Créé par Alban Catry alban.catry@gmail.com
De 3 à 6 joueurs
Durée 90 minutes

Découverte de Scrum

Guide de l'organisateur

Vous allez faire découvrir quelques principes de bases de l'agilité et de Scrum à un petit groupe de joueurs. En particulier, voici les messages-clés qui vont être passés dans ce jeu :

- L'organisation d'un sprint et ses cérémonies : planification, démonstration et rétrospective.
- Le triangle de l'agilité, notamment, éviter, même en période de stress, d'être tentés de jouer sur les couts et les délais.
- L'importance de livrer régulièrement, de maitriser la dette technique, de satisfaire le client.
- La découverte de quelques pratiques d'ingénierie du logiciel : intégration continue, pair programming, …
- Le concept de MVP (Minimum Viable Product).

Il y a bien d'autres choses dans l'agilité, et il a fallu faire des compromis pour que le jeu reste simple. Ce jeu n'est donc pas du tout complet, et parfois approximatif, mais c'est déjà pas mal. Il faudra compléter avec ce qui n'est pas du tout abordé dans ce jeu (les daily scrums, les méthodes d'estimation, les rôles et responsabilités, le management visuel, …). **Ce jeu peut être la première étape d'une formation sur les méthodes agiles et sur Scrum, car il n'y a aucun prérequis, même si quelques connaissances en informatique peuvent aider et améliorer l'expérience de jeu.**

Sommaire

Planning

Positionner un pion sur le sprint 1
Positionner un pion sur l'étape « Planification »
Mélanger les cartes Évènements et les placer ici

