Examen intra – LOG3000 – Automne 2014

- Jeudi le 23 octobre 2014.
- Durée : 13h45 à 15h35 (total 1h50).
- Local : L-4812.
- Total des points : 20.
- Pondération de l'examen dans la note finale : 30%.
- Sans documentation, sans calculatrice.
- Remettre le questionnaire à la fin de l'examen.

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

Présentez deux utilités des processus de développement logiciel (1 point). Justifiez chaque utilité en expliquant en quoi elles peuvent être importantes pour un projet de développement logiciel (1 point).

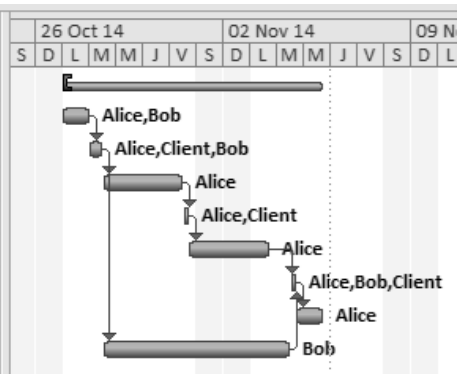
Parmi les réponses possibles :

- Communication interne à l'équipe : Pour que les nouveaux puissent comprendre comment le développement logiciel s'exécute. Important pour l'intégration des nouveaux et afin qu'ils sachent les informations disponibles pour leur travail et les informations qu'ils doivent fournir aux autres.
- 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. Important parce que cela permet d'éviter de refaire constamment les mêmes erreurs.

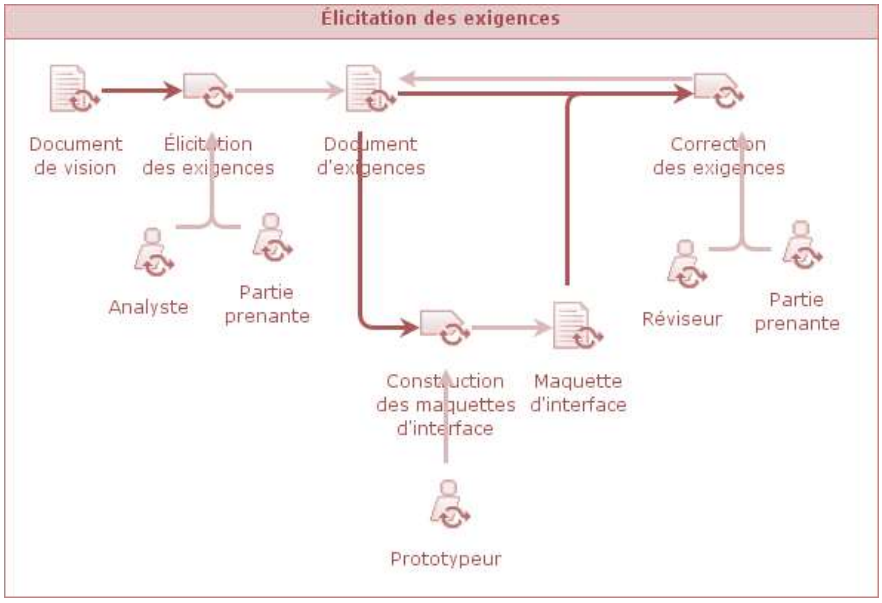
2. *Question sur la différence entre processus et projet (3 points)*

Votre patron a monté le plan de projet suivant. Afin de mieux comprendre la manière dont le logiciel est construit, montez le modèle de processus associé au plan de projet. Votre modèle de processus doit contenir entre trois et sept activités.

Nom de la tâche	Durée	Préd	Noms ressources
Élicitation des exigences			
Lire le document de vision	1 jour		Alice,Bob
Première rencontre avec client	3 hr	2	Alice,Client,Bob
Écrire les exigences	3 jours	3	Alice
Deuxième rencontre avec client	3 hr	4	Alice,Client
Corriger les exigences	1 jour	5	Alice
Troisième rencontre avec client	3 hr	6,9	Alice,Bob,Client
Finaliser les exigences	1 jour	7	Alice
Faire des maquettes d'interface	5 jours	3	Bob



Solution possible :



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.

