GESTION DE PROJET EN METHODES AGILES

R3.10 BUT2 Année universitaire 2022/2023

INTRODUCTION

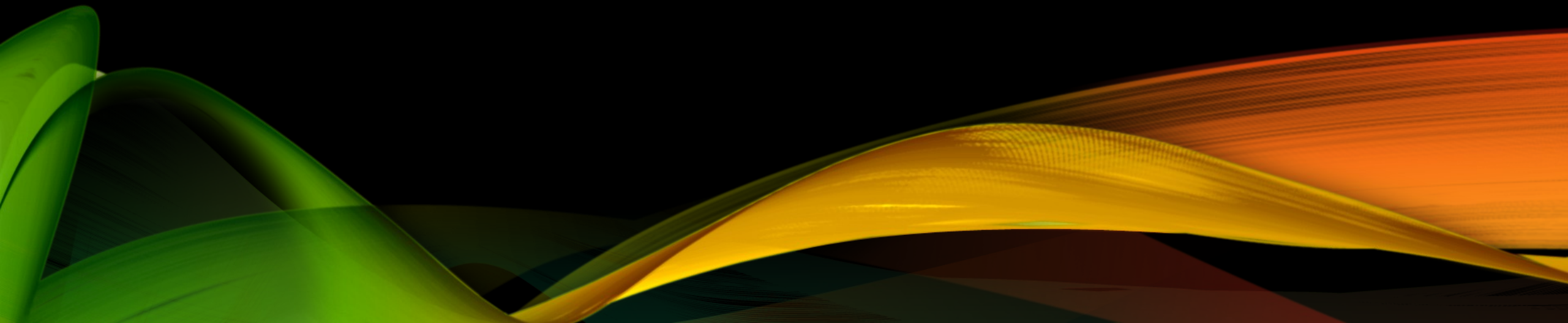
- Environ 20% des projets informatiques n'aboutissent pas
- Dans les 4 années suivantes, 80% des projets vont disparaître
- 50% du code produit n'est jamais utilisé
- Dans un logiciel, environ 20% des fonctions sont utilisées

→Pourquoi? Quels problèmes?

https://app.wooclap.com/events/JZAASB/questions/6316f060557813057cbee29a

- www.wooclap.com/**JZAASB**

LES METHODES TRADITIONNELLES DE GESTION DE PROJET



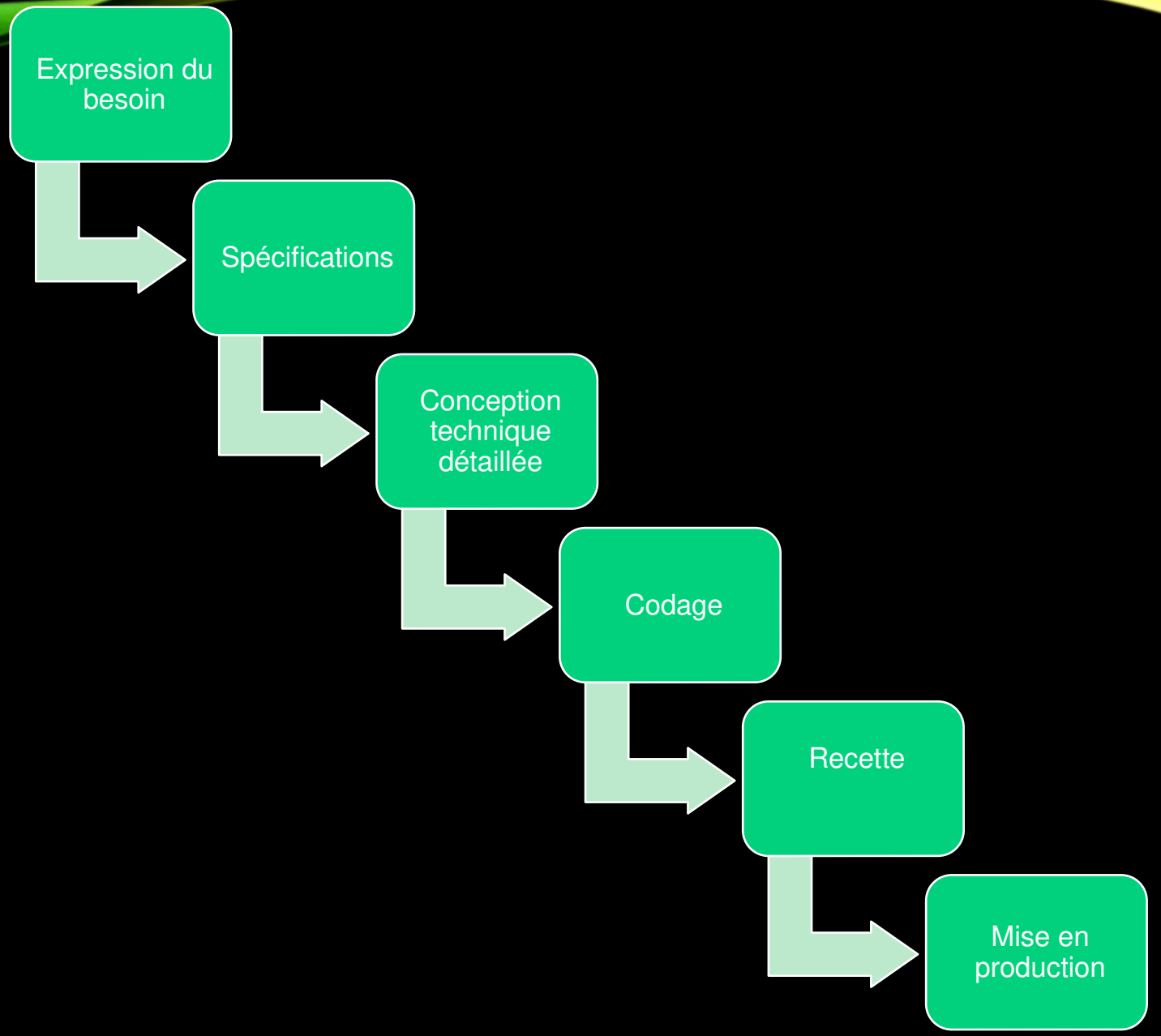
1) LE MODÈLE DE GESTION DE PROJET EN CASCADE

- Chaque phase est appliquée l'une après l'autre:

1) Expression du besoin selon un formalisme défini par l'entreprise. Recensement des exigences fonctionnelles des utilisateurs
2) Rédaction des spécifications fonctionnelles qui décrivent le fonctionnement de l'application du point de vue utilisateur
3) Phase de conception technique
4) Codage sur la base des spécifications fonctionnelles et techniques
5) Recette de l'application (tests)
6) Livraison de l'application

1) LE MODÈLE DE GESTION DE PROJET EN CASCADE

Quel est le problème ?



