Défis dans l'évolution collaborative d'un langage de preuve et de son écosystème

Soutenance de thèse

Théo Zimmermann

Hugo Herbelin (directeur) Yann Régis-Gianas (co-directeur)

Université de Paris, IRIF
Inria Paris, équipe-projet πr^2

Jeudi 12 décembre 2019

Coq : un assistant interactif à la preuve

- Un langage qui permet d'**énoncer des théorèmes** :

```
Theorem pow_add a n m  : a^n * a^m = a^(n+m).
```

Coq : un assistant interactif à la preuve

- Un langage qui permet d'**énoncer des théorèmes** :

```
Theorem pow_add a n m  : a^n * a^m = a^(n+m).
```

- Un langage qui permet de **construire** leurs preuves :

```
induction n.

(* Deux sous-buts :
   - Cas de base : a^0 * a^m = a^(0+m)
   - Cas récursif : a^(n.+1) * a^m = a^((n.+1)+m)
     sachant : a^n * a^m = a^(n+m)
*)
```

Coq : un assistant interactif à la preuve

- Un langage qui permet d'**énoncer des théorèmes** :

```
Theorem pow_add a n m  : a^n * a^m = a^(n+m).
```

- Un langage qui permet de **construire** leurs preuves :

```
induction n.

(* Deux sous-buts :
   - Cas de base : a^0 * a^m = a^(0+m)
   - Cas récursif : a^(n.+1) * a^m = a^((n.+1)+m)
     sachant : a^n * a^m = a^(n+m)
*)

all: simpl; lia.
```

Coq : un assistant interactif à la preuve

- Un langage qui permet d'**énoncer des théorèmes** :

```
Theorem pow_add a n m  : a^n * a^m = a^(n+m).
```

- Un langage qui permet de **construire** leurs preuves :

```
induction n.

(* Deux sous-buts :
   - Cas de base : a^0 * a^m = a^(0+m)
   - Cas récursif : a^(n.+1) * a^m = a^((n.+1)+m)
     sachant : a^n * a^m = a^(n+m)
*)

all: simpl; lia.
```

- Un **noyau qui vérifie** les preuves ainsi construites :

```
Qed.
```

Coq : applications et utilisateurs

Coq sert à :

- vérifier certaines preuves difficiles de **théorèmes** ;
- prouver que des **logiciels critiques** font ce qu'ils sont censés faire (systèmes embarqués, cryptographie, finance...) ;
- enseigner la logique et les preuves mathématiques.

Coq : applications et utilisateurs

Coq sert à :

- vérifier certaines preuves difficiles de **théorèmes** ;
- prouver que des **logiciels critiques** font ce qu'ils sont censés faire (systèmes embarqués, cryptographie, finance...) ;
- enseigner la logique et les preuves mathématiques.

Utilisé par :

- nombreux chercheurs dans (très) gros **projets académiques** ;
- petites équipes au sein de grands groupes (ex. Amazon) ;
- nombreuses start-ups de **blockchain**...

Coq : applications et utilisateurs

Coq sert à :

- vérifier certaines preuves difficiles de **théorèmes** ;
- prouver que des **logiciels critiques** font ce qu'ils sont censés faire (systèmes embarqués, cryptographie, finance...) ;
- enseigner la logique et les preuves mathématiques.

Utilisé par :

- nombreux chercheurs dans (très) gros **projets académiques** ;
- petites équipes au sein de grands groupes (ex. Amazon) ;
- nombreuses start-ups de **blockchain**...

Défi 1 : besoins nouveaux en termes d'accessibilité et de stabilité.

Coq : un logiciel complexe

- ∼295 000 lignes d'OCaml,
- dont un « petit » noyau de ∼33 000 lignes d'OCaml (+ ∼3000 lignes de C).
- ∼185 000 lignes de Coq dans la bibliothèque standard.
- Tactiques dépendent d'heuristiques parfois mal spécifiées.

Coq : un logiciel complexe

- ~295 000 lignes d'OCaml,
- dont un « petit » noyau de ~33 000 lignes d'OCaml (+ ~3000 lignes de C).
- ~185 000 lignes de Coq dans la bibliothèque standard.
- Tactiques dépendent d'heuristiques parfois mal spécifiées.

Défi 2 : haut niveau d'expertise requis pour contribuer à son développement...

... *mais* certains utilisateurs ont une telle expertise.

Coq : un vieux logiciel

- Créé en 1984 par Gérard Huet et Thierry Coquand, rapidement rejoints par Christine Paulin-Mohring.
- Historique des versions dans le dépôt actuel remonte à 1999.
- Nombreux chercheurs/thésards y ont contribué (*turnover*).
- Fonctionalités expérimentales inachevées / non-documentées, dette technique, code peu documenté, etc.

Coq : un vieux logiciel

- Créé en 1984 par Gérard Huet et Thierry Coquand, rapidement rejoints par Christine Paulin-Mohring.
- Historique des versions dans le dépôt actuel remonte à 1999.
- Nombreux chercheurs/thésards y ont contribué (*turnover*).
- Fonctionalités expérimentales inachevées / non-documentées, dette technique, code peu documenté, etc.

Défi 3 : quel compromis entre stabilité et évolution ?

Coq : un logiciel libre

Logiciel libre = librement utilisable, **distribuable**, **modifiable**.

Coq est :

- Distribué avec des **extensions** contribuées par des utilisateurs.
- Distribué sous **licence LGPL** 2.1 depuis 1999 (version 6.3.1).
- Développé entièrement sur **GitHub** depuis 2017.
- Maintenant : + de 150 contributeurs.

Coq : un logiciel libre

Logiciel libre = librement utilisable, **distribuable**, **modifiable**.

Coq est :

- Distribué avec des **extensions** contribuées par des utilisateurs.
- Distribué sous **licence LGPL** 2.1 depuis 1999 (version 6.3.1).
- Développé entièrement sur **GitHub** depuis 2017.
- Maintenant : + de 150 contributeurs.

Défi 4 : comment faciliter et évaluer les contributions des volontaires ?

Génie logiciel

Science des processus et des outils permettant d'augmenter la **productivité** des développeurs et la **qualité** des logiciels produits.

Génie logiciel

Science des processus et des outils permettant d'augmenter la **productivité** des développeurs et la **qualité** des logiciels produits.

