MGL7460

Réalisation & Maintenance logiciels

Projet Individuel

Nasreddine Salem

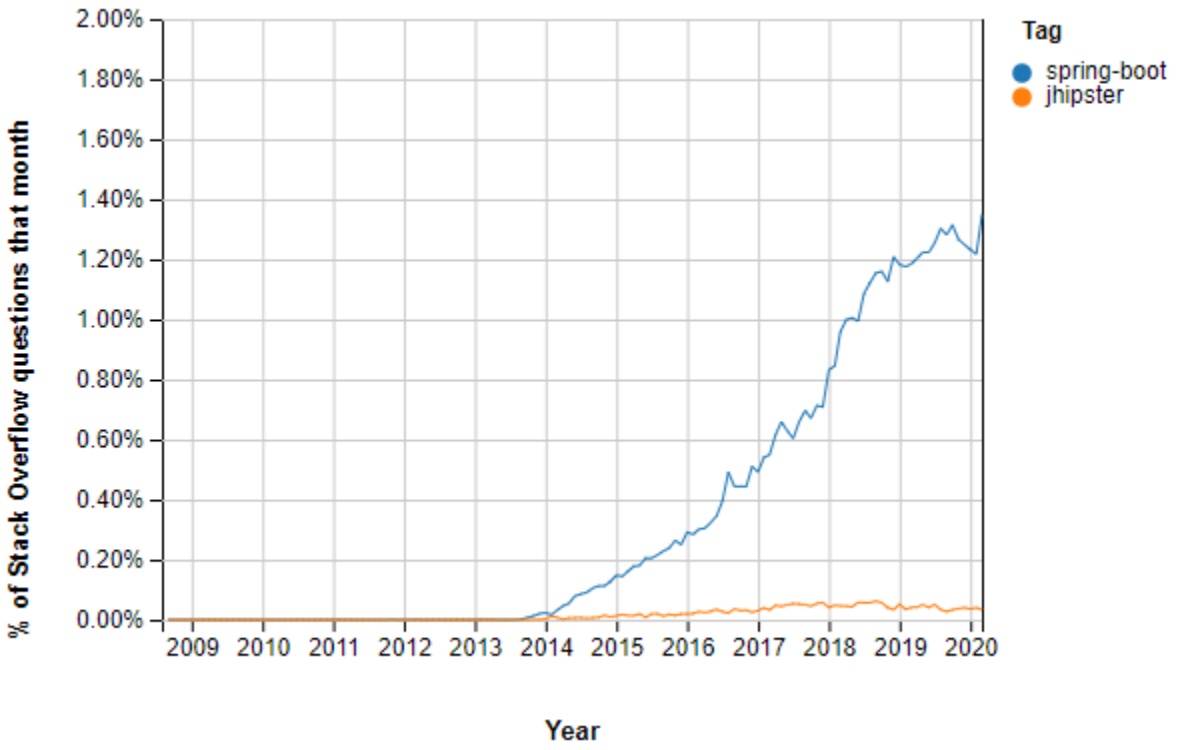
18 Janvier 2020

INTRODUCTION

TP1- RÉALISATION & MAINTENANCE LOGICIELS

Le cours sur la *réalisation et maintenance de logiciels* se conclut avec ce travail pratique. Lors des dernières cours, les concepts tels que la gestion de version et les tests unitaire et intégré lors de la réalisation d'un logiciel et leur importance ont été présentés. Ces pratiques et concepts d'architecture qui présentent des solutions aux problèmes les plus communs complètent la formation théorique.

Ce devoir est une application de ces concepts, mon choix est porté sur le logiciel open source Spring Initializer. C'est un outil développé par l'équipe de Spring boot afin de facilité la génération d'une première structure de projet basé sur du Spring boot, un générateur de projet utiliser par des plusieurs développeurs comme nous l'indique ce site de comparaison avec un outil similaire jHipster.
https://insights.stackoverflow.com/trends?tags=spring-boot%2Cjhipster



Nous remarquons clairement l'utilisation de spring-boot initializer depuis 2014 est on continue progression par rapport à l'un de ces homologues sur le marché.
Ce rapport présentera les différentes dimensions d'analyse suivie, afin de mieux comprendre la phase de réalisation de ce logiciel et sa maintenance. Ces derniers fournissent les détails nécessaires de la contribution des développeurs via l'analyse de Git. Ce faisant, il sera possible de démontrer la bonne forme de l'architecture de logiciel et sa maintenabilité ou pas.

SOMMAIRE D'ANALYSE

Je travaille sur l'analyse et l'étude d'un logiciel en libre-service, c'est un générateur de code pour les projets de types Spring-Boot.

L'analyse sera dirigée principalement par les différentes dimensions suivantes :

- Dimension - Équipe de développement
- Dimension - Architecture logicielle
- Dimension - Tests

Dans cette synthèse, je vais répondre minimalement à une question par dimension, afin de couvrir le plus d'aspects pour ce produit, et comprendre un peu plus sur sa phase de réalisation que ça maintenance.

DIMENSION — ÉQUIPE DE DÉVELOPPEMENT

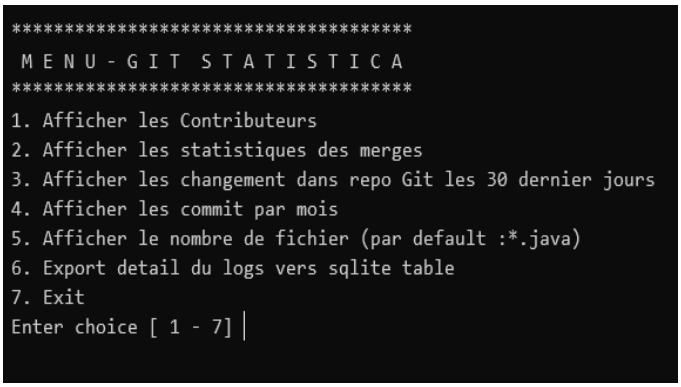
Dans cette phase d'analyse, nous intéresserons principalement aux contributeurs actifs, et leurs contributions, par l'analyse des toutes les actions enregistrer au niveau du gestionnaire de version GIT.

