Rapport d'organisation

Chakib Benkebir
June Benvegnu-Sallou
Antoine Ferey
Emmanuel Loisance
Christophe Planchais
Youssouf Roudani

https://github.com/Kwodhan/SynthLabC

Présentation de l'équipe

Pour ce projet de développement en mode agile, qui met en valeur des notions telles que transparence, inspection ou adaptation, le cadre Scrum nous propose les rôles suivant : Product Owner, Scrum Master et bien-sûr l'équipe de développement.

Les rôles étaient distribués de la façon suivante :

- Product Owner : Noël Plouzeau
- Scrum Master : Jildaz Layec
- Équipe de développeurs : Antoine, Chakib, Christophe, Emmanuel, June et Youssouf

Scrum

Nous étions 6 développeurs, ce qui représente la taille idéale pour une équipe de développement *Scrum* (min: 3, max: 9).

Quant au Scrum Master (M.Layec), il a été le coach de notre équipe. Son rôle était de mettre en route et d'animer le développement de SYNTHLABC. Pour avoir une vision d'ensemble, le Scrum Master animait des Daily meetings. Les Daily étaient effectués le midi pendant 15 minutes. Chaque Développeur expliquait ce qu'il avait fait depuis le dernier daily et les tâches dont il allait s'occuper après le daily.

Il faut mentionner la durée courte des sprints (1 semaine) car le temps alloué pour ce projet était seulement de 3 semaines. À la fin de chaque sprint, avait lieu une rétrospective. Cette réunion permettait de mettre en évidence les points positifs et négatifs du sprint passé. Après un vote de l'équipe pour sélectionner les 3 points négatifs les plus critiques, nous cherchions des solutions afin de les résoudre au sprint suivant.

Sprint 1 : La mise en place et la découverte

Nous avons sélectionné les User Story à embarquer pour le sprint 1 en fonction des objectifs donnés par PO. Vue que nous connaissions mal le domaine métier, nous n'avons pas hésité à chiffrer fortement les US en rapport au domaine métier et à attribuer 8 points de complexité à la catégorie "Risque". Chaque point de complexité représente 2 heures de travail d'un développeur.

Backlog

US	**Complexité**
Création de Connexion	8
Création de module de sortie	20
Création de module VCO type A	20
Création de plan de montage global	8
Manipulation de signaux	3
Evaluation du travail par un rapport	2

Capacité

		30/1	31/1	1/2	2/2	3/2	4/2	5/2
June		0	3	2	4	0	0	3
Emmanuel		0	3	2	4	0	0	3
Antoine		0	3	2	4	0	0	3
Christophe		0	3	2	4	0	0	3
Youssouf		0	3	2	4	0	0	3

Chakib		0	3	2	4	0	0	1
Fait	0	0	18	12	24	0	0	16
RAF	70	70	52	40	16	16	16	0

8	Risque
0	Suivi
0	Intégration continue et gestion configuration
1	Revue
61	Capacité de réalisation
0	Non alloué

Développement

Le code qui a été produit remplissait les objectifs du sprint 1. Cependant, le code n'étant pas optimisé pour les futures modifications et US à implémenter, nous avons décidé de passer par une phase de refactoring. A partir de ce code propre, les US suivantes ont été plus aisées à réaliser. Nous avons préféré perdre un peu de temps avec cette phase pour gagner en efficacité et qualité de code par la suite.

Outil de suivi

Nous avons commencé notre suivi *agile* avec le support de *Taiga*. Ce dernier ne remplissait pas nos attentes :