L'expertise en génie logiciel (notamment en **maintenance et évolution** de logiciel) est de première importance pour un logiciel :

- complexe ;
- ancien ;
- dont dépendent beaucoup d'utilisateurs.

Génie logiciel

Science des processus et des outils permettant d'augmenter la **productivité** des développeurs et la **qualité** des logiciels produits.

L'expertise en génie logiciel (notamment en **maintenance et évolution** de logiciel) est de première importance pour un logiciel :

- complexe ;
- ancien ;
- dont dépendent beaucoup d'utilisateurs.

L'étude de la **collaboration ouverte** et des **écosystèmes libres** est un sous-domaine de première importance si l'on veut :

- impliquer au mieux les utilisateurs / volontaires.

Génie logiciel

Science des processus et des outils permettant d'augmenter la **productivité** des développeurs et la **qualité** des logiciels produits.

L'expertise en génie logiciel (notamment en **maintenance et évolution** de logiciel) est de première importance pour un logiciel :

- complexe ;
- ancien ;
- dont dépendent beaucoup d'utilisateurs.

L'étude de la **collaboration ouverte** et des **écosystèmes libres** est un sous-domaine de première importance si l'on veut :

- impliquer au mieux les utilisateurs / volontaires.

Le génie logiciel **empirique** a pour but de dépasser les croyances et d'établir des connaissances solides en génie logiciel, basées sur des **expériences** et de la **fouille de données**.

Recherche-action

La recherche-action a pour double but de **conduire un changement** et de **produire des connaissances** scientifiques.

Les étapes :

- *Identification* de problèmes concrets auxquels les développeurs / utilisateurs sont confrontés.

Recherche-action

La recherche-action a pour double but de **conduire un changement** et de **produire des connaissances** scientifiques.

Les étapes :

- *Identification* de problèmes concrets auxquels les développeurs / utilisateurs sont confrontés.
- *Proposition* d'une solution (en collaboration avec les acteurs), éventuellement basée sur la littérature, inspirée par d'autres projets, etc.

Recherche-action

La recherche-action a pour double but de **conduire un changement** et de **produire des connaissances** scientifiques.

Les étapes :

- *Identification* de problèmes concrets auxquels les développeurs / utilisateurs sont confrontés.
- *Proposition* d'une solution (en collaboration avec les acteurs), éventuellement basée sur la littérature, inspirée par d'autres projets, etc.
- *Application* de la solution (implémentation d'outil, mise en place de processus, rédaction de documentation).

Recherche-action

La recherche-action a pour double but de **conduire un changement** et de **produire des connaissances** scientifiques.

Les étapes :

- *Identification* de problèmes concrets auxquels les développeurs / utilisateurs sont confrontés.
- *Proposition* d'une solution (en collaboration avec les acteurs), éventuellement basée sur la littérature, inspirée par d'autres projets, etc.
- *Application* de la solution (implémentation d'outil, mise en place de processus, rédaction de documentation).
- *Validation* (écoute des retours, évaluation empirique).

Types de contributions académiques

- Identification de **nouvelles questions de recherche** en génie logiciel.
- Présentation de **modèles nouveaux** pour répondre à certaines questions.
- Outils réutilisables. Recherche reproductible.
- Application d'une méthodologie d'**analyse causale** issue de l'économétrie (peu connue en génie logiciel empirique).

git : un logiciel de gestion de versions

- Préserve l'**historique** des modifications.
- Permet de travailler **en parallèle** sur plusieurs changements indépendants (additions, correctifs, etc.).
- Permet de partager publiquement le code et son historique.
- Facilite la **collaboration**.

git : un logiciel de gestion de versions

- Préserve l'**historique** des modifications.
- Permet de travailler **en parallèle** sur plusieurs changements indépendants (additions, correctifs, etc.).
- Permet de partager publiquement le code et son historique.
- Facilite la **collaboration**.
- git a été créé en 2005 mais des logiciels similaires existent depuis plus de 30 ans.
- Utilisé dans le projet Coq depuis 2013 (et avant cela SVN, CVS...).

Pousser ou tirer des modifications ?

Pousser :

- permission accordée à un **nombre limité** de développeurs ;
- les autres peuvent envoyer des **patchs**.

Pousser ou tirer des modifications ?

Pousser :

- permission accordée à un **nombre limité** de développeurs ;
- les autres peuvent envoyer des **patchs**.

Tirer (modèle popularisé par GitHub, depuis 2008) :

- chacun pousse sur sa **copie personnelle** (*fork*) ;
- on ouvre une discussion lorsqu'un changement est prêt (*pull request*) ;
- les **mainteneurs intègrent** (tirent) les modifications.

Choix d'utiliser les *pull requests*

Facilite les contributions :

- Plus **pratique** pour les contributeurs extérieurs.
- Modèle devenu la **norme** pour contribuer aux logiciels libres.

Choix d'utiliser les *pull requests*

Facilite les contributions :

- Plus **pratique** pour les contributeurs extérieurs.
- Modèle devenu la **norme** pour contribuer aux logiciels libres.

Améliore la qualité :

- Encourage la **validation des modifications** par les pairs (*revue de code*).
- Facilite la transmission de connaissance sur le code.
- Compatible avec l'utilisation de **tests systématiques** pour tout changement (*intégration continue*).

Problème : présence de doc, tests, etc.

Pour tendre vers une documentation complète et des tests aussi complets que possible, je pousse les développeurs à inclure :

- de la **documentation**,
- des **tests**,
- une entrée dans le **journal des modifications**,

dans toutes les *pull requests* pour lesquelles c'est pertinent.

Problème : présence de doc, tests, etc.

Pour tendre vers une documentation complète et des tests aussi complets que possible, je pousse les développeurs à inclure :

- de la **documentation**,
- des **tests**,
- une entrée dans le **journal des modifications**,

dans toutes les *pull requests* pour lesquelles c'est pertinent.

Question : comment faire en sorte que ce soit le cas sans intervenir dans toutes les *pull requests* ?

