Projet Individuel MGL7460-90

Rapport intermédiaire

Réalisé par

Gary Rivera

(RIVG23057109)

Table de matières

Introduction

J'ai choisi d'analyser la dimension de l'équipe de développement du projet angular. Ce dernier est un cadriciel libre disponible dans GitHub. La période analysée est du 14 septembre 2014 au 17 janvier 2020. Dans l'analyse, je ne remets pas en question l'importance ou la pertinence du code fusionné dans la branche master. Je prends les informations du message du *commits* et du ticket créé dans GitHub.

Aussi, dans certaines extractions, j'ai corrigé les noms ou les adresses courriels des contributeurs car dans certains cas, il était évident que c'est la même personne qui avait changé de courriel.

Méthodologie d'analyse

Je procède de la façon suivante; durant l'analyse de la dimension, je fais des extractions du log du projet angular de la branche master. Les extractions sont transposées en question. Elles sont travaillées avec l'aide des scripts PowerShell. Par la suite, j'importe les données dans Excel pour être avalisées grâce aux tableaux croisés dynamiques.

Pour chaque réponse aux questions, je fais référence à l'extraction utilisée pour y répondre. Ainsi, tous les requêtes et les fichiers Excel d'analyse se trouvent dans un répertoire identifié question/réponse.

De plus, pour éviter de travailler en vain, j'établi des paramètres pour réduire le nombre de variable. Au début de chaque réponse, on trouve la définition des paramètres qui explique son utilité.

Pour chaque question il y a un répertoire avec les scripts PowerShell et les fichiers Excel.

Choix d'outils d'analyse

Pour l'extraction des informations dans le log, j'utilise les commandes git pour interroger, mais aussi il y a de l'information exploitable dans un format présentable sur le site web:

https://github.com/angular/angular/graphs/contributors

Ensuite, toutes les données seront analysées et compilées grâce aux logiciels Excel et code-maat. Ce dernier est utilisé et mentionné dans le livre *Code as a Crime Scene*. Il permet de résumer certaines données.

Dimension prise en charge par l'analyse

La dimension abordée dans ce projet est "l'Équipe de développement". Je vais répondre aux questions sur l'équipe des contributeurs qui a participé à la période analysée. L'analyse est concentrée davantage sur les contributeurs qui ont réalisé un nombre de *commits* assez élevé durant la période.

Voici les questions que j'essaie de répondre:

1-Qui sont les développeurs principaux du projet ?

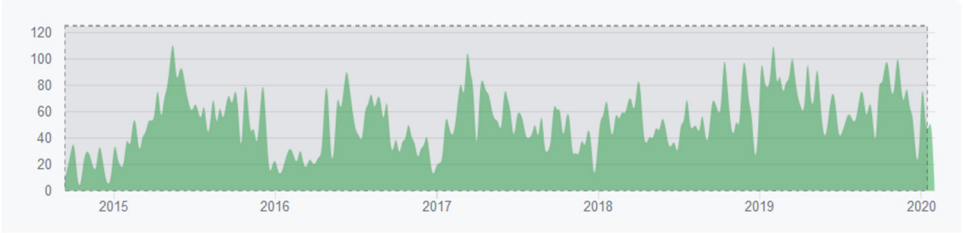
Paramètres: j'analyse les développeurs qui ont réalisé dix *commits* et plus par année ainsi que les contributeurs de 500 *commits* et plus durant la période.

Les développeurs principaux selon ma recherche sont : Peter Bacon Darwin, George Kalpakas, Victor Savkin, Victor Berchet, Igor Minar, Tobias Bosch, MiÅ¡ko Hevery, Alex Eagle, Alex Rickabaugh et Paul Gschwendtner.

Le tableau suivant contient le nombre de *commits* réalisé par les développeurs à ce jour.