1) PROBLEMES DU MODÈLE WATERFALL

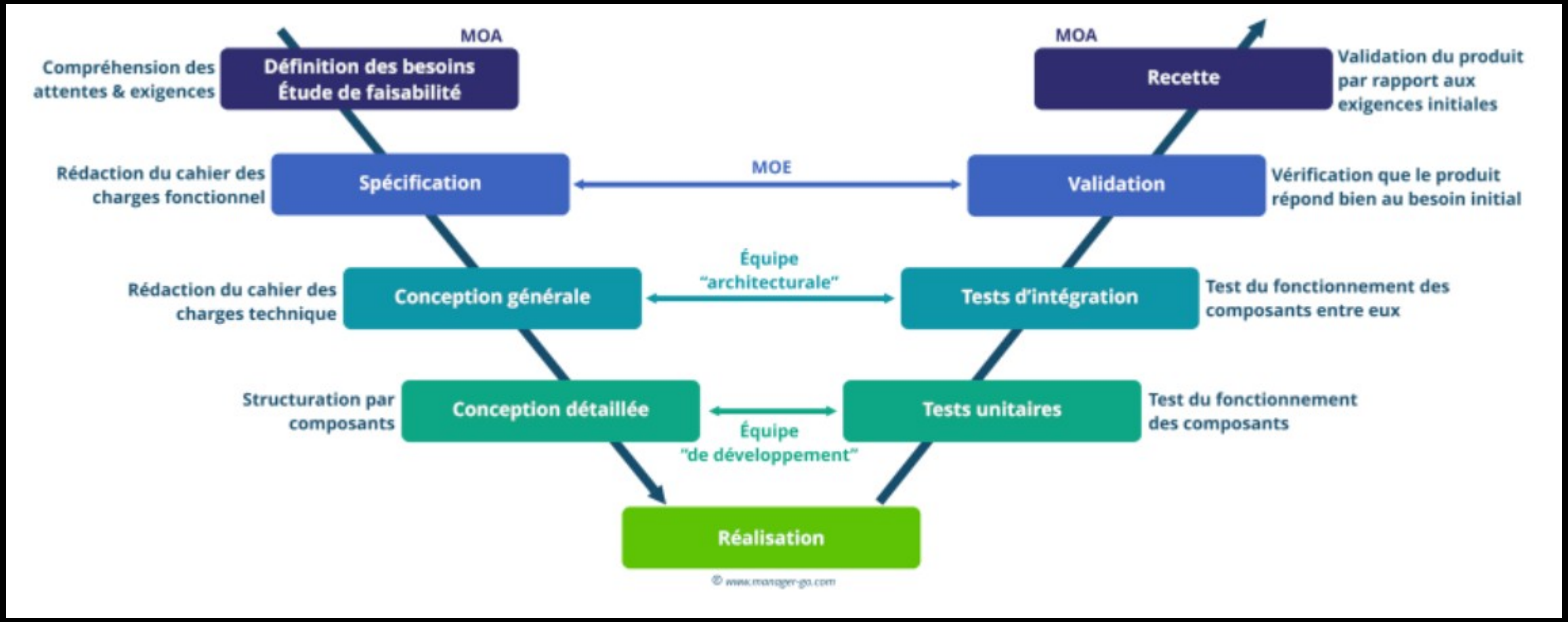
- EXEMPLE: si rédaction des spécifications guidée par une mauvaise compréhension des besoins (étape 3) ✉ détection de l'erreur lors des tests fonctionnels (étape 5)

✉ Il faut alors refaire toutes les étapes alors que la phase 4 du codage a pu prendre beaucoup de temps

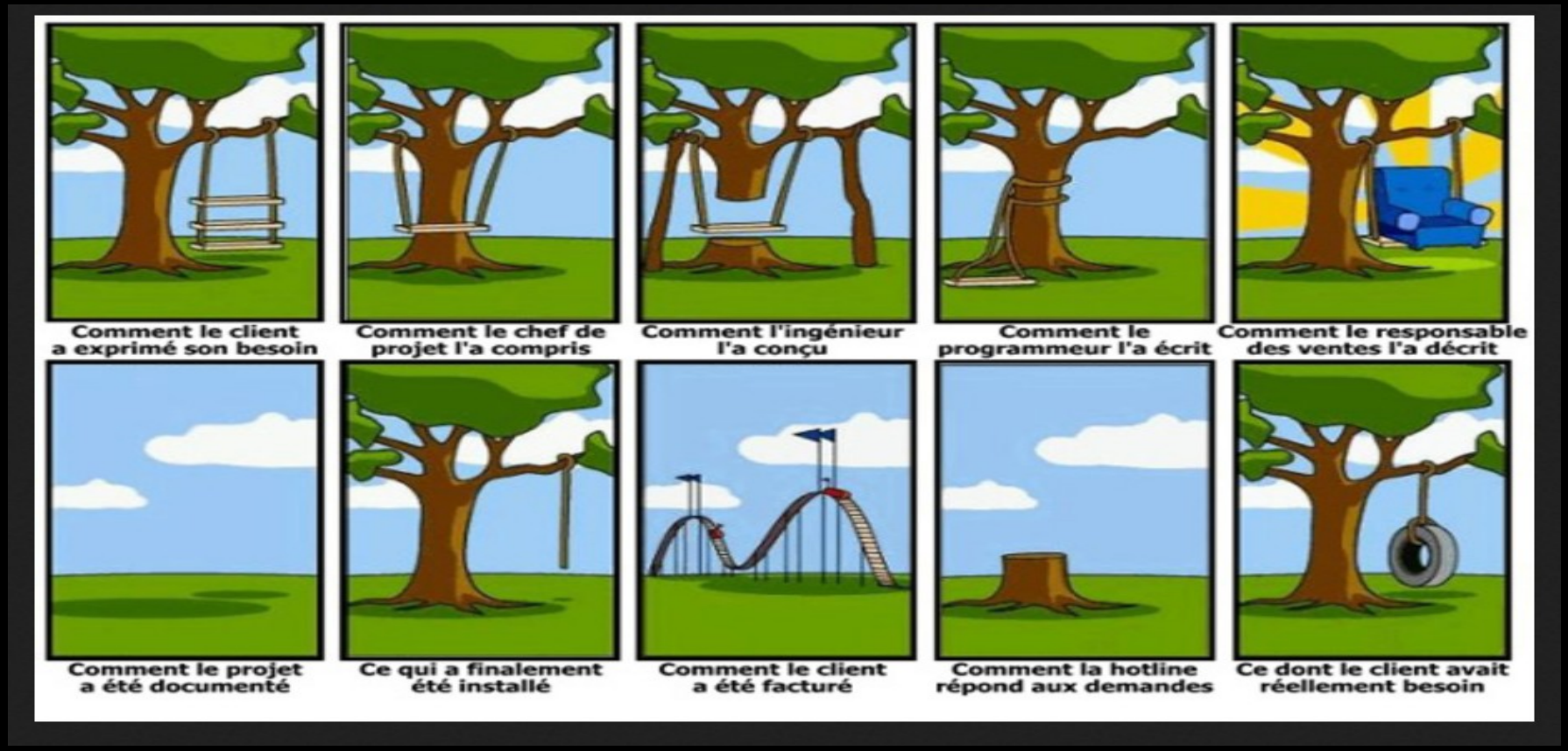
2) LE MODÈLE EN V

- Etape 1 Formalisation de l'expression des besoins. Le client explicite son besoin via une description générale des fonctions du logiciel qu'il imagine
- Etape 2 Ecriture des spécifications fonctionnelles ✉ description complète de tous les cas d'utilisation
- Etape 3 Ecriture des spécifications techniques. Transcription des spécifications fonctionnelles en langage technique. Il y a des briques fonctionnelles (modules)
- Etape 4 Développement
- Etape 5 Exécution des tests unitaires on s'assure que chaque brique fonctionnelle est en ordre de marche (TU avec définition un peu différente dans les méthodes agiles)
- Etape 6 Exécution des tests d'intégration
- Etape 7 Exécution des tests de validation
- Etape 8 Recette et mise en production

2) LE MODÈLE EN V



2) LE PROBLÈME DU MODÉLE EN V: L'EFFET TUNNEL



2) LE PROBLÈME DU MODÉLE EN V: L'EFFET TUNNEL

BESOIN: JEUNE FEMME ÉNIGMATIQUE AVEC DES CHEVEUX LONGS BRUNS DEVANT UN PAYSAGE

Ce que le client voulait

Ce qui lui a été livré

2) LE PROBLÈME DU MODÉLE EN V: L'EFFET TUNNEL

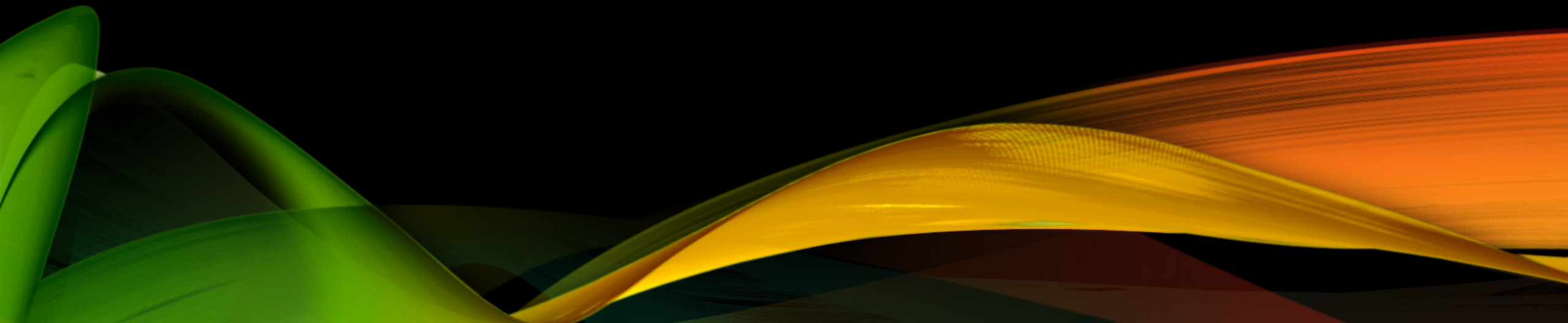
- Le MOA est fortement impliqué au départ dans la rédaction des besoins mais ensuite il est davantage mis à l'écart

