QCM Génie Logiciel

Chapitre I : Introduction au Génie Logiciel

1) Les méthodes Agiles :

a. Est un style de développement centré sur les outils
b. Met l'accent sur la satisfaction du client
c. Basé sur une documentation claire.
d. Consiste à suivre un plan et ne répond pas aux modifications

2) eXtreme Programming XP :

a. Fait une large place aux aspects techniques (règles de développement, prototypes, test…)
b. Est Itératif
c. Distingue clairement les phases Projet
d. Le planning est fait indépendamment du client

3) En Scrum :

a. Ce qui compte, la gestion des ressources matérielles.
b. Ce qui compte la gestion des ressources humaines.
c. Ce qui compte la gestion du temps.
d. Ce qui compte les besoins du client.

4) Le Scrum master :

a. Est le chef du projet
b. Favorise la communication entre les membres de l'équipe.
c. Est le représentant du client.
d. Supervise l'avancement du projet

5) Les 4 phases de RUP sont :

a. Initialisation, élaboration, construction et translation.
b. Initialisation, élaboration, construction et transition.
c. Initialisation, construction, élaboration et transition.
d. Initialisation, construction, élaboration et translation.

6) Les principales étapes du cycle de vie d'un logiciel sont :

a. La spécification, l'analyse, la conception, l'implémentation, test et maintenance.
b. La spécification, l'analyse, la conception, le développement, test et maintenance.
c. La spécification, l'analyse, la conception, le codage, test et maintenance.

7) Le génie logiciel :

a. Est une méthodologie de développement logiciel.

b. Est un ensemble de programmes informatiques.
c. Est un ensemble de documentations.
d. Est une science qui étudie les bonnes pratiques de développement logiciel.

8) Un atelier de génie logiciel :
a. Peut être un système de gestion de base de données.
b. Peut être un outil utilisé lors de la spécification d'un logiciel.
c. Peut être un outil utilisé lors de la conception d'un logiciel.
d. Peut être un outil utilisé lors du test d'un logiciel.

9) Le client participe dans la production d'un logiciel développé suivant :
a. Un modèle en « v ».
b. Un modèle en cascade.
c. Un modèle par prototypage.
d. Un modèle en spirale.

10) Le processus unifié est dit piloté par les cas d'utilisation car :
a. Chaque phase du processus unifié est centrée sur un cas d'utilisation.
b. Chaque cas d'utilisation est analysé, conçu et implémenté dans une itération du processus unifié.
c. Les activités du processus unifié sont réalisées en se référant aux cas d'utilisation.
d. Les cas d'utilisation servent de fil conducteur tout au long du projet.

11) Les 5 activités du processus unifié :
a. Peuvent être réalisées dans une itération.
b. Ne peuvent pas être réalisées dans une itération.
c. S'enchainent dans différentes itérations (chaque activité doit être réalisée dans une itération à part).
d. Peuvent s'enchainer dans les différentes phases du processus unifié.

12) Le rôle du Scrum Master consiste à :
a. Affecter les priorités aux users stories du Product Backlog.
b. Définir les tâches associées à chaque user story dans le Sprint Backlog.
c. Faciliter la mise en pratique de la méthode Scrum dans une équipe.

13) Le génie logiciel :
a. Est une méthode de développement logiciel.
b. C'est la collecte des besoins du client.
c. Utilise des procédures et des outils pour l'automatisation des besoins du client.
d. Un ensemble de méthodes et de langages pour produire des composants logiciels.

14) Une méthode de développement logiciel est :
a. Un processus de production de logiciels.
b. Un modèle de développement logiciel.
c. Une norme de développement logiciel.

d. Un atelier de génie logiciel.

15) Le cycle de vie d'un logiciel représente :

a. La durée de vie d'un logiciel.
b. L'ensemble des étapes pour la réalisation puis la destruction d'un logiciel.
c. Un processus de réalisation d'un logiciel.
d. Un modèle de développement de logiciels.