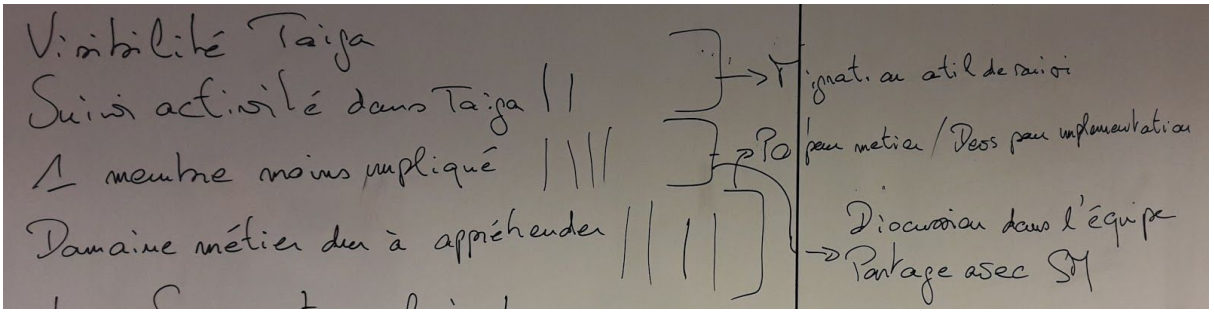
- Pas de possibilité de renseigner son temps passé sur chaque tâche.
- Difficile de réaliser un burndown chart.
- Peu de visibilité agile.

La saisie du temps réalisé s'est faite à l'aide d'un fichier tabulé (Excel), ce qui a été peu pratique. Nous avons décidé lors de la rétrospective de nous intéresser à un autre outil plus complet.

Rétrospective

Il y a eu, à la fin du Sprint 1, une peur qu'un des membres se sente moins impliqué par le projet. Une décision d'en parler dès le début du sprint 2 pour éviter que cela ne

devienne un problème fut prise. Après discussion et une meilleur répartition des tâches, il n'y eut plus de problème pour les sprints suivants.



Sprint 2 : Appropriation

Bien que nous ayons validé l'objectif du sprint 1, les US en rapport n'étaient pas totalement implémentées. Notre première tâche en début de sprint a été de finir les US en cours (soulignées dans la tableau de Backlog). Nous nous sommes ensuite attaqués aux US liées à l'objectif du sprint 2.

Backlog

US	Complexité
Création de connexions (Sprint 1)	2
Création de module VCO type A (Sprint 1)	2
Création de plan de montage global (Sprint 1)	8
Manipulation de signaux (Sprint 1)	3
Création de module VCA	13
Création de module EG	5
Création de module oscilloscope	13
Création de module réplicateur	3

Capacité

		5/2	6/2	7/2	8/2	9/2	10/2	11/2	12/2

June		0	4	4	4	2	0	0	3
Emmanuel		0	4	4	4	4	0	0	3
Antoine		0	4	4	4	4	0	0	3
Christophe		0	4	4	4	4	0	0	3
Youssouf		0	2	4	2	4	0	0	3
Chakib		0	4	4	4	4	0	0	3
Fait	0	0	22	24	22	22	0	0	18
RAF	108	108	86	62	40	18	18	18	0

8	Risque
2	Suivi
2	Intégration continue et gestion configuration
3	Migration outils de suivi
8	Revue
61	Capacité de réalisation
0	Non alloué

Même si la capacité théorique était de 108, nous avons évalué les US avec une capacité de 83. En effet, cette capacité réduite a été calculée à partir de notre vélocité du sprint 1.