→Entre la phase de conception et la phase de livraison, il peut y a voir eu beaucoup de temps et les besoins utilisateurs peuvent avoir évolué (en raison des évolutions de leur métier, de nouvelles lois…) ou il se peut qu'il ait eu une mauvaise interprétation

→Produit voulu ≠ produit livré

→Comment faire?

L’ESPRIT DES METHODES AGILES



LES VALEURS DES METHODES AGILES

- Le manifeste Agile 2001
- 4 valeurs

→Favoriser les individus et leurs interactions plutôt que la mise en place de processus d'outils

→Favoriser la collaboration avec le client plutôt que la négociation contractuelle

→Faire en sorte d'obtenir un logiciel opérationnel plutôt d'investir sur des documents inutiles

→Etre réactif face au changement plutôt que de respecter un plan

LES 12 PRINCIPES AGILES

- Accorder une haute priorité à la satisfaction du client à travers des livraisons de logiciels rapprochées et continus
- Accepter le changement de besoins même tard dans le développement
- Livrer fréquemment un logiciel qui marche à échéances régulières de deux semaines à deux mois avec une préférence pour les petites périodes de temps
- Faire travailler ensemble quotidiennement les utilisateurs et les développeurs
- Construire les projets autour de personnes motivées. Leur donner l'environnement et le support dont elles ont besoin et leur faire confiance
- Privilégier l'information face à face qui est le moyen le plus efficace pour transmettre de l'information aux équipes de développement
- Considérer les versions opérationnelles du logiciel comme étant les mesures principales de progrès

LES 12 PRINCIPES AGILES

- Considérer les procédés Agiles comme les moteurs d'un développement viable . Sponsors, développeurs et utilisateurs doivent pouvoir maintenir un rythme constant indéfiniment
- Améliorer une attention continue à l'excellence technique et à la bonne conception afin d'améliorer l'Agilité
- Privilégier la simplicité c'est-à-dire l'art de maximiser le travail à ne pas faire
- Considérer que les meilleures architectures, besoins et conceptions émergent d'équipes auto organisées
- Réfléchir à intervalle régulier à la façon de devenir plus efficace et agir sur le comportement de l'équipe en conséquence

✉ Beaucoup de conclusions de théories de RH + lean mangement + amélioration continue

AVERTISSEMENT

Les méthodes Agiles ≠ anarchie

Comment livrer un logiciel opérationnel en 4 semaines si chacun fait n'importe quoi, n'importe quand et dans n'importe quel sens????!!!!!!!!

Il y des processus, de la documentation, des plannings … mais ils sont présents pour être utiles

PLUSIEURS MÉTHODES AGILES

- RAD (rapid-application development)
- Lean
- Kan-ban
- eXtreme Programing
- Scrum

Nous allons voir différentes méthodes rapidement et nous verrons la méthode Scrum dans le détail car c’est celle la plus utilisées dans les entreprises

LE LEAN A LA SOURCE DES AUTRES METHODES

- Le lean s'appuie sur le juste à temps et les méthodes mises en place par Toyota dans les années 60's
- Ici le lean management est mis en application sur l'activité informatique comme elle avait été d'abord appliquée à l'industrie puis aux services plus classiques

→Le but est d'analyser tous les processus de production

→Méthode basée sur 14 principes

→De cette méthode vont découler les autres méthodes Agiles

LES 14 PRINCIPES DU LEAN

La philosophie long terme

1. **Penser sur le long terme :** Fondez vos décisions sur une philosophie à long terme, même au détriment des objectifs financiers à court terme.

LES 14 PRINCIPES DU LEAN

Les bons processus donneront les bons résultats

2. **Fluidité :** Organisez les processus en flux pièce à pièce pour mettre au jour les problèmes.

3. **Flux tirés :** Utilisez des systèmes tirés pour éviter la surproduction.

4. **Production constante et lissée :** Lissez la production.

5. **Automatisation avec une touche humaine :** Créez une culture de résolution immédiate des problèmes, de qualité du premier coup.

6. **Tâches standardisée :** La standardisation des tâches est le fondement de l'amélioration continue et de la responsabilisation des employés.

7. **Contrôles visuels :** Utilisez le contrôle visuel afin qu'aucun problème ne reste caché.

8. **Technologies et méthodes fiables :** Utilisez uniquement des technologies fiables, longuement éprouvées, qui servent vos collaborateurs et vos processus.

LES 14 PRINCIPES DU LEAN

Apporter de la valeur à l'organisation en développant les personnes

9. **Cultiver les leaders :** Formez des responsables qui connaissent parfaitement le travail, vivent la philosophie et l'enseignent aux autres.

10. **Faire monter en compétence les personnes de qualité :** Formez des individus et des équipes exceptionnels qui appliquent la philosophie de votre entreprise.

11. **Respecter et motiver ses partenaires :** Respectez votre réseau de partenaires et de fournisseurs en les encourageant et en les aidant à progresser.

LES 14 PRINCIPES DU LEAN

La résolution continue de problème est un moteur d'apprentissage pour l'organisation

12. **Aller toujours sur le terrain :** Allez sur le terrain pour bien comprendre la situation .

13. **Prendre les décisions en consensus :** Décidez en prenant le temps nécessaire, par consensus, en examinant en détail toutes les options. Appliquez rapidement les décisions.

14. **Amélioration continue :** Devenez une entreprise apprenante grâce à la réflexion systématique et à l'amélioration continue.