Solution : l'utilisation d'un modèle

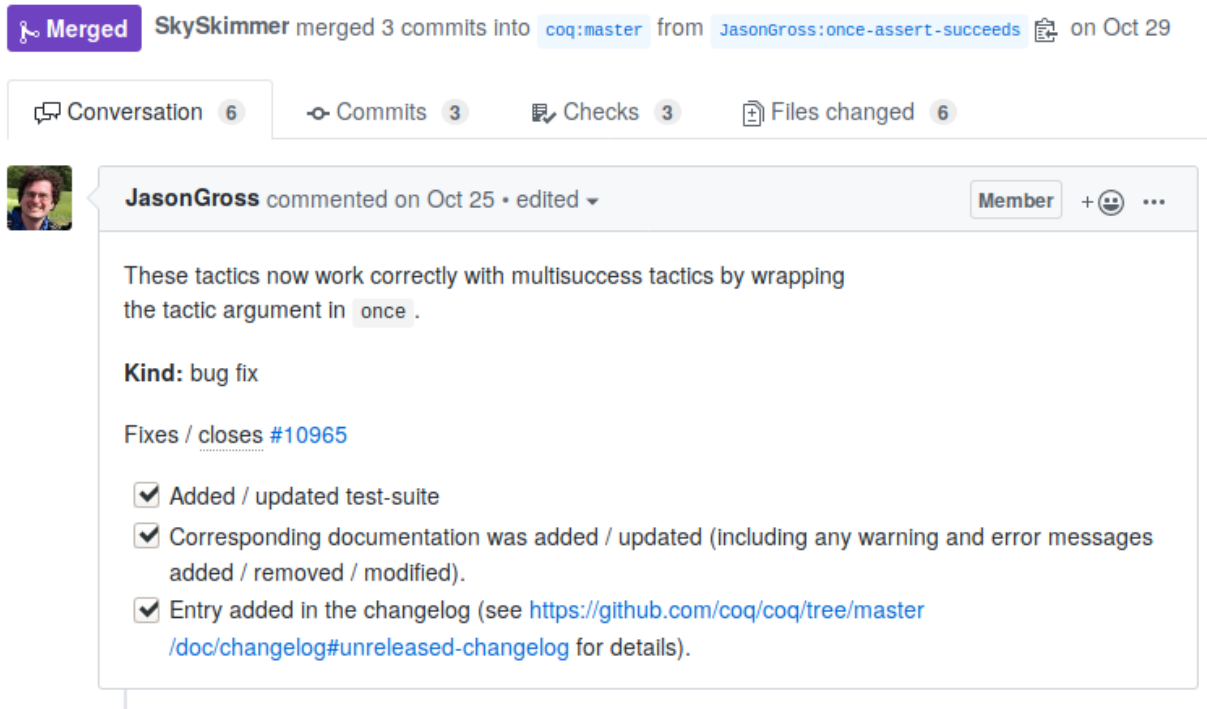


Figure 1: Un contributeur décrit sa *pull request* à l'aide du modèle fourni.

Le modèle contient des **cases à cocher** rappelant de mettre à jour la suite de tests, la documentation et le journal des modifications.

Validation empirique quantitative

On estime la **régression sur discontinuité** suivante :

$$\textit{Doc dans PR}_i = \gamma_0 + \gamma_1 \times \textit{Date relative}_i + \epsilon_i$$

(pour les i tels que $\textit{Date relative}_i < 0$)

$$\textit{Doc dans PR}_i = (\gamma_0 + \gamma_2) + (\gamma_1 + \gamma_3) \times \textit{Date relative}_i + \epsilon_i$$

(pour les i tels que $\textit{Date relative}_i \geq 0$)

Les coefficients γ_2 et γ_3 indiquent si l'introduction du modèle a eu un impact (changement au niveau de la discontinuité).

Résultats

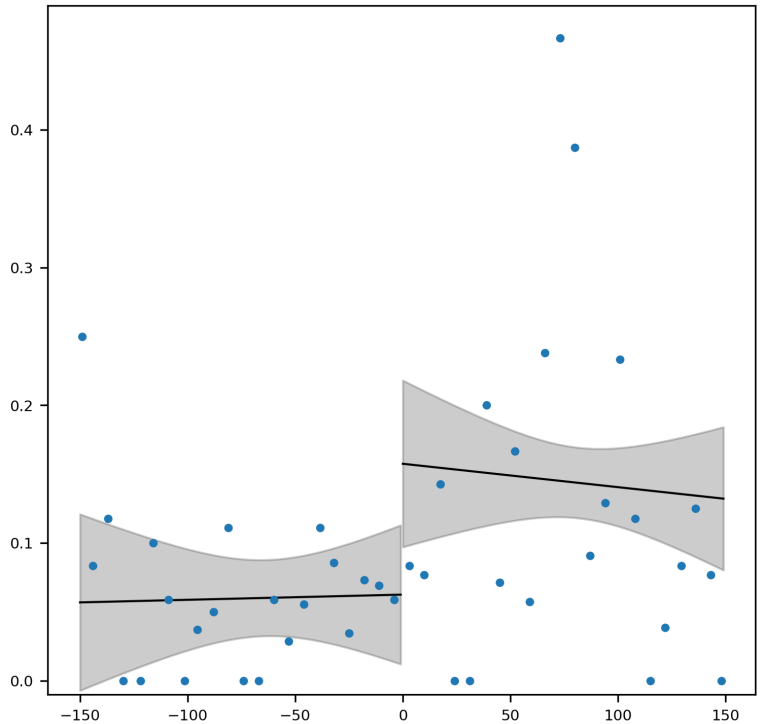


Figure 2: Proportion de *pull requests* (un point par semaine) incluant une mise à jour de la documentation (avec droites de régression et intervalles de confiance).

Le saut γ_2 (la proportion fait plus que doubler) est statistiquement significatif avec $p < 0.05$.