Cérémonies

Mélanger les cartes Idées sans explication et les placer ici

Scrum Master

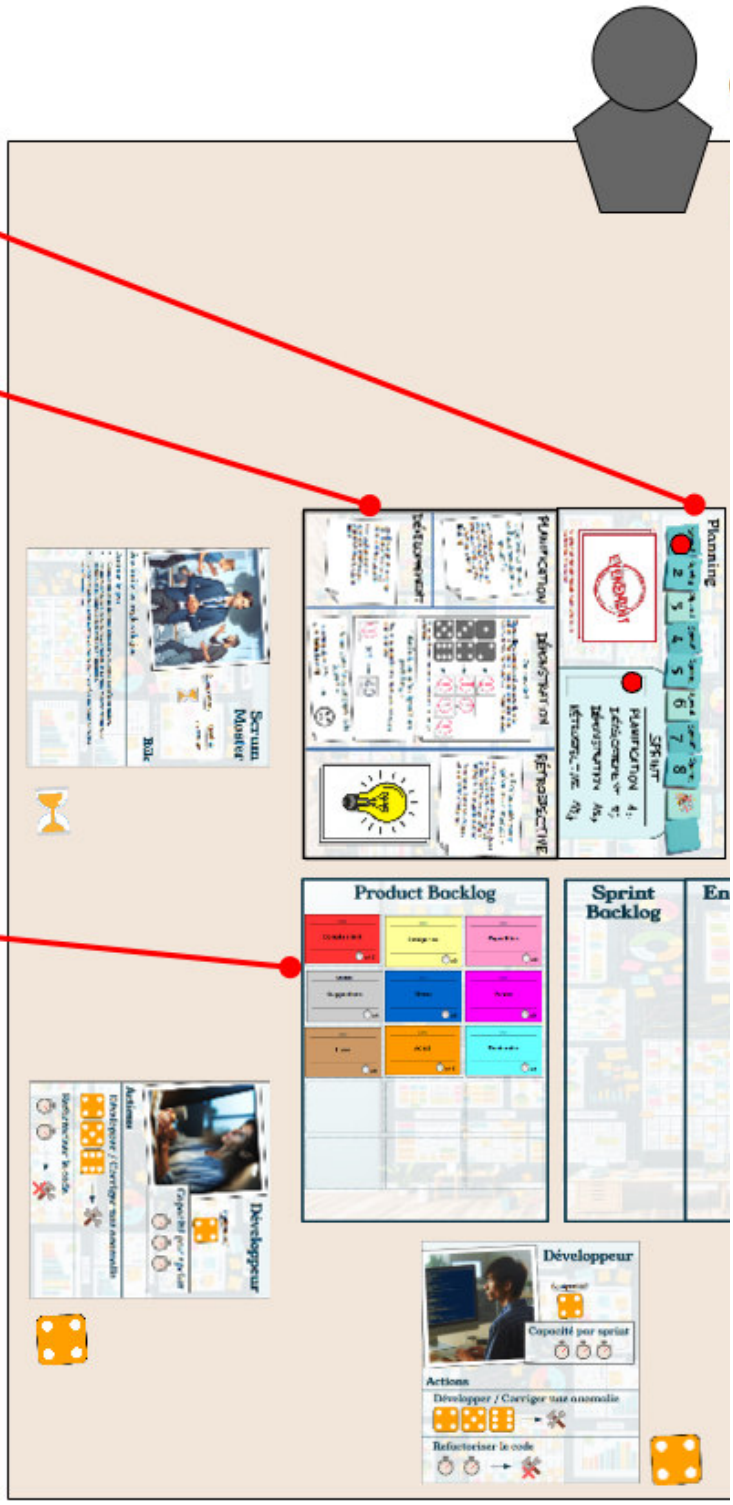
Fiche de personnage
Sablier

Product Backlog

Mettre les 9 Épic initiales (toutes sauf Évaluations), chaque tas étant constitué de la carte Épic et de ses User Stories (de 2 à 7 User Stories).
Par la suite, les joueurs pourront librement déplacer les cartes, voire prendre plus de place sur la table pour avoir toutes les User Stories en visibilité directe.

Développeur

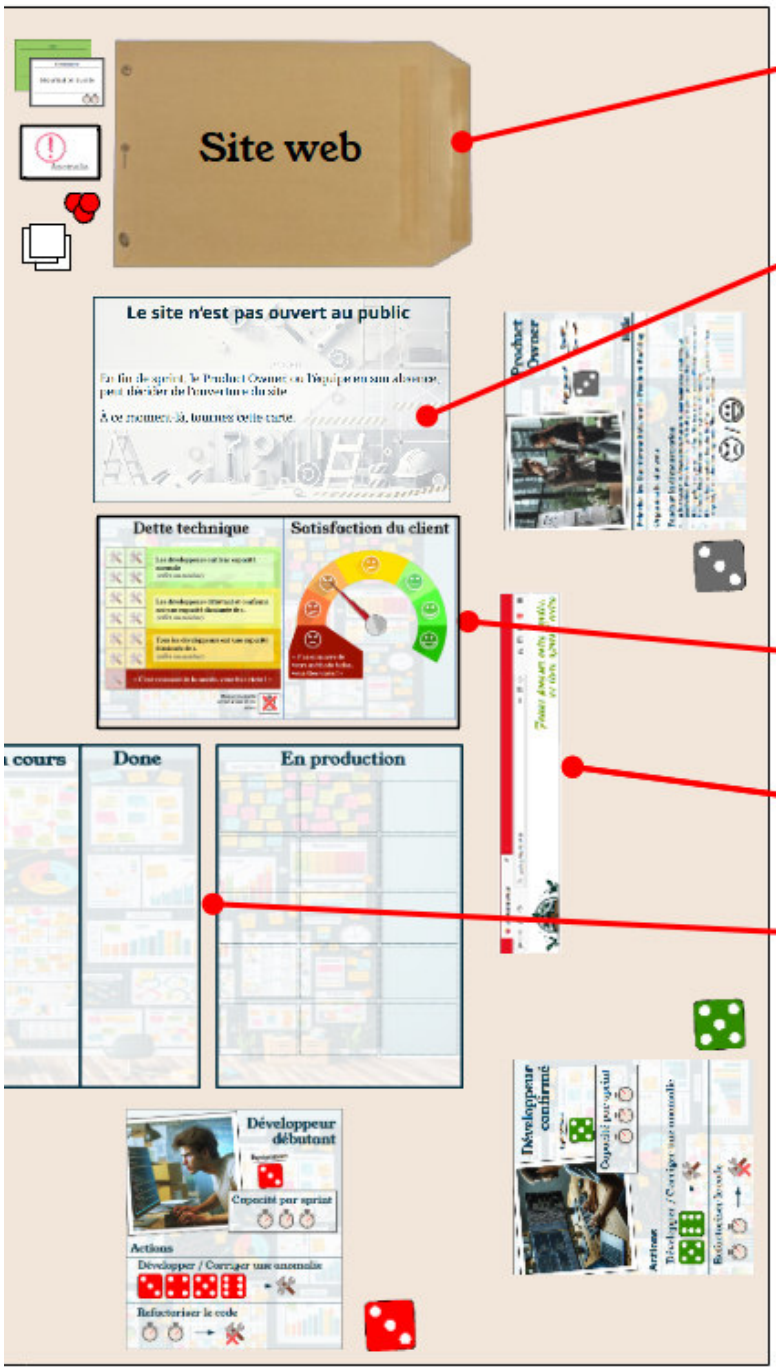
Fiche de personnage
Dé orange



Développeur

Fiche de personnage
Dé orange

Organisateur



Enveloppe contenant le site web
User stories des évènements
Réserve de pions

Ouverture du site
Placer ce plateau sur la face « Le site n'est pas ouvert au public »