- Le lean pose les bases des méthodes Agiles

LE KANBAN

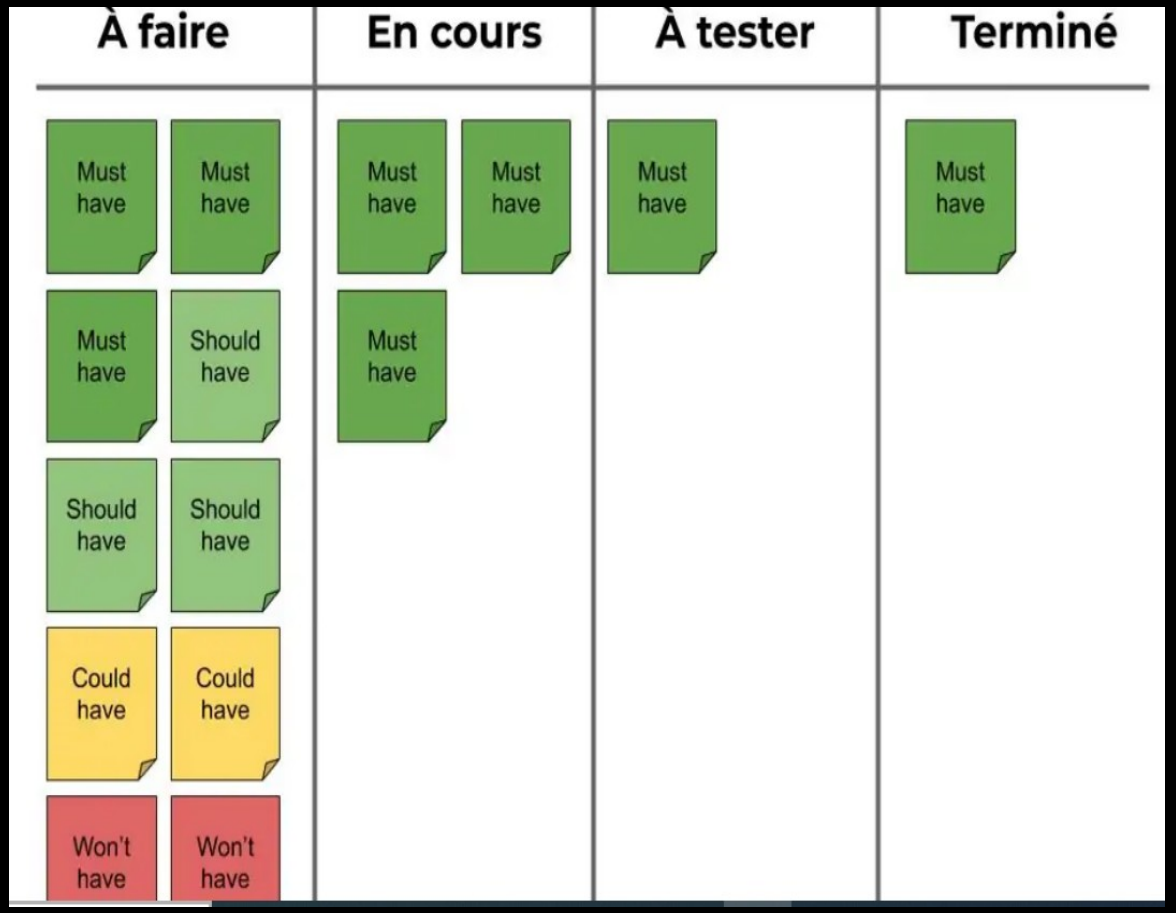
Le kanban est un système permettant de **partager l'information** et de **maîtriser visuellement le flux de travail.**

A l'origine, Kanban veut dire « étiquette ». Celle-ci allait d'un service aval de la production à un service amont pour lui signaler que le service aval allait avoir besoin d'une certaine quantité de pièces pour produire car il est à la fin de son stock.

Chaque service ne peut pas traité qu'une quantité limitée d'étiquette (car il ne peut pas produire tout à la fois) et donc est obligé de prioriser et de connaître le nombre maximal de tâche qu'il peut accepter

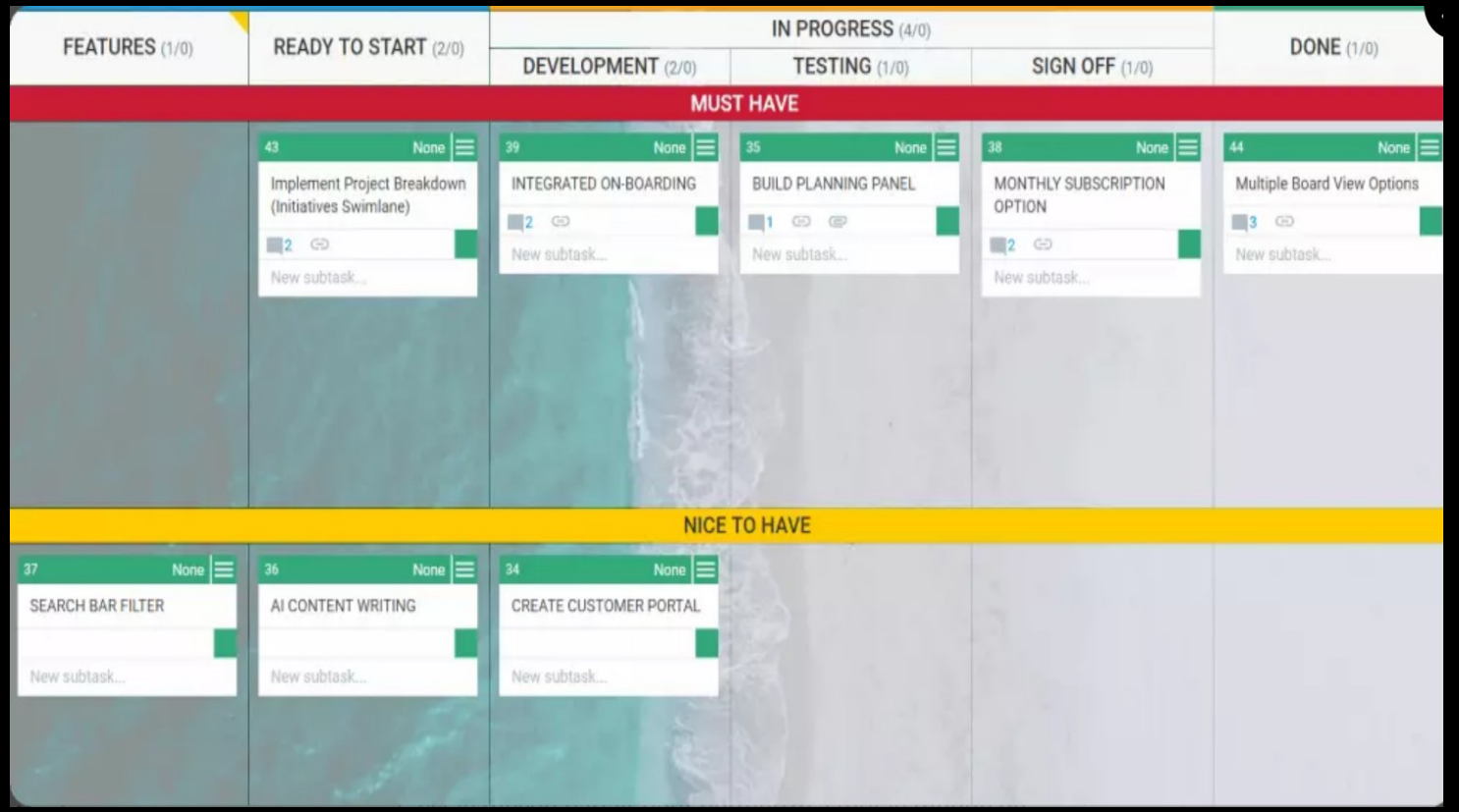
LE KANBAN DANS LE DÉVELOPPEMENT LOGICIEL

- Visualisation du workflow: on découpe le travail en morceau que l'on écrit sur des cartes mises sur un tableau divisé en colonne représentant chaque étape du développement
- On limite le nombre de cartes possibles à chaque étape
- On mesure le « lead time » (temps de cycle pour traiter une carte et on essaie de l'améliorer



LE KANBAN DANS LE DÉVELOPPEMENT LOGICIEL

- → On voit le workflow de bout en bout
- → On peut les compléter par des graphes de suivi des lead times par exemple



METHODE XP OU EXTREME PROGRAMMING

- La qualité est non négociable
- Le nombre de fonctionnalités développées dépend du budget et du temps alloué au projet par le client. Le client est le seul responsable du nombre de fonctionnalités développées en fonction de son budget et du temps imparti.
- 13 PRATIQUES

→Livraisons fréquentes découpage de l'application en module qui vont être livrés à un rythme régulier (itération)

→Rythme durable pour les équipes . Travail constant et durable

→Client sur le site afin de suivre le projet car il a la responsabilité des fonctionnalités développées

METHODE XP OU EXTREME PROGRAMMING

- 13 PRATIQUES

→Conception simple. On répond de la façon la plus simple au besoin du client

→Mise en place de règles de codage afin que le code soit lisible par toute l'équipe

→L'équipe est responsable du code c'est-à-dire le code produit par un membre doit être exempt de bugs et optimisé afin de s'intégrer dans le reste du code de l'équipe

→Utilisation des tests unitaires TDD (test driven development). Ces tests sont rejoués à chaque modification afin de s'assurer de la non régression technique

→Test de recette afin de vérifier la conformité de l'application aux besoins fonctionnels.