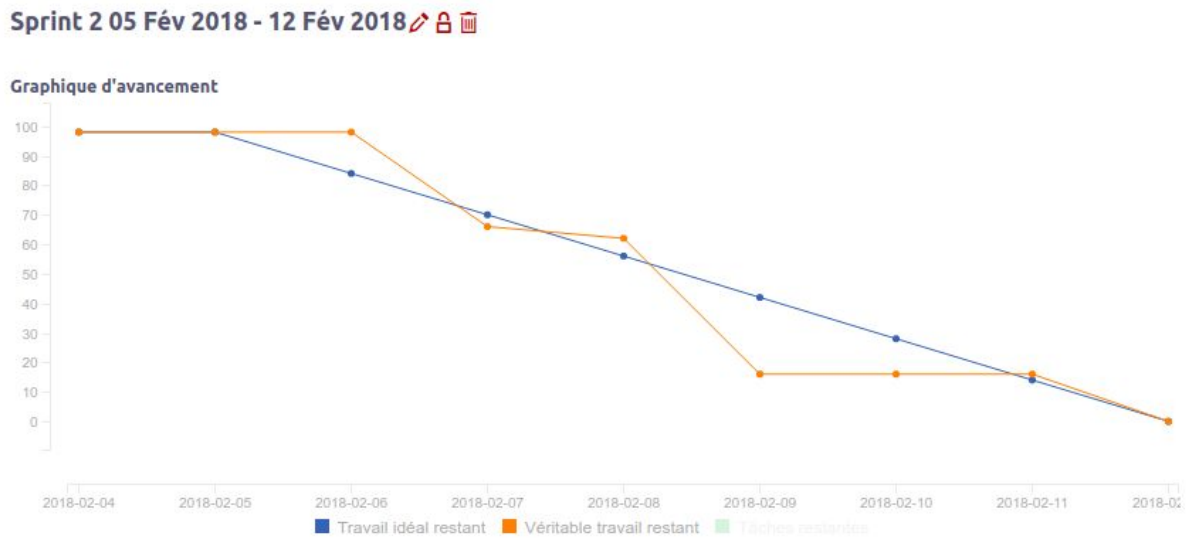
Développement

Le groupe était plus à l'aise au niveau métier et au niveau architecture (grâce au refactoring). Nous avons pu finir le développement des US le vendredi. Toute la journée du lundi fut consacrée à des tests IHM et à de la revue de code.

Malgré les fluctuations dans la présence des membres de l'équipe (entretiens, maladie, etc), nous n'avons pas ressenti d'impact au niveau charge de travail. Nous avions donc bien anticipé lors du Sprint Planning.

Outil de suivi

Nous avons commencé par tester l'outil *Jira*, seulement ce dernier n'était gratuit que pour une période de 7 jours. Certains développeurs connaissaient l'outil de suivi Redmine, nous avons donc opté pour la solution de EasyRedmine qui permet d'utiliser l'outil hébergé dans le cloud, gratuitement pendant 30 jours ce qui correspondait à notre besoin pour le projet.



Un burndown chart (graphique d'avancement) est une représentation graphique du travail restant sur une période de temps donnée. Il fait parti des techniques de management, et permet de prédire la date de fin du sprint. Il est utilisé avec Scrum car donne une interprétation de l'efficacité de l'équipe.

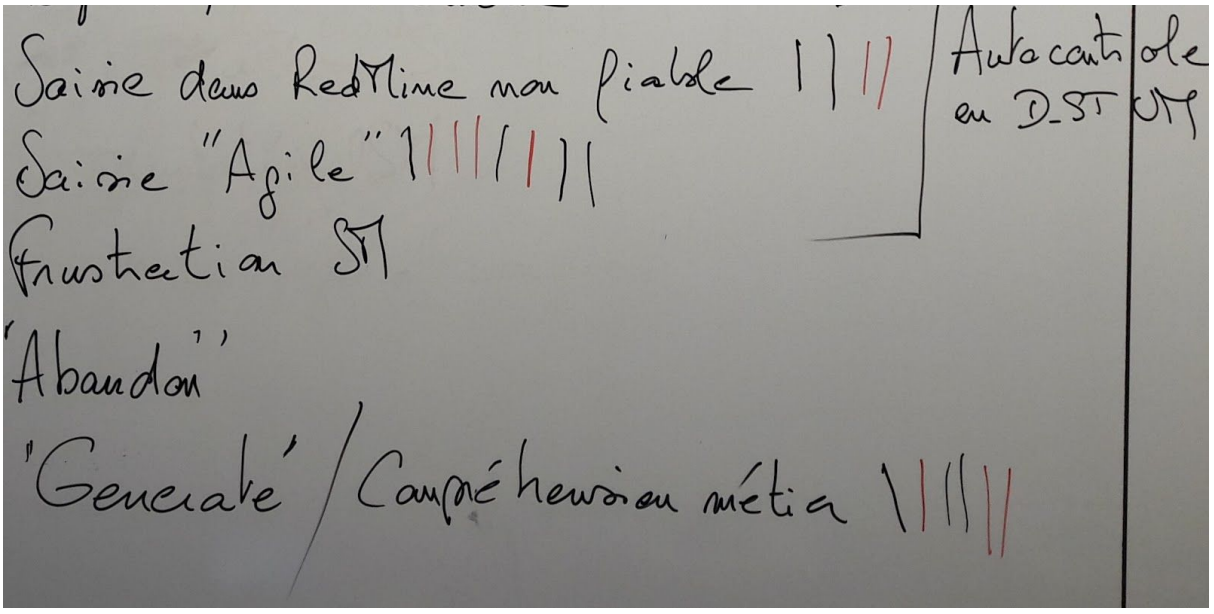
On remarque que la courbe orange représentant notre avancée réelle est bien en-dessous de celle théorique pour le jour du vendredi (09/02).

