UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MÉMOIRE DE SYNTHÈSE SUR LA MAINTENABILITÉ D'UN LOGICIEL

JENKINS

MGL7460-90 – RÉALISATION & MAINTENANCE DE LOGICIELS

VERSION INITIALE

PAR LILIAN OLOIERI - OLOL21017102

HIVER 2020

Table des matières

Introduction

Jenkins est un logiciel libre, multi plateforme, d'intégration et de livraison continue (CI/CD), ce qui permet aux développeurs de créer et tester les projets logiciels en continue, d'intégrer plus facilement les modifications au projet et facilite aux utilisateurs d'obtenir une nouvelle version du logiciel. Il vous fournissent des moyens puissants pour définir vos pipelines de construction et en s'intégrant à un grand nombre de technologies de test et de déploiement. A ce jour, c'est un application principale de type open-source pour cette tâche (vs Travis, Azure DevOps).
L'application a été fondée en 2006. L'auteur principal et fondateur du projet est Kohsuke Kawaguchi (koshuke). Le nom initial du projet était Hudson, qui a été ensuite transformé dans Jenkins. Le premier commit dans Git date du 5 novembre 2006. Donc on a fait un analyse de développement du logiciel à partir de cette date.

Dimension "Équipe De Développement"

Qui Sont Les Développeurs Principaux Du Projet ?

Suite à l'analyse des résultats de nombre des commits par développeur on a sorti 10 principaux participants au projet. On a du faire la somme par nom et nom utilisateur (les logs de git contient les deux).

Nom	Commits	Pourcentage %	Période
Kohsuke Kawaguchi	17033	57.96	
Jesse Glick	7509	25.55	
Daniel Beck	1277	4.35	
Oleg Nenashev	1249	4.24	
Olivier Lamy	1097	3.73	
Seiji Sogabe	786	2.67	
Stephen Connolly	663	2.25	

Christoph Kutzinski	620	2.1	
mindless - alainharder	593	2.01	
Oliver Gondza	582	1.98	

A faire - à analyser les mêmes stats par lignes des codes par dev et aussi par type de fichier.

L'équipe De Développement Est-elle Stable ?

Kohsuke Kawaguchi et Jesse Glick sont les deux principaux auteurs qui ont mis les bases du projet en 2006, et ont resté actives jusqu'au présent.

Ces deux, avec le proéminence de Kohsuke Kawaguchi, ont été les principaux commiteurs jusqu'à la fin de l'année 2008. Le troisième, Visizikov, a commencé à travailler avec eux, mais a cessé ces activités peu après.

A faire - un analyse des évolution plus récent d'équipe.

Comment Est Répartie La Paternité Du Code Source Dans L'équipe ?

Question à développer - étendre la fonctionnalité des scripts.

Comment Les Développeurs Communiquent Entre Eux ?

Côté communication, la communauté Jenkins est assez bien organisée. Il y a des cannaux divisés par sujet. L'équipe de développement organise des rencontres prédéfinies et ouvertes pour ses membres toutes les deux semaines. J'ai bien apprécié la quantité de documentation sur tous les aspects du système.

