

L'impact du renommage sur les métriques de procédés

Pierre Chanson

Encadrants: Jean-Rémy Falleri et Matthieu Foucalt

La prédiction de bugs

- Etudes sur l'historique de création d'un logiciel. « *MSR* »*
- *Gestionnaire de versions (VCS)*
- Prédiction de bugs dans une prochaine version.
- Un modèle de prédiction : Les métriques

* Kim Herzig and Andreas Zeller. "Making Software : What Really Works and Why We Believe It, chapter Mining Your Own Evidence", O'Reilly Media, 2010.

Un modèle de prédiction

- Métriques logicielles : Version fixée d'un logiciel.
(Lignes de codes, complexité du code)
- Métriques de procédés : Evolution du logiciel, suite de versions.

Bird et al. "Don't touch my code ! : examining the effects of ownership on software quality", ESEC/FSE 2011.

Nagappan et al. "Use of relative code churn measures to predict system defect density", ICSE 2005.

Métriques de procédés

- NoD (Number of Developers): nombre de développeurs
- NoC (Number of Changes): nombre de modifications
- CC (Code Churn): nombre de lignes ajoutées ou supprimées

* Radjenovic *et al*,
“Software fault prediction metrics : A systematic literature review.”, Information and Software Technology, 2013

Exemple métriques et renommage

Test.php

```
echo "Hello World";
```

version 1
Bob

Test.php

```
echo "Hello " . $_GET['name'];
```

version 2
Joe

Hello.php

```
// Say hello to user.  
echo "Hello " . $_GET['name'];
```

version 3
Dan

NoD,

NoC,

CC

Problématique



Le **renommage** d'élément du code source peut-il fausser le calcul des métriques de procédés ?

Exemple

calcul de métriques sans détection de renommage

Test.php

```
echo "Hello World";
```

version 1
Bob

Test.php

```
echo "Hello " . $_GET['name'];
```

version 2
Joe

Hello.php

```
// Say hello to user.  
echo "Hello " . $_GET['name'];
```

version 3
Dan

NoD = 1, NoC = 1, CC = 2

Exemple

calcul de métriques avec détection de renommage

Test.php

```
echo "Hello World";
```

version 1
Bob

NoD = 1, NoC = 1, CC = 1

Test.php

```
echo "Hello " . $_GET['name'];
```

version 2
Joe

NoD = 2, NoC = 2, CC = 1

Hello.php

```
// Say hello to user.  
echo "Hello " . $_GET['name'];
```

version 3
Dan

NoD = 3, NoC = 3, CC = 2

Analyse des études antérieures

Radjenovic et al, "Software fault prediction metrics : A systematic literature review", Information and Software Technology, 2013.

- Des études et expérimentations pour la prédiction de bugs, qui utilisent les métriques de procédés :
- 11 projets open source
- 15 projets industriels

maven



ArgoUML

Microsoft®

Questions de recherche

- Quelle est la quantité de renommage dans les projets ?
- Ce renommage a-t-il un impact réel sur les métriques de procédés ?

Etude préliminaire traitement du renommage par les VCS

- CVS : Aucun
- SVN, Mercurial : Manuel*
- **Git** : Automatique et optionnel

Niveau de granularité : Fichier

* Kim et al,

“A field study of refactoring challenges and benefits”, *Foundations of Software Engineering*, 2012.

Corpus

5 projets open-source :

Projet	Language dominant	Taille (LoC)	Nombre de développeurs	URL
Jenkins	Java	200851	454	github.com/jenkinsci/jenkins
JQuery	JavaScript	41656	223	github.com/jquery/jquery
PHPUnit	PHP	21799	152	github.com/sebastianbergmann/phpunit
Pyramid	Python	38726	205	github.com/Pylons/pyramid
Rails	Ruby	181002	2767	github.com/rails/rails

