Université du Québec à Montréal

MGL7460-90

Réalisation et Maintenance de Logiciels

Projet Individuel

Mémoire de synthèse sur la maintenabilité d'un logiciel

Irma Cristina Cruz

Hiver 2020

1 INTRODUCTION

Dans ce rapport, une analyse du dépôt du code source du logiciel « React » est présenté ainsi que les résultats d'investigation sur le développement du logiciel au long de son cycle de vie. Ceci est dans le but de montrer l'état du logiciel vis-à-vis à sa maintenabilité.

L'analyse effectué couvre les dimensions qui parlent de « Équipe de développement », la Dimension de « Code Source » et le « Déploiement et Livraison ».

Du côté de l'Équipe de Développement**,** on répond aux questions sur les développeurs qui contribuent au projet ainsi que leur implication à partir de l'investigation et les statistiques obtenus au moyen des outils suivantes : commandes « Git », l'outil GitHammer et l'outil « Code-Maat » (7Annexe Outils). Des questions concernant les « commits » que l'équipe cœur et les principaux développeurs réalisent ainsi que la détermination de la paternité de code-source par l'équipe sont répondus.

Pour la Dimension de Code source, nous présentons la liste de outils et dépendances du projet et nous explorons les différents aspects qui nous permettront déterminer l'impact sur la maintenabilité. Pour la Dimension Déploiement et Livraison nous expliquons la procédure que l'équipe React a mis en place et de quelle manière ceci aide à la maintenabilité du produit. Finalement une conclusion est donnée qui engloba les 3 dimensions explorés et qui nous donne un point de vue sur la maintenance du logiciel.

1.1 REACT

React est un libraire JavaScript pour créer des interfaces utilisateur. React est né au sein de Facebook et leur créateur est Jordan Walke. Il a été mise en mode « open-source » en mai 2013 et depuis un équipe cœur et la communauté font évoluer le produit (Facebook, 2020), (Wikipedia, 2020).

Le dépôt de code source est trouvé à l'adresse suivante : https://github.com/facebook/react

1.2 METHODOLOGIE

Suivant les recommandations principalement du livre « Your code as à crime Scene » (Tornhill, 2015), à partir du dépôt de code, on obtiendrait les données sur l'historique du code source. Par la suite les commandes correspondantes seront appliquées pour répondre aux questions dont des données statistiques sont nécessaires et qui s'appliquent au produit analysé. Pour certains questions, l'utilisation des formules et graphiques Excel sera utilisé, ainsi que de requêtes SQL.

Selon le type de question, les données peuvent être décortiquées en sous-ensemble pour comparer différents périodes dans l'historique du développement. Certaines réponses seront complémentées avec les informations qui ont été recherché pour se familiariser sur le produite et mieux interpréter les résultats.

Deux outils seront utilisés à fin d'obtenir les données statistiques du dépôt de code sources, Code Maat et GitHammer.

Les fichiers de commandes, des données et d'analyse utilisé dans ce travail se trouvent dans le dépôt GIT public https://github.com/CrisCruzC/MGL7460-ProjetIndividuel/.

1.3 UTILISATION DE CODE MAAT

Code Maat est l'outil présenté dans le livre « Your code as à crime Scene » (Tornhill, 2015). Nous allons utiliser la version qui se trouve dans son dépôt du Code. Cet outil nous permettra d'utiliser certaines commandes recommandées pour les analyses à réaliser selon le livre. La section d'Annexes des Outils 7.1 Code Maat.

Le dépôt Git de Code Maat se trouve à l'adresse https://github.com/adamtornhill/code-maat. La page README du dépôt explique comment exécuter l'outil soit par « leiningen» ou à partir d'un fichier « standalone » généré.

Après avoir obtenu, le dépôt du code avec la commande « clone » de Git, la commande « lein uberjar » a été exécuté pour générer un fichier jar. Le fichier « CommandesOutilCodeMaat.txt » avec les commandes GIT et les commandes Code Maat utilisés se trouvent dans le dépôt de code source de ce Projet sur le dossier Data.

Pour pouvoir exécuter l'analyse, il faut premièrement, générer un fichier de log du dépôt de code source. Ce fichier peut être généré pour le dépôt au complet ou pour un période de temps selon les besoins. À partir du dépôt local du logicielle à analyser la commande suivante a été exécuté :

```
git log --numstat --date=short --pretty=format:'--%h--%ad--%aN' >
maat_evo_react_complet.log
```

Le fichier maat_evo_react_complet.log a été généré pour être utilisé pour les différentes statistiques générés pour l'analyse.