PRÉREQUIS :

Avant d'analyser les données, j'ai mis en place un utilitaire batch, qui permet l'extraction des données à partir du log Git, puis les importer dans une base de données de type SQLITE, pour pouvoir exploiter plus aisément les données.

IL faudra avoir ces outils installés préalablement sur le poste de travail avant de commencer l'exploitation de cet utilitaire, plus de détail sur le dépôt Git : https://github.com/nsalem-it/MGL_7460

GIT-STATISTICA



- Source :https://github.com/nsalem-it/MGL_7460
- Version : 1.0.0

INSTALLATION:

Une fois le dépôt GIT du projet et cloner sur le post de développement, à partir de la racine de projet, lancez le fichier batch suivant :

```
[redacted] 253LVL /mnt/c/Uqam/MGL-7460/projet/initializr (master) $ ../MGL_7460/gitStatistica/gitStatistica.sh
```

L'outil propose des fonctions qui permet l'affichage de certaines statistiques de contribution dans le projet, mais pour une meilleure analyse des données, l'option d'exportation vers une base de données SQLITE est proposée pour des besoins de forage de données plus détailler.

La base de données est composée de quatre tables :

- ComitParAnneeParAuteur : Total de commit par auteur regrouper par année.
- ContributionParFichier : Nom des fichiers avec leur contributeur.
- DetailedGitStatsTbl : Pour chaque contributeur on retrouve le nombre de commit et de ligne insérée et supprimer.
- ImplicationTbl : Pour chaque contributeur on calcule son taux d'implication.

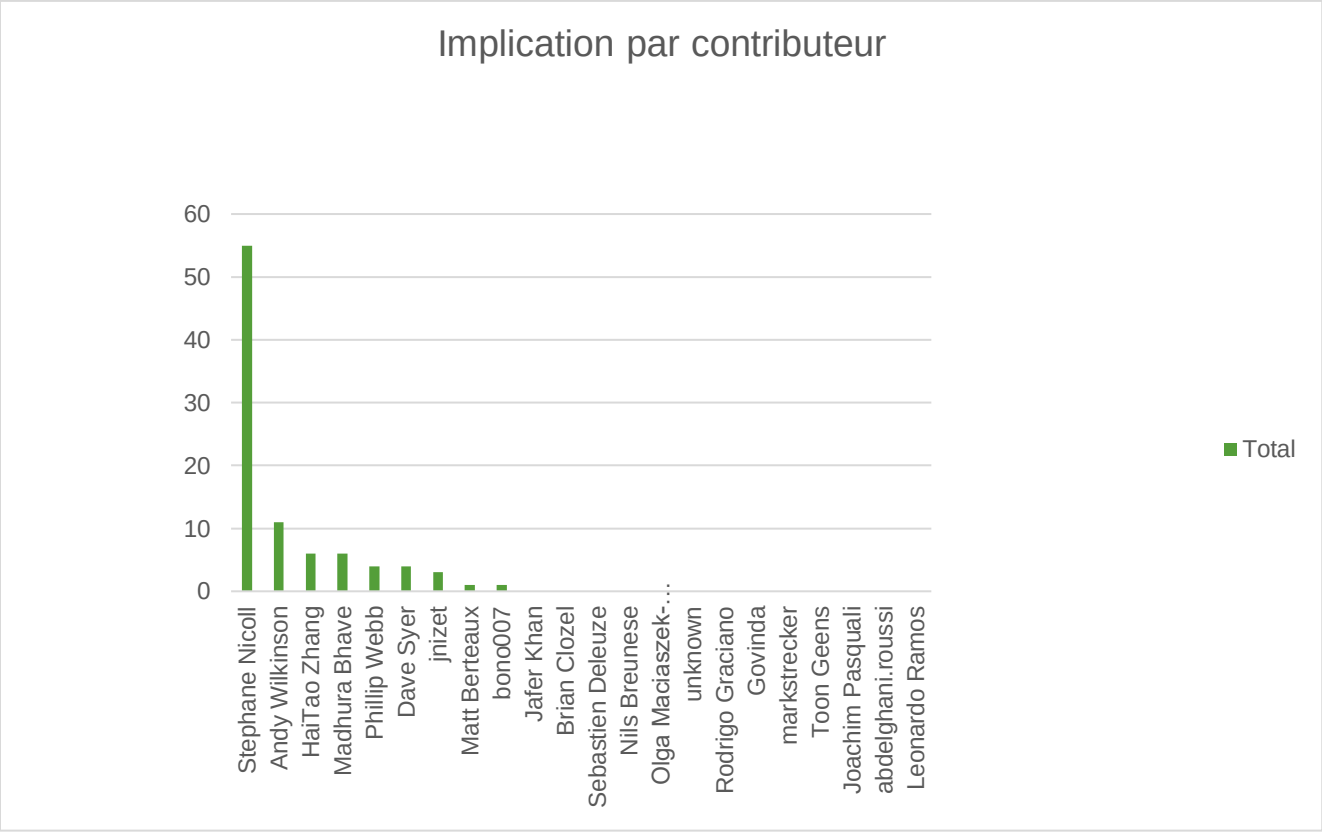
IMPLICATION DES CONTRIBUTEURS

Qui sont les développeurs principaux du projet ? pour répondre à cette question, nous allons utiliser un script qui calcule le taux d'implication des développeurs dans ce projet selon la formule suivante :

$$Implication\ (dev) = \frac{|C|}{|T|} \times 100$$

|T| = Nombre de fichiers complet

|C| = Nombre de fichiers sur lequel « dev » à contribuer



Contributeur	Implication	Emplacement	Rôle	Compagnie
Stéphane Nicoll	55%	Liège, Belgium	Commiter	Spring Team
Andy Wilkinson	11%	Southampton, UK	Développeur	Spring Team
Madhura Bhave	6%	San Francisco, CA	Développeur	Spring Team
Phillip Webb	4%	s-o	s-o	s-o
Dave Syer	4%	London	Senior Consulting Engineer	Spring Team