16) Le cycle de vie en cascade :

a. Représente le déroulement d'activités parallèles pour le développement de logiciels.
b. Ne prend pas en charge la vérification à la fin de chaque étape.
c. Ne permet pas de revenir d'une étape à celle qui la précède.
d. Ne livre pas au client les résultats pendant le développement du logiciel.

17) L'expression des besoins dans le processus unifié :

a. Est finalisée pendant la première phase.
b. Est l'activité principale dans la phase Etude d'opportunité.
c. Est finalisée pendant la première itération.
d. Peut être finalisée lors de la dernière phase.

18) Le rôle du Product Owner consiste à :

a. Accepter ou refuser les users stories proposées par les membres de l'équipe.
b. Valider les priorités affectées aux users stories par les membres de l'équipe.
c. Définir les tâches associées à chaque user story dans le Backlog du sprint.
d. Evaluer les travaux réalisés lors des Daily Scrum.

19) Les méthodes lourdes :

a. Est un style de développement centré sur les outils
b. Met l'accent sur la satisfaction du client
c. Basé sur une documentation claire.
d. Consiste à suivre un plan et ne répond pas aux modifications

20) eXtreme Programming XP :

a. Fait une large place aux aspects techniques (règles de développement, prototypes, test…)
b. Est Itératif
c. Favorise la communication directe
d. Le planning est fait indépendamment du client

21) Le cycle de vie en cascade :

a. Propose de dérouler les phases projet de manière séquentielle
b. Les scénarii de test sont préparés dès le début.
c. Ce qui compte c'est la gestion du temps.
d. Ce qui compte ce sont les besoins du client.

22) Le Scrum Master dans une équipe est :

a. L'expert
b. Le chef de projet
c. Un membre de l'équipe
d. Un facilitateur

23) RUP est:

a. Centré sur L'architecture
b. Une méthode
c. Une méthodologie.
d. Constituée de 5 phases
e. Constituée de 4 phases
f. Incrémental
g. Itératif

24) Un logiciel est :

a. Un ensemble de composants logiciels.
b. Un code exécutable
c. Un cahier de spécification
d. Un manuel d'utilisation
e. Des interfaces ergonomiques.

25) Quel(s) modèle(s) se base(nt) sur les prototypes ?

a. W
b. Linéaire
c. Cascade
d. Spirale

26) Un client veut modifier une tache dans le sprint en cours:

a. On ne modifie pas le sprint en cours mais on prend en considération les changements dans le sprint prochain.
b. En Scrum le client est le roi, alors on modifie le sprint
c. Le client a déjà spécifié ses besoins alors il n'a pas le droit de les changer maintenant

27) C'est le rôle d'un chef de projet :

a. de programmer les composants d'un logiciel.
b. de vérifier le bon déroulement des tâches.
c. d'organiser l'enchaînement des tâches.
d. de fournir une visibilité globale sur un projet.
e. d'écrire la spécification du logiciel.

28) Dans SCRUM, le sprint 0 contient :

a. de la conception
b. du test
c. du codage
d. Isolation des features et des user stories.

29) Il est obligatoire de coller sur les tableaux blancs :

a. Les features
b. Les user stories
c. Les tâches

30) RUP est une démarche de développement :

a. Piloté par l'architecture, centré sur les cas d'utilisation, Itératif et incrémental.
b. Piloté par les cas d'utilisation, centré sur l'architecture, Itératif et incrémental.
c. Piloté par les itérations, centré sur les cas d'utilisation, incrémental

31) Le génie logiciel :

a. Est un modèle de développement logiciel.
b. Consiste à analyser les besoins des clients.
c. S'intéresse aux bonnes pratiques pour le développement de logiciels.
d. Un ensemble de procédures et d'ateliers pour produire des composants logiciels.

32) Une méthodologie de développement logiciel s'appuie sur :

a. Un processus de production de logiciels.
b. Une méthode de développement logiciel.
c. Une norme de développement logiciel.
d. Un outil de développement logiciel.

33) Le client ne participe pas activement dans la production d'un logiciel développé suivant :

a. Un cycle de vie en spirale.
b. Un cycle de vie en cascade.
c. Un cycle de vie par prototypage.
d. Un cycle de vie en « V ».

