IUT Sénart-Fontainebleau, UPEC
Méthodo
IUT Info 2A

Florent Madelaine

2023-24

Le projet agile et son évaluation

Travail en équipe

Vous allez être entre 4 et 5 par groupe. Pour rendre l'exercice plus intéressant, j'ai fabriqué les groupes pour que vous ne soyez pas forcément avec vos amis et que vous sortez de votre zone de confort.

Les groupes.

- groupe 1. Mme BOUSSON Héloïse, M. CHOUX-BEAUREGARD Terence, M. DUFOUR Kohsey, M. ILANCHEJIAN Shiva, M. SEGUY Matthieu.
- groupe 2. M. DRIYEJ Driss, M. FAUVET Matthis, M. GÉNIQUE Florian, M. LELONG Maxime, M. SAMY Florian.
- M. MORIN Lionel, M. PIETROIS Axel, M. SERAZIN Axel, M. VALLAT Guillaume.

Date de rendu

Il n'y a pas vraiment de date de rendu. Vous devez dès aujourd'hui créer un dépôt git intitulé ProjetAgileBUT2FA2324GR1 si vous êtes le groupe 1, et m'ajouter les droits en lecture. Tout travail du groupe doit être reflété dans le git. Le projet s'arrête à la fin des cours au bout de 3 semaines.

Consignes

De manière générale tous les projets que je vous donne doivent suivre les excellentes consignes de Luc Hernandez disponibles ici : `https://www.iut-fbleau.fr/sitebp/doc_consignes/`.

Le prétexte

Le prétexte permettant de mettre en oeuvre les méthodes agiles consiste à coder une application inspirée de *Boulder Dash*[1]. Ici le bonhomme est un lapin qui creuse des terriers, mange des carottes et cherche la sortie.

Comme nous avons fait pas mal de travail sur les aspects IHM en DEV, l'accent portera moins voir pas du tout sur cet aspect.

Il faut essentiellement mettre en oeuvre un boulder dash minimal pour pouvoir visualiser ce que fait un bot qui communiquera avec le jeu via une API.

Le prétexte est important au sens où il faut avoir un travail réel à faire pour vraiment essayer de mettre en pratique des méthodes de travail en groupe. La qualité de ce qui est livré et la méthode collective pour arriver à ce résultat sera plus important que la quantité de travail réalisé.

Il n'y aura normalement pas besoin de travailler en dehors des 3 après-midi de cours à ce projet. Par contre, il faudra travailler exclusivement à ce projet et en groupe pour ces trois après-midi.

Le client

Je joue le rôle du client. Nous ne ferons pas de cahier des charges à proprement parler car vous avez ou allez avoir une assez bonnes expériences à ce sujet dans d'autres cours et avec le projet tuteuré.

1. voir `https://en.wikipedia.org/wiki/Boulder_Dash_(video_game)`.

Le language

Je veux du java pour que tous les groupes travaillent avec un même langage objet et puisse avoir une API.

Ce qui sera évalué

En passant, le code produit va jouer sur la note mais les aspects méthodologiques vont jouer pour une très grande part. En particulier, si le produit demandé n'est pas complètement terminé, ce n'est pas un soucis, tant que vous avez joué le jeu de travailler sur le projet avec une méthode agile adaptée de la méthode scrum.

On attend évidemment du code bien écrit, commenté et accompagné d'une documentation raisonnable (par exemple produite avec javadoc). Pour être considéré comme un incrément, tout code correspondant à une *user story* doit être accompagné autant que possible de tests, soit de tests unitaires réalisés avec junit, soit de description de tests manuels pour l'interface graphique.

Si vous le souhaitez, vous pouvez tenter de partir sur une approche d'*extreme programming* et faire du *test driven*.

Si vous souhaitez faire des diagrammes de classes ou tout autre diagramme UML, c'est votre choix en tant qu'équipe de dev. En tant que client, je n'ai pas besoin de voir ces diagrammes. En tant qu'évaluateur de votre travail non plus. Si cela vous aide pendant un TP que je lise/discute de tels diagrammes avec vous, je le ferais avec plaisir.

Par rapport à l'organisation scrum, je vais vous noter en partie sur ce que j'observe pendant les TPs. Je vais faire une brève soutenance informelle pour chaque groupe en fin de période. Je vais avoir besoin d'un petit rapport de 3 ou 4 pages expliquant les aspects méthodologiques de l'équipe, ce qui a marché/ pas marché etc. Il s'agit essentiellement de ce que l'équipe doit produire au moment de la *sprint retrospective*.

Détails sur le jeu

Le jeu fonctionne avec des cases qui contiennent soit du vide, soit de la terre, soit un rocher, soit une carotte, soit un navet. Il y a aussi une case avec une porte et notre héros qui est un lapin.