1. On distingue clairement un groupe de contributeurs qui sont les plus actifs dans ce projet, puis je constate clairement que **M. Stépahen Nicoll** et le contributeur principal de ce projet, d'ailleurs avec une petite recherche sur le web, je retrouve facilement cet auteur comme développeur principal et initiateur de ce projet, dans l'équipe de développement de Spring.
2. L'autre point intéressant est que les contributeurs les plus importants font partie de la même compagnie, c'est-à-dire Spring-Team, et que sont géographiquement séparés, donc on se demande dans ce genre de situation, comment la collaboration n'est pas affectée, par la distance ou la langue?
3. Contributeurs externes : je remarque que le nombre de contributeurs externe et leurs contributions est relativement faibles comparer à la contribution des développeurs qui travail chez Spring. Ceci peut s'expliquer par le fait que ce projet reste un outil de génération de code pour le Framework Spring boot, donc assez spécifique pour le langage de programmation et très limité à un groupe de développeur.

ANALYSE DES COMMITS

Je commence par analyser les commit totaux effectuer pendant chaque année, et cela depuis la première année à ce jour :

Somme de nombreDeCommit	**Étiquettes de colonnes**								
Étiquettes de lignes	**2013**	**2014**	**2015**	**2016**	**2017**	**2018**	**2019**	**2020**	**Total général**
Stephane Nicoll		49	131	196	248	217	296	6	1143
Dave Syer	23	49	30	21	25	7			155
Andy Wilkinson			6	1	2	17	28		54
Madhura Bhave						9	40		49
Brian Clozel		1	9	7	4	1	5		27
Claus Ibsen				6	10	6			22
Sebastien Deleuze		2		5	9	1	1		18
Phillip Webb						11	1		12
Spencer Gibb				3	4	3			10
Ryan Baxter					4	6			10
Spring Buildmaster					3	2	4		9
Craig Walls				2	5	2			9
Josh Long		1		4	2	1			8
HaiTao Zhang							8		8
Roy Clarkson				4		3			7
Stéphane Nicoll					5	1			6
weiping						5			5
Johannes Edmeier						5			5
jnizet							5		5
Total général	**23**	**102**	**176**	**249**	**321**	**297**	**388**	**6**	**1562**

1. On remarque un changement dans le classement des contributeurs, Mr.Dave Syer prend la deuxième place comme commiter dans cette équipe, et sa contribution est assez stable chaque année jusqu'à 2019 ou il a arrêté de contribuer, tandis que Mr Andy Wilkinson et Mr.Madhura Bhave, leurs courbes de contribution a subitement connu une hausse progressive depuis 2018, on peut imaginer qu'un évènement au sein de l'équipe a eu lieu ou un changement majeur dans l'architecture de projet a évolué, comme les changements de rôle ou d'affectation.

2. On peut déjà à ce niveau d'analyse confirmer que l'équipe ayant contribué le plus dans la création de ce projet n'est pas la même tout au long de son évolution, sauf pour le contributeur principal, et la contribution en continu progression a chaque année.

Nous allons creuser un peu plus pour comprendre ce changement. On passe à l'analyse de type de fichier sur lesquels ont travaillé ces trois contributeurs : Dave Syer, Andy Wilkinson et Madhura Brave :

Extrait du fichier Analyse de Initialzr.xsl — onglet : Analyse Dave vs Andy et Madhura

- Dans cette visualisation, nous avons grader seulement les trois développeurs Andy, Dave et Madhura, afin d'observer leurs contributions dans chaque classe du projet, et le point qui ressort de ça est que Dave et Mahdura on contribuer dans les mêmes classes, tandis que Andy a travaillé sur des classes séparer et complément nouveau et principalement arrivé après que Dave à arrêter le codage.

- Nous observons aussi que la tendance du graphe précédent nous mène à la conclusion que Andy et le remplaçant de Dave, mais à ma grande surprise en analysant le fichier des commit pour ces trois contributeurs, j'ai remarqué que Madhura a fait la révision de tous les fichiers sur lesquels Dave à travailler les dernières années, alors que Andy à continuer dans le projet comme nouveau mainteneur.

Un contrôle de version est un livre d'histoire qui raconte le vécu d'une équipe, et ces interactions, ainsi que les changements dans son vécu, car le but qui les réunit avant tous c'est la contribution dans le même projet.

Afin de confirmer cette histoire, nous allons continuer notre investigation dans l'analyse de données Git, les plus détaillés tel que les nombres de lignes insérer et les dates du premier et dernier commit pour chaque contributeur.

Dans ce graphe la chronologie dans le temps des différentes interventions des insertions et suppression de ligne dans le code, nous démontre que Mr.Dave a participé activement pendant la période entre 2013 et 2018, tandis que Mr Madhura a repris la modification d'une grande partie des fichiers sur lesquelles à contribuer Dave, alors que Andy a beaucoup plus ajouter des lignes dans le code, ce qui le qualifie de nouveau mainteneur dans ce projet.