1.4 UTILISATION DE GITHAMMER

Git Hammer (Kangasharju, 2020) est un outil qui permet de sortir des statistiques à partir d'un dépôt de code source Git. L'outil utilise une base de données SQLLite pour sauvegarder les données extraites du dépôt. Cette base de donnes contiendra les statistiques de contributeurs et « commits » par contributeur. Les informations sur l'installation et les commandes à exécuter sont détaillés dans la section 7.2 GitHammer.

2 DIMENSION ÉQUIPE DE DÉVELOPPEMENT

L'auteur d'origine de React est « Jordan Walke », mais il n'est plus un contributeur actif du projet. Le dernier commit que l'on trouve date du 29 mars de 2014.

Présentement, les contributeurs de React sont l'équipe cœur, normalement composé des ressources de Facebook et la communauté qui contribue à travers le dépôt public sur GitHub.

À l'aide de l'outil « Code Maat » nous avons généré le fichier de log « `maat_evo_react_complet.log` » contenant l'historique du dépôt Git (Voir Annexe 7.1 Code Maat). Sur ce fichier nous exécutons la commande Code Maat pour générer un rapport sommaire :

```
java -jar ~/ProjectsGit/code-maat/target/code-maat-1.1-SNAPSHOT-
standalone.jar -l maat_evo_react_complet.log -c git2 -a summary >
maat-summary-react.csv
```

Selon le Rapport Sommaire obtenu (`maat-summary-react.csv`), 1488 contributeurs ont participé dans le projet depuis son lancement en 2013 jusqu'à la date de notre analyse du 28 février 2020. Au 28 février 2020, 10 contributeurs font partie de l'équipe cœur. Il y a 31 contributeurs mentionnés dans le site de React et si on prend comme hypothèse qu'ils ont fait partie de l'équipe coeur au moment donné, nous pouvons obtenir les données suivantes pour avoir une estimation du nombre des contributeurs externes[1].

Catégorie	Nb contributeurs	% Total Commits
Équipe Cœur	8	36.225
Autres contributeurs	22	36
Contributeurs Externes	1459	27.775

2.1 ÉQUIPE CŒUR DU PROJET

Selon la page web de React (Facebook, Équipe Coeur React, 2020) l'équipe cœur du projet est une équipe composée des 10 développeurs qui travaille temps complet chez Facebook. Cette équipe lead le développement et travaille principalement sur les API Core de React.

Il y a eu des membres de l'équipe cœur dans le passé qui ne travaillait pas nécessairement chez Facebook à ce moment mais qui avait été déjà contributeurs externes selon l'article (TheDevCouple, 2017). De plus, il pourrait ne pas être à 100% un critère pour faire partie de l'équipe cœur d'avoir été un contributeur. Par exemple, l'actuel manager de l'équipe Yuzhi Zheng, ne semble avoir aucune contribution (selon son usager GitHub @yuzhi).

Nous verrons plus l'implication de l'équipe cœur dans l'analyse de la stabilité dans la Section 2.2.

2.2 STABILITÉ DE L'ÉQUIPE DE DÉVELOPPEMENT

Pour analyser la stabilité de l'équipe, nous avons déterminé diffèrent périodes de la vie du produit basé sur les Releases importantes. Le « Tableau 1 Périodes d'analyse » montre les périodes considérés pour analyser la stabilité de l'équipe.

Tableau 1 Périodes d'analyse

	Début	Fin	Milestone
Période 1	2013-05-28	2014-11-21	Version 0.12
Période 2	2014-11-22	2016-04-07	Version 15.0
Période 3	2016-04-08	2017-09-26	Version 16.0
Période 4	2017-09-27	2020-02-08	Date d'analyse

[1] Ces données sont obtenues à partir de la base de données GitHammer une petite différence des 2 contributeurs entre ces données et ceux de Maat. Par exemple, un contributeur est deux fois dans la base de données comme Dan et Dan Abramov. Ceci a été corrigé dans le fichier source.

À partir des données qui se trouvent dans la base de données git-hammer.sqllite, nous avons extrait les commits par contributeur de ces différents périodes. Le fichier « AnalyseReactFichierDeTravail.xlsx » qui se trouvent dans le dossier Data du dépôt Git du projet, contient les résultats obtenus.

Les graphiques suivants montrent les contributeurs qui dans la période visée ont été entre les premièrs 20 selon le nombre de Commits effectués.

