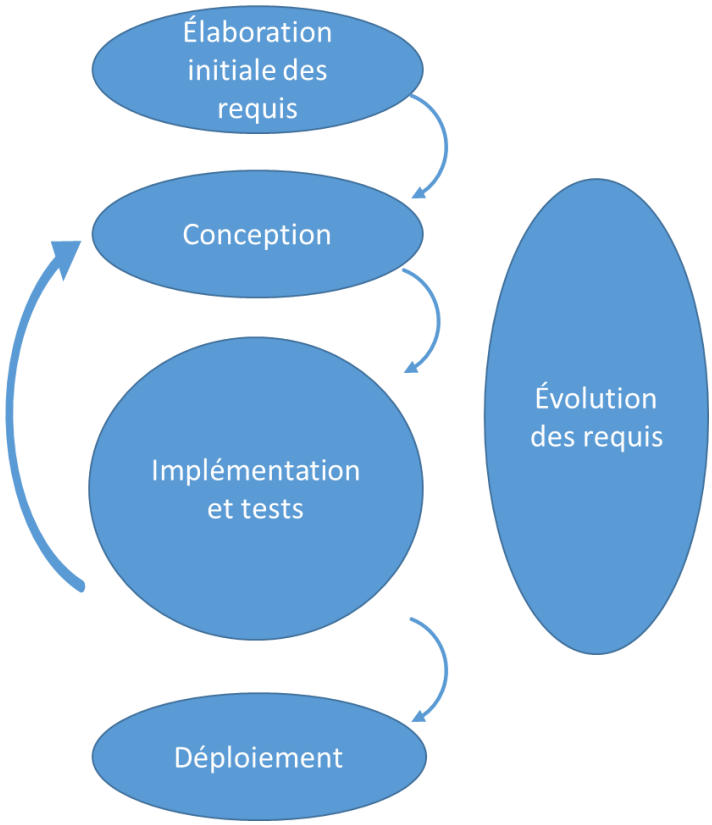
Laboratoire 1 - Création d'un processus de bonne qualité

INF8301 : Ingénierie de la qualité en logiciel

Simon Delisle
Félix Gingras-Harvey
François Pierre Doray
Alexandre Vanier

École Polytechnique de Montréal
24 janvier 2014

Cycle de vie



Le cycle de vie est basé sur le modèle incrémental. Les requis seront revisités tout au long du projet.

Notre projet consiste à participer à la compétition «Laval Virtual». Nous modifierons donc nos requis en nous fixant des objectifs ambitieux au fur et à mesure que nous découvrirons les technologies utilisées au cours des phases de conception, d'implémentation et de tests.

L'implémentation et les tests sont mis dans une seule phase puisque les développeurs devront réaliser les plans de tests et exécuter ceux-ci simultanément avec l'implémentation. Nous encourageons le Test-Driven Development, où les objectifs du développement sont fixés à l'avance par des tests.

Modélisation du processus

Voir le fichier ProcessEdit remis avec ce travail.

Tel que décrit à la question 1, notre projet aura 5 phases. Les phases de conception, d'implémentation et de tests et validation seront répétées à chaque itération et engendreront une évolution des requis.

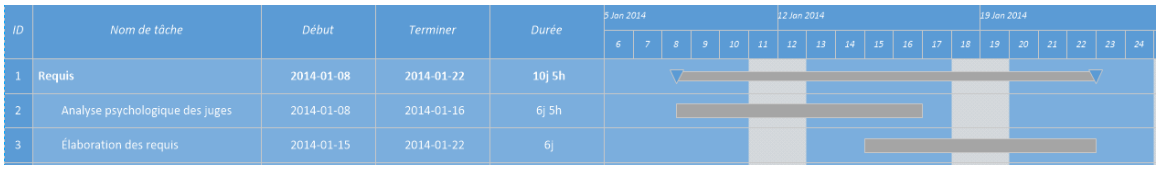
Les livrables remis à la fin de chaque phase sont :

- Élaboration initiale des requis : SRS
- Conception : Document d'architecture
- Implémentation : Programme
- Déploiement : Livrable final (présentation à la compétition «Laval Virtual»)

Planification de projet sommaire

Le diagramme de Gantt complet pour la planification de la session (avec dates précises) a été remis en annexe à ce travail.

Élaboration initiale des requis

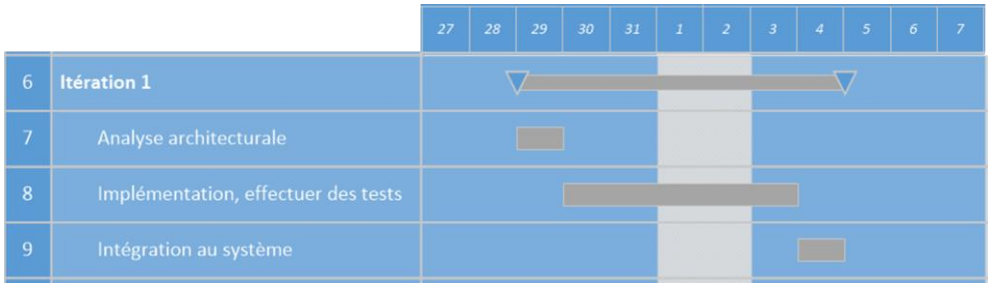


Itération

Chaque itération sera divisée ainsi :

- 1 jour : Planification de l'itération, analyse architecturale
- 3 jours : Implémentation, effectuer des tests.
- 1 jour : Intégration.

L'évolution des requis est une activité constante.



Révision et finalisation

Tel que montré dans le diagramme de Gantt remis avec ce travail, 3 jours complets seront consacrés à cette activité.

3.1 Questions sur la structure et le contenu du processus

1.