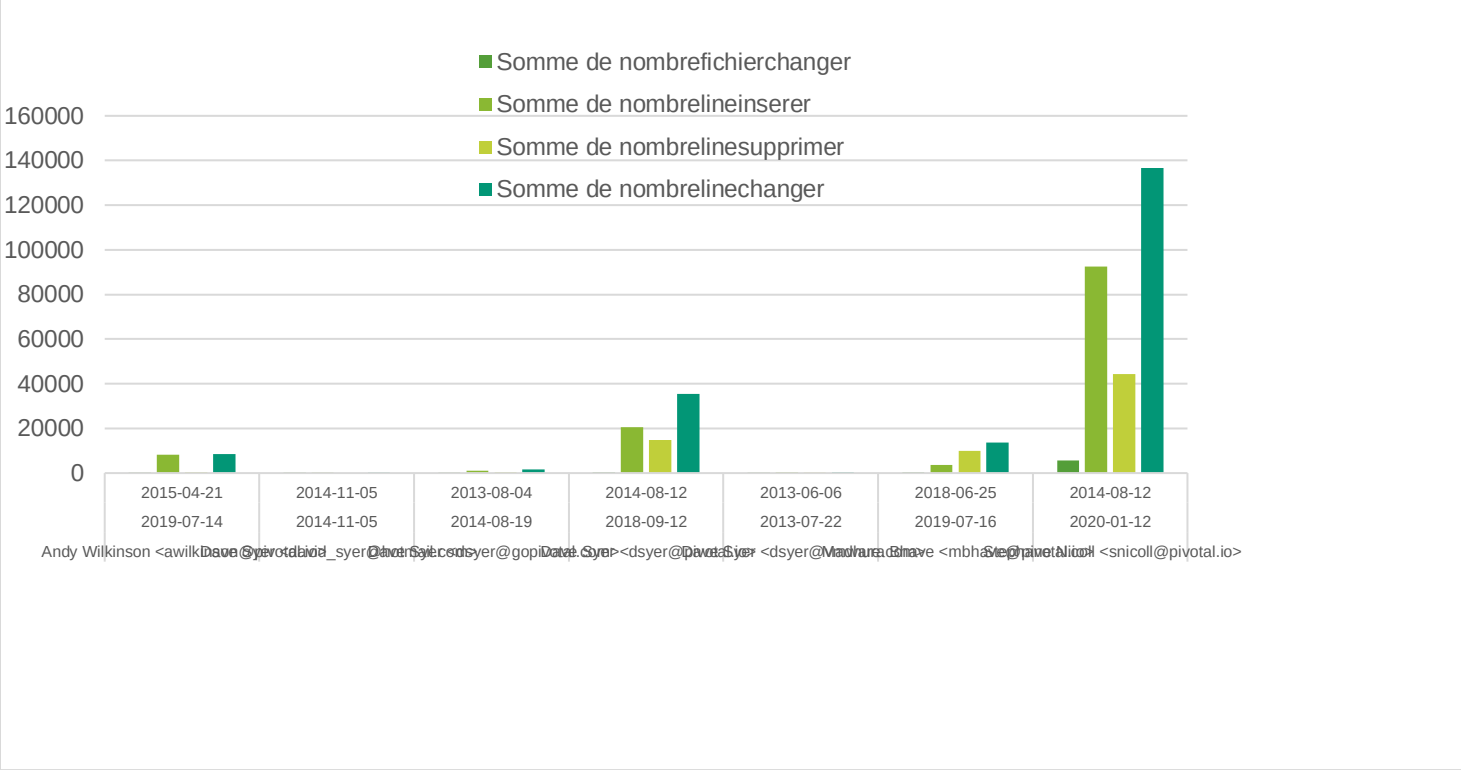


Figure : Implication détailler

- Les autres points qui ont attiré mon attention sont le changement de statut de l'entreprise Spring, car on voit clairement qu'entre 2013 et 2014 c'était goPivotal puis à partir de 2014 elle devient Pivotal. Aussi la parcours de Mr.Dave qui passe de vmware a pivotal, avec probablement une phase de consulting avant de rejoindre définitivement l'équipe spring.

ANALYSE DES MERGE

Dans cette vue on remarque clairement les mainteneurs les plus importants dans ce projet, ce qui confirme notre théorie sur le fait que Andy Wilkinson prend un rôle important dans la continuité de l'évolution du produit, et cela par l'occupation de la deuxième place après Mr.Stéphane.

```
Enter choice [ 1 - 7] 2
% of Total Merges              Author  # of Merges  % of Commits
            89.52      Stephane Nicoll          188         16.44
             6.19       Andy Wilkinson           13         24.07
             1.90     Stéphane Nicoll             4         66.66
             1.42        Madhura Bhave            3          6.12
             0.95            Dave Syer            2          1.29
Press [Enter] key to continue...
```

CONCLUSION

Ce je pourrai conclure de cette première analyse, et que ce projet fait l'objet d'une révision continu et d'amélioration même si les développeurs principaux quittent leur rôle, ou même si l'entreprise qui les réunis connaît des changements dans le statut. Cela a un impact positif pour la maintenance de ce projet dans le futur.