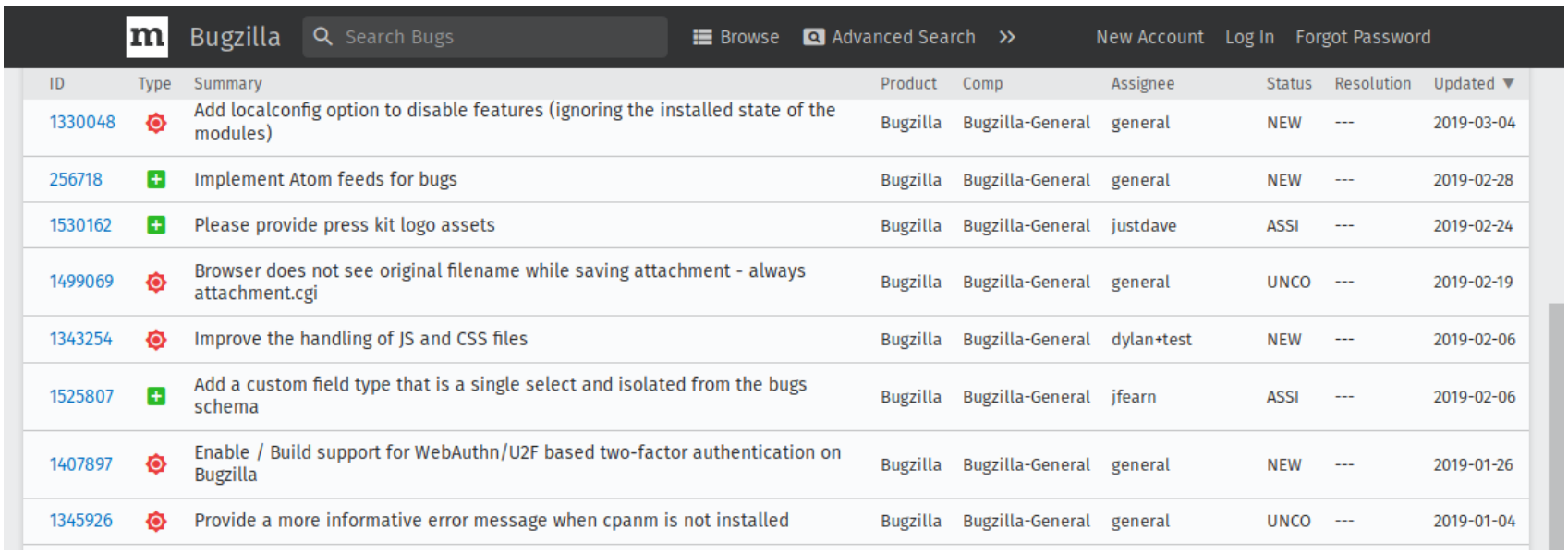
Résultats

- Augmentation importante et significative également sur la proportion de *pull requests* incluant des **tests**.
- Augmentation importante et significative également sur la proportion de *pull requests* incluant une entrée dans le **journal des modifications**.
- Très forte augmentation (statistiquement significative) sur la proportion de *pull requests* proposant des **additions** et incluant une entrée dans le **journal des modifications**.

Plateformes de gestion de tickets

Une plateforme de gestion de tickets :

- permet d'**enregistrer** la liste des bugs connus ;
- peut être utilisée directement par les utilisateurs pour rapporter des **bugs** ou émettre des **suggestions**.



ID	Type	Summary	Product	Comp	Assignee	Status	Resolution	Updated ▼
1330048		Add localconfig option to disable features (ignoring the installed state of the modules)	Bugzilla	Bugzilla-General	general	NEW	---	2019-03-04
256718		Implement Atom feeds for bugs	Bugzilla	Bugzilla-General	general	NEW	---	2019-02-28
1530162		Please provide press kit logo assets	Bugzilla	Bugzilla-General	justdave	ASSI	---	2019-02-24
1499069		Browser does not see original filename while saving attachment - always attachment.cgi	Bugzilla	Bugzilla-General	general	UNCO	---	2019-02-19
1343254		Improve the handling of JS and CSS files	Bugzilla	Bugzilla-General	dylan+test	NEW	---	2019-02-06
1525807		Add a custom field type that is a single select and isolated from the bugs schema	Bugzilla	Bugzilla-General	jfearn	ASSI	---	2019-02-06
1407897		Enable / Build support for WebAuthn/U2F based two-factor authentication on Bugzilla	Bugzilla	Bugzilla-General	general	NEW	---	2019-01-26
1345926		Provide a more informative error message when cpanm is not installed	Bugzilla	Bugzilla-General	general	UNCO	---	2019-01-04

Figure 3: Une capture d'écran des tickets portant sur Bugzilla sur la platforme Bugzilla de Mozilla.

Bugzilla : l'un des choix les plus populaires

Utilisé par :

- Mozilla (Firefox, Thunderbird, Bugzilla, etc.) ;
- Linux (noyau) ;
- GNOME (gestionnaire de bureau et collection de logiciels) ;
- Apache (serveur Apache, Ant, etc.) ;
- Eclipse (plateforme de développement et projets associés).

Nombreuses fonctionnalités avancées :

- **Champs** pour le produit, le composant, la version, le système, le type, la priorité, la sévérité, la cible...
- Gestion fine de la vie d'un ticket (nombreux **statuts**).
- Etc.

Problème : utilisation sous-optimale

- Changement de plateforme entre les tickets (Bugzilla) et les *pull requests* (GitHub).
- Pas de formattage, commentaires ne peuvent pas être édités.
- Besoin de créer un compte spécifique.

$\Longrightarrow$ Développeurs **créent peu de tickets**.

$\Longrightarrow$ Développeurs **réagissent peu aux tickets** ouverts.

Des problèmes d'**administration du serveur** servent de déclencheur :

$\rightarrow$ j'ai pris en charge la **migration des tickets** existants pour permettre un passage au système de tickets de GitHub.

Questions de recherche

Quel est l'impact du passage de Bugzilla à GitHub pour le projet :

- sur le **niveau** d'activité sur la plateforme de tickets (RQ1)
- sur la **qualité** de l'activité sur la plateforme de tickets (RQ2)
- sur l'**audience** de la plateforme de tickets (RQ3)

ICSME 2019 : TZ et Annalí Casanueva Artis, *Impact of switching bug trackers: a case study on a medium-sized open source project*.

Variables mesurées

Nombre de :

- tickets créés par jour,
- acteurs différents qui ouvrent des tickets par semaine,
- commentaires postés par jour,
- acteurs différents qui postent des commentaires par semaine.

Analyse hétérogène de l'impact sur les principaux dévelopeurs / les autres participants.

Les principaux développeurs sont definis comme ceux qui ont créé plus de 100 commits.

Validation empirique quantitative

On estime la **régression sur discontinuité** suivante :

$$\textit{Nombre de tickets}_{t<0} = \gamma_0 + \gamma_1 \times \textit{Date relative}_t + \epsilon_t$$

$$\textit{Nombre de tickets}_{t\geq 0} = (\gamma_0 + \gamma_2) + (\gamma_1 + \gamma_3) \times \textit{Date relative}_t + \epsilon_t$$

Les coefficients γ_2 et γ_3 indiquent si le changement de plateforme de tickets a eu un impact.

