Préparation à l'examen d'*Architecture Logicielle*

Wery Benoît

9 novembre 2017

Chapitre 1

Vrai ou Faux

1. *L'assembleur est un langage de programmation de bas niveau dont les instructions dépendent du type de microprocesseur/microcontrôleur.*

2. *L'architecture d'un système logiciel décrit les spécifications des différentes procédures/fonctions/-méthodes présentes dans le code.*

3. *L'analyse fonctionnelle d'un système logiciel identifie la structure à donner au système.*

4. *L'architecture d'un système logiciel identifie la structure à donner au système.*

5. *On peut décrire l'architecture d'un système logiciel de manière graphique avec des boites et des flèches.*

6. *Parmi les parties prenantes autour d'un système logiciel, il revient au développeur d'intégrer le système dans l'entreprise.*

7. *Parmi les parties prenantes autour d'un système logiciel, il revient au gestionnaire de l'infrastructure de déployer le système.*

8. *Le choix d'architecture d'un système logiciel peut avoir une influence sur sa qualité.*

9. *Le style d'architecture en couches permet de diminuer la complexité et la modularité pour augmenter la réutilisabilité et la maintenance.*

10. *Le style d'architecture en niveaux augmente le couplage pour diminuer la cohésion.*

11. *Le choix d'architecture peut avoir une influence sur la sécurité du système logiciel.*

12. *Un design pattern est un template de code applicable automatiquement étant donné la spécification d'une procédure/fonction/méthode.*

13. *Le design pattern du GoF Singleton permet de créer des instances d'une classe une à la fois.*

14. *Le design pattern du GoF Builder est de type construction.*

15. *Pour appliquer le design pattern du GoF Facade, il faut impérativement rendre toutes les procédures/fonctions/méthodes des sous-systèmes à cacher privées.*

16. *Le design pattern du GoF Template permet d'implémenter un algorithme incomplets avec des « trous » à remplir (hooks).*

17. *La programmation impérative met l'accent sur le comment un programme fonctionne.*

18. *La programmation déclarative met l'accent sur le comment un programme fonctionne.*

19. *La programmation fonctionnelle est plus proche de l'impérative que de la déclarative.*

20. *La programmation fonctionnelle est plus éloignée de l'impérative que de la déclarative.*

21. *Un bon système distribué doit être plus fiable que le même système en centralisé unique.*

22. *Dans une architecture client-serveur, on a toujours un unique serveur et un ou plusieurs clients.*

23. *Une architecture de type broker peut être vue comme orientée service.*

24. *Dans le cadre de services web, UDDI est un langage de description de services.*

25. *Dans le cadre de services web, WSDL est un langage de description de services.*

26. *Le standard REST définit les règles précises à appliquer pour obtenir des services web RESTful.*

27. *L'architecture d'un compilateur est typiquement orientée flux de données.*

28. *L'architecture centrée données consiste en un data store passif et des clients actifs.*

29. *L'architecture blackboard consiste en un data store passif et des clients actifs.*

30. *Le cloud computing consiste simplement à installer des logiciels sur des serveurs plutôt que des desktop stations ou laptops afin de les rendre accessibles à tout le monde via internet.*

31. *Selon la définition du NIST, l'une des caractéristiques du cloud est d'avoir une très grande élasticité et une adaptation très rapide.*

32. *Avec l'IaaS, le client doit gérer de lui-même l'installation et la mise à jour de son application.*

33. *Google docs est un service dans le cloud de type PaaS.*

34. *La complexité software de Halstead offre une mesure de la structure d'un code.*

35. *La complexité cyclomatique offre une mesure de la structure d'un code.*

Chapitre 2

Questions ouvertes

Petite question rapide à répondre en une ou deux minutes maximum, sans devoir donner de détails, juste pour s'assurer que vous avez compris le concept de la question.

1. *Définissez ce qu'est un design pattern, comment le caractériser et à quoi il sert.*

2. *Que sont les concurrency patterns ? Donnez quelques exemples.*

3. *Décrire ce qu'est le test driven development (TDD).*

4. *Définissez ce qu'est un bad smell et donnez un exemple.*

5. *Définissez ce qu'est le refactoring et à quel moment il peut être utilisé dans le processus de développement.*

6. *Définissez la notion de paradigme de programmation, ainsi que la programmation impérative et déclarative.*

7. *Définissez la notion de système distribué en reprenant rapidement les cinq buts.*

8. *Définissez la notion de middleware. Dans quel type d'architecture les retrouve-t-on ?*

9. *Donnez la différence entre un client léger et lourd, dans une architecture client-serveur.*

10. *Qu'est-ce-que CORBA et quel type d'architecture supporte-t-il ?*

11. *Quelles sont les différentes étapes de l'appel d'un service web ?*

12. *Expliquez brièvement les six contraintes d'une architecture REST ?*

13. *Exposez brièvement les différences entre IaaS, PaaS et Saas.*

14. *Exposez brièvement les différences entre IaaS, PaaS et Saas.*

15. *Comment se calcule la complexité Fan-in Fan-out et quels sont ses avantages et inconvénients ?*

16. *Définissez les notions de couplage afferent et efferent. Comment construit-on l'instabilité à partir de ces métriques.*

17. *Qu'est-ce-que la distance from main sequence et que permet-elle de mesurer ?*