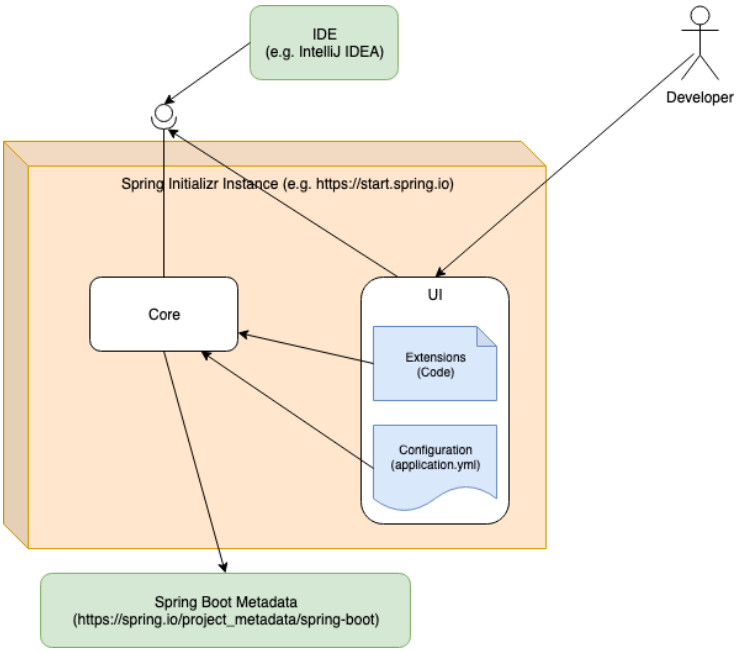
L'autre point qui me semble négatif, c'est le fait que peu de contributeurs externes de l'entreprise Spring, participe dans ce projet, même si c'est un open source, car nous pouvons imaginés que si ce Framework perd sa popularité actuelle, la rien ne pourra garantir l'évolution de cet outil, alors que s'il y avait une plus grande communauté libre sera plus rassurons de savoir qu'une relève et probable.

ARCHITECTURE LOGICIELLE

Référence :https://docs.spring.io/initializr/docs/current-SNAPSHOT/reference/html/

Dans cette partie du document, je vais explorer la dimension architecturale de ce produit, je vais principalement m'intéresser aux différents modules composant ce produit, et les relations entre ces modules, mon analyse portera aussi sur la contribution des développeurs dans ces modules.

VUE GLOBALE D'ARCHITECTURE



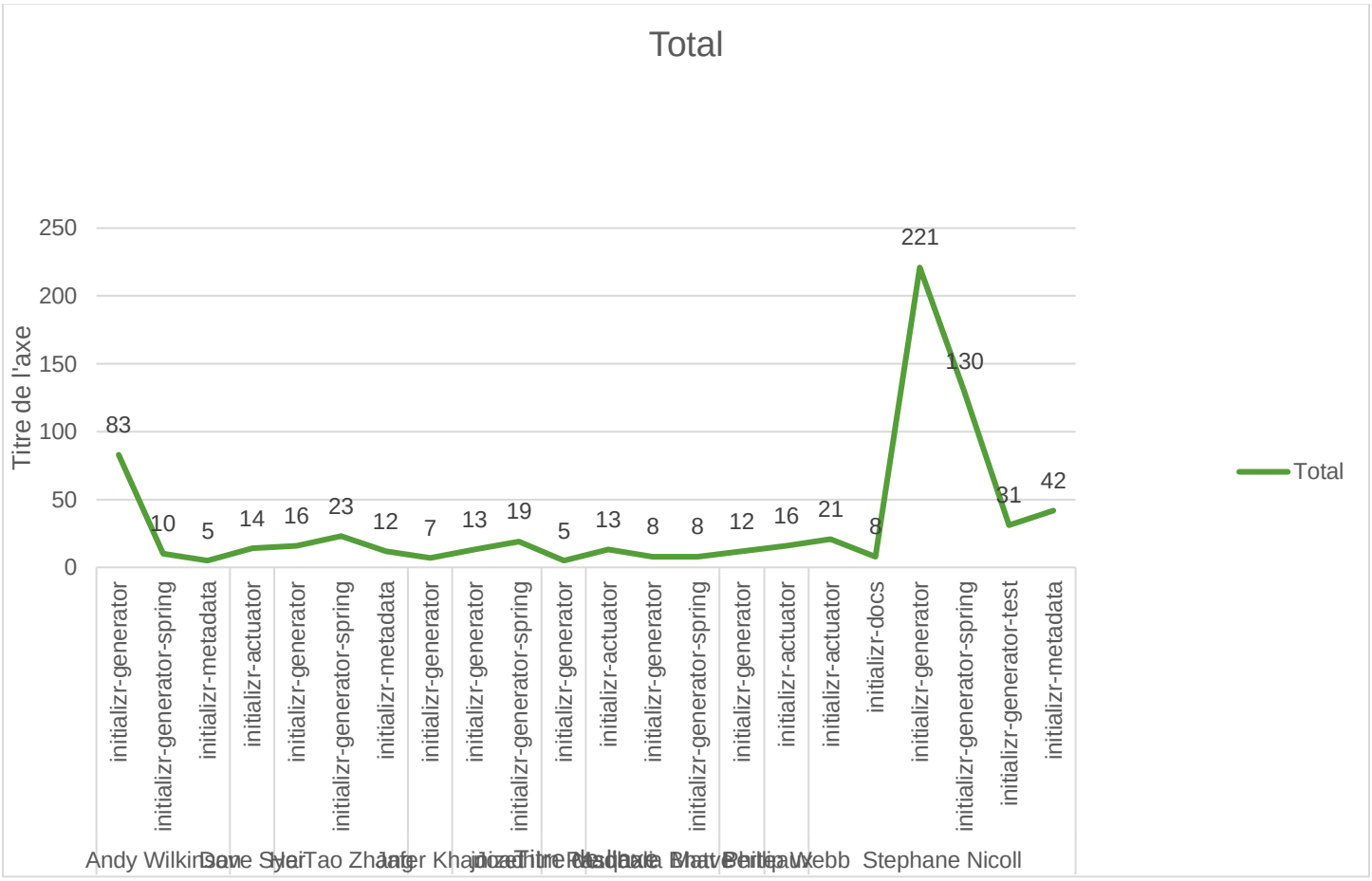
Référence : https://medium.com/digitalfrontiers/how-to-customize-the-spring-initializr-2439ecabb069

Nous avons plusieurs modules qui composent cette application :

- `initializr-actuator`: Module optionnel pour fournir des informations et des statistiques supplémentaires sur la génération de projets
- `initializr-bom`: Fournis une nomenclature pour une gestion plus facile des dépendances dans votre projet.
- `initializr-docs`: Documentation.
- `initializr-generator`: Core principal du projet.
- `initializr-generator-spring`: Module optionnel définissant les conventions pour un projet Spring Boot typique. Peut être réutilisé ou remplacé par vos propres conventions.
- `initializr-generator-test`: tester la composition des modules pour la génération de projets.
- `initializr-metadata`: Infrastructure de métadonnées pour divers aspects du projet.
- `initializr-service-sample`: présente une instance personnalisée de base.
- `initializr-version-resolver`: Module optionnel pour extraire les numéros de version d'un POM. arbitraire.`initializr-web`: Points de terminaison Web pour les clients tiers.