Création de tickets (par jour)

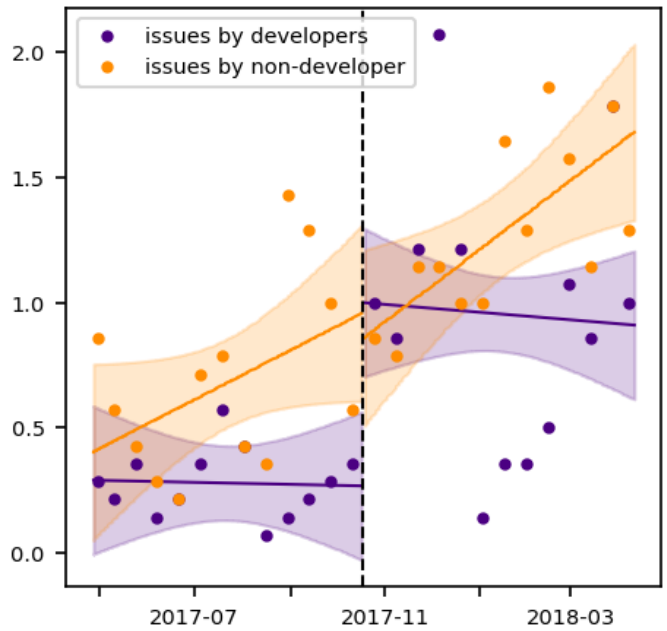


Figure 4: Nombre de tickets créés par jour avant et après le changement de plateforme (avec les droites de régression et les intervalles de confiance, les points correspondent à des valeurs moyennes sur des périodes de deux semaines).

Le saut γ_2 dans le nombre de tickets créés par jour par les **développeurs** est statistiquement significatif ($p < 0.01$).

Auteurs de commentaire (par semaine)

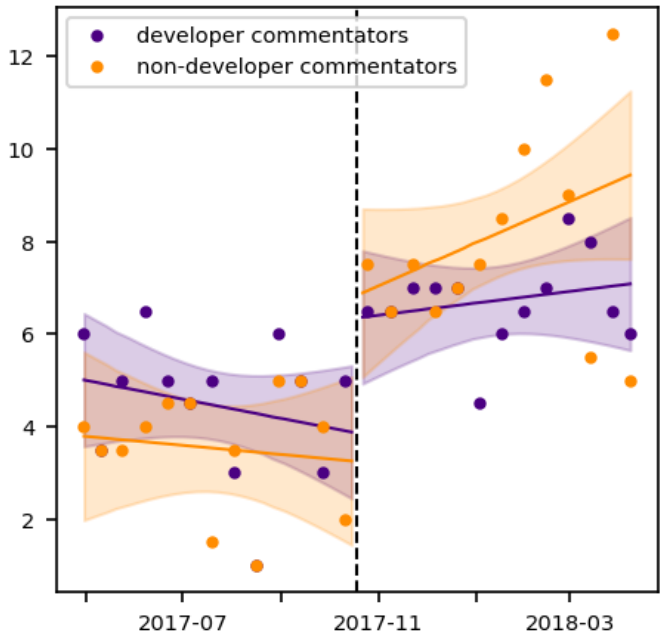


Figure 5: Nombre d'auteurs de commentaire par semaine avant et après le changement de plateforme (avec les droites de régression et les intervalles de confiance, les points correspondent à des valeurs moyennes sur des périodes de deux semaines).

Le saut γ_2 dans le nombre d'auteurs de commentaire par semaine est statistiquement significatif pour les **développeurs** ($p < 0.05$) et les **autres participants** ($p < 0.01$).

Autres résultats (quantitatifs)

- Augmentation significative du nombre de commentaires postés par les **développeurs** principaux.
- Augmentation significative du nombre de **développeurs** principaux distincts qui ouvrent des tickets chaque semaine.

Évaluation qualitative / interprétation

- 9 entretiens semi-structurés avec les principaux développeurs ;
- après l'étude quantitative ;
- questions sur les motivations et les risques, l'impact personnel, l'interprétation des résultats, etc.
- Cf. manuscrit.

Problème : un processus trop long

Version	Date	Durée depuis la précédente	Durée de la beta
8.2	2009-02	–	8 mois
8.3	2010-10	20 mois	8 mois
8.4	2012-08	22 mois	8 mois
8.5	2016-01	**41 mois**	**12 mois**

Trop de temps pour publier une nouvelle version:

$\Longrightarrow$ Trop de changements cassant la compatibilité.

$\Longrightarrow$ Décourageant pour les contributeurs.

Solution : des cycles plus courts

Version	Date	Durée depuis la précédente	Durée de la beta
8.2	2009-02	–	8 mois
8.3	2010-10	20 mois	8 mois
8.4	2012-08	22 mois	8 mois
8.5	2016-01	**41 mois**	**12 mois**
8.6	2016-12	11 mois	1 mois
8.7	2017-10	10 mois	1 mois
8.8	2018-04	6 mois	1 mois
8.9	2019-01	9 mois	2 mois
8.10	2019-10	9 mois	5 mois
8.11	2020-01	3 mois	1 mois

Correspond aux observations sur d'autres logiciels libres dans la thèse de Martin Michlmayr.

Solution : des cycles plus courts

Version	Date	Durée depuis la précédente	Durée de la beta
8.2	2009-02	–	8 mois
8.3	2010-10	20 mois	8 mois
8.4	2012-08	22 mois	8 mois
8.5	2016-01	**41 mois**	**12 mois**
8.6	2016-12	11 mois	1 mois
8.7	2017-10	10 mois	1 mois
8.8	2018-04	6 mois	1 mois
8.9	2019-01	9 mois	2 mois
8.10	2019-10	9 mois	5 mois
8.11	2020-01	3 mois	1 mois

Correspond aux observations sur d'autres logiciels libres dans la thèse de Martin Michlmayr.

Défi : besoin d'efficacité dans la production des nouvelles versions.

