|  |  |  |  |
| --- | --- | --- | --- |
| **Prénom :** | Adiara | | |
| **NOM :** | SALL | **Filière :** | DBE:P5 |

**EVALUATION GENIE LOGICIELLE**

Exercise1 :

1.La méthode Agile est une approche de gestion de projet basée sur des cycles courts (sprints) et une adaptation continue. Contrairement aux méthodes traditionnelles, elle favorise la flexibilité, la collaboration et l'amélioration progressive

Citons moi un exemple de methode agile : EXTREME PROGRAMMING

EXTREME PROGRAMMING (XP) est un processus agile né en 1999 qui a pris tout son essor au début du siécle grace a l’avenement des sites web .

2.Le testeur qui est l’assistant du client et ,le plus souvent , un informaticien . Il code les tests de recette le plus tot possible , valide le fonctionnement du code et vérifie la nom régression .Les notions relatifs aux tests seront détaillées lors du chapitre consacré aux projets de tests

3. Un repository dans Git, c'est comme un grand classeur où on range et on suit tout le code d'un projet.

4.Étapes pour ajouter un fichier à la branche master :

git add fichier1 : Prépare le fichier.

git commit -m "Description" : Sauvegarde les modifications.

git push origin master : Envoie les modifications à la branche master.

4.Avantages de Git : Travail collaboratif facilité, sauvegarde historique des modifications, meilleure gestion des versions.

EXERCISE2 :

Correspondances :

Collaboration avec le client →iii) Négociation du contrat

Répondre au changement → ii) Suivre un plan

Individus et interactions →i) Processus et outils

Logiciel fonctionnel → iv) Documentation complète

EXERCISE3

Exercice 3 : Les affirmations vraies : a) Les testeurs doivent collaborer avec les développeurs tout en restant objectifs. e) Une sélection d'utilisateurs peut faire des tests bêta après plusieurs itérations.

EXERCISE4

Exercice 4 :

Responsabilités cohérentes avec l'approche Agile :

c) Les testeurs travaillent avec les clients pour créer des tests d'acceptation.

d) Toute l’équipe est responsable de la qualité du produit.

e) Les développeurs testent les exigences non fonctionnelles.

**Exercice 5 :**

**a) Différence entre agrégation et composition :**

* **Agrégation :** Relation où les objets peuvent exister indépendamment. Par exemple, une bibliothèque peut avoir plusieurs livres, mais si un livre est retiré, la bibliothèque continue d'exister.
* *Composition* : Les objets sont liés étroitement. Si l'objet principal disparaît, l'autre aussi. Exemple : Un informaticien et son ordinateur

**b) Différence entre prototypage non jetable et modèle par incréments :**

* *Prototypage non jetable* : On crée un prototype qu'on améliore directement pour devenir le produit final.
* *Modèle par incréments* : Le produit est construit et livré petit à petit, par étapes. Chaque étape ajoute quelque chose au produit.

**c) Documents créés dans les différentes phases :**

* *Manuel d'utilisation* : Quand le produit est prêt pour les utilisateurs (déploiement).
* *Conception architecturale* : Lorsqu'on dessine la structure du produit (phase de conception).
* *Plan d’assurance qualité* : Lorsqu'on planifie comment garantir la qualité (phase de planification).
* *Code source* : Quand les développeurs écrivent les programmes (phase de développement).
* *Cahier des charges* : Dès le début du projet, pour expliquer ce qu'il faut réaliser.

**Combinaison cascade/V et spirale :**

* *Cascade ou modèle en V* : Permet de bien organiser les étapes, de manière linéaire.
* *Modèle en spirale* : Ajoute des cycles d'itération pour tester, analyser les risques, et améliorer avant d'avancer.
* *Combinaison* : On suit les étapes du modèle en cascade ou en V, mais avec des itérations du modèle en spirale pour chaque phase, afin de mieux gérer les risques et affiner le produit.
* *EXERCISE6*: les qualités manquantes dans ce logiciel sont :
* **Fonctionnalités** : Le logiciel ne fait pas tout ce qu'on attend de lui.
* **Facilité d'utilisation** : Il est difficile à apprendre ou à utiliser.
* **Exactitude** : Parfois, les résultats sont faux.
* **Efficacité** : Il utilise trop de ressources (comme le CPU) pour des tâches simples.

EXERCISE7 :

Proposition : Modèle en cascade, car les exigences sont claires et peu risquées.

Avantages : La méthode est basée sur une structure simple, qui est répétée dans chaque projet . Les membres de l'équipe peuvent facilement comprendre ce que l'on attend d'eux. La pression générale sur les employés est moindre. La formation est plus facile, surtout pour les nouveaux employés.

