiede un impegno maggiore nella costruzione rispetto agli scenari cartacei
Test di prodotto:
Simile al test di prototipo, eccetto per il fatto che viene utilizzata una versione funzionale Del sistema. Il test può essere affrontato solo dopo che una buona parte del sistema sia stata sviluppata.

Elencare i diversi tipi di test di unità e descrivere una delle tecniche
Il testing di unità si concentra sui blocchi base del sistema: gli oggetti e i sottosistemi.
Ci sono vari tipi di test di unità:
Equivalence Testing: è una tecnica black-box che minimizza il numero di casi di test. I casi di test sono divisi in classi di equivalenza e viene preso un solo caso di test da ogni classe.
Boundary Testing: è un caso specifico di test di test di equivalenza, che si focalizza sui Casi limite delle classi di equivalenza.
Path Testing: è una tecnica white-box che identifica fault nell'implementazione di una componente. L'assunzione dietro al path testing è che provando tutti i possibili percorsi del codice almeno una volta tutte le fault scateneranno delle rispettive failure.
State-based Testing: è una tecnica di testing recente per rilevare failure relative ad caratteristiche degli ambienti ad oggetti, come polimorfismo, binding dinamico e distribuzione di funzionalità su un grande numero di piccoli metodi.

Elencare i diversi tipi di test di integrazione e descrivere una delle tecniche
Big-Bang Integration: consiste nel testare tutte le componenti individualmente e poi unirle; questa tecnica è sconsigliata perché in caso di fault non si può capire facilmente in quale sottosistema risiede.
Bottom-Up Testing: Testa prima i sottosistemi risiedenti agli strati inferiori, avvalorandosi dell'uso dei Driver per simulare quelli superiori, poi vengono testati quelli che chiamano i sottosistemi.

Quest'approccio non è consigliato per sistemi decomposti funzionalmente, perché testa i sottosistemi importanti solo alla fine.
Top-Down Testing: Questa strategia testa prima i sottosistemi che risiedono nello strato superiore e poi l'insieme di quelli testati e quelli che sono chiamati dal componente testato Sandwich Testing: Combina le strategie Bottom-Up e Top-Down. Si seleziona un livello target (minimizzando il numero di stub e di driver) poi vengono eseguiti I test sui livelli superiori e sui livelli inferiori contemporaneamente e successivamente integrati con il livello target.
Sandwich Modificato: Viene testato anche il livello di target, e quindi il target con il livello superiore(che viene sostituito al driver) ed il target con il livello inferiore(che viene sostituito allo stub).

Elencare i diversi tipi di test di sistema e descrivere una delle tecniche
Testing di unità e di integrazione si focalizzano sulla ricerca di bug nelle componenti individuali e nelle interfacce tra le componenti. Il testing di sistema assicura che il sistema completo sia conforme ai requisiti funzionali e non funzionali. Attività :
Testing funzionale. Test dei requisiti funzionali
Pilot Testing. Test di funzionalità comuni, fra un gruppo selezionato di utenti finali, nell'ambiente target
Testing di prestazioni. Test dei requisiti non funzionali
Testing di accettazione. Test di usabilità, delle funzionalità e delle prestazioni effettuato dal cliente nell'ambiente di sviluppo
Testing di installazione. Test di usabilità, delle funzionalità e delle prestazioni effettuato dal cliente nell'ambiente operativo

Descrivere i diversi tipi di test di accettazione
Benchmark test. Il cliente prepara un insieme di test case che rappresentano le condizioni tipiche sotto cui il sistema dovrà operare
Competitor testing. Il nuovo sistema è testato rispetto ad un sistema esistente o un prodotto competitore
Shadow testing. Una forma di testing a confronto, il nuovo sistema e il sistema legacy sono eseguiti in parallelo ed i loro output sono confrontati. Se il cliente è soddisfatto, il sistema è accettato, eventualmente con una lista di cambiamenti da effettuare.