METHODE XP OU EXTREME PROGRAMMING

- 13 PRATIQUES

→Mise en place de l'intégration continue: le développement terminé est intégré automatiquement au produit final

→Réaliser du refactoring de code: le but est de remanier le code perpétuellement pour le rendre plus efficace

→Programmation en binôme le développeur chargé de coder (le driver) + le partner qui suggère des améliorations ou détecte des problèmes. Le but est d'améliorer la communication et la compétence collective de l'équipe

→Estimation collective des efforts de développement avec le planning poker

→Utilisation de métaphores et d'analogies

METHODE XP OU EXTREME PROGRAMMING

Les différentes étapes:

1Le client écrit ses besoins sous forme de scénario

2 Les développeurs évaluent le coût de chaque scénario avec le client

3 Le client choisit les scénarios à intégrer à la prochaine livraison

4 Chaque développeur prend la responsabilité d'une tâche pour la réalisation d'un scénario

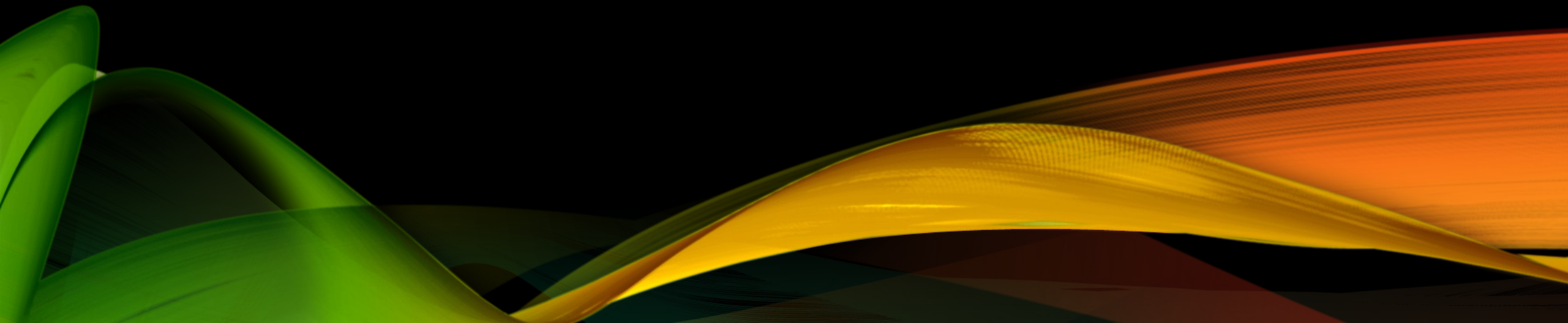
5 le développeur choisit un partenaire

6 le binôme écrit les tests unitaires correspondant au scénario

7 Le binôme prépare et procède à l'implémentation

8 Le binôme intègre ses développement à la version d'intégration

METHODE SCRUM

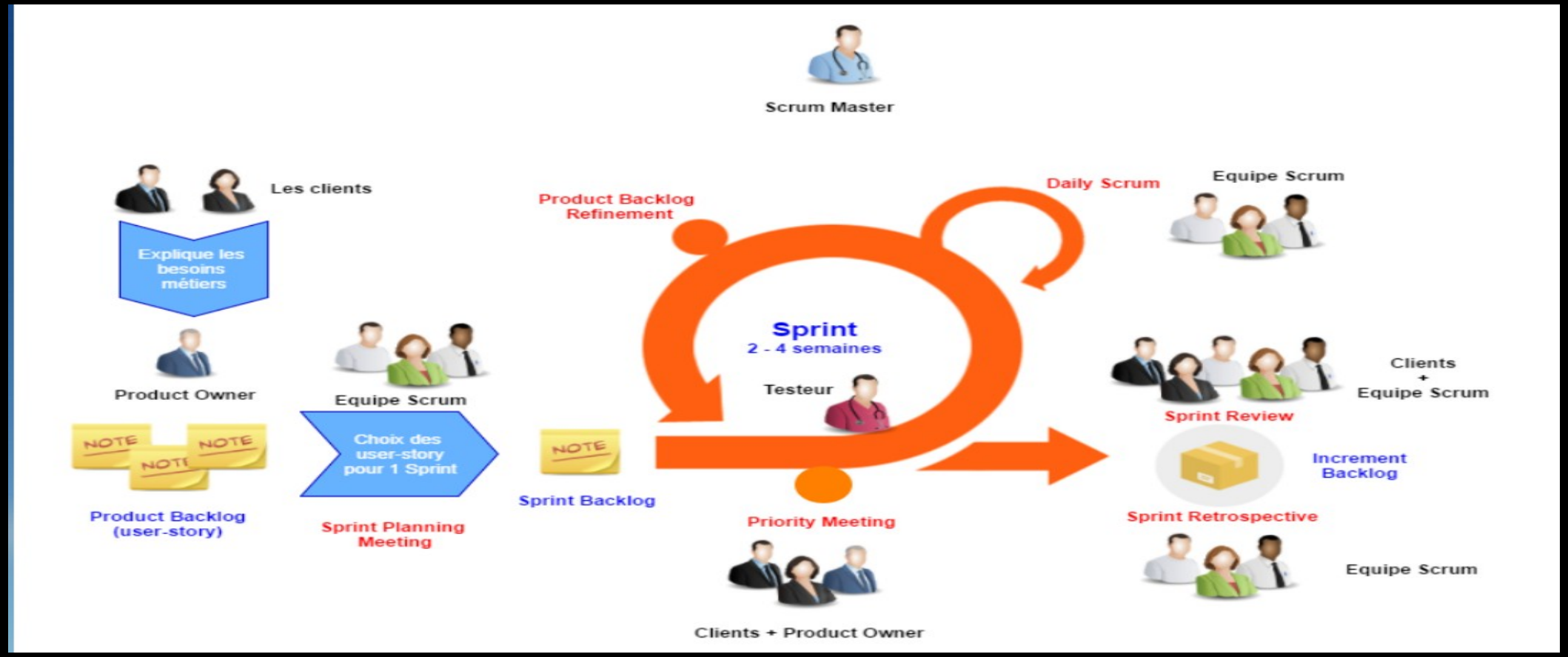


INTRODUCTION

- La base: une équipe unie ayant pour but commun de faire avancer la balle dans le camps adverse
- https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-French.pdf
- https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

- Méthode utilisée majoritairement dans les entreprises et qui empruntent aux autres en développant des concepts propres

LE SCHÉMA DE LA METHODE SCRUM



TRAVAIL SUR LE VOCABULAIRE SCRUM

Les acteurs

→Le scrum master

→Le product owner

→La scrum team

Les évènements

→Le sprint

→Le sprint planning meeting

→Le daily meeting

→Le sprint review

→Le sprint retrospective

Les artefacts

→Le product backlog

→Le sprint backlog

→Le increment backlog

LES EVENEMENTS

- **Le sprint**: durée pendant laquelle est achevé un objectif de production (De 2 semaines à 4 semaines)

- **Le sprint planning meeting** : réunion pendant laquelle on choisit du but du sprint et que l'on divise la user story en tâches (8 heures max pour un sprint de 1 mois)

- **Le daily meeting** : réunion debout quotidienne pendant laquelle chaque membre de l'équipe répond à 3 questions

→Qu'a-t-il fait hier?

→Que va-t-il faire aujourd'hui?

→Quels problèmes a-t-il rencontrés?

- **La sprint review**: réunion pendant laquelle l'équipe présente la production du sprint au PO et pendant laquelle on discuste des changements à apporter (4h max pour un sprint de1 mois)

- **La sprint retrospective** : réunion dont le but est de planifier les moyens d'améliorer la qualité et l'efficacité. L'équipe se pose trois questions:

→Qu'est ce qui a bien fonctionné pendant ce sprint?

→Qu'est ce qui peut être amélioré?

→Sur quoi pouvons nous engager comme piste d'action pour le prochain sprint?

L'équipe va prendre des engagements pratiques pour améliorer la qualité et l'éfficacité.