https://wiki.jenkins.io/#recently-viewed

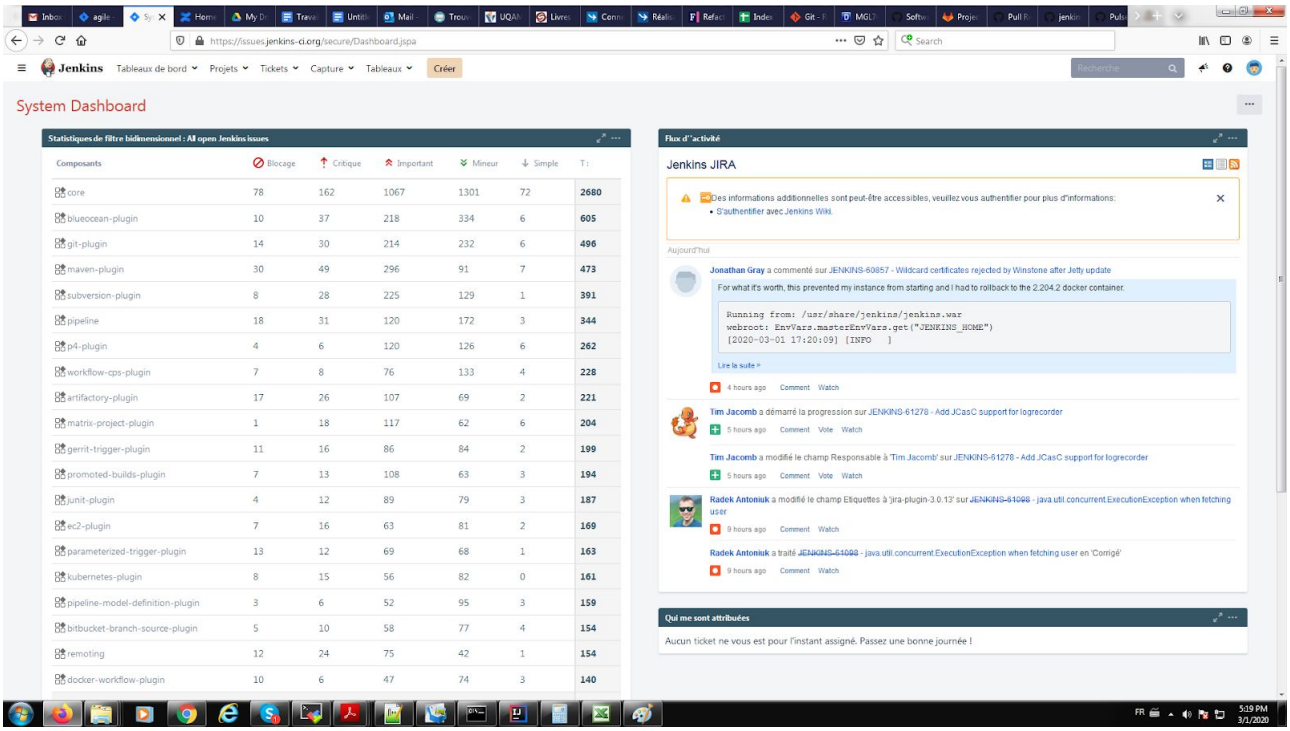
A faire - participer a un de ces rencontres d'équipe.

https://jenkins.io/chat/

Dimension "Exigences"

Comment Le Code Source Est-il Tracé Aux Exigences ?