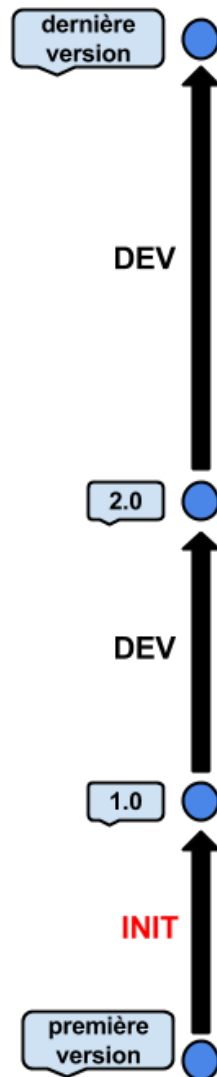
- Git
- Langages généralisés
- Matures

Première expérience



- Quantité de renommage
- Localisation

Première expérience



- Quantité de renommage
- Localisation

Première expérience

1. Lister les fichiers existant à la fin de la période.
2. Pour chacun de ces fichiers, extraire sa séquence de modifications durant la période en activant la détection de renommage (commande `git log -M`).
3. Calculer le pourcentage de fichiers %FR qui inclue au moins un renommage.

Première expérience

1. Lister les fichiers existant à la fin de la période.
2. Pour chacun de ces fichiers, extraire sa séquence de modifications durant la période en activant la détection de renommage (commande `git log -M`).
3. Calculer le pourcentage de fichiers %FR qui inclue au moins un renommage.

```
commit 3d87c26845095438b6c946dc4e1029280593fb91
Author: Aaron Patterson <aaron.patterson@gmail.com>
Date:   Fri May 2 11:52:37 2014 -0700

    push up bind params on "simple" subquery calculations
    bind parameters we not being propogated to simple subquery
    calculation calls. This fixes it

activerecord/lib/active_record/relation/calculations.rb      | 6 ++++--
activerecord/test/cases/associations/has_many_associations_test.rb | 8 ++++++--
2 files changed, 11 insertions(+), 3 deletions(-)
```


Première expérience

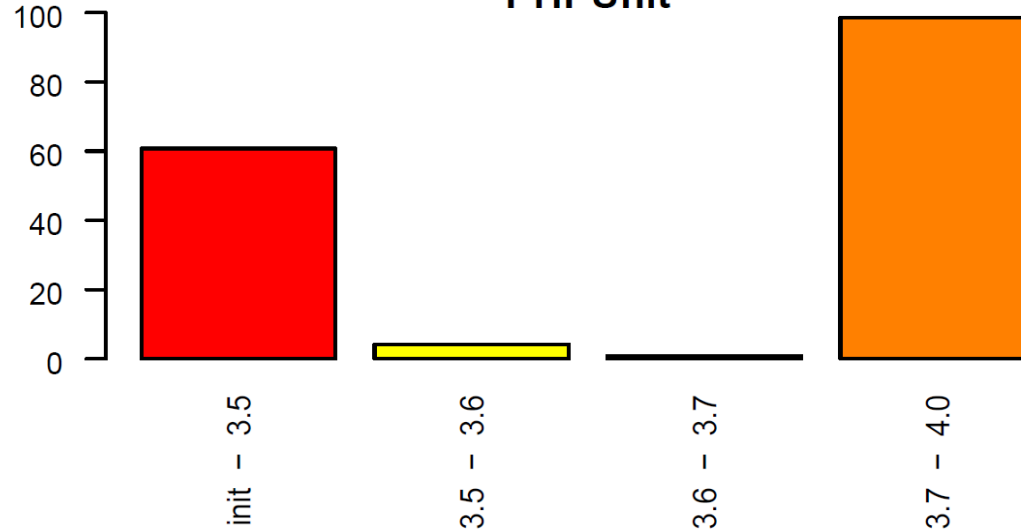
1. Lister les fichiers existant à la fin de la période.
2. Pour chacun de ces fichiers, extraire sa séquence de modifications durant la période en activant la détection de renommage (commande `git log -M`).
3. Calculer le pourcentage de fichiers %FR qui inclue au moins un renommage.