Figure 1 Contributeurs avec plus de 10 « Commits » Période 1

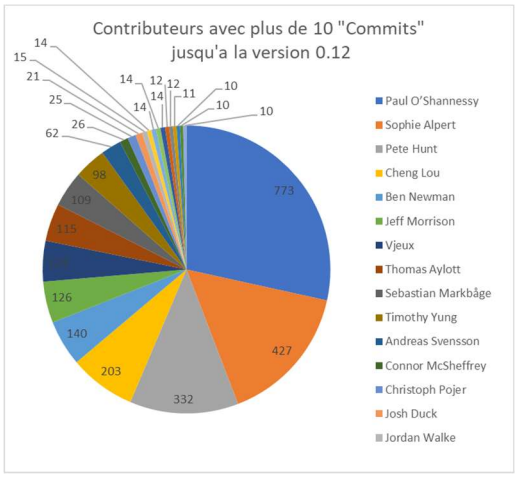


Figure 2 Contributeurs avec plus de 10 « Commits » Période 2

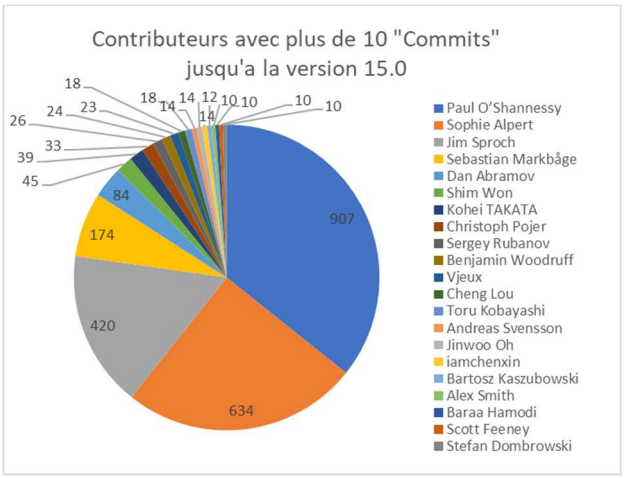


Figure 3 Contributeurs avec plus de 10 « Commits » Période 3

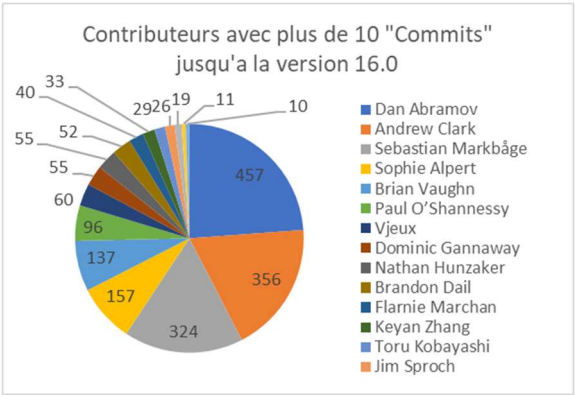
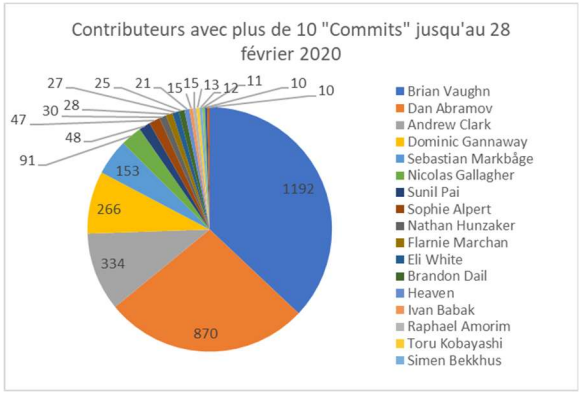


Figure 4 Contributeurs avec plus de 10 « Commits » Période 4



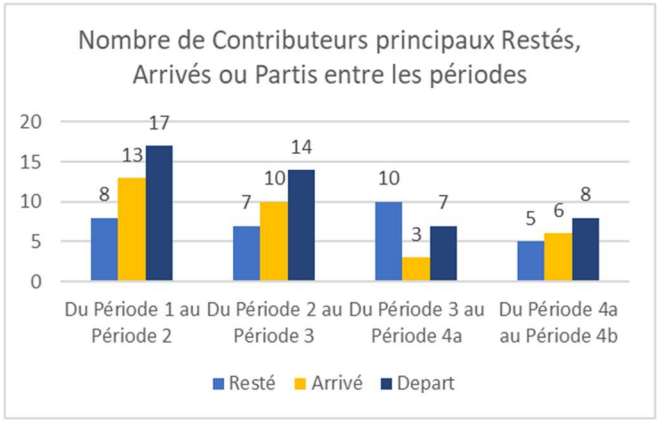
Selon les graphiques montrés, voici les points plus importants à considérer :

- Paul O'Shannessy (membre de l'équipe cœur dans les périodes de majeure contribution, selon un article de ReactRally de juillet 2016 (Rally, 2016)) et Vjeux ont été des top contributeurs selon le nombre des commits pendant les 3 premiers périodes analysés.
- Sebastian Markbåge fait partie de l'équipe cœur et Sophie Alpert l'était jusqu'au début janvier 2019 mais continue a été une contributrice active après la version 16.0.
- Dan Abramov est dans le top des contributeurs du projet depuis la version 15.0 mais contribuait déjà depuis le début.