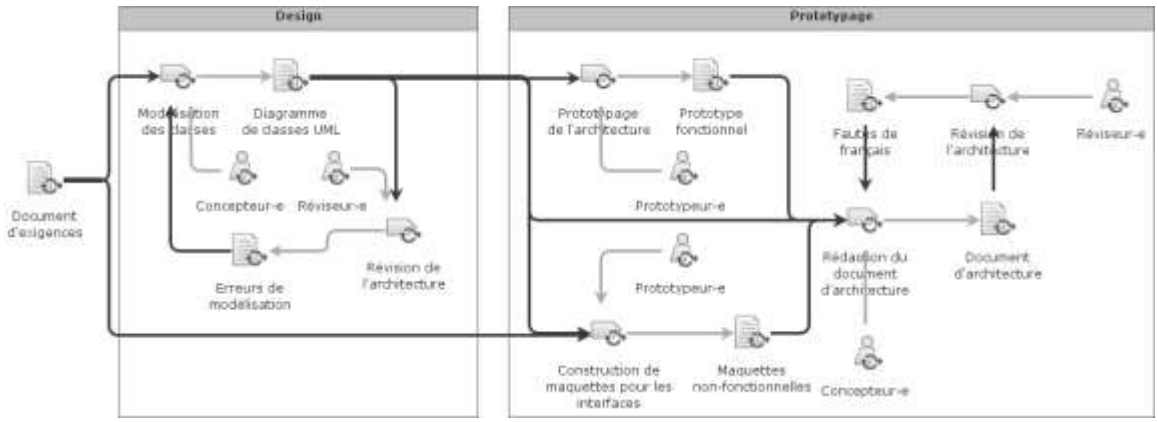
Décrivez brièvement chaque type de cycle de vie (1 point). Donnez un exemple pour chaque type de cycle de vie où ce type de cycle de vie serait approprié (1 point).

- Cascade, modèle en V ou en W : Où les phases sont effectuées séquentiellement, avec peu de retours en arrière. Utile lorsque tout est connu et que le risque est minimal.
- Incrémental : Où le logiciel est construit en itérations et où chaque itération construit un sous-ensemble d'exigences. Utile lorsqu'il est possible d'obtenir une bonne base d'exigences et que celles-ci peuvent facilement se scinder en blocs.
- Transformationnel : Où le logiciel est construit à travers le raffinement progressif d'un prototype. Utile lorsque les exigences sont fragmentaires. Utile pour des projets de recherche où les exigences liées au produit final peuvent être très floues.
- Spirale : Où le logiciel est construit à travers un processus complexe et peu réutilisable impliquant plusieurs itérations cascades qui ajoutent progressivement de la complexité à la solution produite. Utile 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 d'analyse et conception (3 points)

Certains de vos amis qui font le projet intégrateur de 4e année ont un problème. Ils ont passé beaucoup de temps en analyse et conception, mais sans les résultats escomptés. En travaillant sur l'implémentation, ils se sont rendu compte que leur architecture est trop lourde. L'ajout de fonctionnalités demande beaucoup de temps, et de toute manière le logiciel est trop lent pour être utilisable. Pourtant ils ont travaillé très fort sur leur architecture ! Que s'est-il passé ?

Vos amis vous donnent le processus utilisé pour obtenir l'architecture du logiciel. Présentez un problème majeur que vous voyez dans ce processus (1 point). Justifiez en quoi ce problème peut expliquer les ennuis vécus par vos amis (1 point). Proposez une solution à ce problème majeur (1 point).



Ils construisent directement le diagramme de classe sans faire une analyse plus poussée. Ils ont probablement fait une fixation sur une solution particulière. Il n'y a pas de génération ni de choix de la solution, et les révisions restent très superficielles. Le prototype fonctionnel n'a pas été testé, donc son utilisation a probablement été superficielle.

Les étudiants sont donc partis sur une solution inappropriée et ils ne se sont pas rendu compte du problème avant l'implémentation. L'absence de tests de charge sur le prototype, par exemple, aurait permis de découvrir le problème de performance rapidement.

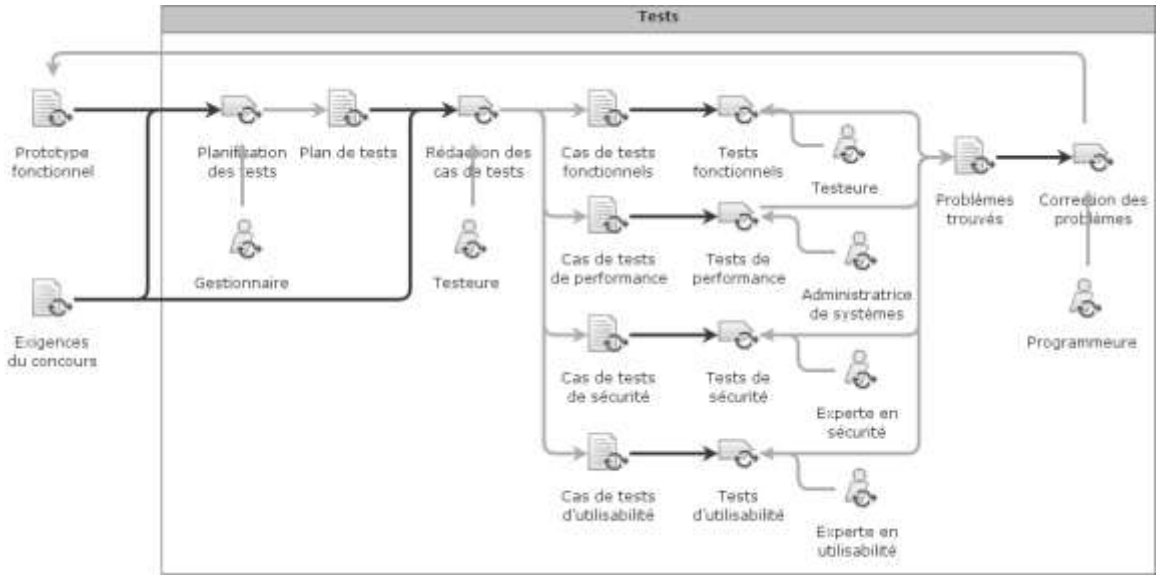
Il aurait fallu avoir une ou plusieurs activités d'analyse, afin de générer différentes solutions et choisir la meilleure, plutôt que de se fixer sur la première solution qui nous vient en tête. Il aurait aussi fallu que les révisions soient plus approfondies, ou bien que le prototype soit testé plus en profondeur, avec par exemple des tests de charge.

