Samuel Rondeau (1723869)

et

Pacome Bondet de La Bernardie (1617489)

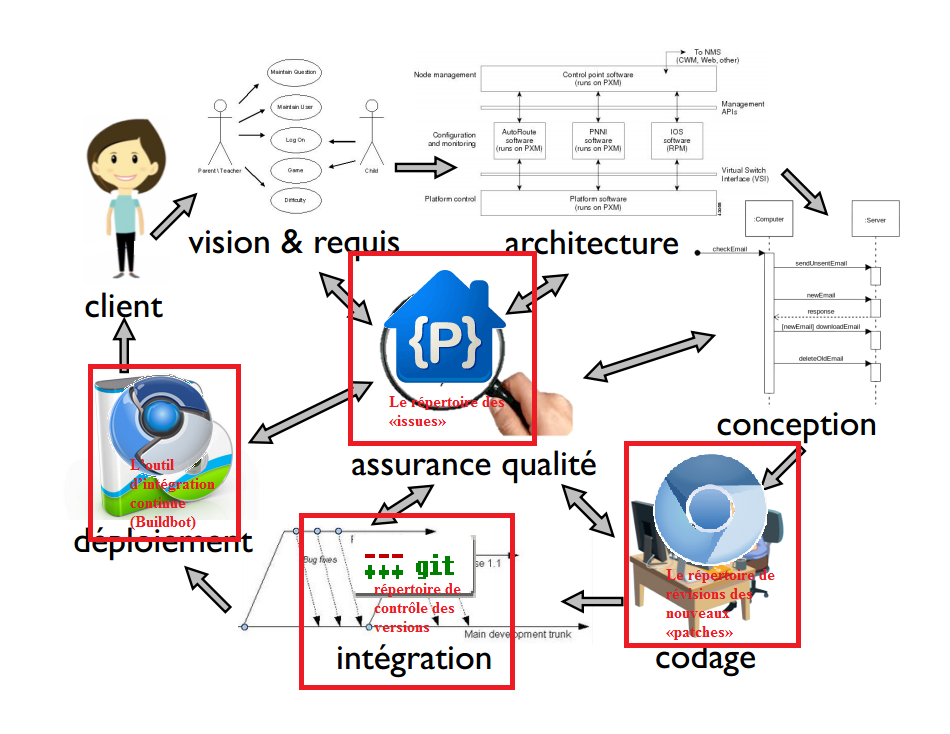
LOG1000 - Ingénieurie Logicielle

TP3

Rapport

**E1) Exploration des outils à votre disposition pour la production de Chromium. [/26]**

**E1.1)**



Évidemment, tout est relié à l’assurance qualité, car l’assurance qualité est importante dans toutes les étapes. Tout bogue est rapporté dans le répertoire des “issues.” Le codage est évalué via codereview, avant d’être officiellement ajouté à l’intégration (git). Ensuite, le buildbot se charge de prendre ce code et de le compiler dans le produit à jour.

**E1.2.A)**

Lors de la soumission d’un bogue, l’utilisateur doit confirmer avoir recherché dans la banque de bogues pour trouver si son bogue a déjà été rapporté. On peut donc dire que le mécanisme est un simple checkbox.

Explication: Si Alice désire rapporter qu’elle n’a pas de son lorsqu’elle est sur Youtube, elle cherche d’abord si quelqu’un a déjà rapporté ce bogue. Si oui, elle pourra se tenir au courant des corrections et contribuer. Sinon, elle pourra alors rapporter son bogue.

Cette méthode ne détermine pas si un bogue est réellement un bogue, mais évite les doublons.

**E1.2.B)**

Non. Son statut est “Available”. Si elle était prise en charge, son statut serait “Assigned”.