Nous voyons que dans la dernière période plusieurs membres de l'équipe cœur ont changé mais des contributeurs qui sont depuis le début restent dans le projet. Ce qu'on voit c'est qu'est un élément important pour considérer l'équipe de développement stable est que dans tous les périodes analysés, les top contributeurs ont été membres de l'équipe cœur, ce qui confirme que l'Équipe cœur lead le développement du produit.

Dans les graphiques suivants il est montré le nombre de contributeurs en considérant ce avec le plus de « commits » qu'ont Resté, Arrivé ou Parti enter les périodes. La période 4 décrit avant a été divisé en Période 4a et 4b.

Figure 5

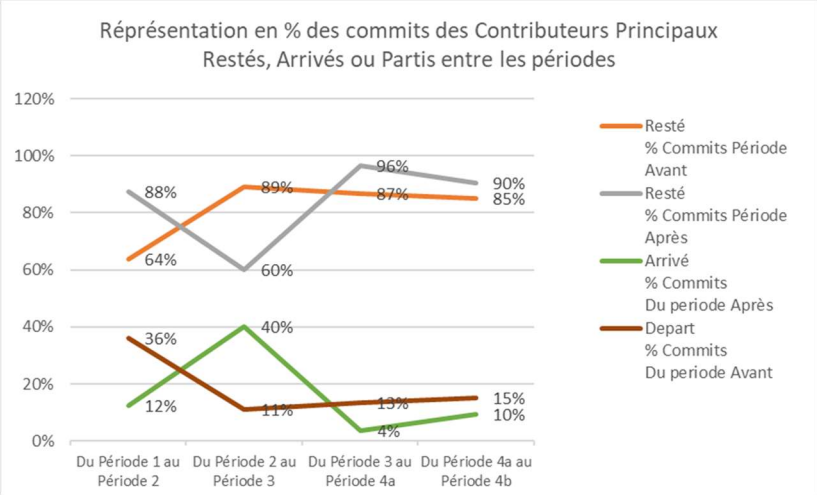


On peut observer qu'il a eu beaucoup plus de changements entre 2013 et 2016 où les arrivés ou sortis de la liste des contributeurs principaux dépassent le nombre des contributeurs qui restent.

Cependant, la Figure 7 montre les données du point de vu de l'implication des contributeurs (le pourcentage par catégorie selon le total de commits de la liste de contributeurs principaux par période). Il est possible de voir que la contribution des contributeurs qui restent dans le top de chaque période est plus important, parce qu'elle représente normalement plus de 60%, en opposition de ceux qui sort de la liste ou qui arrivent comme principaux contributeurs.

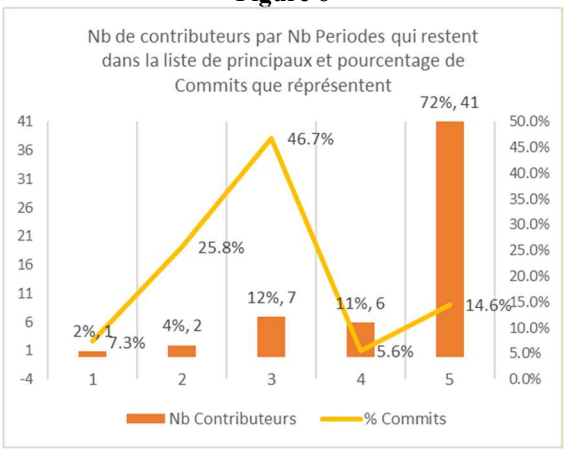
Ces données donnent un autre indicateur de la stabilité de développement par rapport à l'impact des changements dans l'équipe. La Figure 6 complément ce point. Elle montre pour combien des périodes et com combien des contributeurs restent dans la liste des principaux. Nous pouvons voir que bien que le nombre des contributeurs qui restent uniquement 1 période est haut (41, soit un 72% du total des contributeurs de tous les périodes) si on regarde en termes de pourcentage de commits ceci représente uniquement un 14.6 %.

Figure 7



D'autres données pourraient être sorti pour renforcer ces conclusions, comme valider si les contributeurs qui ne sont plus dans les principaux restent encore actifs ou combien d'entre eux font partie de l'équipe cœur et combien les externes.

