UNIVERSITÉ DE SHERBROOKE

Faculté de génie

Département de génie électrique et génie informatique

**Projet S5**

**Assurance qualité**

Présenté à

Jean-Baptiste Michaud

Présenté par

l’équipe P4

Sherbrooke – 23 février 2017

# Table des matières

[Table des matières 1](#_Toc474835264)

[1 Assurance Qualité 2](#_Toc474835265)

[1.1 Plan de l’AQ 2](#_Toc474835266)

[1.1.1 BUTS DE L’ASSURANCE QUALITÉ DANS LE PROJET 2](#_Toc474835267)

[1.1.2 Fonctionnement de l’assurance qualité pour notre équipe 2](#_Toc474835268)

# Assurance Qualité

## Plan de l’AQ

### BUTS DE L’ASSURANCE QUALITÉ DANS LE PROJET

* S’assurer, de manière efficace, que les spécifications techniques sont respectées.
* Faciliter la gestion (et résolutions) d’erreurs des modules techniques.

### Fonctionnement de l’assurance qualité pour notre équipe

***On utilise la plateforme Lean Testing (app.leantesting.com)***

Préalablement à la conception de chaque module, il faut établir un plan de test unitaire pour chaque fonctionnalité dudit module.

Pour les tests, on a :

* L’équipe qui conçoit les tests doivent être différents de l’équipe qui les exécute.
* Chaque test doit contenir les trois sections suivantes :
  + Étapes préalables (*Preconditions*).
  + Étapes à exécuter (*Steps to execute*).
  + Résultat attend (*Expected result*).
* Il y a deux types de tests : régression et de module. Afin de spécifier un test de type régression, on met la priorité à un niveau critique (*Priority -> Critical)* et **on ajoute un R dans le nom du test**.
  + Environ 20 % des tests d’un module doivent avoir la mention de régression.
    - Ces derniers sont les tests clés qui s’assurent que le module fonctionne.
* Il faut que chaque test soit très clair ; l’équipe d’exécution ne doit rien déduire lors de la procédure.
* Nommer le test comme d'un nom significatif au test. Ne pas oublier si le test est de type régression.

Pour l’exécution de test, on a :

* Trois passes d’exécution possibles (par module) :
  + Première passe :
    - Première exécution du plan de test pour un module X.
    - Contient tous les tests pour le module.
    - Si un test échoue, il faut continuer l’exécution, corriger toutes les erreurs trouvées et refaire la passe.
  + Deuxième passe :
    - Il faut que la première passe soit réussite à 100 % pour un module X.
    - Le module X est dépendant au module Y. Ainsi, lorsque le module Y est implémenté, il faut exécuter la deuxième passe du module X.
    - Contient seulement les **tests de régression** pour le module X.
    - Si un test échoue, il faut continuer l’exécution, corriger toutes les erreurs trouvées et refaire la **première** passe.
  + Troisième passe :
    - Il faut que, pour chacun des modules, la première et deuxième passe soient réussites à 100 %.
    - Contient seulement les **tests de régression** de tout le projet.
* Lorsqu’un module subit des changements importants, il est essentiel de refaire la première passe.
* Il faut créer une *Test Run*.
  + On la nomme comme suit :
    - <#\_de\_la\_Passe> - <Nom\_du\_module>
  + Mettre une version si besoin est.
  + Sélectionner le module à tester (ne pas oublier que si passe 2, seulement sélectionner les tests de régression).
* Laisser un commentaire significatif à la fin du test.

Pour les résultats des tests, on a :

* Trois résultats possibles :
  + Réussite (*Pass*).
    - Les résultats attendus sont obtenus lors de l’exécution.
  + Échec (*Fail*).
    - Les résultats attendus ne sont pas obtenus lors de l’exécution.
    - Assigner un ticket d’erreur.
  + Bloqué (*Could not test*).
    - Un test préalable à ce test a échoué.
  + À tester de nouveau (*Not applicable*).
    - Impossibilité de tester en ce moment (manque de matériel, etc.).

Pour le ticket d’erreur, on a :

**Création :**

* Lorsqu’il y a un échec, il faut créer un ticket d’erreur (*bug report*)*.*
  + Un même ticket d’erreur peut être la cause d’échec de plusieurs tests.
* La structure d’un ticket d’erreur est la suivante (juste remplir ce qui est listé ici, le reste est facultatif) :
  + Titre du bogue (*Bug title*).
    - <Nom\_du\_module> - <#\_du\_test> - <Court\_titre\_descriptif>
  + Priorité (*Priority*).
    - Si critique, cela veut dire que l’erreur doit être résolue le plus vite possible (blocker)
  + Type.
    - Type de l’erreur (fonction, texte, visuel, etc.).
  + Sévérité (*Severity*)
    - Si critique, cela veut dire que l’erreur est très grave.
  + Reproductivité (*Reproducibility*)
    - Combien de fois l’erreur arrive (à quelle fréquence).
  + Description
    - Décrire l’erreur de manière précise.
    - Finir avec la date de test
    - Ajouter le numéro du test case.

**Résolution d’un ticket (s’assurer que c’est une équipe différente de ceux qui ont testé) :**

* Changer le statut du ticket d’erreur à ‘Acknowledged’.
* Tester de nouveau pour confirmer l’erreur.
  + S’il n’y a pas d’erreur, discuter avec le reporteur du bogue et déterminer la suite à l’interne.
  + Sinon, changer le statut à ‘Confirmed’.
* Changer le statut à ‘In progress’ lorsqu’on travaille sur un correctif.
* Lorsque l’erreur est résolue, assigner le ticket à l’équipe qui a trouvé l’erreur et changer le statut à ‘Resolved’ (on peut ajouter la version dans laquelle le correctif est présent, si souhaité).
* L’équipe qui a maintenant le ticket doit tester de nouveau pour voir si le correctif est valide.
  + Si oui, on met le statut à ‘Closed’ et on refait la passe.
  + Sinon, on ajoute des commentaires expliquant pourquoi l’erreur n’est pas résolue, on réassigne le ticket à l’équipe qui a trouvé le correctif et on met le statut à ‘Feedback’.
* Si le statut est à ‘Feedback’, on recommence le processus comme si le ticket était à ‘Confirmed’.
* **Prendre note qu’il ne faut jamais supprimer un ticket d’erreur à des fins de traçabilité.**