université de BORDEAUX

Impact of Renaming on Software Change Metrics

Pierre Chanson

Mémoire de stage de Master2
Encadrants: Jean-Rémy Falleri et Matthieu Foucault

LaBRI, UMR 5800
F-33400, Talence, France

Email: pierre.chanson@etu.u-bordeaux.fr, {falleri,mfoucaul,xblanc}@labri.fr

14 mai 2014

Table des matières

1 Introduction

L'accès aux dépôts logiciels a rendu possible de nombreux travaux de recherche sur l'évolution logicielle. Plus particulièrement, les dépôts de code source gérés par des outils de contrôle de versions (Version Control System, VCS, comme SVN, Mercurial ou encore Git) contiennent l'historique de construction d'un logiciel. Au cours de son histoire, un fichier peut être renommé et/ou déplacé, actions qui doivent être prises en compte par les études qui analysent l'évolution des fichiers de code source. Le refactoring est present dans le developement des logiciels à succès d'aujourd'hui, mais sous estimé. Lors de mon stage, les résultats de mes expérimentations aménéront à une publication dans la conference ICSME, article en annexe.

2 Etat de l'art

2.1 Evolution logiciel et refactoring

Pourtant, nombre d'article parlent en introduction de l'importance du refactoring, ce qui inclue le renommage, dans les projets à succès (TODO) mais on obtient pas de chiffres précis.

2.2 Métriques de procédés et évolution logiciel

Les métriques de procédés (change metrics) permettent de calculer à quel point une entité de code source à été modifiée au cours d'une période donnée. On les utilises usuellement pour prédire les bugs entre deux révisions d'un logiciel, souvent la dernière période. L'objectif étant de prédire les bugs qui apparaitront lors de la prochaine release, elles ne considère que les entités étant toujours présentes à la fin de la période. Elles vont donc se concentrer sur les entités actives lors de la période et que l'on retrouvera à la fin.
Un gestionnaire de versions offre plusieurs moyens de calculer ces métriques car il stocke les informations sur les entités modifiés à chaque nouvelle version, l'auteur de ces modifications, la date etc. De plus il permet la récupération du contenu de chaque entité et de l'ensemble d'un projet à une version donnée. Pour calculer ces métriques, il est donc possible d'analyser chaque entités modifiés lors d'une période puis de ne garder uniquement les entités toujours présente à la dernière version de notre période.
Nous avons choisi de nous concentrer sur les trois métriques de procédés les plus connues identifiés par Radjenovic et al (TODO) pour mesurer l'impact du renomage : Le nombre de développeurs (NoD), le nombre de modifications (NoC), et le Code Churn (CC).
En exemple nous allons considerer A, l'ensemble des entités existantes entre deux versions. TODO
NOD :
NOC :
CC :

2.3 Métriques et Renommage

Comme on l'a expliqué plus tôt, un gestionnaire de versions est très pratique pour calculer les métriques de procédés. Par ailleurs, il faut noter qu'un gestionnaire de version identifie une entité par son chemin + nom de fichier. On en déduit qu'un renommage de cette identité, dans le chemin ou le nom de fichier, aura un impact sur le calcul des métriques. Pour expliquer cet impact, on présente un exemple d'historique d'un logiciel figure 1. Ce projet ne contient qu'une entité, Test.php, qui est renommé en Hello.php dans la dernière version. Dans cet exemple nous calculons NoD, NoC, CC entre la version 1 et 3.
La dernière version ne contient qu'une entité. C'est donc cette entité uniquement qui sera considérée. De plus l'identité exacte de cette entité n'aparait que lors de la version 3. Le calcul des métriques est donc trivial, NoD = 1, NoC = 1, CC = 2.