Figure 6



2.3 ANALYSES DE LA RÉPARTITION DE LA PATERNITÉ DU CODE SOURCE DANS L'ÉQUIPE

Pour déterminer la paternité du Code nous nous servons de l'outil « Code Maat ». Les commandes « main-dev » et « entity-ownership » ont été exécutés pour obtenir une analyse de la paternité des éléments du code source.

```
java -jar ~/ProjectsGit/code-maat/target/code-maat-1.1-SNAPSHOT-
standalone.jar -c git2 -l maat_evo_react_complet.log -a main-dev >
main_devs_react.csv
```

```
java -jar ~/ProjectsGit/code-maat/target/code-maat-1.1-SNAPSHOT-
standalone.jar -c git2 -l maat_evo_react_complet.log -a entity-
ownership > entity_ownership_react.csv
```

Le fichier `main_devs_react.csv` et `entity_ownership_react.csv` ont été généré respectivement. Le premier contient le nombre des lignes ajoutés et supprimés par le développeur qui a la plus grande paternité sur le fichier appelé entité dans les données résultantes. Le deuxième contient le

nombre de lignes ajoutés et supprimés pour chaque un des contributeurs par entité. Les résultats ont été transféré dans le fichier de travail « AnalysePaternite_react.xlsx » pour en sortir des graphiques.

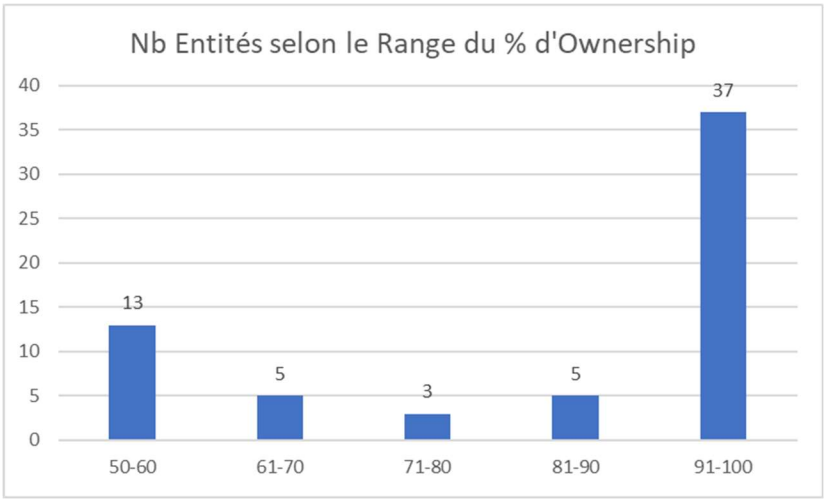
Une des composantes importantes à valider est « React Core » qui contient les principales API's utilisé dans React. Ces composantes sont importantes puisqu'elles sont les composantes dont l'équipe cœur est supposé leader principalement le développement et comme eux l'on mentionné, la simplicité et le contrôle des changements est clé pour la stabilité du produit (TheDevCouple, 2017). La composante se trouve dans le répertoire du dépôt « packages/react » (Facebook, 2020).

À partir de résultats obtenus par l'outil « Code Maat » nous obtenons le nombre des contributeurs qu'on ajouté ou supprimé des lignes des code dans ces entités.

La «

Figure 8 **Nombre d'entités avec un Ownership supérieur à 50%**» représente le sommaire de « ownership » des entités étudiés. Nous observons que d'un total de 83 entités, il en an 63 dont la paternité est de plus de 50%, c'est-à-dire, pour la plupart des entités. Ceci nous permet de conclure que la paternité des éléments concernant cette composante importante n'est pas partagée en temps normal par plusieurs développeurs, ce qui contribue à éviter des problématiques de communication plus complexe entre les contributeurs.

Figure 8 Nombre d'entités avec un Ownership supérieur à 50%



2.4 CONTRIBUTION, COMMUNICATION ET FUTURE DÉVELOPPEMENT

React a une procédure établie pour la contribution de projet. Tout développeur qui désire contribuer doit adhérer à un code de conduite et suivre les consignes décrites dans le page sur « Comment contribuer » (Facebook, How to Contribute, 2020). Dans la même page, il met les liens sur les différentes formes d'interagir avec la communauté.

Ajouter d'issues sur le dépôt est le meilleur moyen qu'ils proposent pour le suivi des propositions ou changements. Cette approche semble très adéquate puisqu'il permet à l'équipe cœur de suivre les

changements mineurs effectués par les contributeurs externes ainsi que les propositions aux composants Core.

L'équipe cœur prends soin d'éviter que des contributeurs effectuent « pull requests » des changements importants dans les composantes du produit sans avoir une connaissance auparavant. Pour ce faire ils demandent d'effectuer un « Request for comments » de sorte qu'ils puissent discuter des éléments de conception importantes avec le contributeur qui le proposent.