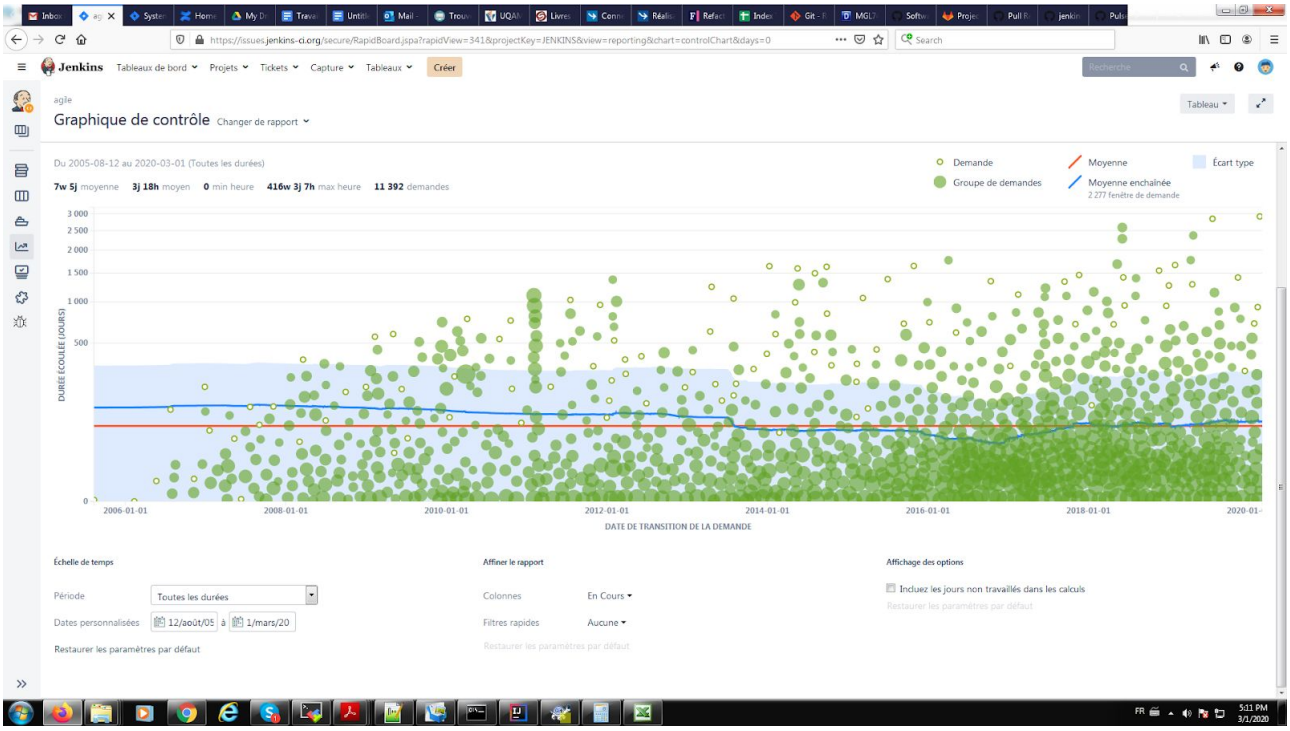
L'équipe de Jenkins utilise un JIRA assez bien configuré. Je me suis fait un compte pour pouvoir extraire des données et analyser leur dynamique.



Quelle Est La Vélocité De L'équipe De Développement Sur La Maintenance Corrective ? Quel Est Le Volume D'exigence Traité Dans Le Projet ?

Pour la toute la période d'existence du projet ils ont traité 11 392 demandes, avec un moyen de 7 semaines 5 jours par demande, que c'est assez bien pour un équipe open source.

A faire - participer au développement d'au moins quelque tache pour valider la dynamique d'équipe.



Dimension "Architecture Logicielle"

Quelles Sont Les Composantes Principales De L'application ?

Il y a trois composantes principales de l'application : Module Core, module client et partie web.

Comment Est Gérée La Modularité De L'application ?

Si on prend en considération la panoplie des plugins développés autour du core du système, on pourra coter Jenkins comme une application très modulaire. Il y a à présent une très bonne documentation sur le développement des plugins pour Jenkins.

Comment Est Documentée L'architecture De L'application ?