34) Le modèle de cycle de vie incrémental :

a. Représente le déroulement d'activités parallèles pour le développement de logiciels.
b. Teste tout le produit logiciel à la fin du développement.
c. Permet de revenir d'une étape à celle qui la précède.
d. Livre au client les résultats pendant le développement du logiciel.

35) La principale activité dans la phase de construction du processus unifié est :

a. L'expression des besoins.
b. L'implémentation.
c. Le test.

d. La conception.

36) Le rôle du Scrum Master dans la méthode Scrum consiste à :

a. Accepter ou refuser les user stories proposées par les membres de l'équipe.
b. Valider les priorités affectées aux users stories par les membres de l'équipe.
c. Définir les tâches associées à chaque user story dans le Backlog du sprint.
d. Evaluer les travaux réalisés lors des Daily Scrum.

37) Les méthodes Agiles sont :

a. Des solutions de gestion de projet
b. Basées sur des pratiques et des valeurs
c. Efficaces sur les projets complexes
d. Facile à mettre en place

Chapitre II : La spécification

38) Le processus de spécification logiciel :

a. Permet la collecte des besoins des clients.
b. Est l'ensemble des étapes pour élaborer un cahier de charges.
c. Traite les modifications des besoins après la validation du cahier de charges.

Permet de filtrer les besoins des utilisateurs

39) Le processus de spécification logiciel :

a. Est l'ensemble des étapes permettant la réalisation du logiciel à partir du cahier de charges.
b. Permet de raffiner les besoins des clients.
c. Traite les modifications des besoins avant la validation du cahier de charges.
d. Permet de reformuler des exigences spécifiées.

40) Le processus de spécification logiciel :

a. Est l'ensemble des étapes permettant l'analyse du cahier de charges.
b. Permet d'éliminer les ambiguïtés dans les besoins des clients après leur spécification.
c. Permet d'éliminer les ambiguïtés dans les besoins des clients avant leur spécification.
d. Permet de suivre l'évolution des besoins au cours d'un projet.

41) Le cahier de charge contient :

a. Les besoins fonctionnels et non fonctionnels
b. Les exigences fonctionnelles et non fonctionnelles.
c. Une liste de documents applicables et de références
d. Le contexte du projet

42) Le but de la gestion de configuration :

a. est de maximiser la possibilité d'erreur en minimisant la productivité

b. est de gérer les modifications du logiciel en développement et d'assurer la maintenance
c. est de créer les versions
d. est de tester le code et détecter les erreurs

43) La gestion des versions permet de :

a. Contrôler les modifications apportées à chaque fichier ou composant d'un logiciel.
b. Archiver les états de livrés successifs.
c. Suivre les évolutions dans le temps de la configuration.

S'assurer que chacun des états livrés est cohérent et complet.

44) La spécification des exigences est une phase qui nécessite une validation :

a. Oui
b. Non

45) Un cahier de charge contient obligatoirement :

a. La totalité des besoins du client exprimés de manière non ambiguë
b. La totalité des besoins du client.
c. La majorité des besoins du client.

46) La validation de la spécification vise à :

a. Montrer que les besoins explicités correspondent à ceux des utilisateurs du système
b. S'assurer que les besoins sont exprimés sous une forme vérifiable et avec plus d'ambiguïtés possibles
c. Détecter les erreurs d'incompatibilité et d'incohérence entre les éléments de la spécification.

Chapitre III : La conception

47) La conception logicielle

a. Permet d'analyser les problèmes d'implémentation.
b. Permet de s'assurer que le système réponde aux exigences.
c. Permet d'identifier la meilleure façon d'implémenter un logiciel.
d. Permet d'assurer un travail collaboratif dans l'efficacité et l'organisation.

48) La conception architecturale :

a. Décompose le logiciel en composants définis par leurs interfaces uniquement.
b. Décompose le logiciel en composants définis par leurs fonctions uniquement.
c. Spécifie pour chaque composant logiciel les structures de données et les algorithmes utilisés.
d. Identifie les composants logiciels.