Une autre façon d'interagir et veiller pour le saine développement du produit est l'expérimentation. L'équipe a un dépôt spécifique pour proposer et expérimenter certains propositions, idées de changements de sorte que la conception soit solidifiée et expérimentée et choisir la meilleure solution avant de l'inclure dans le développement du produit (React Community, 2020).

2.5 FUTURES ANALYSE

D'autres éléments suggérés dans le livre « Your code as a Crime Scene » pourrait être considérés. Nous pouvons faire une analyse de la paternité para périodes, pour voir comment les développeurs travaillent pour une livraison donnée et nous pourrions explorer la détection de hotspots et comment ceci a de l'influence dans les contributeurs.

À partir de données de la base de données GitHammer nous avons le nombre de lignes ajoutés et supprimés. Avec ceci nous pouvons déterminer la contribution de l'équipe cœur et les contributeurs externes à l'ajout ou modificat

3 DIMENSION CODE SOURCE

Dans cette dimension nous explorons les outils utilisés pour le produit autant pour la construction et pour l'implémentation.

3.1 OUTILS DE CONSTRUCTION

Pour contribuer sur React il est nécessaire avoir les outils suivants :

- Node v8.00 et plus
- Yarn version 1.2.0 et plus
- Gcc ou autre compilateur

3.2 BRANCHE

React n'utilise pas des branches séparées pour le développement. Chaque contributeur créera sa branche mais les changements vont être mergé directement dans la branche « Master ». Ainsi, React profit du « Feature Branche Workflow »

Le choix de ce workflow permet à React de profiter des avantages nécessaires pour leur type de projet :

- Ayant une unique branche principale, « master » ils s'assure qu'elle n'a pas de code non fonctionnel le plus possible (Facebook, How to Contribute, 2020)
- Facilite la collaboration et permet et aide à la livraison continue et l'intégration de nouvelles fonctionnalités rapidement (Git, 2020), (Vallandingham, 2015)

3.3 TESTS

Les Unit tests sont placés dans un repertoire _Tests_ placé dans l'emplacement où se trouve les fichiers qui tests. Les tests sont exécutés avant d'effectuer des merges. L'équipe a construit un « DOM Fixture » pour certains modules avec de tests manuels pour attraper les problèmes des navigateurs. Parce que React est très attaché au site Facebook.com, il a l'avantage d'effectuer plusieurs validations (TheDevCouple, 2017):

- Lorsque la version de React est mise à niveau chaque deux semaines, un serie des unit tests de Java Script en lien avec React sont exécutes
- De test d'intégration avec Selenium sont exécutés
- Par le sytème de déploiement en continue, de mises à niveau sont effectué dans les outils facebook utilisé par les employés pour détecter des bugs avant de déployer la version à tous les utilisateurs.

4 DIMENSION DÉPLOIEMENT ET LIVRAISON

Pour la livraison il est organisé en Milestone. Les Milestones sont déterminés par l'équipe cœur. Au fur et à mesure ils livrent des changements mineurs. Pour une livraison avec des changements plus importantes, le temps entre une livraison pu aller jusqu'à 2 ans.

Les features inclus dans une livraison sont identifiés par l'équipe cœur selon les besoins de Facebook mais aussi ce qui est signalé par la communauté, ce qui pourrait avantager le développement des applications avec React ou les problèmes plus récurrents dans l'ensemble des usagers. En date de 2017, il n'utilisait pas un Roadmap par plusieurs années mais l'équipe mentionne que lors de l'approchement de la planification de la livraison ils demandent de la retro alimentation à la communauté pour les features identifiés.

4.1 GESTIONNAIRE DE PACKAGES

React livre leur version à travers de NPM (https://www.npmjs.com/package/react) gestionnaire des packages pour des logiciels JavaScript, ceci permet de gérer les dépendances du logicielle et différentes versions du Release.

Présentement, React travaille avec 3 « Release Channels » :

- Latest : est la version stable et celle recommandé pour utilisation
- Next : provenant de la branche Master, cette branche représente un « Release Candidate »
- Experimental : provenant de la branche Master, ceci inclus des API's d'expérimentation est « Feature Flags » additionnels activés.

La création de ces 3 différents livraisons permet assurer un produit stable à différentes étapes de la livraison. React encourage le test vers les applications qui l'utilisent valider à l'avance si un futur changement peut affecter les applications dépendantes. En plus, ceci fait possible d'utiliser UNPKG https://unpkg.com/, qui es un outil pour livrer facilement des paquets qui sont dans NPM.

React utilise le gestionnaire de packages livré par Facebook en 2016 Yarn (https://classic.yarnpkg.com/en/), qui permet travailler ensemble avec npm pour le coté client, donnant une expérience client centré sur la performance.