Détacher le processus de contribution de la production des nouvelles versions

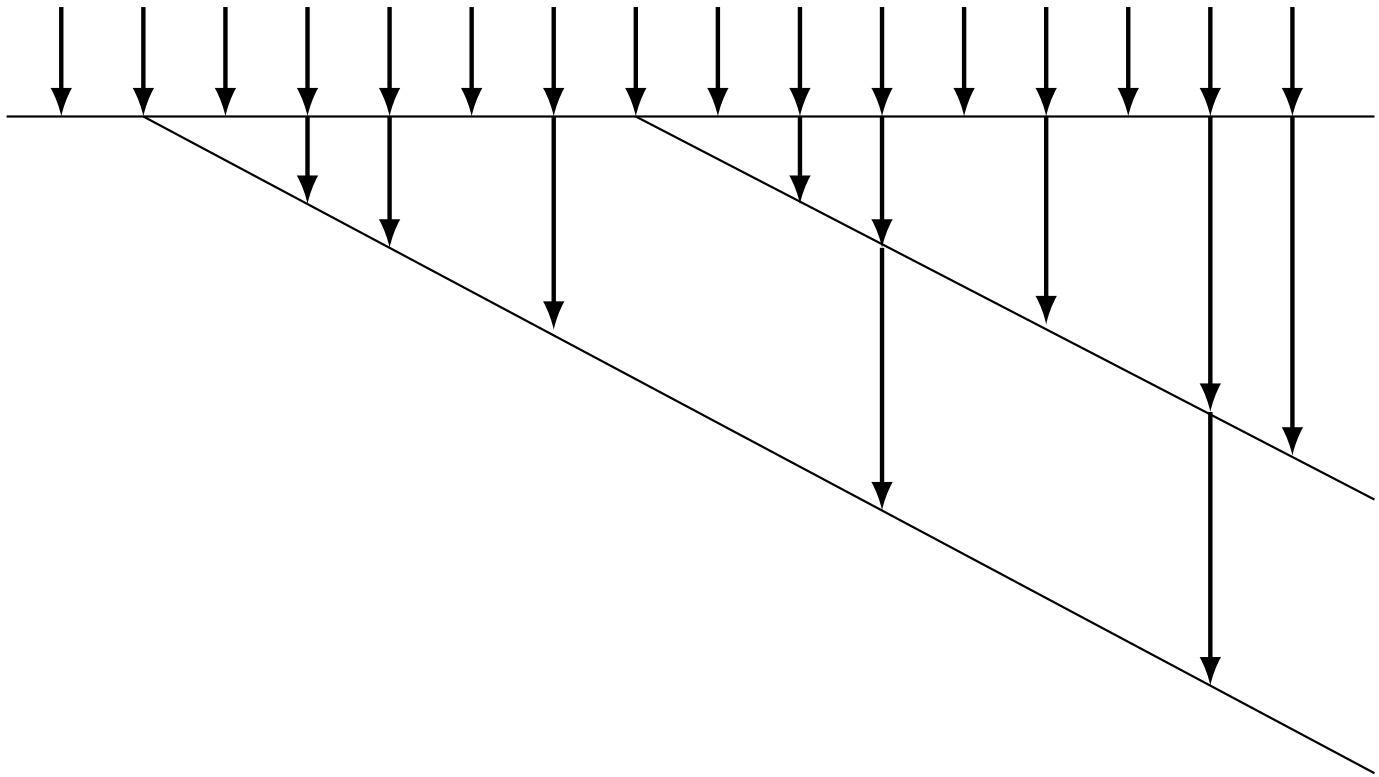


Figure 6: Flot des modifications : les *pull requests* sont intégrées dans la branche *master*, avant qu'une sélection soit appliquée sur les branches stables qui sont activement maintenues.

Avantages

- Plus simple pour les **contributeurs** : toutes les *pulls requests* visent la branche *master*.
- Plus simple pour les **intégrateurs** : valident seulement la qualité du changement proposé.
- Plus pratique pour les **utilisateurs** de la branche *master* : les correctifs arrivent plus vite.
- Plus simple pour le responsable de la publication de la nouvelle version (***release manager***) : contrôle absolu sur la branche stable.

Aussi : amélioration de la gestion du journal des modifications.
Cf. manuscrit.

