Examen intra – LOG3000 – Hiver 2014

- Vendredi le 28 février 2014.
- Durée : 08h30 à 10h00 (total 1h30).
- Local : B-415.
- Total des points : 20.
- Pondération de l'examen dans la note finale : 35%.
- Sans documentation, sans calculatrice.

1. Question sur l'utilité des processus (2 points)

Présentez et expliquez deux utilités des processus de développement logiciel.

Parmi les réponses possibles :

- Communication interne à l'équipe : Pour que les nouveaux puissent comprendre comment le développement logiciel s'exécute.
- Communication externe avec les parties prenantes : Pour les certifications comme le CMMI, qui permettent de convaincre les clients de la maturité de l'entreprise.
- Évaluation et diagnostic de problèmes : La production d'artéfacts déficients peut être due à des activités manquantes ou mal faites dans le processus. Un processus défini peut permettre de trouver le problème et le corriger.
- Amélioration du processus existant : Introduire des nouvelles pratiques dans un processus existant, comme le TDD (*test-driven development*).
- Structuration du travail à faire : Le processus peut servir de base pour la planification de projet sur diagramme de Gantt.

2. Question sur la différence entre processus et projet (1 point)

Quelle est la différence entre un modèle de processus (ex.: modèle fait utilisant le langage SPEM 2.0) et un modèle de projet (ex.: diagramme de Gantt) ? Quelles informations ne sont visibles que dans le modèle de processus ? Quelles informations ne sont visibles que dans le diagramme de Gantt ?

Le modèle de processus se concentre sur la transformation de l'information. Le modèle de projet se concentre sur l'utilisation des ressources.

Les artéfacts et les compétences requises pour les rôles ne sont visibles que dans le modèle de processus. Les ressources (temps, personnes, matériel) ne sont visibles que dans le diagramme de Gantt.

3. Question sur les cycles de vie (2 points)

Durant le cours, nous avons vu les types de cycle de vie suivants :

- Cascade, modèle en V ou en W,

- Incrémental,
- Transformationnel,
- Spirale.

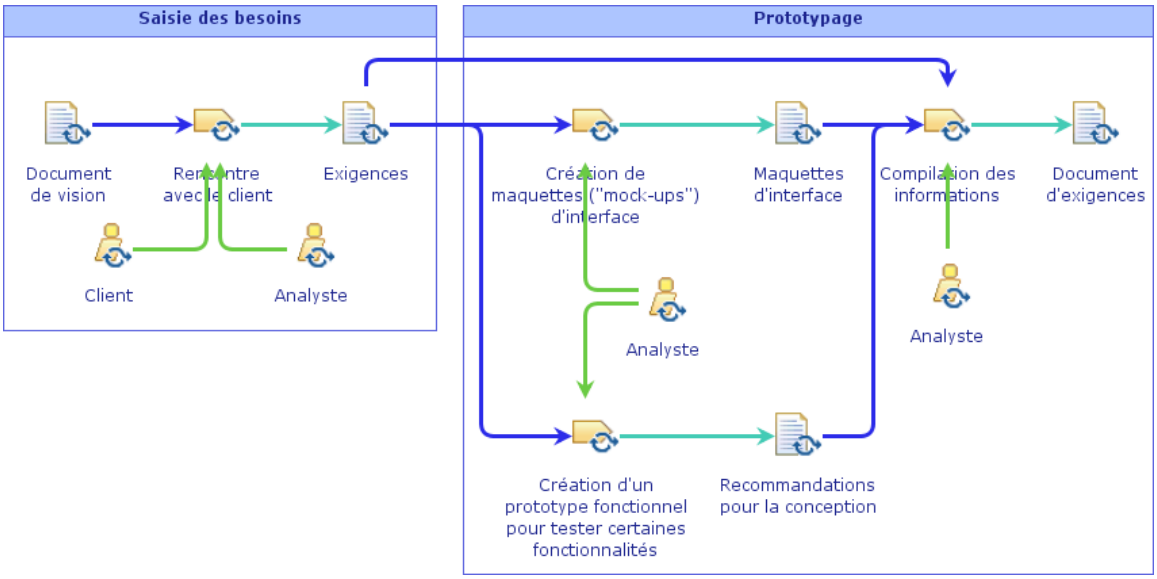
Pour chacun de ces types de cycle de vie, décrivez dans quel cas il serait le plus approprié.

- Cascade, modèle en V ou en W : Lorsque tout est connu et que le risque est minimal.
- Incrémental : Lorsqu'il est possible d'obtenir une bonne base d'exigences et que celles-ci peuvent facilement se scindées en blocs.
- Transformationnel : Lorsque les exigences sont fragmentaires. Utile pour des projets de recherche où les exigences liées au produit final peuvent être très floues.
- Spirale : Pour des projets très gros et très complexe. L'approche spirale permet d'aborder un problème complexe par étapes de raffinement progressif.