Product Owner

Fiche de personnage
Dé noir

Dette technique et satisfaction du client
La dette technique commence à zéro
La satisfaction du client commence à -1 (orange)

Base du site web
(laisser le la place en-dessous pour le construire)

Scrum board
En production

Développeur confirmé

Fiche de personnage
Dé vert

Développeur débutant

Fiche de personnage
Dé rouge

Description des éléments

Si vous avez acheté une version physique de ce jeu, le matériel est déjà présent et packagé. Si vous avez téléchargé le jeu, alors il va vous falloir imprimer les plateaux et cartes, dont certains en recto-verso, et également acquérir quelques objets.

Les plateaux et tous les éléments ont été conçus pour pouvoir être stockés dans une boite carrée de dimensions extérieures 27.5 x 27.5 x 6.3 cm et intérieures 26.6 x 26.6 x 6 cm.

Les plateaux imprimables

- Scrum board
- Product backlog
- En production
- Planning
- Cérémonies
- Dette technique et satisfaction du client

Autres éléments imprimables

- 6 cartes personnages
- Une enveloppe contenant les éléments du site web (17 éléments de diverses tailles)
- 7 cartes Evènement
- 9 cartes Idées (sans explication)
- 9 cartes Idées (avec explication)
- 58 cartes Épic ou User Stories
- 15 cartes anomalies
- 3 capacités pour 3 joueurs
- Des pions carrés utilisés comme caches dans diverses situations
- Des feuilles de joueur à photocopier et à donner à chaque joueur
- Un livret de règles du jeu

Matériel non-imprimable

- 13 pions de dette technique
- 5 dés : 1 vert, 1 rouge, 1 noir, 2 orange
- Un sablier de 5 minutes
- 2 pions : un pour désigner le sprint sur le plateau planning, un pour la trésorerie une fois le site en ligne

Mise en place

La page précédent montre un exemple de disposition pour l'espace de jeu, mais rien n'est obligatoire. À partir de 5 joueurs, vous pouvez même laisser au Scrum Master le soin de tout organiser, puisque c'est son rôle !

Voici les spécificités selon le nombre de joueurs :

3 joueurs	**Un développeur débutant, un développeur, un développeur confirmé** Mettre les caches spécifiques sur la capacité de chaque développeur pour faire passer leur capacité à 4 au lieu de 3.
4 joueurs	**Un développeur débutant, deux développeurs, un développeur confirmé**
5 joueurs	**4 développeurs, un Scrum Master.** À partir de 5 joueurs, le Scrum Master a pour mission d'animer le jeu (avec votre soutien pour les points de règles qu'il ne connaît pas). De plus, il a un sablier et est garant que les tours à partir du deuxième ne durent pas plus de 5 minutes.
6 joueurs	**4 développeurs, un Scrum Master, un Product Owner.** Lorsque le Product Owner est là, il gère le dé des anomalies, le site web et la priorisation du backlog. En son absence, ces rôles sont diffus parmi les développeurs.

Comment introduire le jeu ?

Commencez par distribuer les feuilles d'Onboarding à chaque joueur.

Si vous disposez d'un ordinateur et d'un projecteur, vous pouvez introduire le jeu avec la présentation « Onboarding.odp ».

Sinon, vous pouvez introduire le jeu à partie des feuilles Onboarding et du matériel présent sur la table.

- **Expliquez le contexte** : vous êtes une petite équipe qui va développer le site web Permalivre selon les principes de l'agilité.
- **Décrivez l'équipe** : invitez les participants à se présenter (prénom et le rôle qu'ils jouent) et à ce que chacun note le prénom de ses coéquipiers et fasse un petit portrait d'eux. Cela fait à la fois office d'ice-breaker et de tour de table.
- **Décrivez les grands principes de l'agilité**, sur la base de ce qui est dans l'Onboarding, en particulier le triangle de l'agilité, mais sans trop insister (pour qu'ils soient confrontés à des choix difficiles en cours de partie).
- **Montrez les fonctionnalités à développer** en montrant les Epic du Product Backlog, elles-mêmes décomposées en User Stories.
- **Montrez ensuite comment on développe** : chaque développeur a une capacité, et lance le dé de dette technique. Expliquez la dette technique et le mécanisme de refactorisation.
- **Expliquez que le jeu est décomposé en Sprints**, qu'il y en aura 8, et expliquez le déroulement de chaque sprint par ses cérémonies : planification, développement, démonstration, rétrospective.