Rétrospective

L'objectif de sprint a été validé par le Product Owner. En manipulant notre application pendant la rétrospective, le PO a pu valider l'objectif et nous faire des retours sur les US du sprint.

Les points à rectifier suite à la rétrospective :

- Changer l'alignement des labels
- Valeur initiale de l'attaque de l'EG non alignée avec la valeur du slider
- VCO doit avoir une sortie à 5 V



Sprint 3 : La dernière ligne droite

Les US en rapport avec le domaine métier ont des points relativement faibles par rapport aux sprints précédents. La User Story *Sauvegarde et restauration de configuration* nous angoissait le plus car nous n'avions pas prévu notre architecture pour cette fonctionnalité.

Backlog

US	**Complexité**
Création de boucles	1
Création de module VCF type LP	8
Création de module mixer	7,5
Création de module séquenceur	8
Création de module VCF type HP	2,5
Changement d'aspect IHM (#SKIN)	0
Clavier de commande (#KEYB)	1
Sauvegarde du son produit dans un fichier	4
Sauvegarde et restauration de configuration	40
Création de module bruit blanc	1

Capacité

		12/2	13/2	14/2	15/2	16/2	17/2	18/2	19/2
June		0	4	4	4	4	0	0	3
Emmanuel		0	4	4	4	4	0	0	3
Antoine		0	4	4	4	4	0	0	3
Christophe		0	4	4	4	4	0	0	3
Youssouf		0	4	4	4	4	0	0	3
Chakib		0	4	4	4	4	0	0	3
Fait	0	0	24	24	24	24	0	0	18
RAF	114	114	90	66	42	18	18	18	0

8	Risque
3	Suivi
2	Intégration continue et gestion configuration
0	Revue
73	Capacité de réalisation
0	Non alloué

Même si la capacité théorique était de 114, nous avons évalué les US avec une capacité de 86. En effet, cette capacité réduite a été calculée à partir de notre vélocité des sprints 1 et 2.

Développement

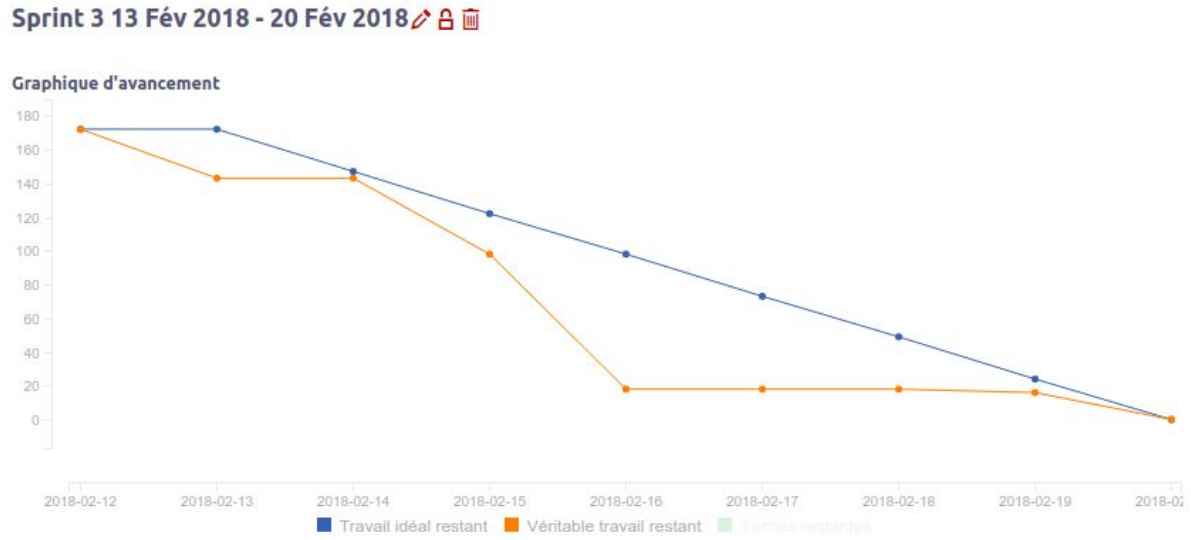
Le design de départ de SYNTHLABC n'a pas été prévu pour la sauvegarde des configurations de nos circuits audios. Ce défaut d'architecture fût la dernière difficulté à surmonter. Nous avions estimé cette tâche à 80 heures. Cependant, en 48 heures elle fût