ANALYSE DE LA CONTRIBUTION PAR MODULE

Source des données : Onglet Contribution par module dans le fichier Excel.

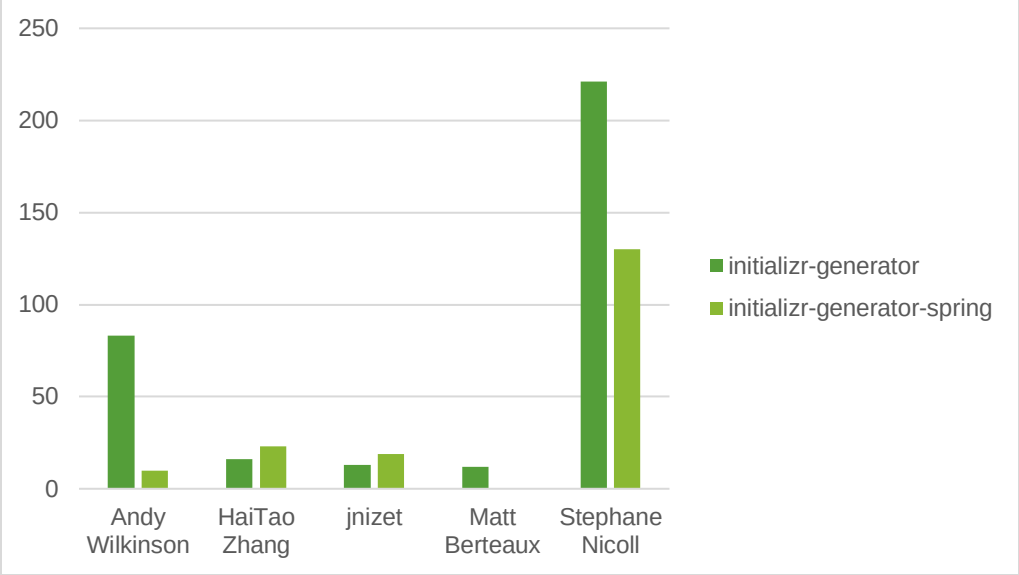


Dans cette courbe nous illustrons la participation de chaque contributeur dans les différents modules, nous avons gardé seulement les contribution supérieure ou égale à 5.

Voici les points les plus pertinents :

- Le module principal est initializr-generator, la plupart des contributeurs ont travaillés sur ce module d'où sont importance dans l'architecture de l'application et sa maintenance.
- Le module documentation, il y a seulement un seul contributeur « Stephan » qui maintien cette documentation, mais elle n'évolue plus depuis 13 mois environ, et je me questionne sur l'impact des changements majeurs non documenté, à date un site pour la communauté permet les échanges sur ce produit est dédié, qui est une bonne chose (https://gitter.im/spring-io/initializr).

Pour la prochaine étape nous allons comparer la contribution des développeurs dans les deux principaux modules, c'est-à-dire le Initializr-generator et Initializr-generator-spring. Afin de restreindre la comparaison a seulement les plus actifs contributeurs, j'ai limité seulement aux contributeurs qui ont faite plus de 10 commit sur les deux modules.



- Stephane et Andy sont les deux principaux contributeurs de ces modules, ce qui confirme notre analyse dans la première phase, ou j'ai conclu que Andy était le nouveau mainteneur du code après le départ de Dave.
- L'autre point intéressant qui sort de cette visualisation, et probablement l'apparition de nouveau contributeur qui ne fait pas partie de l'équipe Spring, mais des développeurs qui contribuent à l'évolution du projet

Alias Git	Fonction	Source
jnizet	Développeur	https://ninja-squad.com/team
HaiTao Zhang	Software Engineer At Uber	https://www.linkedin.com/in/haitao-zhang-3a7b43a6/
Matt Berteaux	Développeur chez Adobe	https://github.com/mattyb678

CONCLUSION :

- Les modules principaux font l'objet d'une plus grande contribution et suivi par les initiateurs du projet, j'ai même remarqué la présence d'un module test « initializr-generator-test » maintenu par Stephane pour valider la bonne intégration des composantes pour la génération des projets, ceci donne une assurance que lors des intégrations de nouvelle version il y a une meilleure assurance qualité , d'autre part savoir que dans l'équipe de Spring Stephane n'est pas le seul mainteneur de ce module principal et qu'une relève est toujours présente pour l'évolution et la correction des issues de ce produit.
- La contribution extérieure des développeurs travaillant pour d'autres entreprises que Spring, et plus importante dans les modules principaux plus précisément le générateur, est un indicateur qui peut être positive pour la maintenance de ce produit, même si elle est à faible participation.
- Je peux déduire aussi que la stabilité de ce produit et basée principalement sur les modules Initializr-generator, initializr-generator-spring et initializr-generator-test, car j'ai remarqué que c'est dans ces modules ou nous observons le plus d'activité au niveau des commit « insertion et suppression de ligne de code », d'où l'importance de révision et maintenance de ces modules.