Déroulement du jeu

Le jeu va se dérouler en 8 sprints. À la fin des 8 sprints, vous regarderez sur la dernière page des règles du jeu comment calculer le score de l'équipe.

Chaque sprint est décomposé de la manière suivante :

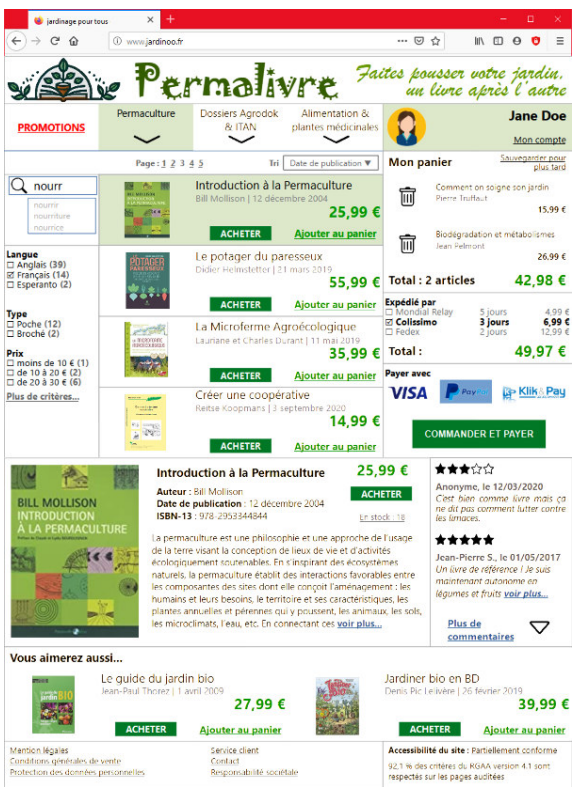
- On tire un évènement et on le résout (sauf au premier sprint)
- Planification
- Développement
- Démonstration
- Rétrospective

Le site à développer

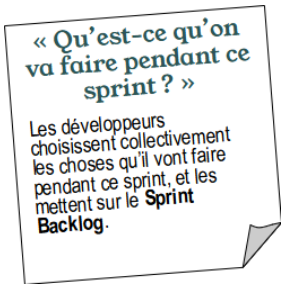
Au début de la partie, sur le plateau de jeu, il n'y a que le bandeau « Base » qui contient l'URL, le logo et le titre du site. Au fur et à mesure que les développeurs vont terminer des Épics et des User Stories isolées, le site va s'enrichir sous forme de puzzle.

Il n'est pas possible de faire le site web en entier, le jeu est fait de telle sorte qu'il est tout juste possible de faire le MVP (voir feuille de score), avec éventuellement quelques ajouts.

Il n'y a pas de bonne ou de mauvaise façon de faire le puzzle. Ce puzzle n'a qu'un rôle cosmétique et immersif, pour donner l'impression qu'on « produit » quelque chose !



Planification

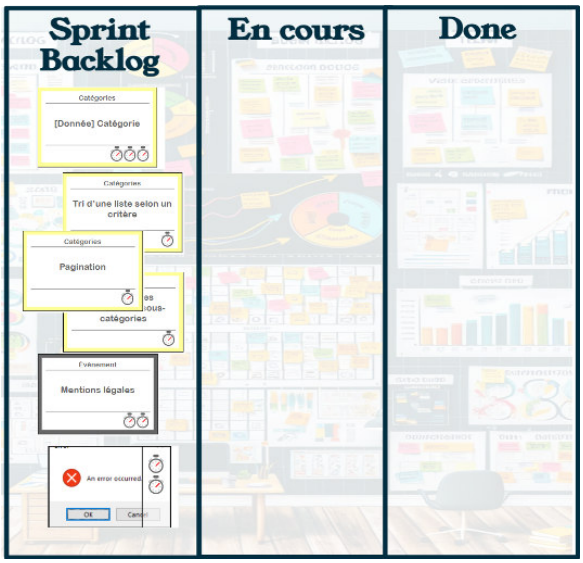


Ces règles sont inspirées du fonctionnement normal de **Scrum** :

- À la fin de cette phase, seul ce qui sera sur le **Sprint Backlog** (user stories + anomalies) pourra être développé pendant la phase de **Développement**. Les développeurs ne pourront pas en rajouter.
- Le **Sprint Backlog** doit être vide à la fin du **Sprint**, sinon il y a une pénalité de satisfaction du client (voir phase **Démonstration**).
- Si le Product Owner est présent, il priorise le **Product Backlog** mais ce sont les développeurs qui choisissent ce qu'ils mettent dans le **Sprint Backlog** (théoriquement, en accord avec les priorités fixées par le Product Owner).