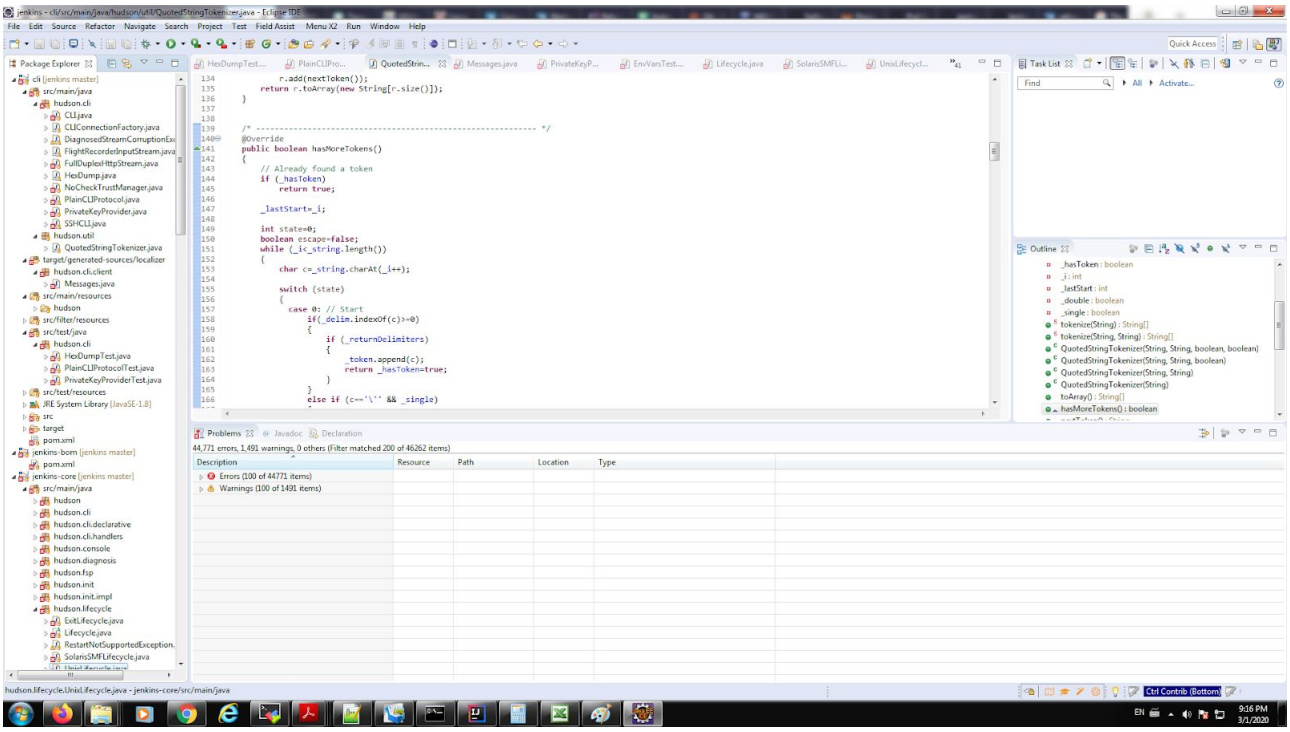
Pour le moment, aucune documentation trouvée sur ces points.

A faire - se clarifier sur l'architecture du projet.

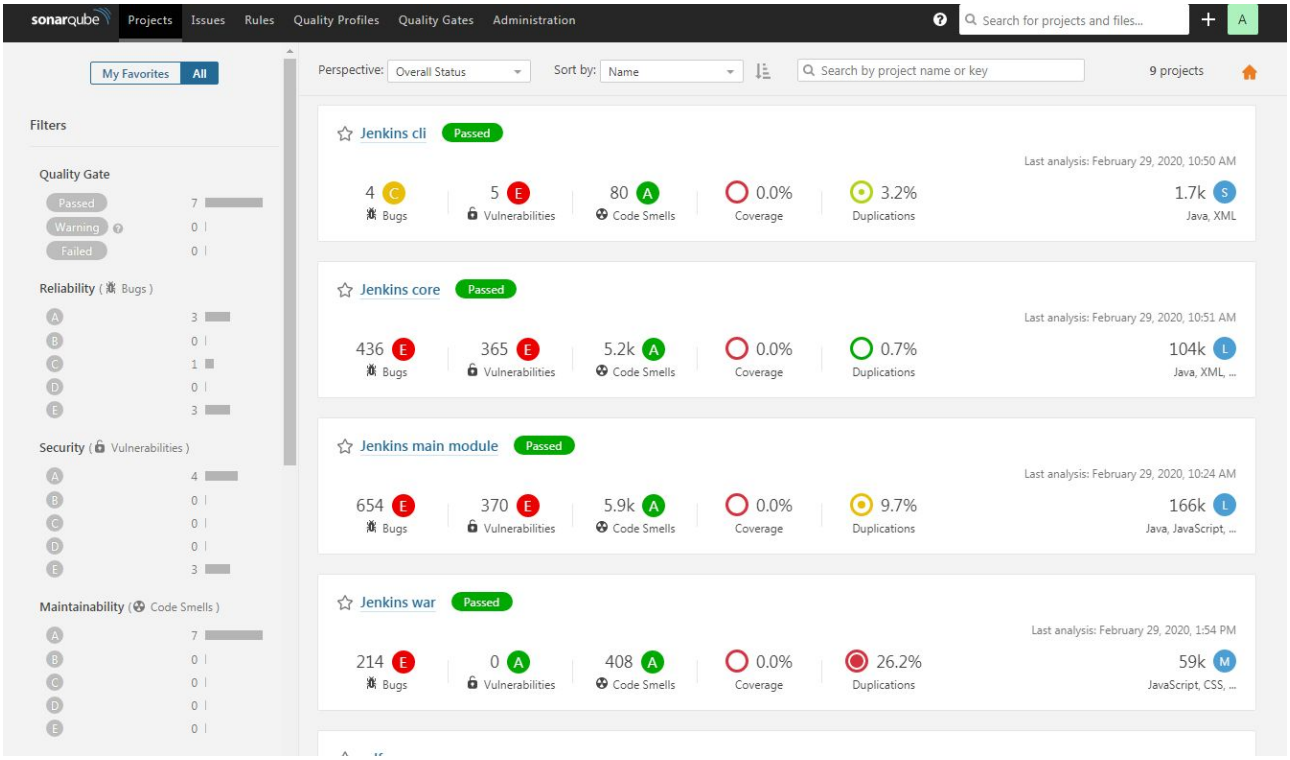
Dimension "Code Source"