4.2 OUTIL D'INTÉGRATION

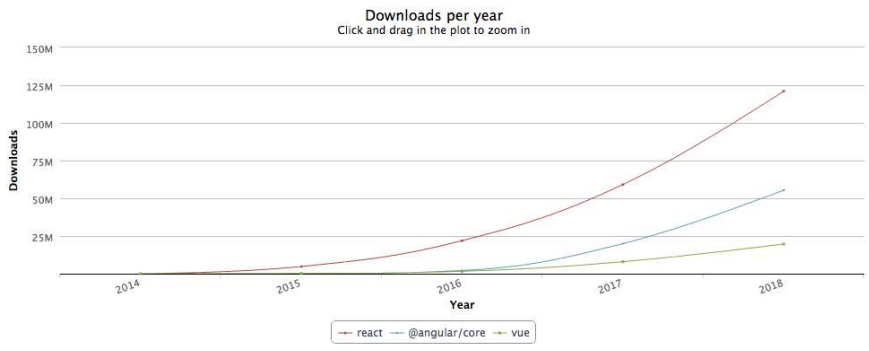
L'outil d'intégration utilisé est CircleCI (https://circleci.com/gh/facebook/react). Circle CI permet d'exécuter les tests pour s'assurer que React a une branche stable et qui s'intègre avec GitHub.

CircleCI est un des outils d'intégration leader dans l'industrie (Altexsoft, 2019). L'outil est utilisé par Facebook et d'autres importantes compagnies comme Spotify.

4.3 UTILISATION DE REACT

L'article … positionne React avec 761,401 WebSites et 259, 772 domains. Il est utilisé par organisations ou marques importantes, mise à part Facebook, par Instagram, Twitter, Netflix, Yahoo!Mail, Khan Academy, WhatsApp, Dropbox, Attlassian, Salesforce, Codecademy, Github et autres (Nadia Hlebowitsh, 2019)).

Au long des années l'utilisation de React a evolué, comme le demontre l'article (Balbier, 2019)



Mais l'évolution peut être aussi en termes de la manière de l'utiliser comme Netflix, qu'en 2017 a arrêté son utilisation de côté client (pour le laisser uniquement de côté serveur) pour améliorer la performance.

5 CONCLUSION

La méthodologie de travail en Équipe de React est bien documenté. L'équipe que développe le logiciel est supporté pour une organisation, Facebook, qui est la plus intéressé à garder une logicielle saine et stable, puisque leur produit y est dépendant.

Le lead que l'équipe cœur a sur le logiciel lui donne de la stabilité. Nous avons vu dans l'analyse de la dimension de l'équipe de développement que la stabilité de l'équipe provient du fait que dans les principaux développeurs il y a toujours des personnes de l'équipe cœur. Bien que le développeur créateur ne soit plus dans l'équipe, plusieurs développeurs sont là depuis le début ou ils ont fait partie d'une longue période dans l'histoire du logicielle.

La répartition de la paternité a été un point intéressant à regarder, ceci nous permettait voir comment la paternité dans la principale composante de logiciel s'est comporté au long du projet. Nous avons constaté que la plupart des entités examinés pour la composante React Core, avaient moins d'un développeur en charge, et que comme React l'établie, c'est l'équipe lead principalement en charge de cette composante.

Quant à la Dimension Code Source, ce que l'on observer est que l'utilisation de feature branche permet d'avoir une gestion des branches qui évite de polluer la branche master, ce qui est importante parce que React s'intègre au cycles de 2 semaines de Facebook. Au même temps, toutes les étapes de tests garanti que le code est validé sur plusieurs points de vue et les anomalies peuvent être détecté à différents moments sur tous ses filtres avant qu'un version soit lancé au public, ce qui donnerais de la confiance aux grandes entreprises d'utiliser ce framework.

Pour le développement en revanche, il est nécessaire être en connaissance de certains outils, mais pour bien être supporté l'équipe met à disposition plusieurs moyennes de communication. Ceci assure que le développement se suit selon les directrices de l'équipe cœur.

Le type de workflow de branche utilisé assure une constante surveillance sur le développement d'une communauté nombreuse. Les différentes étapes de tests, pour le développement en cours, avec les différents channels sur NPM, le code source d'expérimentation et le « Feature Flags » qui doivent être implémenté pour le changements majeurs assure une branche stable.

Le projet React, étant développé pour un grand nombre des contributeurs est confronté aux défis de la maintenabilité au niveau des connaissances du produit par les contributeurs et l'impact des changements non uniquement sur leur propre code mais sur toutes les applications qui l'utilisent. En tant que l'équipe cœur lead la vision et la révision du produit, ceci garanti une stabilité adéquate. Si l'équipe cœur ou en général les contributeurs principaux changent constamment, ceci pourrait affecter la maintenabilité au futur.