Règles additionnelles pour la fluidité du jeu :

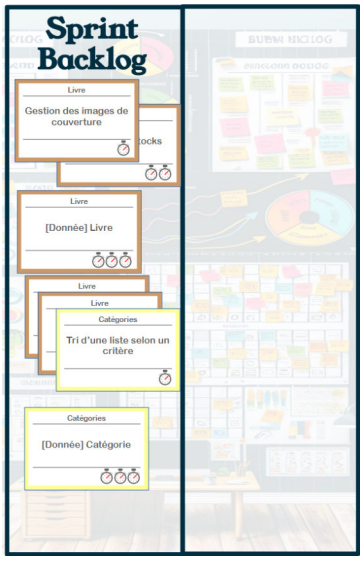
- La refactorisation n'a pas besoin d'être mise sur le **Sprint Backlog**, elle est toujours disponible. Il est donc toujours possible de refactoriser avec de la capacité restante s'il n'y a plus rien à développer.
- S'il est présent, c'est le Scrum Master qui décide de la fin de la phase de planification, sinon, ce sont les développeurs, collectivement.



*Un exemple de **Scrum Board** après la phase de planification. Les développeurs y ont mis pour 10 points d'effort, et prévoient 2 points d'effort dans la refactorisation.*

*Notez que la vélocité de l'équipe est de 12 points d'effort par sprint, il est donc évidemment conseillé de mettre au plus 12 points d'effort sur le **Sprint Backlog**, ou moins si vous souhaitez refactoriser.*

Le premier sprint



Pour le premier sprint, il est recommandé d'imposer ce Sprint backlog. Cela permet d'expliquer les règles au fur et à mesure du sprint et de ses étapes. Le premier sprint sert donc de « tutoriel » et la partie commence au début du deuxième sprint.

Développement

Tour à tour, les développeurs vont dépenser leur capacité pour réaliser des tâches. Chaque développeur a une capacité de 3 points d'effort (4 points d'effort à trois joueurs).

Développer une User Story ou corriger une anomalie

- Le cout en points d'effort est indiqué sur la carte User Story ou Anomalie.
- Le joueur passe la carte sur la colonne **En cours**.
- Il jette le dé pour voir s'il crée de la dette technique.
- Il passe ensuite la carte sur la colonne **Done**.

Refactoriser

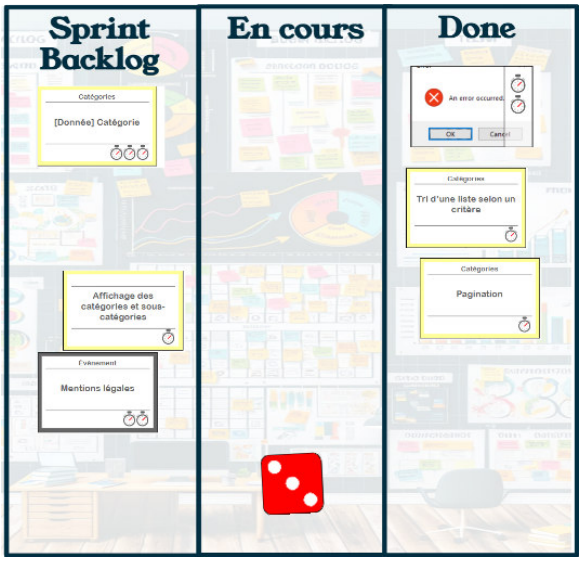
- Le cout en points d'effort est indiqué sur la fiche de personnage (2 en début de partie).
- Le joueur diminue la dette technique.

Quand tous les développeurs ont joué leur tour, on passe à la phase de **Démonstration**.