	À partir du git log	**À partir du site github**
Peter Bacon Darwin	1240	1255
George Kalpakas	1102	1118
Victor Savkin	952	952
Victor Berchet	916	916
Igor Minar	896	912
Tobias Bosch	789	788
MiÅ¡ko Hevery	615	631
Alex Eagle	559	559
Alex Rickabaugh	531	542
Paul Gschwendtner	535	536

Si je prends les développeurs qui ont fait plus de dix *commits*, j'ai une moyenne de 153 *commits*. Alors mon top 10 sont de loin les plus productifs. Ces *commits* sont répartis durant la période analysée.



https://github.com/angular/angular/graphs/contributors?from=2014-09-14&to=2020-01-17&type=c

Le tableau précédent montre les *commits* des contributeurs dans la branche master. La vraie question est si les développeurs principaux ont été toujours les mêmes. La réponse est, à la logique d'un bon informaticien; ça dépend de la façon à laquelle nous faisons parler les *commits*. Si l'on réparti la période analysée par année, on remarquera qu'entre 2014 et 2016 Peter Bacon n'était pas le premier dans les tops 10.

On observe qu'au début Peter était au bas du peloton. Mais avec le temps il a augmenté sa contribution. Un autre exemple est George Kalpakas, il n'était pas de la première moitié du projet et il est arrivé au top de la liste à ce jour.

Tableau de *commits* entre 2014 et 2016 :

Contributeur	À partir du git log
Victor Savkin	932
Victor Berchet	650
Tobias Bosch	577
Igor Minar	406
Misko Hevery	231
Tim Blasi	224
Alex Eagle	221
Pawel Kozlowski	207
Marc Laval	185
Peter Bacon Darwin	169

Les développeurs principaux ont tous en commun un grand nombre de *commits* sur la branche master, des propositions *et des revues* de *pull request* et des ouvertures de problèmes. Ils participent activement au projet[1].

En dernier, ce petit ensemble de développeurs ont aussi le plus grand nombre d'ajout et de suppression de lignes de code dans le projet durant la période analysée. Voir le tableau suivant:

Nom	Nombre de lignes affectées
Peter Bacon Darwin	473,293 ++ 260,197 --
Victor_Savkin	165,431 ++ 117,393 --
Victor Berchet	160,889 ++ 182,997 --
Igor Minar	576,017 ++ 529,026 --
Tobias Bosch	285,203 ++ 245,331 --
Misko Hevery	158,030 ++ 153,748 --
Alex_Eagle	403,091 ++ 261,394 --
Pawel Kozlowski	38,858 ++ 26,165 --
Marc Laval	37,792 ++ 16,413 --

[1] https://github.com/angular/angular/graphs/contributors?from=2014-06-14&to=2015-12-31&type=c

2-L'équipe de développement est-elle stable ?

Oui, l'équipe est stable, malgré une baisse en 2017, l'équipe n'a pas cessé d'augmenter. Il y a eu un phénomène particulier, les premiers développeurs ont eu une baisse de productivité et le deuxième groupe de développeurs, qui était moins productif au début de la période, sont passés les premiers du peloton à la fin de la période.

La première analyse est depuis le début de la période 2014 et 2015. Alors, il ne reste que quatre personnes: Misko Hevery, Marc Laval, Rado Kirov et Jeremy Elbourn. Seulement deux parmi les quatre apparaissent dans le top dix des développeurs.

En revanche, à partir de 2017, les choses ont changé. On a vingt-six développeurs qui forment le cœur de l'équipe. Ils sont : Peter Bacon Darwin , Igor Minar, Matias NiemelÃ¤, George Kalpakas, Alex Rickabaugh, Stefanie Fluin, Alex Eagle, Olivier Combe, Jason Aden, Misko Hevery, Marc Laval, Kara Erickson, MisÌŒko Hevery, Kapunahele Wong,Pawel Kozlowski, Vikram Subramanian, Hans Larsen, Filipe Silva, Trotyl Yu, vikerman, Martin Probst, Fabian Wiles, JiaLi.Passion, Kara, Rob Wormald et Stephen Fluin.