**E1.2.C)**

Elle est à priorité “Normal” (2) ceci est *indiqué à gauche en dessous de Type.*

***“Pri****-2”*

**E1.2.D)**

“chira”

**E1.2.E)**

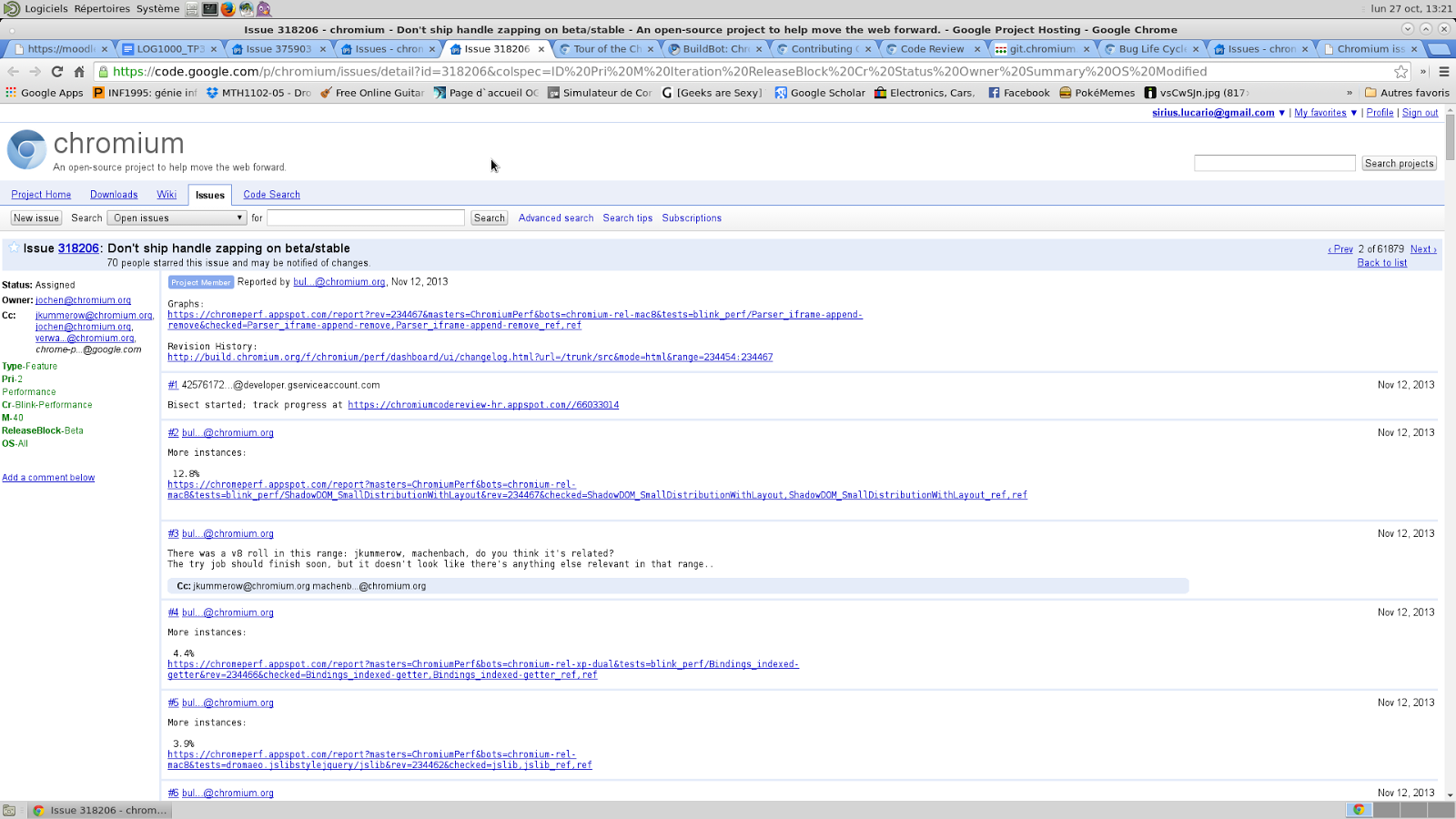


Image I : Capture d’écran d’un bogue assigné à quelqu’un

**E1.3.A)**

La revue de code sert à faire réviser un certain code par d’autres gens.

