POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

LOG1000 : Ingénierie logicielle Processus de développement d'un projet logiciel open source

Travail pratique # 3

Hiver 2019

1 Objectifs

- Exploration du processus d'assurance qualité d'un projet logiciel open source.
- Exploration du processus de contribution à un projet logiciel open source.

2 Sujet

Le logiciel open source qui servira d'exemple dans la première partie est Wireshark, projet qui utilise les logiciels Git, Gerrit et Bugzilla. Ensuite, il sera demandé de contribuer à un projet open source selon un processus semblable à celui de Wireshark.

3 Remise

- Rapport sous format PDF contenant les réponses aux questions, déposé dans votre répertoire Git d'équipe de laboratoire dans un dossier appelé TP3.
- N'oubliez pas de faire « git add », « git commit » et « git push » sur le rapport à la fin de votre TP pour vous assurer que le rapport soit soumis !

- N'oubliez pas de vérifier après la remise si votre rapport est beau et bien visible sur le serveur et pas seulement sur votre copie locale.

La remise doit se faire sur Git avant midi (12h00), le 11 Mars 2019. Il y a une pénalité de 10% par jour de retard.

4 Partie 1

4.1 Révision technique de code source

1. Qu'est-ce que le logiciel Gerrit et à quoi sert-il ?
2. Dans un contexte de contribution à un projet, qu'est-ce qu'une révision (patch) ?
3. Ouvrez la page Gerrit de Wireshark (https://code.wireshark.org/review/). Dans l'onglet All dans le coin gauche de la page, que signifie les sous-sections Open, Merged and Abandoned ?

   Dans la barre de recherche, envoyez la requête Icdf39f23bfcbd711f1b20a6bf7144f9fcff9744e, correspondant à l'identifiant d'une révision technique.
4. Qui est l'auteur de cette révision ?
5. Combien de fichier(s) a/ont subi de changement ? Le(s) quel(s) ?
6. Décrivez brièvement ce que fait cette révision.
7. Sur cette même révision, repérez la section Code-Review. Qu'est-il indiqué et qu'est-ce que cela signifie ?
8. (8.1 à 8.4) Répondez aux questions 4 à 7 inclusivement, mais cette fois avec la révision Id48cd9b876b4d272eb6dbdadf8a0859190cc0ce8.

4.2 Gestion des bogues

Le projet Wireshark utilise Bugzilla pour la gestion des bogues. Accéder au dépôt Bugzilla du projet par https://bugs.wireshark.org/bugzilla/.

9. Trouvez le bogue #15162. Quel est le nom de la personne qui a trouvé le bogue ? Est-ce que ce rapporteur est un être humain ?
10. Quel est le niveau d'importance de ce bogue ?
11. Comment est déterminé le niveau d'importance d'un bogue ?
12. Est-ce que le bogue #15162 risque de compromettre le fonctionnement du logiciel ? Pourquoi ?
13. Combien de personnes ont commenté le bogue, excluant le commentaire initial.
14. Est-ce que la communauté a confirmé la validité du bogue ? Si oui, comment ?
15. Est-ce que le bogue a été résolu ? Qu'est-ce qui vous l'indique ?

4.3 Intégration continue

L'intégration continue est une pratique de développement logiciel qui consiste à intégrer le code pour produire un exécutable régulièrement et automatiquement. L'objectif est de savoir en continu si les changements ("commits") effectués produisent toujours un logiciel fonctionnel qui passent les cas de tests définis. Wireshark utilise un outil d'intégration continue appelé Buildbot.

16. Expliquez l'utilité de Buildbot dans un contexte de projet open source.
17. Identifiez les différents builders disponibles pour ce projet en cliquant sur Builders au haut de la page (https://buildbot.wireshark.org/wireshark-master/).
18. Visitez la page Waterfall. Que signifient les les couleurs vert, rouge et orange ?
19. Identifiez un build qui a échoué ou bien qui possède des avertissements et collez une capture d'écran.
20. Expliquerz la cause des problèmes en accédant aux messages du build.
21. Quel est le numéro de la révision technique de ce build échoué et qui en est l'auteur ?