Exemples

Le premier joueur est un développeur. Avec sa capacité de 3 points d'effort, il fait deux actions :

- *Il corrige une anomalie à 2 points d'effort. Il lance le dé orange et fait 3 : il ne crée pas de dette technique.*
- *Il développe une User Story à 1 point d'effort. Il lance le dé orange et fait 5 : il crée une dette technique. La dette technique augmente donc de 1.*



Le deuxième joueur est un développeur débutant. Avec sa capacité de 3 points d'effort, il fait deux actions :

- *Il développe une User Story à 1 point d'effort. Il lance le dé rouge et fait 3 : comme il est débutant, il voit sur sa fiche de personnage que le 3 lui fait créer une dette technique. La dette technique augmente donc de 1.*
- *Voyant que la dette technique augmente trop, il décide de consacrer ses 2 points d'effort restants à refactoriser le code. C'est une action sans risque qui lui permet de diminuer la dette technique de 1.*

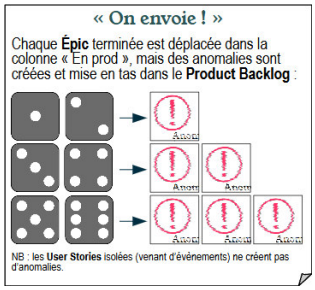
Démonstration

La **Démonstration** est une cérémonie où l'équipe présente ce qui a été réalisé pendant le sprint. Elle y reçoit notamment du feedback des participants. Ici, nous avons décidé d'y intégrer la mise en production et la remontée des anomalies.

0. Les développeurs présentent ce qu'ils ont fait à ce sprint au Product Owner

Vous pouvez scénariser un peu la démonstration en demandant aux développeurs de présenter ce qu'ils ont fait à ce sprint. Cela ajoute à l'aspect formel de la cérémonie.

1. Mise en production



1.1 Création des anomalies

Les Épics terminées et les User Stories isolées (issues d'évènements) sont placées dans la zone **En production**.

Une Épic terminée va générer un tas d'anomalies qui va aller dans le Product Backlog. Ces anomalies seront cachées, c'est-à-dire qu'on ne peut pas encore les corriger (elles n'ont pas encore été trouvées !) et on ne saura pas combien de points d'effort il faudra pour les corriger. C'est le dé noir, lancé par le Product Owner, ou un développeur en son absence, qui détermine le nombre d'anomalies dans le tas (de 0 à 3 anomalies).

1.2 Mise à jour du puzzle

Toutes les Épics et les User Stories isolées correspondent à une pièce de puzzle du site web. Récupérer les pièces correspondant aux Épics et User Stories réalisées et les donner au Product Owner, ou, à défaut, au développeur le plus à droite sur le plan de table.

Nota : le puzzle du site web est purement ludique. Il n'y a pas de bon ou de mauvais assemblage et la forme du site n'a aucun impact sur le score final. Mais il faut bien matérialiser qu'on a produit quelque chose !

1.3 Satisfaction du client

Si au moins une Épic a été terminée à ce tour, la satisfaction du client augmente de 1. Sinon, elle diminue de 1.

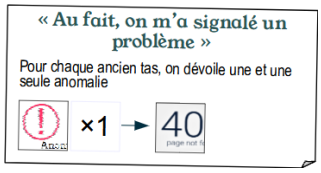
Cette règle n'apparait pas sur le plateau de jeu !

En effet, elle est cachée au début pour les développeurs. Mais l'agilité est vendue comme une méthode itérative et incrémentale. Le client s'attend donc à avoir de nouvelles choses porteuses de valeur à chaque sprint (pas seulement de la réduction de dette technique ou des corrections d'anomalies).
Il faut que vous accompagniez l'augmentation ou la diminution d'un commentaire :

- *Super, je vais pouvoir montrer ça à mes utilisateurs !*
- *Mince, rien de nouveau ? Je croyais qu'on avait de nouvelles choses à chaque fois !*

Ainsi, les développeurs comprendront très vite la mécanique. Ou alors ils sont très bêtes.

2. Révélation d'anomalies

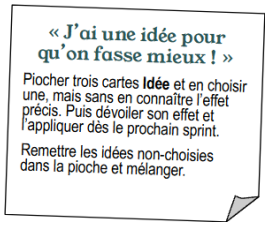


On révèle une anomalie de chaque tas créé aux sprints précédents. On pose l'anomalie, isolée et visible, sur le Product Backlog.

3. Vérification du Sprint Backlog

S'il reste quelque chose sur le Sprint Backlog, le client est mécontent, et la satisfaction diminue d'un cran.