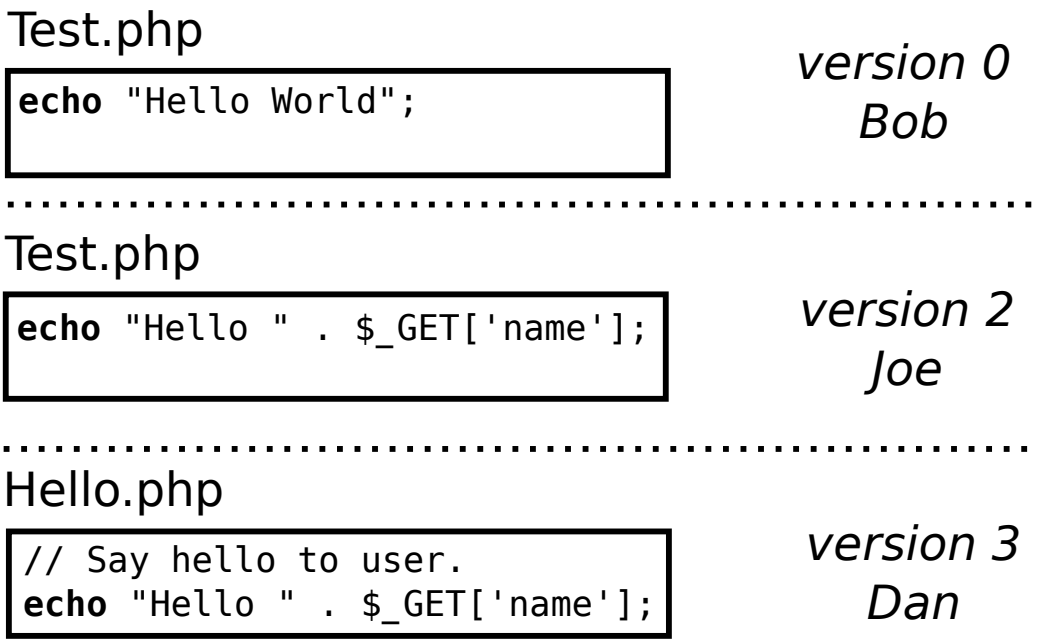

FIGURE 1: Example of a project history. The project is composed of only one file `Test.php` which is renamed to `Hello.php` in the last version.

Par ailleurs, si on prend en compte le fait que ce fichié a été renommé, il y a trois versions à regarder en ce qui concerne l'entité. etc...TODO

2.4 Métriques et Renommage Existant

Nous nous somme donc intéressés aux études passées qui pouvaient traiter les métriques de procédés dans la prédiction de bugs, et vérifié si ces études avaient considérés le renommage de fichiers. L'article (TODO) référence 26 articles qui nous intéresse, 15 sur des projets industriels, 11 sur des logiciels open-source. TODO

2.5 Les outils

Comme nous l'avons vue, il existe un certains nombre de gestionnaires de versions pour effectuer notre détection de renommage, étant donné que nous avons simplement besoin de versions à comparer. Nous avons néhanmoins étudié les VCS potentiels, afin de voir quel VCS sera le plus apte à gérer les renommages. La table1 résume notre étude. TODO Git propose un algorithme de détection automatique de renommage.

2.6 "Origin Analysis"

L'algorithme utilisé par Git est connu sous sous le nom de "Origin Analysis" et est expliqué par Godfrey et al dans les articles, (TODO) (An integrated approach for studying architectural evolution, Tracking Structural Evolution Using Origin Analysis, Using origin analysis to detect merging and splitting of source code entities) Origin Analysis TODO

3 Problématique

Nous n'avons pas réellement trouvé d'études traitant le renommage. Ces études ont-elles volontairement ou non omis le renommage ? La problématique qui se pose est donc, quelle est la quantité

	Renaming handling		
		Automatic	
Tool	Manual	Standard	Optional
CVS			
Subversion	×		
Mercurial	×		×
Git			×

Table 1: Handling of renaming of the main VCS tools.

de renommages ou déplacements de fichiers dans les projets ? Où interviennent-ils ? Ont-ils un réel impact sur les métriques de procédés ?

4 Un ensemble de projet

Nous avons commencés par selectionner un ensemble de projets sur lesquels effectués nos expérimentations. Des projets open-source et conséquents et connues de la communautés MSR. Voir table 2 (TODO)

5 Première analyse à grain fin

Le premier travail réalisé a été de faire une étude manuelle des renommages dans les VCS. Nous avons sélectionnés 100 commits de manière aléatoire et étudié le renommage d'entités dans ces commits. Nous avons définie le changement d'identité (TODO) et diférencié le renommage direct du renommage induit (TODO)

6 Analyse à gros grain

6.1 Première experience

6.2 deuxième expérience

7 Resultats

7.1 resultats première expérience

7.2 resultats deuxième expérience

8 Conclusion

Références