LES ROLES LE PRODUCT OWNER

- C’est une SEULE personne et ce n’est pas un commitee
- Il représente les besoins des consommateurs, des utilisateurs finaux
- Il doit réussir à créer une vision produit
- Il doit gérer le backlog:

→Le créer

→Clairement communiquer sur les items du backlog

→Ordonner le backlog

- Il peut être aidé par des personnes qui connaissent le métier ou des aspects métiers que le PO MAIS le PO reste responsable du backlog
- Il choisit le calendrier de livraison des releases. Une release est un regroupement de plusieurs production de srpint

LES ROLES LE PRODUCT OWNER

- Il doit être avec la team (si il ne peut pas être suffisamment présent, il peut avoir un proxy product owner (PPO))
- Le PO accepte ou non la production du backlog. Il doit dire à l'équipe pourquoi il refuse un increment et ce qui doit être modifié ✉ amélioration continue
- Le product owner:

→Crée le backlog

→Priorise le backlog et donc en quelque sorte décide ce que l'équipe va faire en premier, en second…

MAIS une fois le sprint commencé, il ne peut pas et ne doit pas changer ce que va faire l'équipe= le sprint backlog ne peut pas être changé en cours de sprint

LES ROLES LE SCRUM MASTER

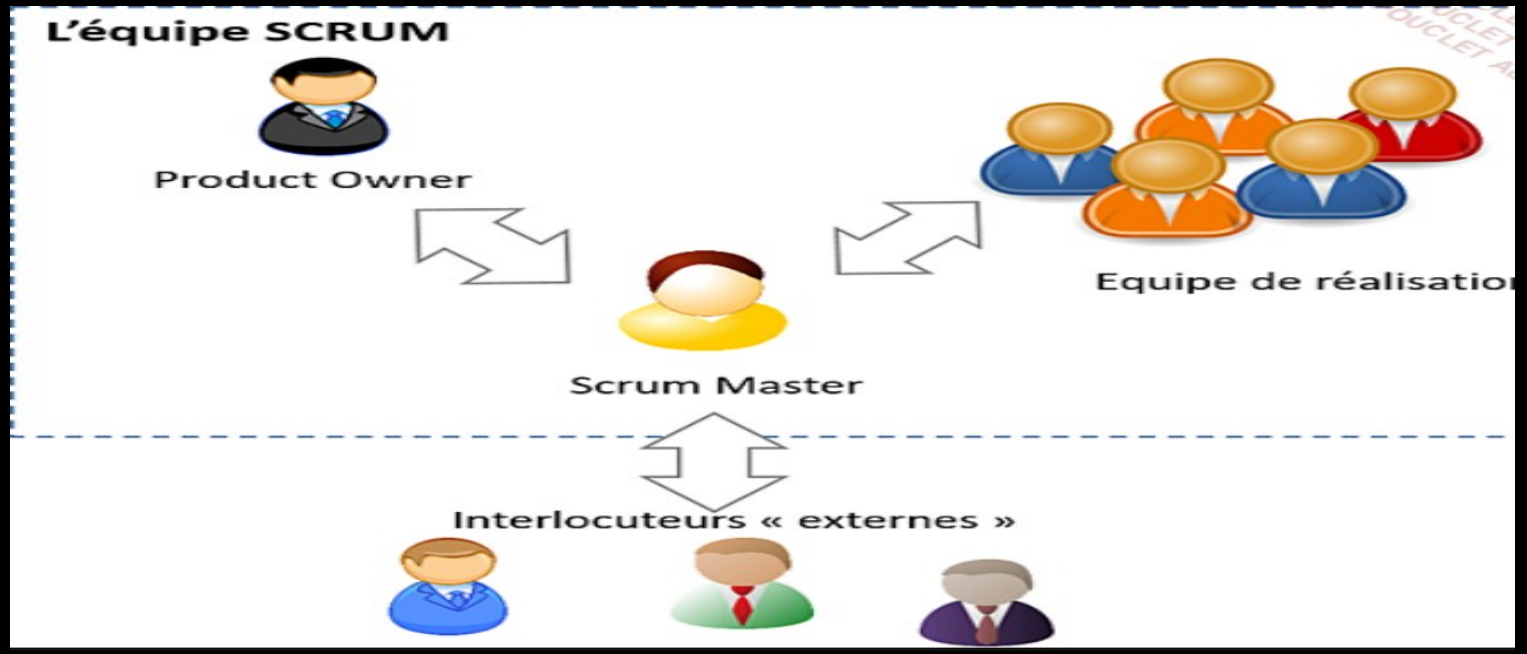
- Le scrum master n’est pas

→Un manager

→Un chef technique

→Un chef de projet

- Le scrum master est un leader serviteur (servant leader)



LES ROLES LE SCRUM MASTER

- Il sert l'équipe scrum
- Il est présent pour vérifier que les processus et les artefacts de la méthodes scrum sont bien respectés
- Il vérifie que les temps sont respectés
- Avec le product owner il crée les users stories et prend soin du backlog.
- Il peut conseiller le product owner mais il ne peut prendre aucune décision à sa place
- Il n'est pas un manager et donc ne dit pas à l'équipe quelles tâches elle doit faire ni comment se les répartir
- Il vérifie en revanche que les méthodes scrum sont bien appliquées
- Il « protège » l'équipe et fait en sorte que l'équipe reste focalisée sur le projet

✉ Il essaie de faire l'intermédiaire et le « facilitateur » entre l'équipe et les autres département

LES ROLES L'ÉQUIPE

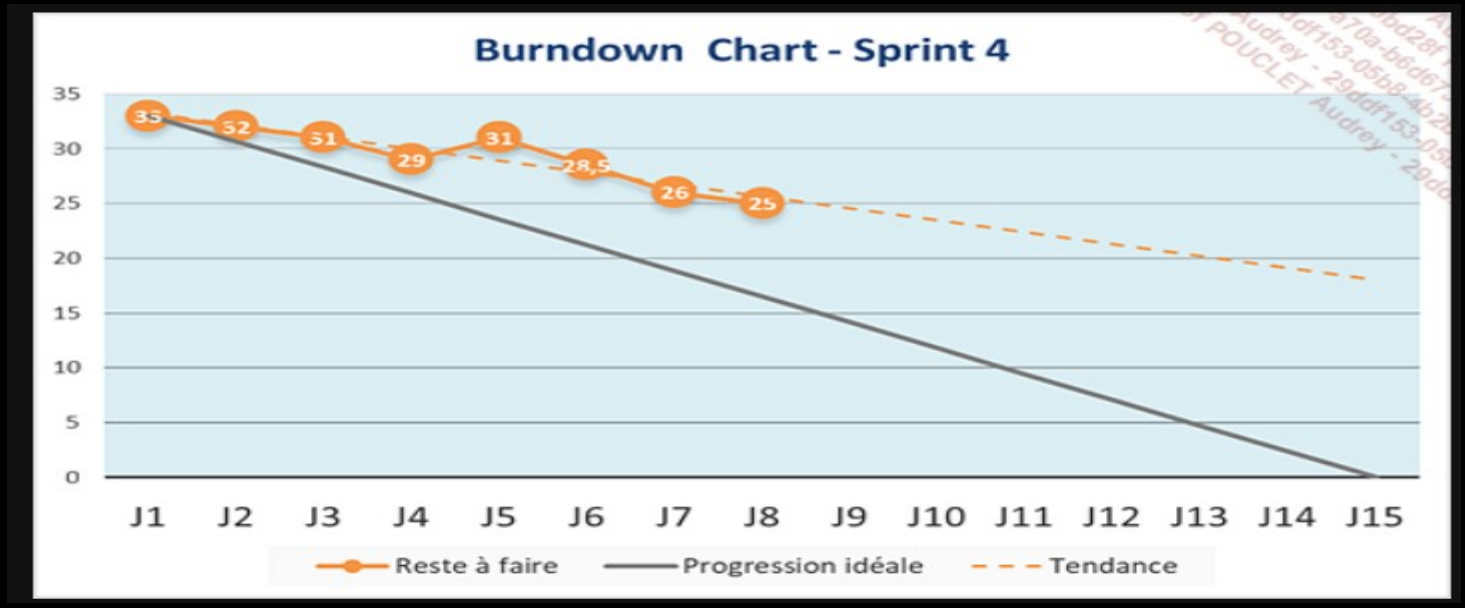
- Equipe de réalisation plutôt que équipe de développement car l'équipe est multitâches
- L'équipe est auto organisée et personne n'a de titre à priori (par exemple personne n'est gestionnaire de la base de données à priori, si quelqu'un s'occupe de cette tâche en particulier c'est parce que c'est utile à l'équipe et au projet).
- L'équipe est multi compétence
- Entre 3 et une dizaine avec des compétences diverses
- Il est nécessaire d'avoir des lieux dédiés avec des outils de management visuels du projet et à défaut tout faire pour s'en rapprocher