18. *Définissez brièvement les principes DRY et WET.*

19. *Définissez le concept d'orthogonalité. En quoi améliore-t-il la qualité d'un logiciel ?*

20. *Expliquez brièvement le principe du code traçant.*

21. *Définissez brièvement la notion de microservice.*

22. *L'architecte est un jardinier ou un planificateur de ville, expliquez.*

23. *Quelles sont les différences entre principe et pratique.*

24. *Qu'est-ce-qu'un contexte borné et quel est le lien avec microservice ?*

Chapitre 3

Réflexion

Pour les différentes questions ouvertes, ce qui est attendu est une discussion argumentée et appuyée par des concepts vus au cours. N'hésitez pas à rappeler les définitions des concepts de base que vous utiliserez dans votre argumentation. Il n'y a pas forcément de réponse unique aux questions de réflexion, et ce qui sera évalué est l'exactitude de vos propos et l'utilisation adéquate d'arguments pertinents.

1. *Il existe un lien fort entre l'architecture et la qualité d'un système logiciel. En particulier, améliorer la qualité logicielle peut se faire en choisissant une architecture pertinente par rapport au cahier des charges du système à développer. Expliquez en donnant des exemples concrets, et argumentez.*

2. *L'architecte logiciel est le garant de l'intégrité conceptuelle du système logiciel. De quoi s'agit-il ? Quelles sont les différents éléments auquel il devra faire attention tout au long de la durée de vie du logiciel ? Illustrez vos explications avec des exemples concrets, et argumentez.*

3. *Décrivez brièvement les six principaux styles d'architecture suivant en donnant, pour chacun, les avantages et inconvénients et un exemple concret utilisant ce style. Comparez ensuite ces styles et déterminez une procédure qui permettrait à un architecte de se diriger vers le style le plus adéquat étant donné un système logiciel à concevoir. Centrée sur les données, flot de données, en couches, en niveaux, invocation implicite et MVC*

4. *Le pattern d'implémentation argue qu'il faut viser l'excellence en programmation en suivant les trois valeurs importantes que sont la communication, la simplicité et la flexibilité. En vous appuyant sur des exemples de systèmes logiciel avec un choix d'architecture le plus adapté, discutez à partir des avantages de l'architecture choisie de pourquoi elle permet de tendre vers l'excellence.*

5. *Un mauvais système logiciel peut se détériorer avec le temps avec pour conséquence qu'il deviendra cher et difficile, voir impossible, à maintenir. L'une des causes majeures est que le code*

a été sous-ingénierié. Qu'est-ce-que cela signifie-t-il et quelles sont les principales raisons pouvant mener à un tel code ? Quelles bonnes pratiques, tant au niveau du code qu'au niveau de l'architecture, peuvent aider à éviter un code sous-ingénieré ? Argumentez.

6. *Lorsqu'il s'agit de choisir un ou des langage(s) de programmation concret pour développer un système logiciel, quelles sont les questions à se poser ? Comment peut-on organiser et structurer la réflexion qui va guider vers le choix de langage ? Expliquez et argumentez.*

7. *Dans les architectures orientée-interaction le système logiciel est découpé en trois partitions principales : données, contrôle et vue. Comparez les trois grandes familles de modèles MV* (MVC, MVVM et MVP) en identifiant comment les trois partitions sont organisées et identifiez les avantages et inconvénients des différentes architectures existantes des trois familles.*

8. *Les architectures de type broker et orientée-service possèdent une série de points communs, notamment qu'elles permettent toutes deux de partager des services. Mais en quoi diffèrent-elles ? Quels sont les avantages et inconvénients de ces deux types d'architecture ? Pour quel type d'applications l'une ou l'autre sera-t-elle plus adaptée ? Argumentez.*

9. *Les architectures prédominantes ont évolué avec les changements business partant de solutions monolithiques vers des solutions actuellement orientées services. Expliquez comment cette évolution s'est passée en reprenant les points forts et faibles de chacune des architectures de la ligne du temps suivante et argumentez.*

10. *Il y a trois principales architectures orientées données que sont le batch séquentiel, les pipes et filtres et le contrôle de processus. Quels sont les points communs et différences entre ces trois architectures, les avantages et inconvénients. Illustrez votre réponse à partir d'exemples concrets. Quelles sont les questions que l'on pourrait se poser afin d'orienter son choix vers l'une des trois architectures sachant qu'on a à réaliser un système logiciel qui doit traiter des données ? Argumentez.*

11. *Afin d'évaluer la complexité d'un système logiciel, on peut procéder à des mesures sur le code de ce dernier. En particulier, on peut utiliser les métriques de Halstead, McCabe et Henry ou Kafura/Shepperd pour mesurer différents types de complexité. Quels sont les aspects de complexité mesurés par ces métriques ? Comment sont elles calculées ? Discutez de la pertinence des mesures ainsi réalisées, par rapport aux variables prises en compte et de l'utilité que l'on peut faire de ces métriques.*

12. *En quoi l'architecture en microservices permet-elle de suivre le principe de responsabilité unique (SRP). Expliquez et argumentez en mentionnant les bénéfices d'une telle architecture. En parti-*

culier, illustrez votre réponse en utilisant la notion d'architecture serverless et à l'aide d'exemples concrets.