Un tableau Kanban pour visualiser l'état

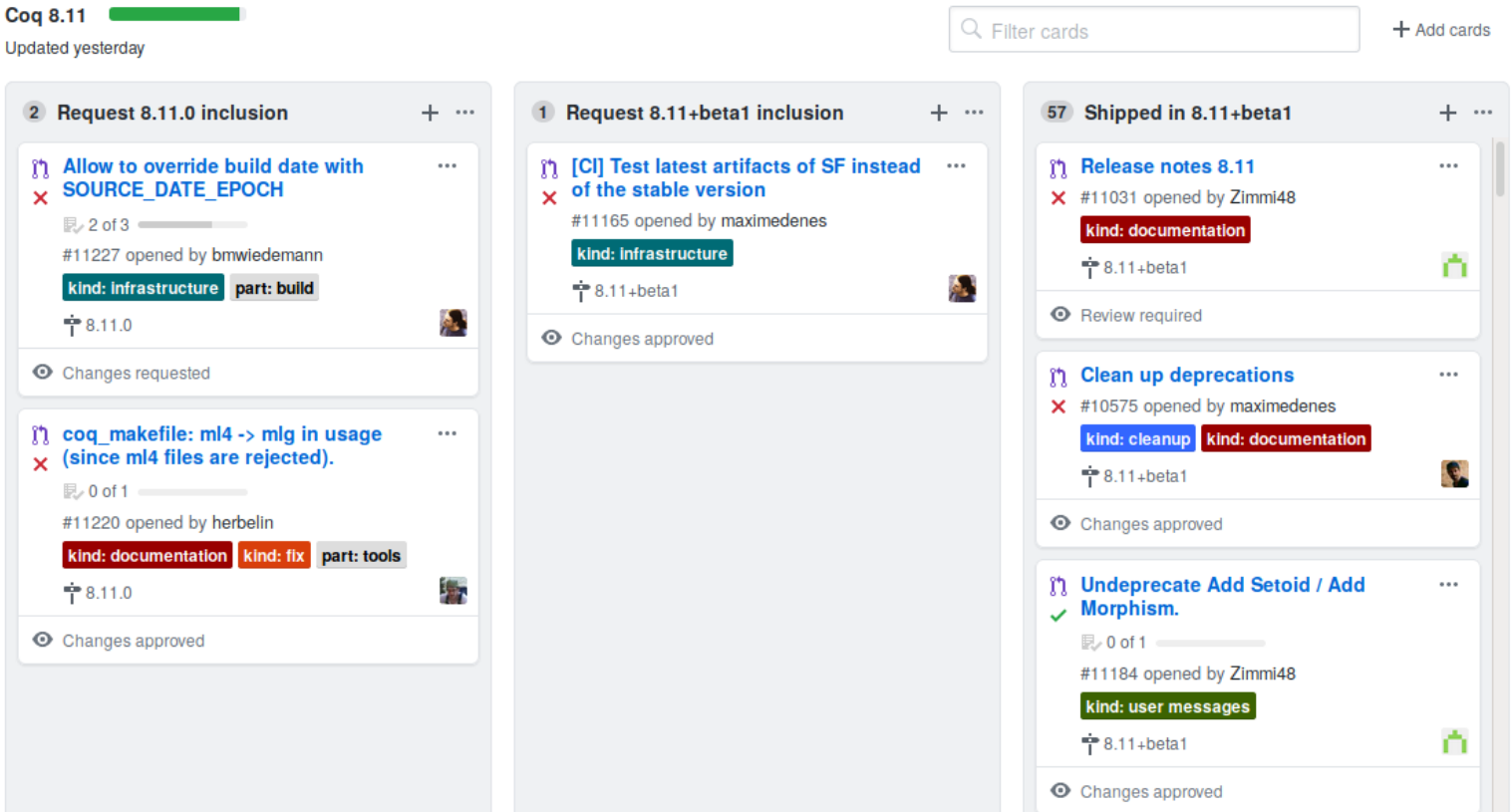


Figure 7: Ce tableau sert au responsable de la branche stable à gérer les *pull requests* devant être appliquées sur celle-ci.

Gestion automatique des colonnes

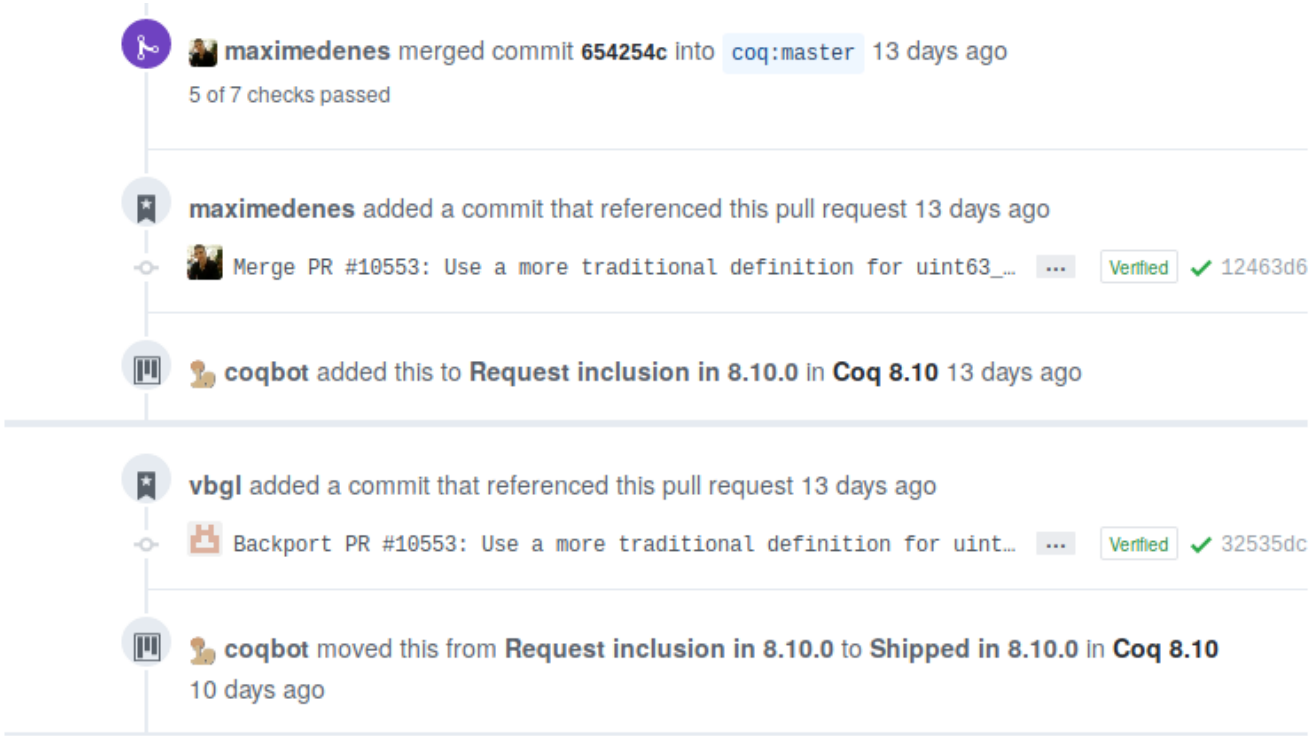


Figure 8: Le bot ajoute automatiquement les *pull requests* au tableau et les déplace entre les colonnes.

Paquets : des fondations partagées

- La réutilisation de code augmente la **productivité**, les **performances**, et **réduit les bugs**.
- Des fonctions, modules, extensions, bibliothèques… sont partagés sous forme de **paquets** via internet.
- Les paquets (et leurs auteurs / mainteneurs) forment des **écosystèmes** de logiciels libres.

