Rapport hebdomadaire semaine 2

Yann DUBOIS

April 16, 2016

Chapter 1

Résumé de la première semaine

Lors de la première semaine, mon tuteur n'étant pas présent, on m'a présenté la mission qui m'était donné et les problèmes qui en découlé. Je n'ai pas vraiment compris sur le moment ce qu'il fallait faire et on m'a donc demandé d'attendre que mon tuteur revienne de son déplacement. J'ai donc passé le reste de ma semaine à programmer des améliorations sur Pillar comme l'ajout de script pouvant être caché (#101>https://github.com/pillar-markup/pillar/issues/101) et aussi l'ajout d'ancre automatique pour les sections (#19>https://github.com/pillar-markup/pillar/issues/19). Ces ajouts m'ont permis de mieux comprendre l'architecture de l'outil, qui est décomposé en phase (Creation -> Parsing -> Transformation -> Exportation) en ayant plus particulièrement travaillé sur la phase de transformation. J'ai terminé cette semaine testant le Mooc de Pharo PharoMooc>https://www.fun-mooc.fr/courses/inria/41010/session01/info et en regardant comment fonctionné sbabook (le premier rapport).

Chapter 2

Semaine 2

2.1 Lundi 11/04/2016

Damien (mon tuteur) est revenu de son déplacement et m'a expliqué plus clairement ce qu'il attendait de nous. La principale problématique est de simplifier Pillar, et plus particulièrement de le désolidariser de Mustache, son système de template actuel. La deuxième tache est de supprimer les dépendances cyclique, en utilisant la console sur Jenkins et un outils présent dans Pharo 5. Il faut aussi faire un exporter vers Scenari, pour le Mooc de l'université. Et enfin, il faut réfactorer les méthodes de tests, il y en a beaucoup trop qui se ressemble. Le reste de la journée, j'ai travaillé en pair programming avec Thibault ARLOING pour concevoir un Makefile dans l'optique de supprimer la dépendance entre Mustache et Pillar.

2.2 Mardi 12/04/2016

J'ai commencé la journée en terminant la partie du Mooc qui m'était attribué, j'ai trouvé quelques coquilles, j'ai donc prévenu par mail la responsable du Mooc. Lors de la réflexion sur la désolidariosation de Mustache, on a remarqué que Mustache était un moteur de template écrit en Ruby à la base, et qu'il pouvait donc être lancé par une commande. Mustache prend en entré un fichier YAML et un fichier de template. Notre première idée a alors été de créé un script bash pour transformer le résultat d'un fichier .pillar vers un fichier .yaml. Mais il se trouve que l'idée du script n'était pas viable et beaucoup trop compliqué à mettre en place, toujours dans l'optique de simplifier

Pillar.

2.3 Mercredi 13/04/2016

Nous avons, avec Thibault, désolidarisé Mustache de Pillar, cette fois, plus une seule ligne de code ne faisait référence à Mustache à l'intéieur du logiciel. Ou du moins c'est ce qu'on pensé, au moment ou nous avons supprimer les méthodes, le système n'a pas compris qu'on les avait enlevé, et nous a lancé plein d'exceptions, et bloqué complétement l'accès à des packages. Le bug a été remonté mais est difficile à reproduire car il arrive de manière plus ou moins aléatoire. On a donc recommencé depuis zéro et de nouveau supprimer la dépendance à Mustache. Une fois fait, la journée était bientôt terminé, on a donc réfléchie à une façon de générer uniquement un fichier YAML à partir de Pillar. Il suffit de "yamlizer" le contenu juste avant que le fichier soit exporter.

2.4 Jeudi 14/04/2016

J'ai codé la fonction qui "yamlize" un fichier, et est cherché la façon la plus jusdicieuse de l'ajouter dans l'exportation. Ce changement à causé énormément de problème sur les tests unitaires, je les ai donc mis à jour directement pour éviter qu'à la fin du prototypage, il y est une centaine voir plus de test à retravailler. Damien nous a fait remarqué qu'il pouvait y avoir des morceaux de configurations dans les fichier .pillar. Il fallait donc trouvé où se faisait la fusion entre la configuration principale et celle du fichier. A notre surprise, c'était le système de template qui s'en occupé, celui que nous avons supprimé. La fin de la journée nous a permis de réfléchir à un moyen de faire une fusion propre en récupérant uniquement les données importantes pour un fichier.

2.5 Vendredi 15/04/2016

Stéphane DUCASSE est revenu de vacance et est venu nous voir pour faire un état de nos avancements. Il semble être content de nos avancé mais n'aime pas l'idée de se rendre dépendant d'un moteur de template qui n'est plus maintenue, c'est à dire Mustache en Ruby. Il aimerait qu'on puisse utiliser

uniquement les fonctionnalité de Pharo quite à devoir avoir une couche intermédiaire entre Pillar et Mustache qui s'occuperai de faire les transformations que l'on gère pour le moment avec le Makefile. On a quand même continué à travailler sur notre prototype dans l'espoir de pouvoir le rendre plus modulaire, de façon à ce que si un jour on veuille utiliser un autre moteur de template, le changement ne soit pas difficile à faire. J'ai donc créé une méthode de fusion entre les configurations, en la testant je me suis rendu compte que l'ont récupérer des informations qui n'était pas nécessaire, j'ai donc ajouté une description pour les metadata qui me renvoie une sous configuration qui elle même contient un dictionnaire de valeur. C'est ce dictionnaire que j'utilise pour récupérer uniquement les informations importantes. On a donc testé la solution sur un petit projet, et on remarque que notre solution est élégante et fonctionne bien. Il reste à l'adapté à ce que Stéphane veut.