- Le lapin peut enlever la terre en se déplaçant.
- Le lapin peut manger une carotte et gagner un point.
- Le lapin peut manger un navet et perdre un point.
- Le lapin peut aussi ne pas bouger.
- Un rocher avec du vide en dessous tombe d'une case vers le bas (les rochers tombent tous à la même vitesse).
- Quand le lapin arrive sur la porte le jeu se termine.

Lien avec le client : product Backlog et product owner

Normalement c'est le travail du *product owner* de gérer avec le client le *product backlog*. Vu le temps que nous avons, nous allons tricher et je vous donne cette partie (voir Figure 1 pour les détails).

Le *product owner* sera responsable de travailler un peu plus sur le *sprint backlog* que les autres, et sera le canal dédiée pour que j'explique ce que je veux à l'équipe.

L'API

Le bot peut demander le contenu des cases à distance 2 de sa position. Le bot peut se déplacer sur une case à distance 1. Parfois ce n'est pas possible et le moteur de jeu doit indiquer si le déplacement a eu lieu.

Le visualisateur permet de montrer l'évolution du jeu à une distance variable qu'on peut choisir (niveau de zoom) autour du lapin.

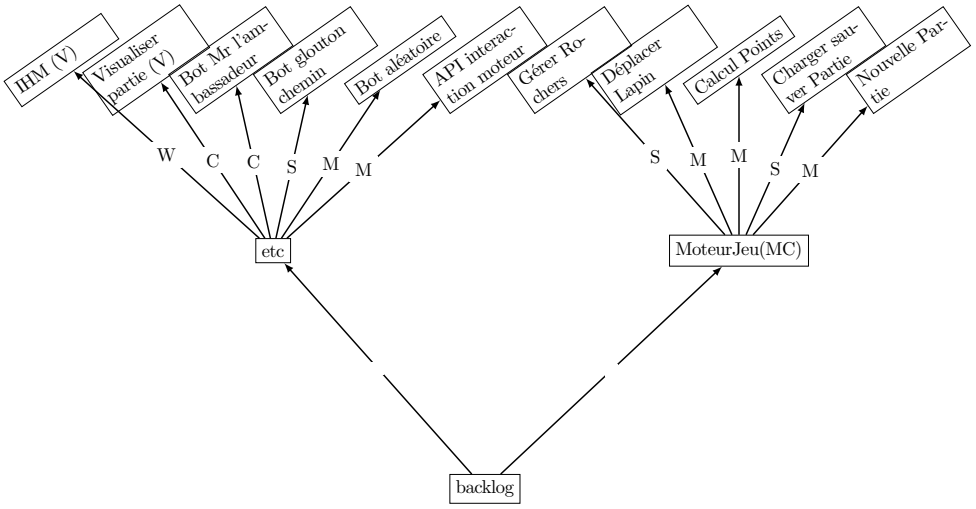

FIGURE 1 – Esquisse du product backlog : les feuilles de l'arbre correspondent à des *use case* auxquels j'ai ajouté une priorité avec la notation MoSCoW. Dans certains cas j'ajoute entre parenthèse une lettre parmi MVC, en référence au découpage d'une application selon ce modèle.

Consortium pour la normalisation des bots

Certains membres de votre groupe vont être amenés à participer à un effort de normalisation des bots et des moteurs de jeu.

Il s'agira dans un premier temps de définir une API pour les bots.

Le Scrum Master

C'est le membre du groupe qui a minima va lire la doc de 20 pages expliquant la méthode scrum et essayer de la digérer pour faciliter sa mise en oeuvre par l'équipe. C'est lui qui va essayer de préparer et d'animer la *sprint retrospective*.

Qui fait quoi ?

C'est l'équipe qui s'organise. Le *scrum spring board* (physique avec des post-its ou virtuelle avec un outil comme trello voir un simple fichier tableur partagé) sert à s'organiser. Dans notre cas, le scrum master et le product owner codent aussi.

Les sprints

Nous ne pouvons pas suivre le rythme classique d'un sprint. Comme c'est intéressant de simuler la fin d'un sprint, je propose l'organisation suivante.

- Semaine de cours 1 : sprint 0 et début du sprint 1.
- Semaine de cours 2 : sprint 1.
- Semaine de cours 3 : sprint 2.

Etc

Je recommande fortement d'utiliser le *pair programming*[2].

Le client change d'avis

Si le client change d'avis, c'est plutôt normal. Le produit est pour lui *in fine* et il faut adapter le backlog à ses demandes. Le *product owner* doit toutefois protéger son équipe en donnant des délais raisonnables au client, selon ce que l'équipe peut réaliser. Le *scrum master* est là pour expliquer et trouver des solutions pour remotiver les membres de l'équipe. Par exemple, lorsque un membre d'équipe vient de travailler sur une partie qui devient obsolète.

2. à ne pas confondre avec *peer* ou *pear*.