L’erreur est humaine, mais deux têtes valent mieux qu’une. La révision par les pairs permet de trouver une solution à un problème plus rapidement et s’assurer qu’un code est bon. Cela suit bien la loi de Linus Torvalds.

**E1.3.B)**

Cette revue concerne une patch, mais pas pour résoudre un bogue. Le but est d’enlever un fichier obsolète.

**E1.3.C)**

Au moment de la capture d’écran, 5 Patches ont été soumis.

**E1.3.D)**

Non. Sergey Ulanov n’a pas encore donné son approbation.On voit que son nom n’est pas affiché en vert dans l’onglet à gauche.

**E1.4.A)**

Dans cette capture d’écran, on voit compiling, update, gclient\_revert, cleanup\_temp, taskkill, svnkill, update\_scripts sont les étapes executés par le Buildbot.

**E1.4.B)**

Les “buildsalves” impliqués sont les suivants:

Win: build51-m1

Mac: vm682-m1

Linux: build14-m1

Linux x64: build48-m1

Android: build7-m1

Ce sont les différentes plateformes où on peut exécuter le programme conçu.

**E1.4.C)**

Oui, car l’arbre est ouvert. Si un slave échoue, l’arbre se ferme. Donc, le dernier build fut un succès.

**E1.4.D)**

Le build courant a commencé à 21:47:41

**E2) Cheminement pour devenir un Googler [/13]**

**E2.1.A)**

La Commande pour cloner un répertoire Git est la suivante:

git clone http://git.chromium.org/chromium/src.git

**E2.2.A)**

<https://code.google.com/p/chromium/issues/detail?id=196399&can=6&sort=-pri&colspec=ID%20Pri%20M%20Iteration%20ReleaseBlock%20Cr%20Status%20Owner%20Summary%20OS%20Modified>