terminée. Comme le Sprint 2, nous avons fini le développement le vendredi et pu tester un maximum de cas sur le temps restant du sprint.

Outil de suivi

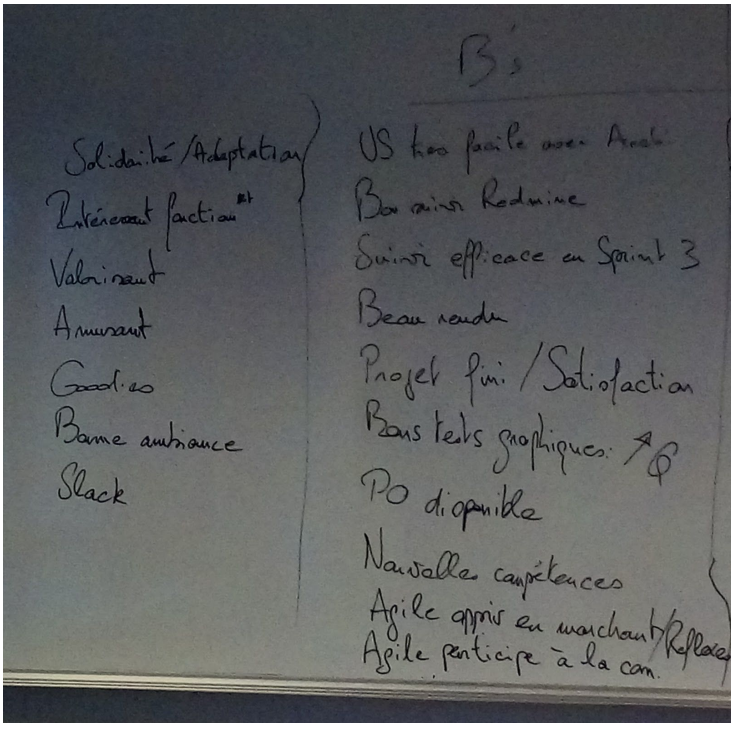
Avec EasyRedmine, la saisie du temps est devenue régulière. Notre équipe a gagné en expérience sur les outils utilisés.

Comme on peut le voir sur le graphique, notre avancée par rapport à la courbe théorique bleue (cf. graphique) a été plus importante durant ce sprint, notamment le vendredi (16/02) où l'écart est le plus important. Il restait quelques tâches non fermées qui correspondaient à l'objectif académique d'évaluation. Le tâches relatives au fonctionnement de l'application étaient toutes closes.