Comment Qualifiez Vous La Qualité Du Code Source ?

Même si j'en trouve des places ou il faudra du refactoring (noms des variables moins intuitives, méthodes trop longs),



de façon général la qualité du code est assez bonne à mon avis. Ce qui est confirmé par les données sorties dans SonarQube :



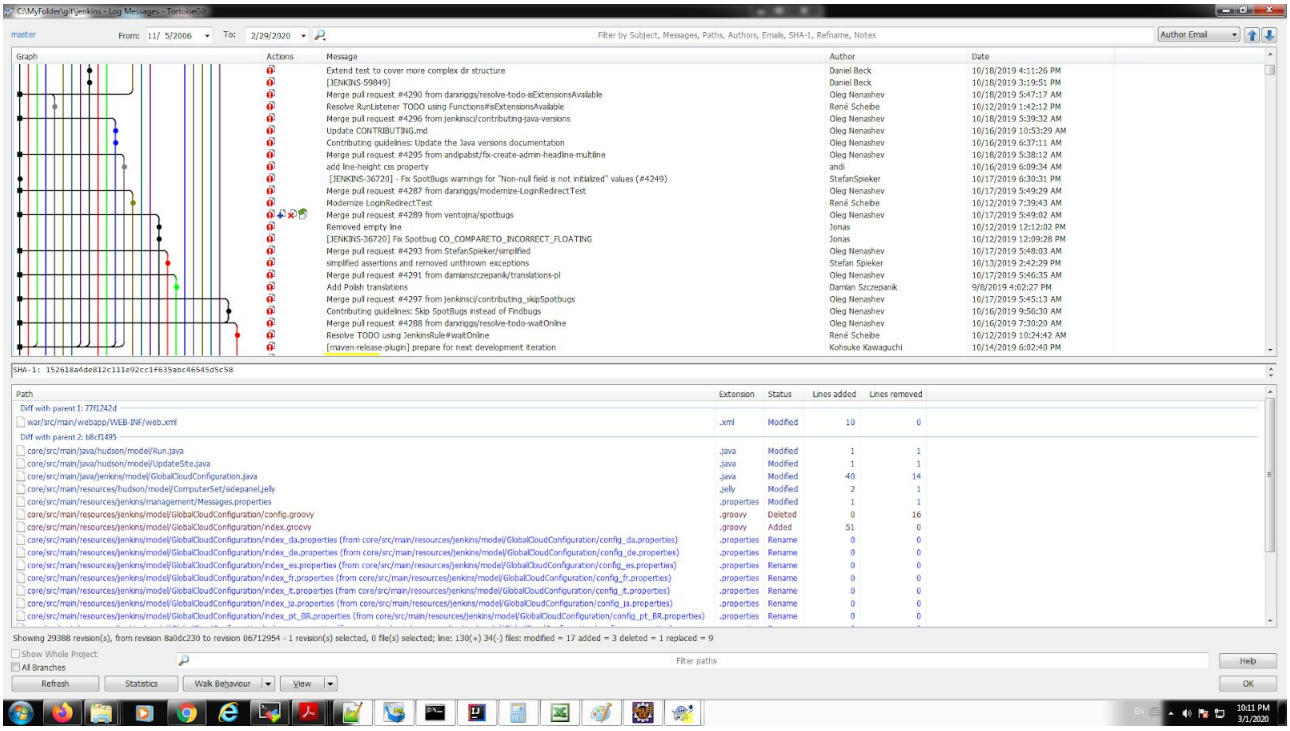
A faire - une analyse plus détaillée de résultats de SonarQube et possiblement d'utiliser un autre outil (PMR, FindBugs)pour faire comparer les résultats de ces deux.