5. Question sur la discipline des tests (3 points)

Vous rencontrez vos amies qui font le projet intégrateur de 4e année. Leur projet est un concours où leur prototype est présenté devant un jury. Leur objectif est de gagner le concours.

Contre toute attente, leur développement s'est terminé une semaine avant le concours et elles ont du temps libre pour faire des tests. Elles ont donc monté le processus suivant qui décrit les tests qu'elles prévoient faire pour la semaine suivante.

Présentez un problème majeur que vous voyez dans ce processus (1 point). Justifiez en quoi ce problème majeur pourra causer des ennuis à vos amies (1 point). Proposez une solution à ce problème majeur (1 point).



Pour seulement une semaine de tests, ce processus est beaucoup trop ambitieux et implique beaucoup de compétences qu'il serait difficile à trouver. Il est fort peu probable que l'équipe soit capable de faire toutes ces activités en une semaine.

Le risque est qu'avec le manque de temps et de compétences, l'équipe fasse seulement des tests moins importants pour le concours. Par exemple, il est peu probable que les aspects sécurité soient importants pour le concours.

La solution serait de simplifier le processus afin de se concentrer sur les tests les plus susceptibles de trouver des problèmes qui seront relevés par le jury du concours.

6. *Question sur la théorie de la mesure (2 points)*

L'approche GQM (*Goal-Question-Metric*) présente comment lier un but de haut niveau avec des métriques concrètes qui peuvent être évaluées sur le terrain. Utilisez l'approche GQM pour développer les buts suivants en au moins deux questions et trois métriques :

- Fiabilité : Code ayant la capacité de fonctionner adéquatement durant une période donnée (1 point),
- Maintenabilité : Code offrant une facilité à trouver et corriger des défauts, à modifier les fonctionnalités existantes et à ajouter de nouvelles fonctionnalités (1 point).

Solution possible :

- G1 : Fiabilité :

    - Q1 : Est-ce que le logiciel contient des bogues ?
        - M1 : Nombre de bogues documentés dans Bugzilla.
        - M2 : Proportion de tests unitaires réussis.
    - Q2 : Est-ce que le logiciel est robuste à la perte de connexion internet ?
        - M3 : Proportion de tests de connexion réussis.
- G1 : Maintenabilité :
    - Q1 : Est-ce que le code est facile à comprendre ?
        - M1 : Proportion de commentaires.
        - M2 : Cohésion et couplage du code.
    - Q2 : Est-ce que le code est facile à modifier ?
        - M2 : Cohésion et couplage du code.
        - M3 : Nombre de sorties de messages de débogage.

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

Vous faites un projet avec une physicienne pour la construction d'un logiciel capable de calculer et modéliser des effets quantiques entre différents types d'atomes. Ce logiciel n'a jamais été construit auparavant, et ni vous ni la physicienne ne savez ce qu'il faut modéliser. Les exigences sont donc très floues au début du projet.

Le projet est donc risqué, étant donné que le travail à faire n'est vraiment pas clair. Étant donné ce risque, vous décidez de monter un processus décrivant la manière dont vous allez développer le logiciel.

Le processus doit produire minimalement un composant de code intégré (*build*) accompagné des jeux de tests effectués. Le processus doit être construit suivant un cycle de vie approprié.

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

- Un cycle de vie non-cascade (+1 point),
- Au moins une activité liée aux exigences (+1 point),
- Au moins une activité d'analyse et de design (+1 point),
- Au moins une activité de programmation et une activité d'intégration (+1 point),
- Au moins une activité de création de tests, une activité d'exécution de tests et une activité de correction des erreurs (+1 point),
- Respect des règles de modélisation (correction négative, -0,25 point par erreur).