Résultats première expérience

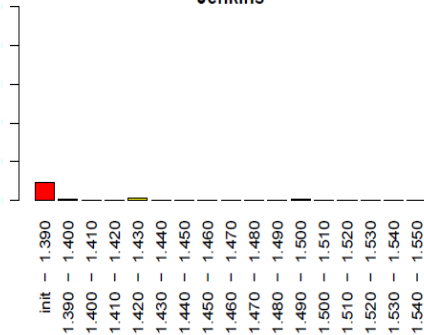
Pourcentage de
fichiers renommés

périodes
■ initiale
■ majeur
■ mineur

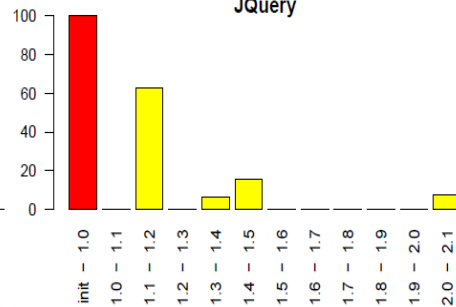
PHPUnit



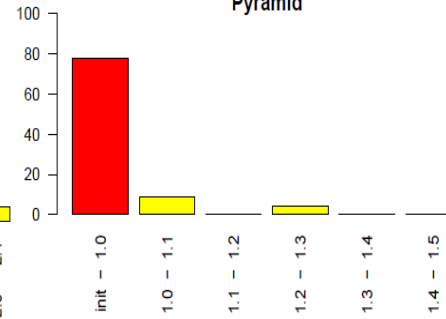
Jenkins



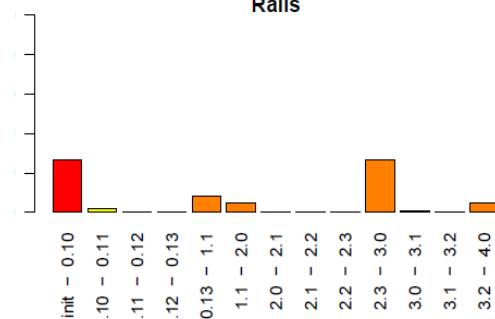
JQuery



Pyramid



Rails



Deuxième expérience Spearman

- Comparer les calculs de NoD, NoC, CC **avec / sans** détection du renommage.
- Corrélation de **Spearman** : Calcul du coefficient de corrélation entre les rangs des valeurs de deux variables.
 - 1: Corrélation parfaite (-1: Corrélation inverse)
 - 0: Aucune corrélation
 - $\approx 0,5$ mauvaise corrélation

Deuxième expérience Méthodologie

Calculs de **NoD**, **NoC**, **CC** sur une période par projet.

Sans détection :		Avec détection :	
Fichiers	Valeurs (NoC/NoD/CC)	Fichiers	Valeurs (NoC/NoD/CC)
A	3	D	5
B	2	C	4
C	2	A	3
D	1	B	2

Calcul du coefficient de corrélation de **Spearman**.

Résultats deuxième expérience

Coefficient de corrélation de Spearman :

Période	$\%F_R$	Métriques de procédés		
		CC	NoD	NoC
Jenkins 1.420 - 1.430	0.86	1	1	1
JQuery 1.1 - 1.2	62.5	0.98	0.08	0.59
PHPUnit 3.7 - 4.7	98.51	0.6	0.38	0.71
Pyramid 1.0 - 1.1	8.69	0.97	1	1
Rails 2.3.0 - 3.0.0	26.49	0.98	0.96	0.93

Limitations

- Dépend de Git: « Origin Analysis »
- 100 renommages aléatoires :
 - 100% précision
 - Rappel inconnu

Conclusion

Le renommage :

- Jusqu'à **100%** des fichiers d'un projet.
- **Fausse** le calcul des métriques de procédés.

Les recommandations :

- Périodes initiales
- Utiliser un algorithme de détection de renommage, Git à défaut
- Attention au niveau de granularité
- Le renommage dans les prochaines études

Prochaine étapes:

- Evaluer la précision des algorithmes de détection de renommage
- Etude des split et merge d'entités

Annexe A

Analyses des études antérieures

Open-source

Erik Arisholm, Lionel C. Briand, and Eivind B. Johannessen. A systematic and comprehensive investigation of methods to build and evaluate fault prediction models. Journal of Systems and Software, 83(1) :2 – 17, 2010. SI : Top Scholars.

Todd L. Graves, Alan F. Karr, J. S. Marron, and Harvey Siy. Predicting fault incidence using software change history. IEEE Trans. Softw. Eng., 26(7) :653–661, July 2000.

T.M. Khoshgoftaar, R. Shan, and E.B. Allen. Using product, process, and execution metrics to predict fault-prone software modules with classification trees. In High Assurance Systems Engineering, 2000, Fifth IEEE International Symposium on. HASE 2000, pages 301–310, 2000.