Quels Outils De Construction (P.-ex. Make, Maven) Sont Utilisés ?

Maven est utilisé comme projet outil de construction. L'utilisation est assez facile et intuitive … ce qui montre le point sur la facilité de construction du projet - un des points importants sur la maintenabilité. Le projet s'installe facilement dans IntelliJ, par contre j'ai encore des petits soucis sur Eclipse.

Quel Modèle De Branche Est Utilisé Dans Le Développement Du Projet ?

Même si l'équipe utilise une modèle assez élaboré en utilisant un branch 'stable', il reste inquiétant le nombres des merge direct dans le master.



Dans Quelle Mesure La Compilation Du Code Est-elle Reproductible ?

La compilation ne pose aucun problème avec la configuration actuelle de Maven. Très bonne et détaillée documentation pour installation et participation au projet. Voir : https://github.com/jenkinsci/jenkins/blob/master/CONTRIBUTING.md

A Quel Point Le Code Est-il Dépendant De Bibliothèques Externes ?

A faire - se clarifier ce point.

Dimension "Tests"

Quelles Sont Les Méthodes De Tests Mise En Place Dans Le Projet ?

Ils sont présentés des tests unitaires, d'intégration et même JavaScript. Par contre à la compilation par Maven j'ai du utilisé -DskipTests, pour pouvoir builder le projet.

A faire - se clarifier avec le problème des tests.

Comment Qualifiez Vous La Qualité Des Tests Mis En Oeuvre ?

La couverture par tests unitaires ne sont pas suffisantes. Il y a des lacunes aussi dans la configuration des tests. Comme résultats, j'ai une couverture de 0% pour tous les projets dans SonarQube.

A faire - se clarifier avec le problème de couverture de Sonar pour sortir des vrais données sur la couverture des tests. A analyser la qualité du code des tests.

Dans Quelle Mesure Les Tests Sont-ils Reproductible ?

Il faut régler les problèmes sur la couverture pour pouvoir répondre à ce point.

Dimension "Déploiement & Livraison"

Quels Outils D'intégration / Déploiement Continus Sont Mis En Place ?
Comment Sont Gérées Les Dépendances Logicielle Dans Le Projet ?
Quelle Méthodologie De Publication (Releasing) Est Mise En Place ?

Conclusion

Maintainability

defined as the probability that a system or system element can be repaired in a defined environment within a specified period of time. Increased maintainability implies shorter repair times (ASQ 2011).

Source SEEBOK

Caractéristiques de la maintenabilité selon ISO25000 :

Modularité
Réutilisabilité
Potentiel d'analyse
Potentiel de changement
Stabilité des modifications
Potentiel de test

faire l'analyse en se basant sur la définition de maintenabilité.