LES ARTEFACTS

- Le backlog produit: c'est l'expression des besoins du product owner traduite sous forme de user stories. Elles sont priorisées.
- Le backlog sprint: liste des user stories qui vont être finies pendant le sprint. Chaque user story est divisée en tâche

+ pendant le sprint il y a le backlog grooming c'est-à-dire que l'on raffine le backlog, on le nettoie, on l'améliore

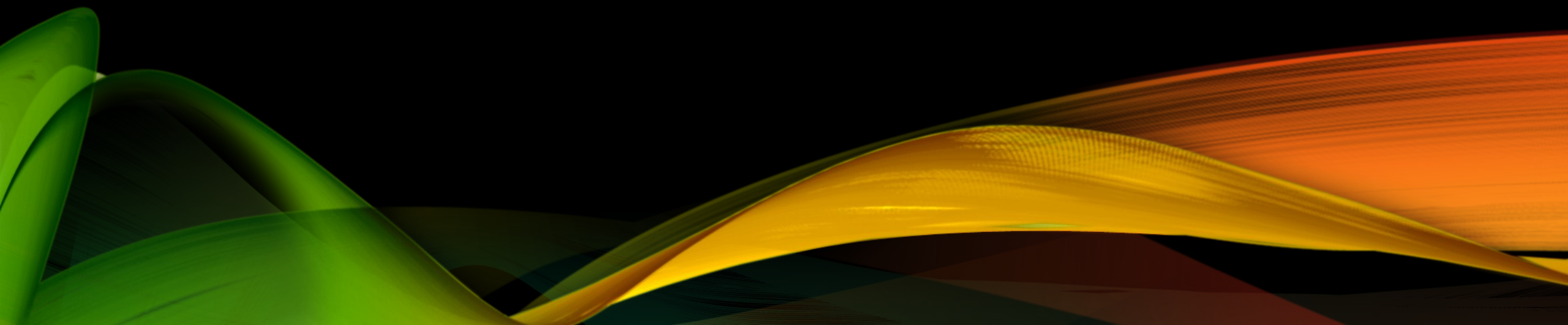
- Burn down charts: c'est un outil de suivi. On mets en abscisse la durée du sprint et en ordonnée la somme des efforts contenus dans le sprint (on a évalué chaque user story en effort)



DÉROULEMENT D'UN CYCLE SCRUM

1) Le product owner rédige les user stories et les place dans le product backlog
2) Il les priorise
3) L'équipe fait la réunion de planification du sprint, décide des user stories qui seront traitées pendant le sprint et les découpe en tâches
4) Sprint
5) Pendant le sprint daily metting et backlog grooming
6) À la fin du sprint démonstration du livrable et ajustement avec le product owner
7) Retrospective du sprint

CONSTRUIRE ET PRIORISER LE BACKLOG



LA USER STORY

- C’est une description simple et compréhensible d’un élément de fonctionnalité à valeur « métier » du système
- La valeur métier est aussi appelée valeur économique. Elle se base sur un rapport coûts et bénéfices. Elle est déterminée par plusieurs caractéristiques :

→la valeur commerciale (ce produit me fait gagner ou économiser de l’argent),

→la valeur marché (clients potentiels),

→La valeur sociale (statut social)

→la valeur temporelle (gain à long terme).

LA USER STORY

- Elle est exprimée du point de vue utilisateur et elle est guidée par ces trois questions:

→Qui fait la demande ou qui bénéficie de la demande?

→Quelle est la demande ? (besoin)

→Quelle valeur métier découle de la réalisation de ce besoin?

Exemples: je veux que la TVA soit automatiquement calculée sur mes factures

Je veux que le compte d'un client non actif pendant un an soit supprimé automatiquement.

- Il y a aussi des Epic story qui sont des ensemble de user stories sur le même thème

Exemples: je souhaite une automatisation des factures

Je souhaite une gestion du portefeuille client

COMMENT REDIGER UNE USER STORY?

- Règles des 3C:

→Carte : Sur une carte de taille assez réduite (genre post it)

→Conversation: Pendant une conversation les détails de la user story seront exprimés

→Confirmation: des tests de validation sont décrits avec la story (ils serviront à valider qu'elle a été réalisée correctement)

Pour les tests de validation, on peut utiliser le « given/when/then »
Ou « étant donné/ quand ou lorsque/ alors »

Etant donné que 'contexte initial'
Quand 'un événement'
Alors 'un certain résultat'

MODÈLE DE USER STORY

Epic/User Story *<n°identifiant>* : **Titre** *<Titre de l'histoire utilisateur>*

Description : *<Description brève de l'histoire utilisateur>*

En tant que …
Je veux …
Afin de …

Critères d'acceptation : *<Description des conditions qui devront être satisfaites pour dire si l'histoire est terminée>*

Etat : *<Avancement de l'histoire dans son cycle de vie>*

Valeur métier : *<Estimation de la valeur>*

Effort de réalisation : *<Estimation en Story Points>*

Itération : *<Numéro de l'itération pour la planification>*

EXEMPLES

- User story 1 **En tant qu**'utilisateur déconnecté, **Je souhaite** pouvoir me connecter au site Web, **Afin que** je puisse accéder à mon profil personnel
- Test : **Scenario** : L'utilisateur système se connecte avec des informations d'identification valides

Étant donné que je suis un utilisateur système déconnecté et que je suis sur la page de connexion, **Lorsque** je remplis les champs «Nom d'utilisateur» et «Mot de passe» avec mes informations d'authentification et que je clique sur le bouton Connexion, **Alors** le système me connecte

EXEMPLES

•**En tant qu**'abonné de la médiathèque **Je recherche** un ouvrage en spécifiant son auteur, son titre ou sa référence **Afin que** l'emprunter ou de le réserver

- Critères d'acceptation sous forme de liste :

✉ Résultat unique, afficher l'ouvrage : auteur/titre/reference

→Pas de résultat, inviter l'abonné à une nouvelle recherche

→Plusieurs résultats, afficher la liste des ouvrages triés par auteur

- Critères d'acceptation sous forme de scenario :
- **Scenario** : l'abonné lance une recherche pour consulter la fiche d'un ouvrage

Étant donné que je suis abonné et que je suis sur la page de recherche d'un ouvrage, **Lorsque** je saisis l'auteur, le titre ou la référence et que je lance la recherche, **Alors** je peux consulter la liste des ouvrages correspondants triés par auteur

LA MÉTHODE INVEST POUR ECRIRE LES USERS STORIES

- **I**ndépendante des autres: une bonne US doit être indépendante des autres
- **N**égociable initialement, plutôt qu'un engagement ferme: ne formuler dans un premier temps que l'essentiel, à savoir l'objectif fonctionnel recherché; on évitera par exemple de spécifier dans une User Story des éléments techniques,
- Source de **Valeur :** elle doit être source de valeur pour le client ou l'utilisateur (par exemple une US avec seulement de la technique n'est pas source de valeur)
- **Estimable** en termes de complexité relative Les conditions de satisfaction doivent être suffisamment précises et restreintes pour que l'équipe de développement puisse quantifier l'effort d'implémentation, sinon dans l'absolu du moins en termes de complexité relative.
- **S**uffisamment petite (en anglais *S*mall) dans l'idéal 5 ou 6 US peuvent être terminées en sprint
- **T**estable en principe, ce qu'on vérifie en écrivant un test « améliorer la performance » par exemple est difficilement testable