EXERCISE8 :

Risques des stratégies :

Acheter une base de données : Coût élevé et complexité du développement.

Modifier un système existant : Risque d’incompatibilité.

Collaborer avec plusieurs universités : Retards possibles dans le développement.

Approche recommandée : Collaborer avec les universités (solution économique et mutualisation des efforts).

Calcul de productivité en LOC/j : Divisez le nombre de lignes de code (LOC) par le temps de travail (en heures/jour).

Données fournies :

• Projet 1 : 7200 LOC

• Projet 2 : 180 LOC

• Une journée normale = 7 heures

• Les réunions ne comptent pas dans le temps de travail effectif

Étape 1 : Calcul du temps de travail effectif par jour

On calcule la durée de travail pour chaque jour en heures, en retirant les interruptions sauf les réunions (car elles ne comptent pas dans le calcul).

01/11 :

• Durée : 16h30 - 08h30 = 8h

• Pause : Déjeuner 1h

• Temps effectif : 8h - 1h = 7h

• Projet : 1

02/11 :

• Durée : 17h00 - 09h00 = 8h

• Pause : Déjeuner 0h30

• Temps effectif : 8h - 0h30 = 7h30

• Projet : 1

05/11 :

• Durée : 17h30 - 09h00 = 8h30

• Pause : Déj 0h30 + réunion 1h (réunion ignorée)

• Temps effectif : 8h30 - 0h30 = 8h

• Projet : 2

06/11 :

• Durée : 12h00 - 07h30 = 4h30

• Pas d’interruption

• Temps effectif : 4h30

• Projet : 2

Étape 2 : Calcul de la productivité par projet

Projet 1 :

• Total LOC : 7200

• Total temps : 7h + 7h30 = 14h30 = 2,07 jours (14,5 / 7)

• Productivité = 7200 LOC / 2,07 j = ~3478 LOC/j

Projet 2 :

• Total LOC : 180

• Total temps : 8h + 4h30 = 12h30 = 1,79 jours (12,5 / 7)

• Productivité = 180 LOC / 1,79 j = ~101 LOC/j

Résultat final :

**Données principales :**

* **Projet 1 :** 7200 lignes de code (LOC), avec un temps total de 14,5 heures (équivalent à 2,07 jours).
* **Projet 2 :** 180 LOC, avec un temps total de 12,5 heures (équivalent à 1,79 jours).

**Productivité calculée :**

* **Projet 1 :** Productivité = 7200 LOC ÷ 2,07 jours ≈ **3478 LOC/jour**.
* **Projet 2 :** Productivité = 180 LOC ÷ 1,79 jours ≈ **101 LOC/jour**.

En résumé :

* **Projet 1** est beaucoup plus productif à environ **3478 LOC/jour**.
* **Projet 2** est moins productif avec environ **101 LOC/jour**.

EXERCISE9 :

1. Date de début du projet

Les données :

• Le projet est suivi jusqu’au 15/04/2015

• La semaine va de dimanche à jeudi

• La tâche 1 devait finir le 25/01, et elle devait durer 5 jours ouvrés

• 25/01/2015 est un dimanche

• 5 jours ouvrés avant : dimanche 18/01/2015

→ Date de début du projet = 18 janvier 2015

2. Calcul des indicateurs d’avancement (au 15/04)

les formules utilisées sont suivantes :

•CBA = somme des travaux estimés

•CBTP = somme des estimations des tâches qui devaient être terminées au 15/04

•CBTE = somme des estimations des tâches effectivement terminées au 15/04

•CRTE = somme des travaux réels des tâches terminées

•VA = CBTE

•IPT = VA / CRTE

•VE = VA - CRTE

•IC VE / VA

•VC = CBTP - CBTE

Étape 1 : Regrouper les données

• CBA = 5 + 25 + 120 + 60 + 40 + 80 = 330 jh

• CBTP = tâches qui devaient être finies avant le 15/04 → tâches 1, 2, 4

→ 5 + 25 + 60 = 90 jh

• CBTE = tâches effectivement finies avant le 15/04 → tâches 1, 2, 4

→ 5 + 25 + 60 = 90 jh

• CRTE = travail réel de celles finies → 10 + 20 + 50 = 80 jh

• VA = CBTE = 90 jh

• IPT = VA / CRTE = 90 / 80 = 1.125

• VE = VA - CRTE = 90 - 80 = 10 jh

• IC = VE / VA = 10 / 90 ≈ 0.11

• VC = CBTP - CBTE = 90 - 90 = 0

Résultat final résumé :

Indicateur Valeur

Date début 18/01/2015

CBA =330 jh

CBTP =90 jh

CBTE =90 jh

CRTE =80 jh

VA =90 jh

IPT =1.125

VE =10 jh

IC =0.11

VC =0