Université de Tlemcen
Faculté des Sciences
Département d'Informatique
Filière : L2 Informatique - Module : Génie logiciel

Année universitaire : 2016/2017
Enseignants :
A. Chikh, D. Malti & S-M Chouiti

TD 2: Cycle de vie d'un logiciel

Objectifs :

Connaître les étapes du cycle de vie d'un logiciel (Etapes de développement, Etapes d'utilisation), la maturité d'un processus de développement logiciel

Exercice 1

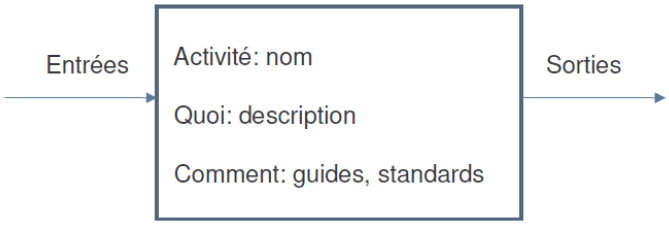
Nous avons vu en cours qu'un processus de développement de logiciel se compose, entre autres, des activités principales suivantes :

- Analyse (Spécification)
- Conception
- Implémentation

Question 1

Décrire pour chacune de ces trois activités la fiche suivante:

- Entrées de l'activité (input) :
- Sorties de l'activité (output) :
- Description de l'activité – quel est le problème à traiter? (What) :
- Standards, guides, « best practices » à appliquer (How) :



Question 2

Même question que la précédente mais applicable aux trois activités restantes suivantes du cycle de vie : Étude d'opportunité, Implantation, Maintenance

Exercice 2

Une entreprise LOG de production logiciel adopte un processus de développement logiciel qui consiste à enchainer les différentes phases de développement : étude de faisabilité, spécification, conception, implémentation, tests et livraison. Les retours en arrière entre ces différentes phases ne sont pas planifiés mais si des erreurs sont détectées pendant les tests, il est possible que l'équipe de développement réadapte la conception et/ou l'implémentation du logiciel. Le succès des projets de développement logiciel de cette entreprise est garanti seulement s'il s'agit de reproduire un projet déjà réalisé.

Question

- Sur la base de cet énoncé déterminer le niveau de maturité du processus de développement de l'entreprise LOG.

Exercice 3

Université de Tlemcen
Faculté des Sciences
Département d'Informatique
Filière : L2 Informatique - Module : Génie logiciel

Année universitaire : 2016/2017
Enseignants :
A. Chikh, D. Malti & S-M Chouiti

Question 1

Donnez des exemples d'instructions (en java) impliquant un couplage fort entres modules (Classes, objets).

Question 2

En java, laquelle des deux relations suivantes : généralisation (héritage) ou la dépendance (utilisation), se traduit par un couplage plus fort ?

public class A extends B { … }

public class C { .. D d; ..}

Question 3

En java, la classe String est référencée à partir de plusieurs autres classes dans la bibliothèque de classes du langage et aussi largement utilisée dans les programmes d'application. La classe String a ainsi un couplage fort.

a. Est-ce une mauvaise conception? Comment les concepteurs y ont remédié. ?
b. En plus, la classe String est une classe immutable (immuable), ce qui cause un problème d'efficacité (Par exemple, lors de la concaténation). Expliquez le problème et la (les) solution(s) apportée(s).

Université de Tlemcen
Faculté des Sciences
Département d'Informatique
Filière : L2 Informatique - Module : Génie logiciel

Année universitaire : 2016/2017
Enseignants :
A. Chikh, D. Malti & S-M Chouiti

Réponses

Exercice 1

- Niveau de maturité de processus « 2- reproductible »

Exercice 2

1. Spécifier

Activités: *spécifier*

Activité	Spécifier
Description	Décrire ce que doit faire le logiciel (comportement en boite-noire). Décrire comment vérifier en boite-noire que le logiciel fait bien ce qui est exigé.
Entrées	Client qui a une idée de ce qu'il veut.
Sorties	Une spécification (description des besoins du client). Des procédures de validation.

Comment :
Une spécification peut suivre de nombreux formalismes: cas d'utilisation, modèles UML, user-stories, exigences …
Les procédures de validation peuvent être des procédures manuelles ou des tests.

2. Concevoir

Activités: *concevoir*

Activité	Concevoir
Description	**Organiser** le logiciel afin qu'il puisse satisfaire les exigences de la spécification. Faire les principaux **choix techniques** pour satisfaire les exigences de la spécification.
Entrées	Une spécification.
Sorties	Une description des décisions de conception. Des procédures de tests qui permettent de vérifier que les décisions de conception sont correctement implémentées en code source et qu'elles contribuent à satisfaire les exigences de la spécification.

Comment :
Une conception peut prendre de nombreux formalismes: description textuelle des décisions de l'architecture, modèles UML, …

3. Coder

Activités: *coder*

Activité	Coder et tester
Description	Écrire le code source du logiciel. Tester le comportement du code source afin de vérifier qu'il réalise les responsabilités qui lui sont allouées.
Entrées	Spécification, conception.
Sorties	Code source. Tests unitaires. Documentation électronique navigable?

Comment :
Le codage peut respecter plusieurs paradigmes : Prog Structurée, POO, patterns, …
Utilisation des IDE (Integrated Development Environment) tel que .NET

Activité : Étude d'opportunité
Description : Etude de d'opportunité du projet (utilité, faisabilité)
Entrées : besoins de l'utilisateur, solutions potentielles, moyens
Sorties : Faisable ou non
Commentaires : Voir des projets similaires, Diagramme de contexte

Activité : Implantation (Installation)
Description : Installation et utilisation du logiciel dans l'environnement de l'utilisateur
Entrées : Code exécutable installable, manuels d'installation et d'utilisation
Sorties : Utilisation du logiciel, feedback utilisateurs validation
Commentaires : installer du logiciel, formation des utilisateurs, recueil feedback users et validation

Activité : Maintenance
Description : Maintenance corrective, évolutive, perfective et adaptative
Entrées : nouveaux besoins, bugs constatés, remarques générales, version actuelle
Sorties : nouvelle version
Commentaires : test de non régression,

Exercice 3

Question 1

Réponse: Héritage, implémentation, déclaration publique, appel de méthode, etc.

Question 2

Réponse : Dans l'exemple ci-dessus, C dépend uniquement de la partie publique de D. La classe A dépend des éléments publique et protégée de la classe B. De même, la classe A doit être conforme pour les changements d'interface à la classe B parce que la classe A est une classe B par contre la classe D peut étendre son interface sans affecter la classe C

Question 3

Réponse: la classe String a effectivement un couplage fort. Une des conséquences est qu'il serait très difficile de modifier l'interface de la classe String sans causer de problèmes dans beaucoup d'autres classes. La classe String est parmi les classes, qui ne sont pas susceptibles de changer afin que le couplage fort ne soit pas un problème.

a. Cette classe est déclarée finale. Ce qui signifie qu'il n'est pas possible de l'étendre. Toute méthode qui reçoit un objet de type String a donc la garantie que les méthodes qu'elle appelle sur cet objet ont bien le comportement nominal : elles n'ont pas pu être surchargées.
b. Parler de StringBuffer et de la création d'objets intermédiaires.

Université de Tlemcen
Faculté des Sciences
Département d'Informatique
Filière : L2 Informatique - Module : Génie logiciel

Année universitaire : 2016/2017
Enseignants :
A. Chikh, D. Malti & S-M Chouiti