5 Partie 2

Dans cette deuxième partie, le processus de contribution à un projet open source publié sur Gitlab sera exploré. L'objectif étant de s'initier au processus de contribution à un projet d'équipe.

5.1 Prérequis

Rendez-vous sur la page de Gitlab (https ://gitlab.com/) et créez un compte de type Libre.

5.2 Développement

22. Après avoir créé un compte Libre sur Gitlab, Allez sur la page Gitlab : https://gitlab.com/somedPoly/log1000_h19.git. Dans ce dernier URL, à quoi correspond log1000_h19 (du point de vue de Gitlab) ?

    Accéder au répertoire du TP3-H19. Si un dialogue d'avertissement s'affiche, cliquer sur set a password pour HTTPS et créez un mot de passe, reconnectez-vous et accéder à la page du répertoire du tp3. Pour ce laboratoire, la connexion ne se fera pas sous SSH.

23. À quoi l'option « fork » sert-elle ? Sur la page des détails du répertoire, faites-un fork du projet pour votre compte Gitlab.
24. Sur cette même page, cliquez sur le bouton possédant le symbole d'addition et puis créez une nouvelle branche en lui donnant comme nom, le numéro de groupe de votre équipe (equipeX).

25. Clonez le répertoire localement sur votre machine à l'aide de Git et accéder à votre branche dans le terminal. Quelle commande avez-vous utilisée ? Le but des modifications de cette partie est d'ajouter, une commande correspondante à la création de l'objet « moteur.o » dans le makefile. Exécutez « make » pour générer les objets et l'exécutable du projet.
26. Ajoutez vos modifications au staging local de votre répertoire Git. Quelle commande avez-vous utilisée ? Faites un commit et un push de vos changements vers le serveur. Retournez sur votre compte Gitlab, **Projects -> Your projects** et puis le répertoire correspondant du TP. Sur la barre de navigation de gauche, accédez à la section Merge Requests

27. Qu'est-ce qu'un Merge Request?
28. Quel est l'utilité d'un Merge Request? Cliquez sur « New merge request », afin de créer un nouveau Merge Request.
29. Quelles sont les branches source et destination correspondantes à votre Merge Request ?

Continuez la création du Merge Request en y ajoutant un titre (supprimez WIP du titre) et une description. Indiquer dans votre titre le numéro de groupe de votre équipe. Laissez les deux options décochées, et faites la soumission du Merge Request.

5.2.1 Révision de code

Chaque équipe doit faire la révision de code d'une autre équipe correspondant à un Merge Request.

30. Choisir une équipe que vous allez réviser (N'oubliez pas de mentionner l'équipe que vous avez choisie dans votre rapport). Il faut choisir une équipe à réviser qui a eu au maximum 2 révisions. Une équipe pourra avoir au maximum 3 révisions.

Faites une révision technique du Merge Request que vous avez choisie.

31. Quel(s) est/sont les commits associées ?
32. Quel(s) est/sont les fichiers qui ont été modifiés ?
33. Est-ce que les modifications demandées sont correctes ?
34. Est-ce que les objets et l'exécutable du projet ont été créés ?
35. Attribuez un score aux modifications et justifiez votre choix.

Ajoutez des commentaires, incluant le score et votre justification et publiez-les dans la discussion du Merge Request.

6 Barème

4. Partie 1 (32pts)

1. [/1]
2. [/1]
3. [/2]
4. [/1]
5. [/2]
6. [/1]
7. [/1]
8. [/5]
9. [/1]
10. [/1]
11. [/1]
12. [/2]
13. [/1]
14. [/2]
15. [/2]
16. [/1]
17. [/1]
18. [/2]
19. [/1]
20. [/1]
21. [/2]

5. Partie 2 (18pts)

22. [/1]
23. [/1]
24. [/1]
25. [/2]
26. [/1]
27. [/1]
28. [/1]
29. [/2]

30. [/1]
31. [/1]
32. [/1]
33. [/1]
34. [/1]
35. [/3]

Total [/50]