Nous avons choisi d'utilise le modèle incrémental comme base pour construire notre cycle de vie. Avec ses itérations courtes et indépendantes les unes des autres, ce modèle peut facilement être utilisé avec une méthodologie agile. Cela est très intéressant car les caractéristiques de notre projet s'accordent bien avec les méthodes agiles (requis très volatils, expérimentation de technologies inconnues, besoin de faire des démonstrations fréquemment).

Nous savons que les méthodes agiles permettent une grande flexibilité et que celle-ci est nécessaire pour un projet réalisé par une petite équipe et dont les requis devront être ajustés en fonction des capacités des nouvelles technologies en jeu. Avec les méthodes agiles, nous pouvons ajuster notre planification rapidement (grâce aux courtes itérations) en fonction de nos découvertes et de nos difficultés. De plus, un processus plus simple nous permet de passer plus de temps sur les activités que nous considérons plus importantes (programmation, conception, tests) tout en gardant une planification assez efficace pour ne pas nous amener des problèmes durant le projet.

2.

Une pratique importante que nous utiliserons est le «pair programming». En tout temps, nous serons deux développeurs par ordinateur pour développer une fonctionnalité. Un développeur aura la responsabilité d'entrer le code au clavier tandis que l'autre fera les schémas aidant à la compréhension du problème et révisera au fur et à mesure le code écrit par son compagnon.

3.

Nous estimons que le «pair programming» est important étant donné que cela permet d'avoir du code qui est souvent plus propre et parfois plus efficace (car chaque programmeur peut voir des erreurs ou des améliorations que l'autre ne voit pas). Également, nous savons qu'il est moins coûteux de déceler des erreurs (de requis, de conception, d'implémentation…) les plus tôt possible. Ainsi, nous comptions sur le «pair programming» pour réduire le nombre d'erreurs sortant de l'activité d'implémentation. Le «pair programming» permet aussi de partager les connaissances, ce qui est utile dans notre cas puisque les notions utilisées dans notre projet sont parfois très avancées et le code peut être compliqué à comprendre.

3.2 Questions sur le suivi et le contrôle du processus

4.

Pour évaluer un cycle de vie, il faut s'assurer qu'à chaque phase, nous sommes capables d'atteindre les objectifs spécifiques de notre projet. Dans notre cas, le développement doit être flexible car il y a beaucoup d'incertitudes par rapports aux requis et aux technologies utilisées. Il faut donc vérifier que le cycle de vie sur lequel nous nous sommes basés peut être bien respecté sans imposer de contraintes nuisibles au déroulement du projet.

Une manière de faire cette vérification est de noter les périodes passées dans chaque phase à chaque itération. Si des phases prennent trop de temps, semblent inexistantes ou encore provoquent des frustrations dans l'équipe, c'est qu'elles ne correspondent pas à la réalité de notre projet.

5.

Pour mesurer la qualité de notre processus, il faut accumuler des données sur notre travail. Ces données doivent ensuite être comparées aux objectifs que nous nous sommes fixés.

Nous présentons ci-dessous des mesures qui nous permettraient d'ajuster notre processus.

- Temps (heures-personne) consacré à chaque activité.
    - Plus de temps que prévu : activité devrait être scindée en plusieurs activités.
    - Moins de temps que prévu : activité insignifiante, à combiner avec une autre.
- Nombre de de défauts répertoriés pour un artéfact (par KLOC, par page…)
    - Trop de défauts : il manque sûrement des activités préliminaires qui aideraient à produire un artéfact de meilleure qualité. Par exemple, si les utilisateurs ont souvent de la difficulté à utiliser les composantes produites, c'est un signe qu'il manque des activités pour évaluer les cas d'utilisation / besoins des utilisateurs.
- Taille des artéfacts produits.
    - Très grande taille par rapport à leur utilisation dans le processus : Trop de temps a été consacré à leur production. L'exécution de l'activité devrait être revue.
- Nombre moyen de méthodes par classe.
    - Trop de méthodes par classe : Indicateur d'une faible cohésion et donc d'un manque d'activités de conception.
    - Ces mesures peuvent aussi indiquer un processus de conception inadéquat : nombre moyen de dépendances envers d'autres fichiers par fichier, nombre moyen de méthodes héritées par classe, proportion de méthodes publiques…
- Flexibilité du processus.
    - Faible flexibilité : Il faut revoir le processus pour permettre des changements plus fréquents, par exemple en réduisant la durée des itérations.

6.

Des exemples de mesures à prendre ont été listés à la question 5. On indique ici comment ces ferait la saisie de ces mesures.

- Temps (heures-personne) consacré à chaque activité.
    - Comme nous l'avons fait dans notre projet intégrateur précédent, nous ajouterons aux commentaires des «commits» le numéro de la tâche associée et le temps qui lui a consacré. L'interface Web nous permettra ensuite de résumer le temps consacré à chaque activité de manière automatique.
- Nombre de de défauts répertoriés pour un artéfact (par KLOC, par page…)
    - Nous associons chaque défaut répertorié à un module. L'interface Web du système dans lequel nous entrons les défauts décelés nous permet donc directement de voir le nombre de défauts par module.

- Taille des artéfacts produits.
    - Le nombre de page de chaque document produit peut être retrouvé de manière triviale. Le nombre de lignes de code peut être calculé par l'interface Web de notre système de contrôle de versions.
- Nombre moyen de méthodes par classe.
    - L'utilitaire «ctags» permet de produire automatiquement un rapport des noms des classes et les méthodes associés en inspectant un répertoire. L'utilisation de la commande «grep» permet ensuite d'obtenir le nombre moyen de méthodes par classe.

4. Question de rétroaction

Nous avons consacré 16 heures-personne à la réalisation de ce laboratoire.