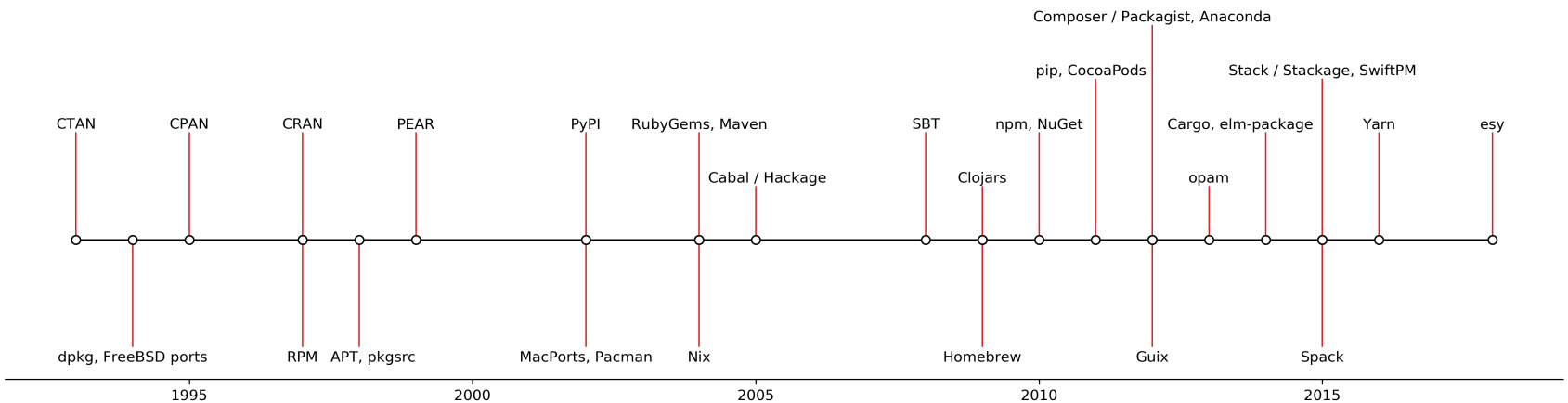


Figure 9: Dates de création des **gestionnaires de paquets** les plus connus. Les gestionnaires spécifiques à un langage sont en haut.

Paquets avec un seul mainteneur

Les gestionnaires de paquets ont rendu facile :

- la création et le partage d'une bibliothèque ;
- l'installation et l'utilisation d'une bibliothèque.

$\Longrightarrow$ Beaucoup de développeurs **partagent** des petites bibliothèques créées pour leurs propres besoins.

$\Longrightarrow$ Beaucoup de développeurs **ajoutent** des dépendances à leurs projets (sans vérifier qui les maintient).

Paquets avec un seul mainteneur

Les gestionnaires de paquets ont rendu facile :

- la création et le partage d'une bibliothèque ;
- l'installation et l'utilisation d'une bibliothèque.

$\Longrightarrow$ Beaucoup de développeurs **partagent** des petites bibliothèques créées pour leurs propres besoins.

$\Longrightarrow$ Beaucoup de développeurs **ajoutent** des dépendances à leurs projets (sans vérifier qui les maintient).

Mais paquet **libre** $\Longrightarrow$ pas de garanties.

Risque : paquet très utilisé dont le seul mainteneur devient **inactif**.

Estimation empirique : Concerne une proportion importante des paquets utilisés comme dépendances. Cf. manuscrit.

Les spécificités de l'écosystème de Coq

- Beaucoup de projets créés par des **thésards** pour leur thèse, ou par des chercheurs pour un **article**.
- Les preuves aussi ont **besoin de maintenance**.
- Même si personne ne dépend d'un théorème, il peut valoir la peine de le maintenir (**connaissance**, valeur intrinsèque).

Les spécificités de l'écosystème de Coq

- Beaucoup de projets créés par des **thésards** pour leur thèse, ou par des chercheurs pour un **article**.
- Les preuves aussi ont **besoin de maintenance**.
- Même si personne ne dépend d'un théorème, il peut valoir la peine de le maintenir (**connaissance**, valeur intrinsèque).
- Le précédent modèle de **distribution** (*contribs*) était aussi un modèle de maintenance, avec des **limites** :
  - flexibilité ;
  - passage à l'échelle ;
  - maintenance active par les auteurs (ex. Math-Comp, Flocq…).
- Le nouveau modèle de distribution (gestionnaire de paquets) ne propose **pas de modèle de maintenance**.

Fork d'un paquet abandonné

Fork (fourche) = copie maintenue de manière autonome.

- Rendu possible par les **licences libres**.
- Peut **redonner vie** à un projet.
- Mais le **risque** d'abandon est encore là si le *fork* n'a lui-même qu'un seul mainteneur.

Fork d'un paquet abandonné

Fork (fourche) = copie maintenue de manière autonome.

- Rendu possible par les **licences libres**.
- Peut **redonner vie** à un projet.
- Mais le **risque** d'abandon est encore là si le *fork* n'a lui-même qu'un seul mainteneur.

Solution : *fork* au sein d'une **organisation** communautaire avec **plusieurs administrateurs**. Permet de nommer un nouveau mainteneur si l'actuel disparait.

Exemples d'organisations communautaires

- **Elm-community** : fondée en nov. 2015, écosystème Elm.
- **Vox Pupuli** : fondée en sept. 2014, écosystème Puppet.
- **Sous Chefs** : fondée en mai 2015, écosystème Chef.
- **Dlang-community** : fondée en déc. 2016, écosystème D.

Exemples d'organisations communautaires

- **Elm-community** : fondée en nov. 2015, écosystème Elm.
- **Vox Pupuli** : fondée en sept. 2014, écosystème Puppet.
- **Sous Chefs** : fondée en mai 2015, écosystème Chef.
- **Dlang-community** : fondée en déc. 2016, écosystème D.
- **Coq-community** : fondée en juil. 2018, influencée par Elm-community.
- **OCaml-community** : fondée en août 2018, influencée par Coq-community et Elm-community.

Figure 10: Le logo de coq-community, créé par Aras Atasaygin du projet Open Logos (http://openlogos.org/)

- 25 projets… dont :
  - 14 anciennes *contribs* ;
  - 3 nouveaux projets ;
  - 1 *fork* (le reste des projets ayant étant transférés).
- 19 mainteneurs principaux.

Récapitulatif des contributions

	Question	Solution	Validation
Labels (tickets et *pull requests*)	x		
Modèles (*pull requests*)	x	x	x
Dépendances inverses (*CI*)	x	x	
Plus d'intégrateurs (*pull requests*)	x	x	
Système de tickets	x	x	x
Visualisation des *backports*	x	x	
Gestion des notes de version	x	x	
Design gestionnaire de paquets	x		
Maintenance des paquets	x	x	

Validation : prend du **temps**, pas toujours les conditions d'une analyse **causale**.

Perspectives

- Poursuivre une approche **au cœur d'un projet libre** ayant pour double but d'aider au succès du projet et de produire des connaissances scientifiques nouvelles.
  - Par exemple, comment **gérer efficacement** et de manière collaborative **un grand nombre de tickets** dans un projet libre, **reposant sur des volontaires**.
  - *Mais aussi* : intégration des *pull requests*, fiabilité des tests, distribution des paquets, documentation, etc.
- Continuer à appliquer des méthodes nouvelles en génie logiciel empirique, par exemple des **méthodes d'économétrie** pour l'analyse causale de **quasi-expériences**.