49) Le principe du faible couplage consiste à :

a. Affecter les responsabilités afin de réduire l'impact des modifications.

b. Mesurer les dépendances entre deux éléments.
c. Eviter de lier deux éléments dont l'existence est indépendante.

50) Le principe de forte cohésion consiste à :

a. Regrouper dans un même ensemble des éléments indépendants.
b. Regrouper dans un même ensemble des éléments dépendants.
c. Dissocier dans des ensembles distincts des éléments dépendants.
d. Dissocier dans des ensembles distincts des éléments indépendants.

51) Un composant est :

a. Un élément de petite taille qui sert, par assemblage à la construction de logiciels
b. Définit comme étant une unité de composition avec des interfaces contractualisées et un contexte de dépendance exprimé explicitement
c. Un système faisant partie d'un système plus grand et ayant une interface bien définie

52) Dans le patron MVC :

a. Le modèle est chargé de l'affichage à l'écran.
b. Le modèle Envoie des événements quand les données sont modifiées.
c. Le modèle Envoie des événements correspondant aux actions de l'utilisateur.
d. Plusieurs contrôleurs peuvent modifier le même modèle.

53) Une conception de qualité est une conception :

a. Faible cohésion et faible couplage
b. Forte couplage et faible cohésion
c. Forte cohésion et faible couplage
d. Forte cohésion et forte couplage

54) MVC est

a. Un patron de conception
b. Un patron d'architecture
c. Une architecture logicielle
d. Une architecture physique

55) La conception architecturale est une :

a. Manière de décomposer le logiciel en composants plus simple
b. Architecture à bas niveau
c. Structure et organisation générale du système à concevoir

56) Un système a une cohésion forte si :

a. Les éléments inter reliés sont groupés ensemble
b. Les éléments dépendants sont dans des groupes distincts

57) Avec le patron d'architecture MVC :

a. On peut avoir plusieurs vue sur le même modèle
b. Un seul contrôleur peut modifier le même modèle

c. Toutes les vues seront notifiées des modifications

58) Les patrons :

a. Favorisent la réutilisation

b. Sont plus orientés implémentation que le Framework

c. Utilisent le Framework

d. Favorisent la capitalisation des expériences

59) La couche présentation de l'architecture logique d'une application :

e. Peut être une interface graphique.

f. Peut être un système de messagerie.

g. Peut être une ligne de commandes.

h. Peut être une table dans une base de données.

60) Le principe de faible couplage consiste à :

a. Regrouper dans un même ensemble des éléments indépendants.

b. Regrouper dans un même ensemble des éléments dépendants.

c. Dissocier dans des ensembles distincts des éléments dépendants.

d. Dissocier dans des ensembles distincts des éléments indépendants.

Chapitre IV : Gestion de configuration logicielle

61) Le but de la gestion de configuration :

a. est de maximiser la possibilité d'erreur en minimisant la productivité

b. est de gérer les modifications du logiciel en développement et d'assurer la maintenance

c. est de créer les versions

d. est de tester le code et détecter les erreurs

62) Le contrôle de concurrence pessimiste :

a. Permet de contrôler l'accès à une version logicielle.

b. Permet à deux développeurs de travailler simultanément sur un même programme.

c. Permet le verrouillage des données en cours de traitement par un développeur.

63) Le contrôle de concurrence optimiste :

a. Permet de contrôler l'accès à une version logicielle.

b. Empêche deux développeurs de travailler simultanément sur un même programme.

c. Permet de vérifier l'état d'un programme avant sa modification.

d. Utilise un verrou pour empêcher la modification d'un programme.

64) Le contrôle de concurrence optimiste :

a. Permet de vérifier si un programme est en cours de modification par un premier utilisateur avant de permettre à un deuxième utilisateur de le modifier.
b. Permet de vérifier si un programme n'a pas été modifié par un premier utilisateur avant d'appliquer les modifications effectuées par un deuxième utilisateur.
c. Empêche deux utilisateurs de travailler sur un même programme.
d. Utilise un verrou pour contrôler la modification d'un programme.