Rétrospective



Le principe général est le suivant :

- Les développeurs piochent trois cartes **Idée**.
- Ils en choisissent une.
- Ils remettent les cartes non-choisies dans la pioche et remélangent.

Le problème, c'est que les développeurs ne connaissent pas l'effet exact de la carte **Idée** qu'ils vont choisir. Tout ce qu'ils ont, c'est un titre et un petit texte humoristique qui peut donner une piste. Cela fait partie du piment du jeu car il y a deux cartes pièges, qui sont en fait de mauvaises idées : le **sprint supplémentaire** et le **développeur en renfort**. Cela fait partie du triangle de l'agilité : c'est le périmètre la variable d'ajustement, pas les délais ou les couts.

L'effet des cartes Idée

Toutes les autres idées ont des effets positifs, qui se manifestent différemment. Lorsqu'une Idée est choisie, l'organisateur sort la carte Idée avec la vraie explication du fonctionnement. Normalement, l'effet est suffisamment bien décrit sur la carte. Mais il faut bien comprendre le système de caches (voir ci-contre).

Les caches

Avec le jeu sont fournis plusieurs pions carrés qui vont faire office de « caches ». Ces caches vont recouvrir des dés, des anomalies, des points d'effort, pour faire évoluer les règles du jeu.

Exemple

*Les développeurs viennent de choisir l'idée « **Intégration continue** », qui permet de créer moins de dette technique. Chaque développeur prend un cache blanc qu'il pose sur le dé de son choix. Ici, le développeur a choisi le 5. Cela fait que, désormais, il ne produira de la dette technique que sur un 4 ou un 6.*

Impacts sur la création de dette technique : intégration continue, tests unitaires automatisés, pair programming

Impacts sur la refactorisation : design patterns

Impacts sur les anomalies : tests fonctionnels automatisés, Product Owner dédié, déploiement continu

Mettre le site en ligne

Comme indiqué sur le plateau « Le site n'est pas ouvert au public », en fin de sprint, le Product Owner ou l'équipe en son absence, peuvent décider d'ouvrir le site au public. Pour cela, ils retournent le plateau. Il faut alors leur expliquer qu'un nouvel indicateur sera suivi : la trésorerie.

Cet acte introduit aussi le concept de MVP : si les fonctionnalités de base pour une expérience client complète ne sont pas présentes, l'entreprise perdra de l'argent.

À partir de ce moment, en fin de sprint, il faudra actualiser la trésorerie. C'est cet indicateur qui permettra de mesurer le niveau de victoire en fin de partie.

FAQ

Pourquoi utilisez-vous le terme de « capacité » pour les développeurs ? Ne parle-t-on pas de « vélocité » ?

La vélocité est effectivement ce que l'équipe peut produire dans un sprint, mais c'est une notion collective ; on évite de parler de productivité individuelle dans une équipe Scrum, mais il a bien fallu faire des règles pour le jeu, c'est pourquoi j'ai pris un autre terme, pour ne pas créer de confusion.

Ce terme de « En production » est troublant !

En effet, on pourrait croire que quelque chose de « en production » n'est pas fini. Mais en informatique, quand quelque chose est « en production », ça veut dire qu'il est terminé et qu'il est disponible pour les utilisateurs. C'est le terme consacré pour l'environnement opérationnel, par opposition aux environnements de tests, dits « de recette », « de pré-production », « de développement », etc.

Est-ce que deux développeurs peuvent chacun mettre 1 ⏱ pour développer une User Story à 2 ⏱ ?

Non ! L'idée d'une décomposition en tâches élémentaires, c'est qu'elles sont faites pour être développées par une seule personne, en un seul sprint.

Que veut dire « effet immédiat » pour l'augmentation de la dette technique ?

Cela veut dire que la capacité est immédiatement modifiée, sauf pour le joueur qui est en train de jouer.

Il se peut donc qu'une perte de capacité soit sans effet. Voici deux exemples :

- Un développeur débutant augmente la dette technique et la fait passer de 4 à 5. Le développeur débutant et le développeur confirmé perdent 1 ⏱ de capacité, mais cela n'a pas d'impact immédiat sur le développeur débutant puisqu'il est en train de jouer. Puis c'est au tour d'un développeur normal, qui n'a pas été affecté. Il refactorise le code, et la dette repasse à 4. Les développeurs débutant et confirmé regagnent leur capacité. Au final, il n'y aura eu aucun impact négatif.
- Un développeur débutant consacre le premier de ses trois ⏱ à développer une User Story à 1 ⏱ . Il augmente la dette technique, ce qui la fait passer de 4 à 5. Il consacre ensuite ses deux derniers ⏱ à refactoriser. La dette repasse de 5 à 4. Le développeur confirmé, qui jouait juste derrière, n'a pas perdu sa capacité de 3 ⏱ .