On peut dire que l'équipe est stable depuis les trois dernières années. Une chose intéressante à remarquer est que le nombre d'anomalies a diminué durant la période de transition de l'équipe. Nous pourrions croire qu'un changement dans l'équipe apporterait des anomalies mais ce ne fut pas le cas.

3-Comment est répartie la paternité du code source dans l'équipe ?

Dans cette question, j'essaie de valider si le code a changé beaucoup depuis sa création durant la vie du projet.

Alors, sur 4871 *commits* analysés, je trouve une moyenne 29% et une médiane de 25% de changement tout type de fichiers confondu. Je me suis attardé sur les modules et les packages car ces derniers possèdent une grande quantité de codes exécutables.

Mille neuf cent quarante-quatre (1944), parmi toute cet ensemble de fichier plus que 29% ont été changé et ces changements ont eu lieu dans une période de trente-neuf jours (39) en moyenne. J'ai ajouté la notion de jour entre la création et les *commits* subséquents. Cette donnée peut nous informer si le changement a eu lieu rapidement après la création du code ou non. Il y a du code en haut de 80% du taux de changement, mais à l'intérieur de deux mois. On peut dire qu'il y a eu une correction importante dans le code.

Pour le reste des changements, ceux entre 30% et 79% du code changé ont en moyenne 46% de changement du code dans ce palier et, nous pouvons reculer presque deux ans entre la création et les *commits.* Nous pouvons affirmer que la paternité de cette portion du code est très touchée car elle a subi beaucoup de changement.

Voici, les paliers de variations depuis le premier *commit* en 2014 :

Palier	Commit	% Changement
1	2935	0-29
2	1920	30-80
3	17	80-96

4-Comment les développeurs communiquent entre eux ?

Le projet Angular possède un système de tickets assez efficace. Il est enrichi de 138 étiquettes (au moment de l'analyse). Ces dernières permettent d'identifier et qualifier un ticket (github.com/angular/angular). Les types possibles de tickets lors d'une ouverture sont:

- ***build**: Changes that affect the build system or external dependencies (example scopes: gulp, broccoli, npm)*
- ***ci**: Changes to our CI configuration files and scripts (example scopes: Circle, BrowserStack, SauceLabs)*
- ***docs**: Documentation only changes*
- ***feat**: A new feature*
- ***fix**: A bug fix*
- ***perf**: A code change that improve performance*
- ***refactor**: A code change that neither fixes a bug nor adds a feature*
- ***style**: Changes that do not affect the meaning of the code (white-space, formatting, missing semi-colons, etc)*
- ***test**: Adding missing tests or correcting existing tests*

Après avoir déterminé le type de ticket, nous pouvons ajouter des attributs comme la sévérité, le *Pull Request*, le risque, la grandeur de l'effort, le composant affecté et son état. Il a y des étiquettes qui s'ajoutent régulièrement dans le projet.

Il y a également une politique de message lors d'un *commit*. Ceci permet d'uniformiser les messages et faciliter la compilation de l'information. Il faut respecter une nomenclature précise pour identifier le message du *commit*. Voir Commit-Message-Guidelines

Le système de tickets compte sur leur système de clavardage, des commentaires et des courriels. Cet ensemble d'outils aide la communication entre les développeurs.

5-Il n'y a pas de sous-équipes pour corriger des problèmes (tout le monde fait tout)

Afin de répondre à cette question, j'ai choisi de me concentrer sur les catégories de *commits* qui ont plus de 1000 occurrences dans la période analysée. Ces catégories sont : build, chore, doc feat, fix et refactor. Le système des tickets d'Angular, nous a permis de retrouver les tickets par catégorie rapidement.

Selon les données récupérées des *commits*, au-delà de 50 *commits* par contributeur. Je remarque qu'un contributeur a participé à presque toutes les catégories mentionnées précédemment.