DIMENSION — TEST

Dans cette dimension, mon analyse s'effectuera principalement sur les tests effectuer dans les différents modules, tels que la qualité des tests ainsi que la qualité du code plus précisément l'analyse de la cohésion et le couplage entre les classes, pour cela je vais m'appuyer sur deux outils :

- SonarQube : pour l'analyse statique des tests.
- CodeMR : un plug-in qui donne des statistiques sur le couplage et la cohésion des classes et composante d'un logiciel.
  https://www.codemr.co.uk/

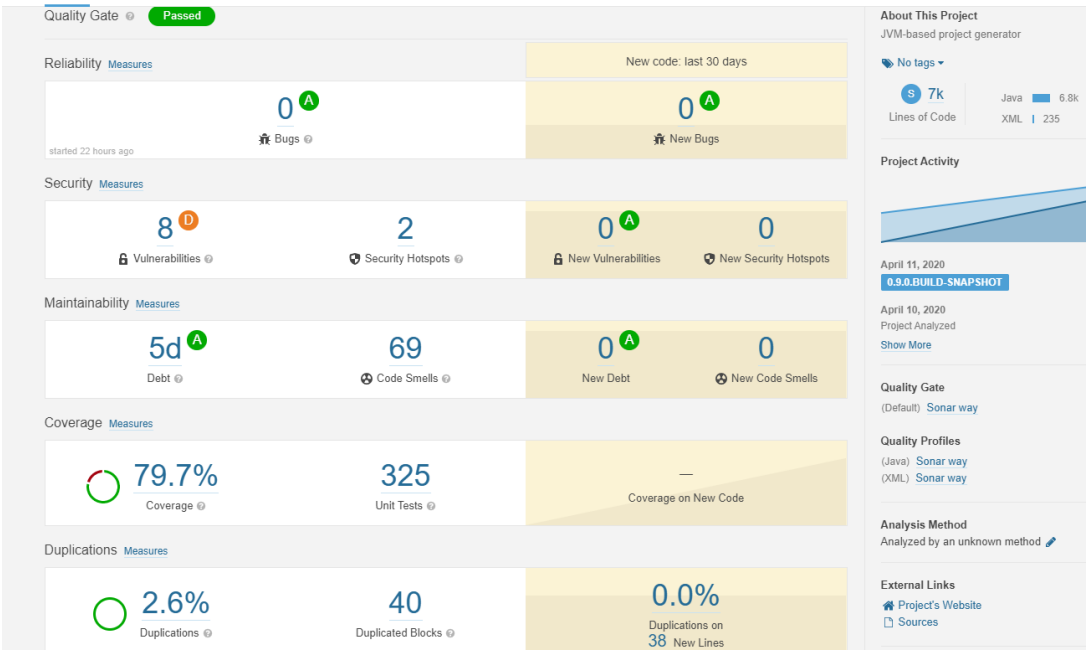
ANALYSE AVEC SONARQUBE

J'ai cloné le projet sur ma machine local, afin que je puisse analyser avec sonarQube cloud, même si le projet est doté d'un pipeline d'intégration continue, ce dernier n'a pas d'analyseur statique du code tel que sonarQube ou autre. Ceci est le lien du pipeline qui complète juste un build puis un déploiement sur la plateforme applicative :

https://ci.spring.io/teams/initializr/pipelines/initializr?groups=Build

Pour cette raison je trouve que l'analyse statique avec sonarQube peut nous aider à mieux comprendre la qualité des tests de ce logiciel, voici les résultats obtenus suite à cette analyse :

https://sonarcloud.io/dashboard?id=io.spring.initializr%3Ainitializr



LES TESTS

Nous avons 325 classes de tests qui couvre l'intégralité du produit, mais pour mieux comprendre la répartition, je vais voir la répartition de ces tests dans les modules :

	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
Spring Initializr							
initializr-actuator/src	404	0	0	0	0	0.0%	0.0%
initializr-docs/src	50	0	0	0	0	0.0%	0.0%
initializr-generator-spring/src	504	0	0	0	0	0.0%	0.0%
initializr-generator-test/src	379	0	0	15	0	68.0%	0.0%
initializr-generator/src	4,074	0	0	42	0	93.3%	4.5%
initializr-metadata/src	572	0	0	6	0	80.4%	3.0%
initializr-service-sample/src/main/java/sample/service	13	0	0	0	1	0.0%	0.0%
initializr-version-resolver/src	1	0	0	0	0	—	0.0%
initializr-web/src	764	0	8	6	1	0.0%	1.3%
pom.xml	235	0	0	0	0	—	0.0%

- Comme attendu la couverture était principalement sur le module central ou cœur « Initialzr-generator », qui est le module principal de ce produit, du coup les autres modules tels que initialzr-actuator ou initialzr-generator-spring ne sont pas couverts par des tests unitaires, cependant on trouve une couverture de type intégration avec le module initialzr-generator-test, ce dernier effectuer des tests d'infrastructure pour couvrir l'ensemble des fonctionnalités de ce produit.

En conclusion, le produit à une assez bonne couverture de test, sans trop avancé sur la qualité, mais le fait d'avoir choisi de coder toute un module pour les tests d'intégration reste un choix de développeur qui peut se justifier par le fais que ce dernier a voulu externaliser ce type de test pour faciliter l'intégration des évolutions des composantes séparément.

Cela dit, les tests d'intégration n'excluent pas le fait de mettre de place des tests unitaires, je trouve dommage que certain de ces modules on ne trouve pas des tests unitaires, du coup tout changement doit passer par les tests intégrer, mais cela ne donne pas l'assurance du bon fonctionnement de la classe ou de l'objet.

ANALYSE AVEC CODEMR

CodeMR est un outil d'analyse qui s'attarde surtout à la qualité de l'architecture. Il évalue de façons statiques, le code et sa qualité. Il fournit plusieurs mesures d'attributs de qualité qui sont dérivés de combinaisons de plusieurs indicateurs.