Crédits et perspectives

Remerciements

Je remercie la SNCF d'avoir autorisé la publication de ce jeu en open source. Je remercie aussi tous les testeurs qui ont éprouvé les versions successives du jeu.

Crédits

Sans compétence de graphiste, j'ai fait au mieux pour décorer le jeu sans violer de copyright. Voici ce qui a été utilisé :

- Police de caractères : YoungSerif https://github.com/noirblancrouge/YoungSerif/blob/master/fonts/ttf/YoungSerif-Regular.ttf , Noto https://fonts.google.com/noto, toutes deux sous licences SIL Open Font License.
- Logo du site, image de fond, photos des équipiers, ampoule : créés par IA générative

Perspectives

Pas mal de pistes ont été abandonnées, et ce afin de garder le jeu simple. Voici en vrac quelques idées qui n'ont pas été retenues mais qui pourraient l'être dans des versions futures :

- Le fait de ne pouvoir livrer les Epics que quand elles sont développées à 100 % n'est pas trop conforme à l'esprit de l'agilité. Il faudrait pouvoir livrer une Epic incomplète, donc en étant capable de définir plus finement ce qui fait partie du MVP ou pas, on en affectant une valeur métier (visible ou cachée), à chaque User Story. Cela pourrait également nécessiter de créer les anomalies au moment du développement de la User Story et pas à la livraison de l'Épic.
- Intégrer un rôle de UX/UI et renforcer le rôle du puzzle. Le rendre potentiellement plus complexe (avec plus de pièces), en ajoutant l'objectif de le garder cohérent. Par exemple, si le tri choisi dans l'image fournie est « date de publication », il faut bien classer les livres fournis, sinon l'organisateur lui rajoute une anomalie. Cela aurait intégré le puzzle au gameplay, mais complexifie à la fois le design et les règles.
- Faire un site web Amazon-like, c'est un peu du déjà-vu de partout. Mais je n'ai pas trouvé d'idée à la fois originale et suffisamment simple pour que tout le monde puisse imaginer un MVP raisonnable.
- Idée « Offshorer les User Stories à 1 ⏱ ». Pour faire croire qu'on va pouvoir faire développer à moindre cout les fonctionnalités simples. Ce serait une mauvaise idée de plus, une équipe Scrum / agile ne fonctionnant vraiment bien qu'en local. Le risque serait que les joueurs piochent trois mauvaises idées et soient contraints d'en choisir une.
- Idée « Stagiaire » ou « alternant » (ça ne coute presque rien !) : il pourrait, après une phase de formation couteuse, apporter quelques ⏱ de plus de capacité. Il faudrait qu'au global, ce ne soit ni rentable ni pénalisant.

To-do list

- **Dessiner la boite de jeu** : illustrer une boite carrée de dimensions extérieures 27.5 x 27.5 x 6.3 cm.

Licence

Fin du jeu

À la fin du huitième sprint, si l'équipe n'a pas été virée à cause du mécontentement du client ou de la dette technique, le jeu est terminé.

On procède à un dernier calcul de la trésorerie, et l'organisateur invite les joueurs à regarder le verso de leur feuillet de joueur.

Selon le fait qu'ils ont développé ou pas le MVP et selon leur trésorerie, ils pourront cocher dans quel niveau de réussite ils se trouvent.

C'est le moment de débriefer, d'expliquer les concepts plus en profondeur et surtout de faire la part des choses entre ce qui a été modélisé dans le jeu et la vraie vie. C'est là que les joueurs vont critiquer les règles du jeu ! C'est normal, mais il faut leur expliquer que le jeu est une vision simplifiée de la réalité, destinée à éprouver certaines situations et découvrir certains concepts. Ce n'est pas grave s'ils ont « perdu » !

Fins possibles

Si le MVP n'a pas été réalisé	💀 Échec critique 💀
Si la trésorerie est négative ou nulle	💀 Échec critique 💀
Si la trésorerie est entre 1 et 7	n succ s miti é
Si la trésorerie est supérieure à 7	n franc succ s **Normalement, pour obtenir une trésorerie supérieure à 7, il faut prendre la décision d'ouvrir le site avant le dernier sprint, et ainsi de passer en mode produit. Le but est de montrer l'impact crucial du TTM (Time To Market).**