En dessous de 30 *commits* par contributeur, on trouve une participation dans toutes les catégories de ticket mais elles sont évidemment moindres. Ici on n'essaie pas d'évaluer l'importance ou la pertinence de l'intervention du contributeur.

En générale, on peut dire qu'il y a une variété d'interventions par les contributeurs. Il est certain qu'il y a une équipe centrale qui est en mesure de tout faire dans le projet mais il n'existe pas une équipe dédiée à la maintenance. Même ceux qui ont moins des *commits*, ont contribué à plusieurs catégories de tickets dans l'ensemble de l'application.

6- Le changement des contributeurs dans l'équipe a-t-il un impact sur l'augmentation des anomalies?

Voici les nombres de *commits* pour les: Feat, Fix et test.

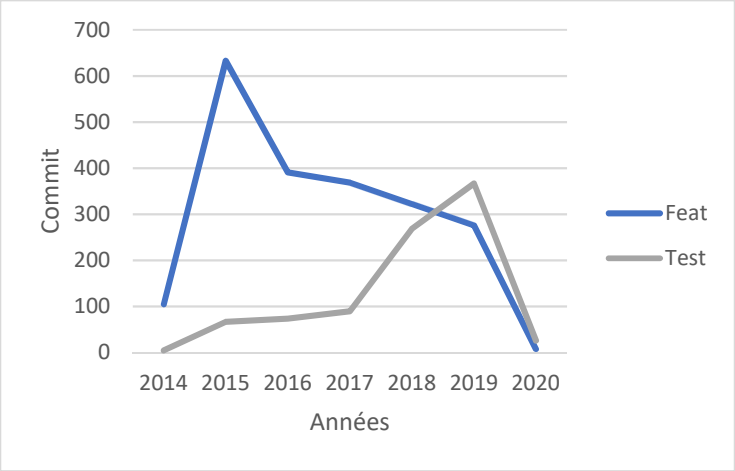
Année	**2014**	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	Total
Feat	105	633	391	369	322	276	8	**2104**
Fix	50	757	839	819	703	986	36	**4190**
Test	5	67	74	90	269	367	26	**898**
Total	**160**	**1457**	**1304**	**1278**	**1294**	**1629**	**70**	**7192**

Je dirais que les anomalies sont restées stables malgré le changement de garde dans l'équipe. J'ai remarqué que durant la période 2017 et 2018, les contributeurs plus actifs ont réduit leur participation pour laisser la place autres contributeurs.

Dans le tableau, je me suis permit d'ajouter les *commits* liés aux tests et fonctionnalités. Il est facile de remarquer que les corrections des anomalies sont presque 200% de plus que les fonctionnalités ajoutées. Sans trop abuser de l'interprétation des chiffres, on pourrait dire que l'équipe passe plus de temps à corriger des anomalies qu'à ajouter des fonctionnalités.

7- Est-ce-que l'équipe est axée sur les tests?

Seulement 36 contributeurs sur 132 ont ajouté ou corrigé des tests durant la période analysée. Ceci veut dire que 27% de l'équipe contribue aux tests.



De plus, 2019 est la seule année que le nombre de tests a été supérieur à l'ajout des nouvelles fonctionnalités. Le ratio des tests est très bas dans cette cadriciel qui est très utilisé par l'industrie.

Bibliographie

Adam Tornhill, Your Code as a Crime Scene: Use Forensic Techniques to Arrest Defects, Bottlenecks, and Bad Design in Your Programs. Pragmatic Bookshelf; 1 edition (April 9 2015) 220 pages

Annexe

Liste des portés qu'on peut travailler sur le projet Angular

- animations
- common
- compiler
- compiler-cli
- core
- elements
- forms
- http
- language-service
- platform-browser
- platform-browser-dynamic
- platform-server
- platform-webworker
- platform-webworker-dynamic
- router
- service-worker
- upgrade
- zone.js