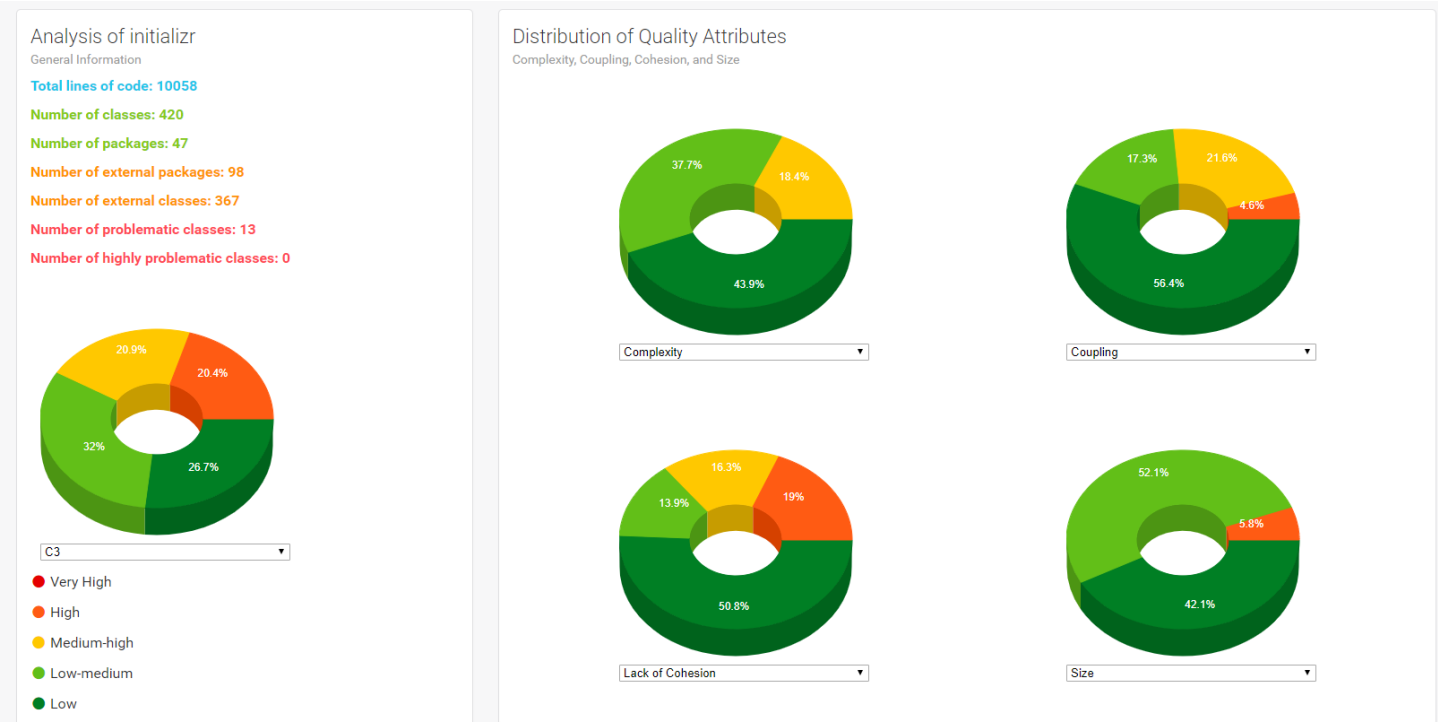
Le couplage entre les éléments du logiciel, le manque de cohésion et le degré de complexité sont des éléments fournis par CodeMR, et qui ont permis de mieux évaluer ce produit.

Source codemr.zip : https://github.com/nsalem-it/MGL_7460/tree/master/documents

Classes with high coupling (#3)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	CBO	CBO APP	CBO LIB	RFC
1	MavenBuildWriter					319	30	29	1	111
2	InitializrAutoCon...					85	22	13	9	86
3	ProjectMetadataCo...					61	21	16	5	99

- Dans cette illustration on remarque la présence de trois classes très fortement coupler avec d'autre classe, qui est tout à fait acceptable, car ces des classes de configuration pour mettre un contexte applicatif.



Sur l'ensemble, l'architecture de ce produit et assez bonne, car je remarque qu'il n'y a pas de classe avec un fort couplage et une faible cohésion, ce qui rendrait la maintenance très complexe

En résumer pour cette phase, ces données pouvant faire l'objet d'une analyse assez profonde pour détecter les bugs ou des problèmes qui peuvent surgir à tous moments avec une bonne corrélation avec les métriques de sonarQube tel que la complexité cyclomatique ou l'indice d'instabilité.

SYNTHÈSE D'ANALYSE

En générale, je trouve que ce produit est très bien fait sur le plan de la conception architecturale, la réalisation sous forme de plusieurs modules séparer par fonctionnalités est une bonne approche pour la maintenance des modules ou la gestion des versions, car cela ne nécessite pas la maintenance de tous les modules qui sont assez stable et peu évolutive. Voici quelques points que je retiens pour ce projet :

- La communauté des développeurs externe qui contribuent à l'évolution de ce produit et assez faible, elle st composer principalement des développeurs de Spring.
- Récemment ce projet a intégré la technologie Kotlin, qui est un bon point pour l'évolution future, sauf que ce dernier reste assez restreint niveau technologie.
- La qualité du code et assez bonne et continue à s'améliorer à chaque découvert de nouveau bug, un suivi de ces bugs et bien gérer au niveau du gît.
- Les changements des développeurs dans l'équipe de Spring n'ont pas beaucoup d'influence sur le code ou sa maintenance.
- L'absence d'un pipeline pour la qualité ou l'analyse statique du code.
- Les tests unitaires ne sont pas présents dans tous les modules.