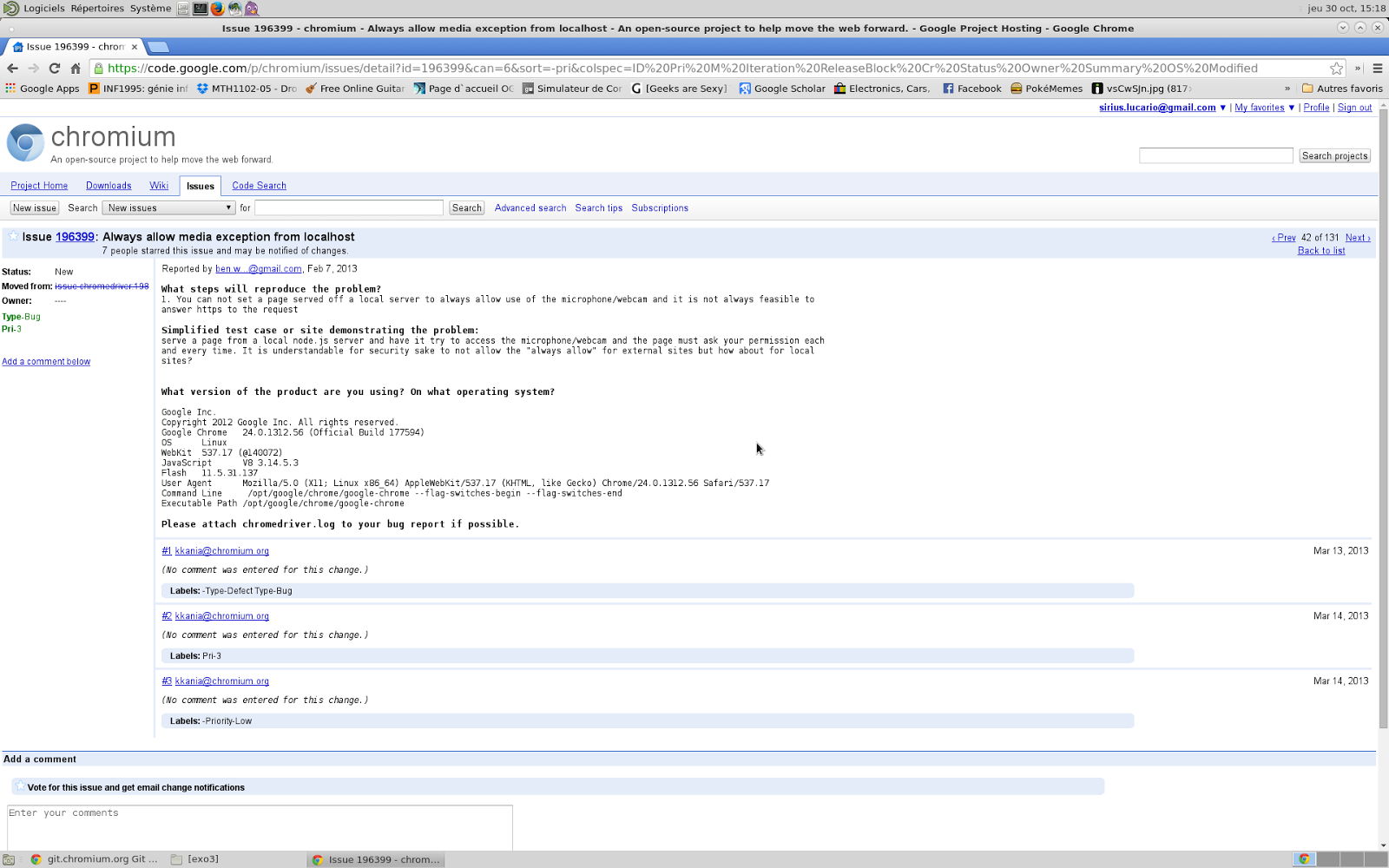


Image II : Capture d’écran d’un bogue nouveau à prendre en charge pour un débutant

On choisit un bogue qui est nouveau, c’est-à-dire qu’il a été soumis récemment (“New”). De plus, ce Bogue n’est pas à haute priorité (“Pri-3”). Ce serait bon pour un débutant.

**E2.2.B)**

Upload ne fait que déposer le code dans le répertoire de révisions (<https://codereview.chromium.org/>) pour être révisé par ses pairs. Commit soumet le code au serveur du gestionnaire de révisions.

**E2.2.C)**

Déposer le code dans le serveur des révisions et bien le documenter.

On peut aussi *flagger* un réviseur pour nécessairement obtenir son *LGTM*.

**E2.2.D)**

Non, ceci n’est pas possible. Il faudrait soit déjà être en train de travailler pour Google ou être approuvé par un employé qui travaille pour Google, et obtenir un compte “@chromium.org”. Ceci est généralement fait après avoir écrit quelques patches qui ont bien été approuvées.

**E3) Votre vie de Googler [/6]**

**E3.1.A)**

Non. L’approbation de tous les réviseurs impliqués est requise pour que l’arbre soit vert. Lorsque l’arbre est vert, le code peut être soumis.

**E3.2.A)**

Oui, en passant par Reitveild, on coche “commit” et les changements seront ajoutés au “commit queue”, qui est un bot de commit. (http://www.chromium.org/developers/testing/commit-queue).

**E3.2.B)**

* Le plus récent essai sur le serveur a été approuvé
* On est présent sur le IRC, ou au moins disponible par messagerie instantanée
* Être disponible durant le processus du build

Références images:

Bram Adams (notes du cours LOG1000)

<https://git.chromium.org/gitweb/>

<https://www.iconfinder.com/icons/88447/projecthosting_icon#size=128>

<http://commons.wikimedia.org/wiki/File:Chromium_11_Logo.svg>

<http://alien.slackbook.org/blog/new-chromium-now-supports-netflix-natively-without-wine-plugins/>