6 REFERENCES

Altexsoft. (2019, Février 19). *Altexsoft Blog*. Retrieved from Comparison of Most Popular Continuous Integration Tools: Jenkins, TeamCity, Bamboo, Travis CI and more: https://www.altexsoft.com/blog/engineering/comparison-of-most-popular-continuous-integration-tools-jenkins-teamcity-bamboo-travis-ci-and-more/

Balbier, P. (2019, Juillet 2). *What Is the React.js Framework? When and Why Should I Use React.js in My Project?* Retrieved from netguru: https://www.netguru.com/blog/what-is-react-js

Facebook. (2020, Février 28). *Codebase Overview - React Core*. Retrieved from React: https://reactjs.org/docs/codebase-overview.html#react-core

Facebook. (2020, Février 28). *Équipe Coeur React*. Retrieved from React: https://reactjs.org/community/team.html

Facebook. (2020, 02 28). *How to Contribute*. Retrieved from React: https://reactjs.org/docs/how-to-contribute.html

Facebook. (2020). *React*. Retrieved from React: https://reactjs.org/

Git. (2020). *Git Feature Branch Workflow*. Retrieved from Attlasian.com Git tutorials: https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow

Kangasharju, J. (2020, février 28). *GitHub*. Retrieved from Dépôt GitHammer: https://github.com/asharov/git-hammer/blob/master/README.md

Nadia Hlebowitsh. (2019, June 10). *2019 Stats on Top JS Frameworks : React, Angular & Vue*. Retrieved from TECLA: https://www.tecla.io/blog/2019-stats-on-top-js-frameworks-react-angular-and-vue/

Rally, R. (2016, Juillet 07). *Write your own React with Paul O'Shannessy*. Retrieved from https://medium.com/: https://medium.com/@ReactRally/write-your-own-react-with-paul-oshannessy-beb4353458d2

React Community. (2020, février 28). *The future of react*. Retrieved from Repository react-future: https://github.com/reactjs/react-future

TheDevCouple. (2017, Octobre 9). *We Interviewed the React.js Team at Facebook About WordPress & Gutenberg!-Q #4: How Does React Approach Breaking Changes?* Retrieved from https://thedevcouple.com/: https://thedevcouple.com/interview-react-team-facebook-wordpress-gutenberg/

Tornhill, A. (2015). *Your Code as a Crime Scene.*

Vallandingham, J. (2015, Juillet 20). *A GIT WORKFLOW WALKTHROUGH – FEATURE BRANCHES*. Retrieved from Blog Jim Vallandingham: https://docs.gitlab.com/ee/gitlab-basics/feature_branch_workflow.html

Wikipedia. (2020, février 28). *Wikipedia*. Retrieved from React (web framework): https://en.wikipedia.org/wiki/React_(web_framework)

7 ANNEXE OUTILS

7.1 CODE MAAT

Le dépôt Git de Code Maat se trouve à l'adresse https://github.com/adamtornhill/code-maat. La page README du dépôt explique comment exécuter l'outil soit par « leiningen» ou à partir d'un fichier « standalone » généré.

Après avoir obtenu, le dépôt du code avec la commande « clone » de Git, la commande « lein uberjar » a été exécuté pour générer un fichier jar.

Le fichier « CommandesOutilCodeMaat.txt » avec les commandes GIT et les commandes Code Maat utilisés se trouvent dans le dépôt de code source de ce Projet sur le dossier Data.

Pour pouvoir exécuter l'analyse, il faut premièrement, générer un fichier de log du dépôt de code source. Ce fichier peut être généré pour le dépôt au complet ou pour un période de temps selon les besoins. À partir du dépôt local du logicielle à analyser la commande suivante a été exécuté :

```
git log --numstat --date=short --pretty=format:'--%h--%ad--%aN' > maat_evo_react_complet.log
```

Le fichier maa_evo_react_new.log a été généré pour être utilisé pour les différentes statistiques générés pour l'analyse.

7.2 GITHAMMER

Le dépôt de l'application GitHammer se trouve dans https://github.com/asharov/git-hammer. La page README explique les étapes pour la configuration et installation.

Pour créer la base de données il faut suivre les indications de « Création de Projet ». Pour les besoins de ce travail nous avons créés une base de données contenant les informations du dépôt de React avec les commandes suivantes.

```
Python -m githammer init-project ReactAnalysis ~/ProjectsGit/react
```

Une base de données git-hammer.sqlite a été créé avec les données sur les commits et les contributeurs extraite du dépôt de react.

Le fichier « CommandesOutilGitHammer.txt » avec les requêtes SQL utilisés se trouvent dans le dépôt de code source de ce Projet sur le dossier Data.