4. Question sur la discipline des requis (3 points)

Certains de vos amis qui font le projet intégrateur de 4e année ont un problème. Ils viennent de livrer leur document d'exigences au client mais ce dernier refuse de le lire parce qu'il est trop mal écrit. Le client affirme que le document est bourré de fautes de français, qu'il se contredit d'une section à l'autre et qu'il utilise un jargon technique incompréhensible. Vos amis sont découragés : Ils ont mis beaucoup d'efforts dans ce document d'exigences ! Pourquoi est-il d'aussi mauvaise qualité ?

Vos amis vous donnent le processus qu'ils ont utilisé pour obtenir les exigences. Ce processus est présenté ci-dessous. Que feriez-vous pour éviter le problème vécu par vos collègues ? Justifiez.

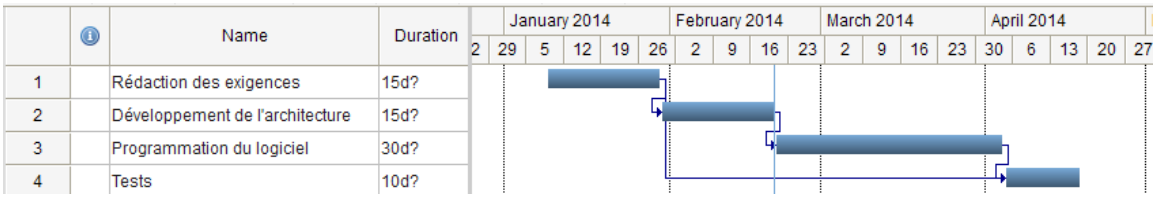


Ils ont mis beaucoup de temps sur le document d'exigences, en particulier à travers la création de deux prototypes relativement complexes. Il est donc probable qu'ils aient manqué de temps pour travailler sur la finition du document.

Des activités de révision en fin de phase aurait permis d'allouer du temps dans la planification de projet, ce qui aurait assuré au moins une autre passe sur le document. Une révision finale aurait pu au moins détecter les fautes de français.

5. Question sur la planification de projet (2 points)

Certains de vos amis qui font le projet intégrateur de 4e année viennent de terminer la planification du projet dans un diagramme de Gantt. Ce diagramme de Gantt est présenté ci-dessous au complet.



Présentez et justifiez deux risques majeurs introduits par cette manière de gérer le projet.

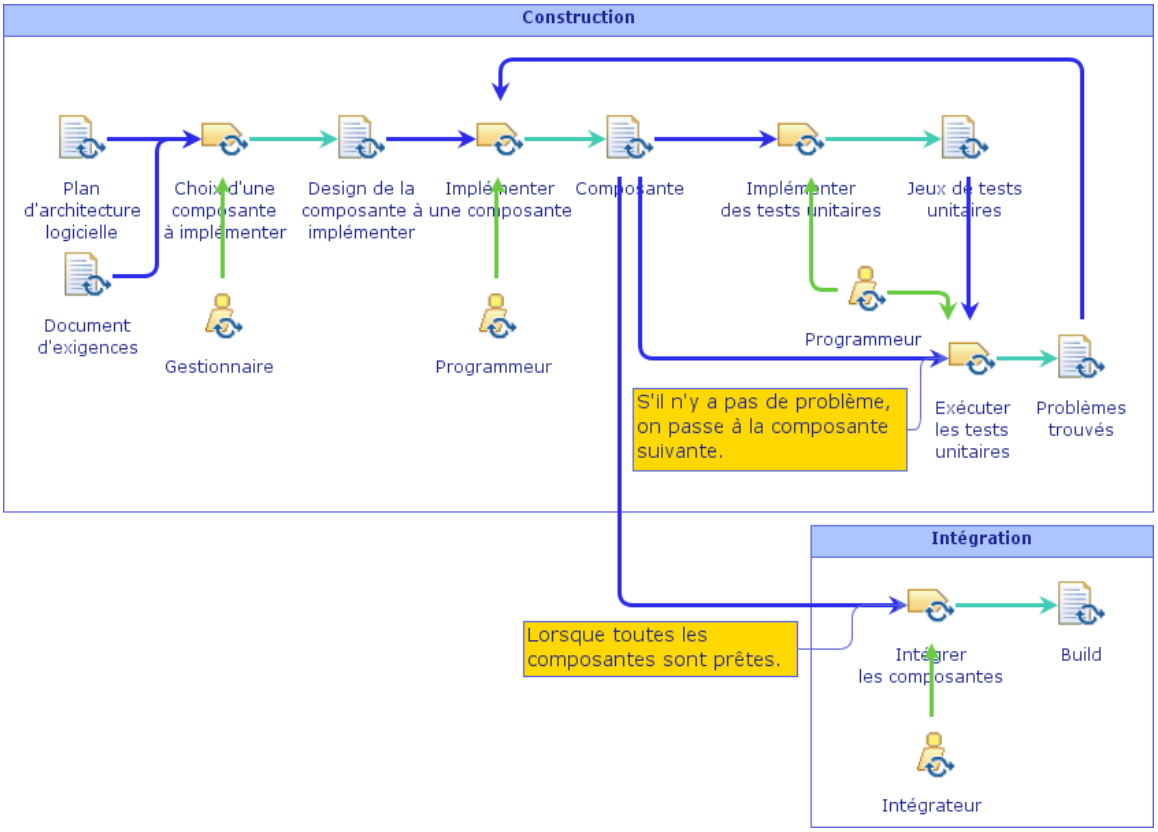
Les tâches sont trop longues pour un projet aussi court. Il est difficile de faire un suivi avec des tâches aussi longues : Par exemple, on ne saura pas si les exigences sont en retard avant le 25 janvier, et à ce moment-là, il sera trop tard pour corriger le tir.