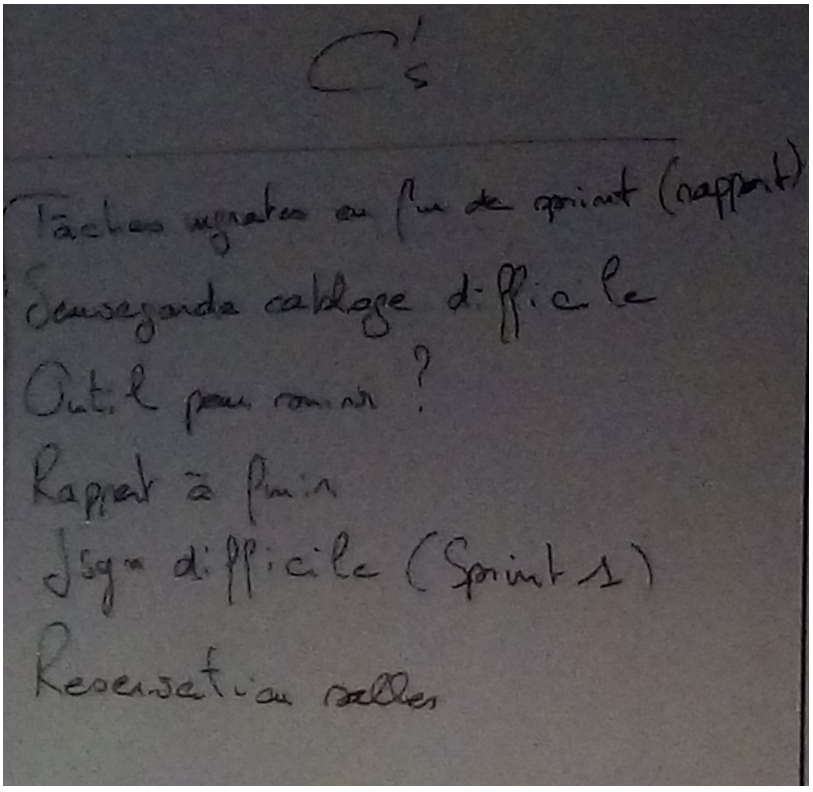


Rétrospective

L'objectif du sprint 3 a été validé par le PO. Nous avons profité de cette rétrospective pour faire le bilan de ce dernier sprint et également du projet en général.



Les points positifs du projet



Les points négatifs du projet

Organisation

Git

Nous avons utilisé au maximum le potentiel des branches de Git. Tous nos développements se sont réalisés sur la branche *dev*. Dès qu'il y avait une fonctionnalité cruciale ou un refactoring, nous n'avons pas hésité à créer d'autres branches. Cela nous a permis d'avoir toujours une application fonctionnelle sur notre branche *dev*.

Slack

Slack a constitué un outil privilégié pour communiquer rapidement avec le PO. Nous avons pu poser nos questions et obtenir des réponses dans de brefs délais. Des prises de rendez-vous ont également été réalisées via l'outil. Il a en somme favorisé les pratiques agiles durant le projet.

Revue de code

Pour chaque tâche réalisée par un développeur, un autre développeur se chargeait de tester la réalisation de la tâche en manipulant directement l'application. Une période d'écriture et lancement de tests IHM avec TestFX venait compléter ce processus de validation. De plus, la production de code s'est généralement faite en pair programming (deux développeurs l'un à côté de l'autre sur un même ordinateur).

Lors de moment critique, comme la prise de décision sur des points de l'architecture, l'équipe de développeurs se réunissait. La situation était exposée à l'oral et/ou au tableau puis nous débattions pour parvenir à faire un choix qui serait adopté pour la suite du déroulement du projet.

Lorsqu'un développeur avait des difficultés sur une tâche et le faisait savoir, un autre développeur moins chargé ou qui avait fini ses tâches en cours venait en aide au premier pour ne pas que tâche reste bloquée. De cette façon, nous évitons une perte de temps inutile et l'apparition d'un risque au niveau de la réalisation des US et de l'objectif du sprint.

Redmine

Redmine est un outil de gestion de projet gratuit et open source. Il permet la gestion des utilisateurs (droits d'accès par exemple) et permet à ceux-ci de gérer leurs projets en y associant tâches et sous-tâches.

Les statuts suivants de progressions des tâches sont supportés : *new, in progress, user testing* et *done.* Redmine propose un suivi du temps, permettant de calculer la vélocité de l'équipe et d'anticiper les éventuels retards.

Conclusion

L’équipe, qui était constituée à la base de deux groupes de personnes ne travaillant pas ensemble habituellement, a réussi à collaborer pour réaliser un projet abouti grâce aux efforts mis en place lors du premier sprint.

Au fur et à mesure des sprints, nous avons amélioré notre gestion de projet et nous sommes appropriés les techniques de la méthode SCRUM. Au final, une application fonctionnelle et tous les objectifs fixés par le PO ont été atteints.