Chapitre V : Le test

65) La phase de test :

a. Permet de tester tous les comportements du système.
b. Est la dernière phase du processus de développement logiciel.
c. Permet de montrer l'absence des erreurs dans un livrable.
d. Permet d'identifier les erreurs dans une version logicielle.

66) Le défaut logiciel de type « Segmentation Fault » :

a. Faiblesse dans un système informatique permettant à un attaquant de porter atteinte à l'intégrité de ce système
b. Dysfonctionnement dû à un bug dans des opérations de manipulations de pointeurs ou d'adresses mémoire.
c. Met hors service le logiciel défaillant lors de la tentative de ce dernier d'effectuer des opérations impossibles à réaliser
d. Dysfonctionnement dû à un bug dans les opérations d'allocation de mémoire

67) le test de vérification

a. Est un test Boite Noire.
b. Permet de tester la conformité du système aux documents de spécification.
c. Permet l'amélioration de la productivité des équipes.
d. Permet de montrer que le code est bien construit.

68) Le test unitaire permet de :

a. Valider le bon fonctionnement d'une ou de plusieurs parties développées indépendamment avec le reste de l'application
b. Confirmer que l'application réponde d'une manière attendue aux requêtes qui lui sont envoyées

Tester les fonctions (ou les modules) de code par les programmeurs

69) Un test boîte blanche :

a. Vérifie les détails d'implémentation des composants.
b. Teste l'application dans sa situation réelle d'exploitation.
c. Teste le fonctionnement de l'application dans une configuration spécifique d'un système.
d. Teste l'intégration des différents modules d'une application.

70) Un test de performance permet de :

a. Tester une application avec des données valides.
b. Tester l'application avec des données invalides.
c. Tester l'application avec des valeurs de données aux limites.
d. Tester l'application avec des ressources anormales.

71) L'activité de test ne peut commencer :

a. Qu'après l'implémentation.
b. Qu'avant l'implémentation.
c. Qu'après les spécifications.
d. Qu'après la conception.

72) Un test boîte noire :

a. Vérifie les détails d'implémentation des composants.
b. Teste l'application dans sa situation réelle d'exploitation.
c. Teste le fonctionnement de l'application dans une configuration spécifique d'un système.
d. Teste l'intégration des différents modules d'une application.

73) Un test de robustesse permet de :

a. Tester une application avec des données invalides.
b. Tester l'application avec des données valides.
c. Tester l'application avec des valeurs de données aux limites.
d. Tester l'application avec des ressources anormales.

74) Le défaut logiciel de type « Crush » :

a. Faiblesse dans un système informatique permettant à un attaquant de porter atteinte à l'intégrité de ce système
b. Dysfonctionnement dû à un bug dans des opérations de manipulations de pointeurs ou d'adresses mémoire.
c. Met hors service le logiciel défaillant lors de la tentative de ce dernier d'effectuer des opérations impossibles à réaliser
d. Dysfonctionnement dû à un bug dans les opérations d'allocation de mémoire

75) Le test de validation

a. Est un test Boite Blanche.
b. Permet de tester la conformité du système aux documents de spécification.
c. Permet l'amélioration de la productivité des équipes.
d. Permet de montrer que le code est bien construit.

76) Le test unitaire permet de :

a. Valider le bon fonctionnement d'une ou de plusieurs parties développées indépendamment avec le reste de l'application.

b. Confirmer que l'application réponde d'une manière attendue aux requêtes qui lui sont envoyées.
c. Tester les fonctions (ou les modules) de code par les programmeurs.

77) Le but de la phase de test est de :

a. Minimiser les erreurs
b. Trouver et corriger les erreurs
c. Trouver et expliquer les erreurs
d. Valider un code

78) Un test boîte noire : (cette question est à éliminer)

a. Vérifie le fonctionnement interne d'un système.
b. Test le système sous un environnement réseau particulier.
c. Test le bon fonctionnement d'une ou plusieurs parties développées indépendamment avec le reste de l'application.
d. Teste le code de l'application.

79) Un test peut concerner :

a. L'environnement du produit
b. La spécification du produit
c. La charge auquel peut être soumit le produit
d. La durée de vie du produit