Il n'y a pas d'itérations, et donc pas de temps prévu pour revenir en arrière. Il est fort possible qu'il faille revoir les exigences ou l'architecture. Il faut donc allouer du temps pour cette correction.

6. Question sur la discipline d'implémentation (3 points)

Vous rencontrez vos amis qui font le projet intégrateur de 4e année dans la soirée du 12 avril 2014. Ils ont les yeux cernés et semblent découragés. Entre deux bâillements, un de vos amis vous explique qu'ils n'arrivent toujours pas à faire fonctionner le programme au complet.

Vos amis vous donnent le processus qu'ils ont utilisé pour l'implémentation du code. Ce processus est présenté ci-dessous. Que feriez-vous pour éviter le problème vécu par vos collègues ? Justifiez.



Il faudrait idéalement faire de l'intégration continue. Au minimum, le projet au complet devrait être intégré lorsque chaque composante est complétée. En ne faisant l'intégration qu'à la toute fin, on peut trouver des problèmes majeurs de communication entre les modules qui peuvent prendre beaucoup de temps à corriger.

7. Question sur le CMMI (1 point)

À quoi sert le CMMI ? Pour quelle raison a-t-il été conçu ?

Le CMMI a été créé afin d'évaluer la maturité des processus de développement logiciel, soit le niveau de conscience de l'organisation sur la manière dont ses logiciels sont développés. Son objectif est de trouver des failles dans les processus dans le but de les améliorer.

8. Question sur les tests unitaires (1 point)

Votre patron vient de lire un article sur l'importance des tests unitaires. Il vient donc de renvoyer tout le département de qualité logicielle en disant qu'il est possible de tester tout le logiciel avec les tests unitaires. Votre patron dit qu'il suffit que chaque ligne de code soit associée à un test unitaire pour s'assurer que tout le logiciel fonctionne.

Que lui répondez-vous ?

Les tests unitaires se limitent à vérifier le fonctionnement des modules pris de manière indépendante. Il y a beaucoup d'autres tests à faire : Tests d'intégration, tests de système (validation et vérification), tests d'acceptation, tests de sécurité, tests de charge, etc.

9. *Question sur la modélisation de processus (5 points)*

Notre laboratoire veut développer un nouvel outil de saisie de données qui servira pour nos travaux de recherche. L'objectif de l'outil est de nous permettre d'observer comment les développeurs de logiciel travaillent, en se basant sur un journal quotidien de leurs activités cognitives. Nous avons quelques exigences que nous voulons absolument voir dans le produit final, mais beaucoup de détails sont encore à clarifier.

Modélisez le processus dans le langage SPEM 2.0 basé sur le cycle de vie de votre choix. Le processus doit couvrir minimalement les disciplines suivantes :

- Discipline des requis,
- Discipline de l'analyse et conception,
- Discipline de l'implémentation.

Le processus doit produire minimalement un *build* intégré accompagné d'un jeu de tests unitaires.

Il y a beaucoup de solutions possibles. Ce que nous voulons voir :

- Cycle de vie non-cascade (+1 point),
- Au moins une activité liée aux requis (+0,5 point),
- Au moins une activité d'analyse et de design (+0,5 point),
- Une activité de programmation, une activité d'intégration et une activité liée aux tests unitaires (+2 points),
- Un artéfact *build* et un artéfact jeu de tests unitaires à la fin (+1 point),
- Respect des règles de modélisation du SPEM 2.0 (correction négative, -0,25 point par erreur).

La solution proposée est la suivante :