Industriels

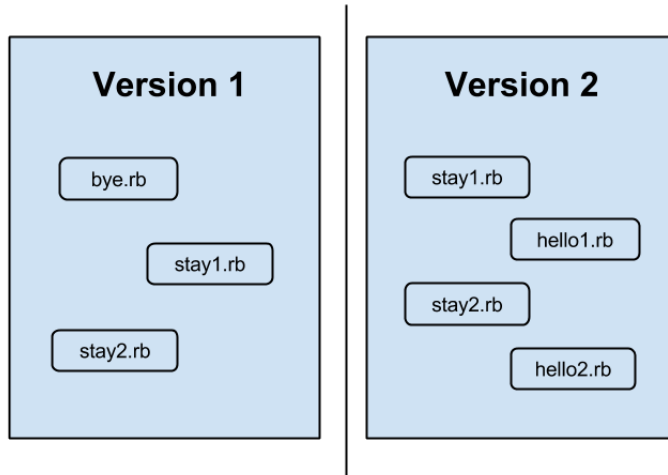
Alberto Bacchelli, Marco D'Ambros, and Michele Lanza. Are popular classes more defect prone ? In Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering, FASE'10, page 59–73, Berlin, Heidelberg, 2010. Springer-Verlag.

Bora Caglayan, Ayse Bener, and Stefan Koch. Merits of using repository metrics in defect prediction for open source projects. In Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, FLOSS '09, page 31–36, Washington, DC, USA, 2009. IEEE Computer Society.

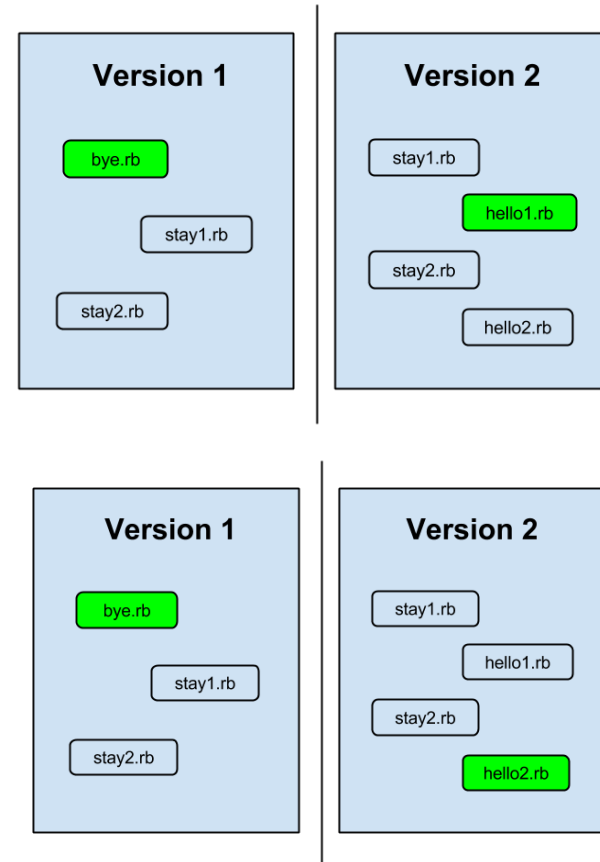
M. D'Ambros, M. Lanza, and R. Robbes. On the relationship between change coupling and software defects. In Reverse Engineering, 2009. WCRE '09. 16th Working Conference on, pages 135–144, October 2009.

Annexe B

« Origin Analysis* »



Composants d'un projets lors de deux versions consécutives.
Mise en corrélation des éléments
« supprimés » et « créés » pour détecter les renommages potentiels.



* Godfrey et al. "An integrated approach for studying architectural evolution", International Workshop on Program Comprehension, 2002.

Annexe C

Corrélation de **Spearman** :

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

Avec (x_i, y_i) les rangs données par les deux valeurs de Métriques (X_i, Y_i)

Annexe D

Les algorithmes de détection de renommage

G. Antoniol, M. Di Penta, and E. Merlo. “An automatic approach to identify class evolution discontinuities”. In Software Evolution, 2004.

T. Lavoie, F. Khomh, E. Merlo, and Ying Zou. “Inferring repository file structure modifications using nearest-neighbor clone detection”, In Reverse Engineering (WCRE), 2012.

Daniela Steidl, Benjamin Hummel, and Elmar Juergens, “Incremental origin analysis of source code files”. Conference on Mining Software Repositories, 2014.

Michael Godfrey and Qiang Tu. “Tracking structural evolution using origin analysis”. International Workshop on Principles of Software Evolution, IWPSE 2002.