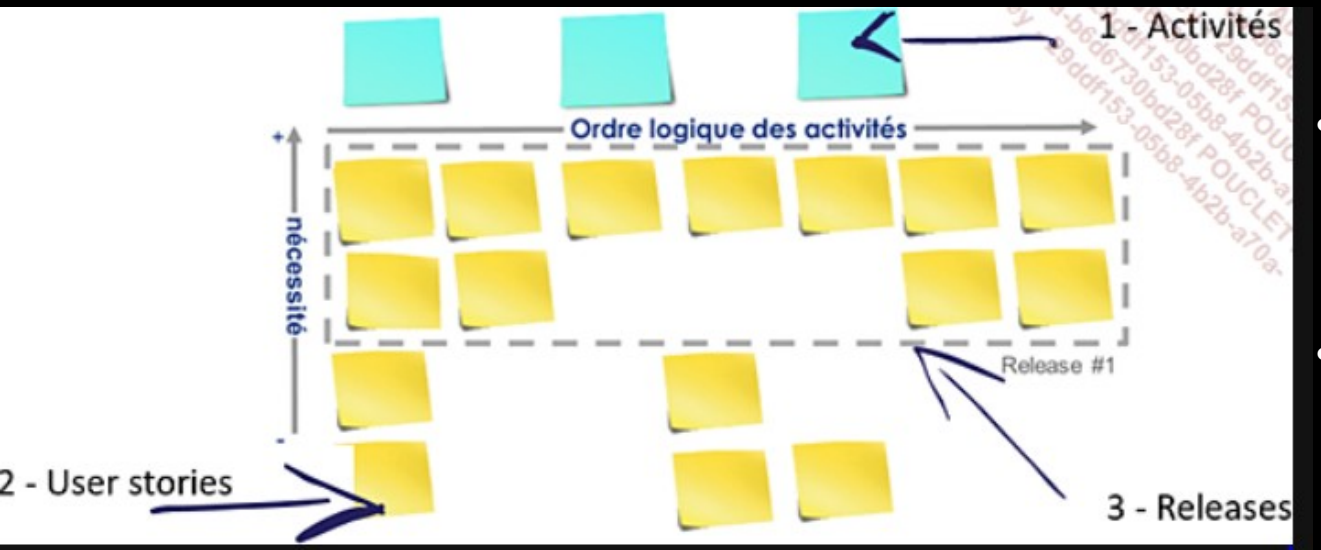
ERREURS À EVITER

- Prendre des utilisateurs trop génériques

Par exemple faire la distinction client abonnés et client standard plutôt que de dire en tant qu'utilisateurs

- Les US qui vont être réalisées doivent être détaillées mais celles qui le seront dans plusieurs sprints n'ont pas à l'être trop et seront revues au cours du temps

CONSTRUIRE LE BACKLOG: LE STORY MAPPING



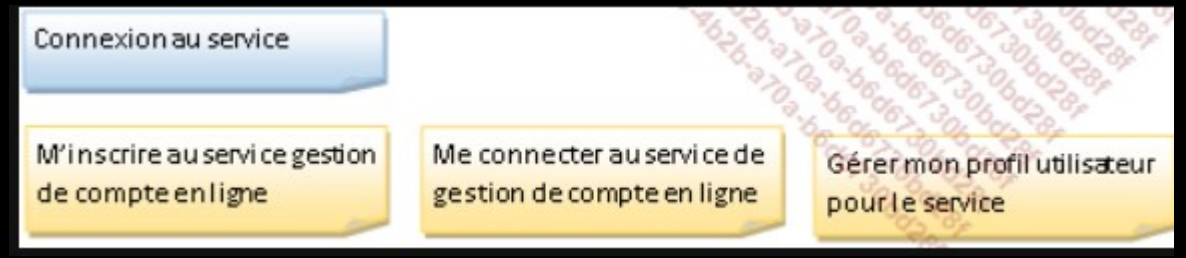
- Activités : ce sont les actions de « haut niveau », ordonnées de gauche à droite selon l'ordre logique (celui qui apparait quand on raconte l'histoire de l'utilisation du système par ses utilisateurs)
- User Stories : les tâches accomplies dans le cadre de chaque activité sont placées en dessous de celle-ci et ordonnées de haut en bas par « nécessité » (les plus indispensables en haut) et de gauche à droite selon l'ordre logique (comme les activités). En général, on a une ligne supérieure composée d' « Epic Stories » qui sont les macrotâches principales.
- Releases : une fois que les Stories ont été ordonnées, on découpe la carte en tranches qui constituent des Releases en partant des Stories de plus grande valeur.

EXEMPLE GESTION DES COMPTES BANCAIRES EN LIGNE

On peut imaginer le scénario typique suivant (vu du client de la banque) : je me connecte, je consulte la situation de mes comptes courants, je gère mes virements, je consulte l'état d'avancement de ma demande de prêt, je m'informe sur les produits financiers d'épargne.

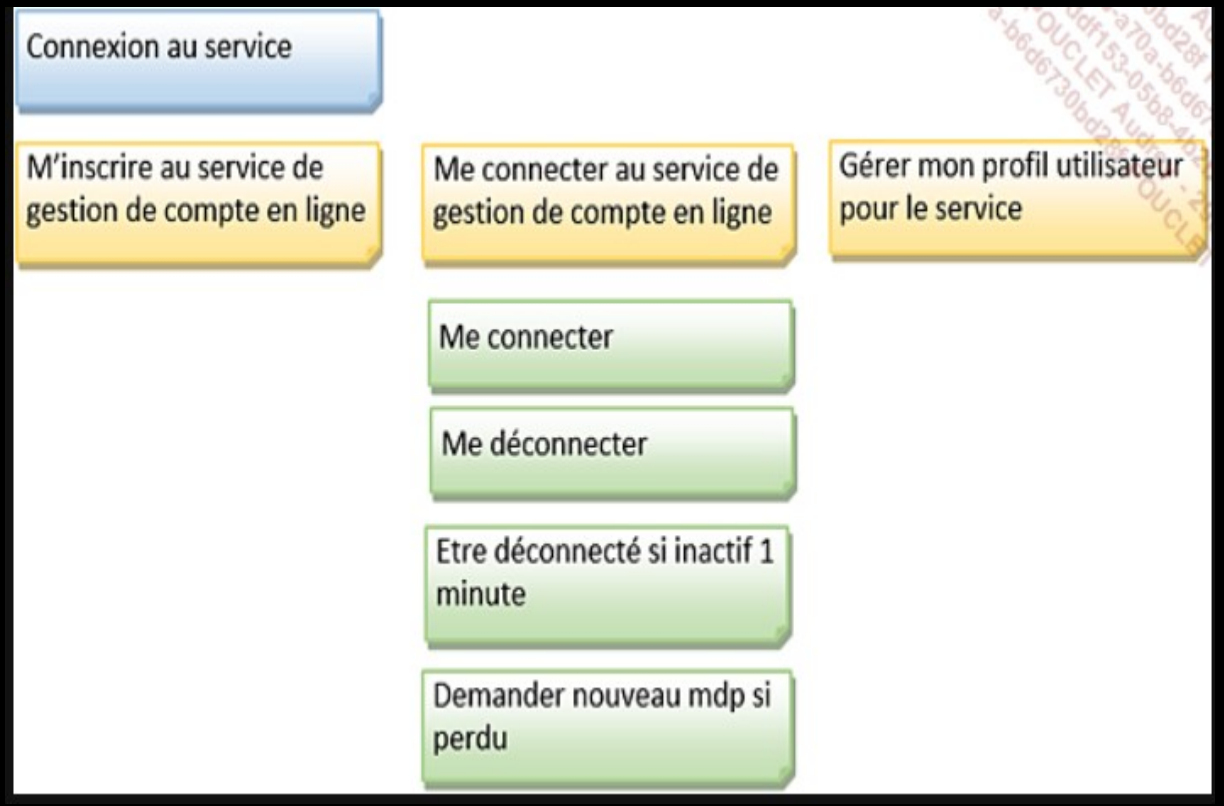
EXEMPLE GESTION DES COMPTES BANCAIRES EN LIGNE

- Dans le cadre de ces activités, en tant que client, on accomplit un certain nombre de tâches. Prenons l'exemple de l'activité « Connexion au service ». Dans cette thématique, je vais : demander mon inscription au service de gestion de compte en ligne, me connecter/déconnecter, gérer mon profil, etc.



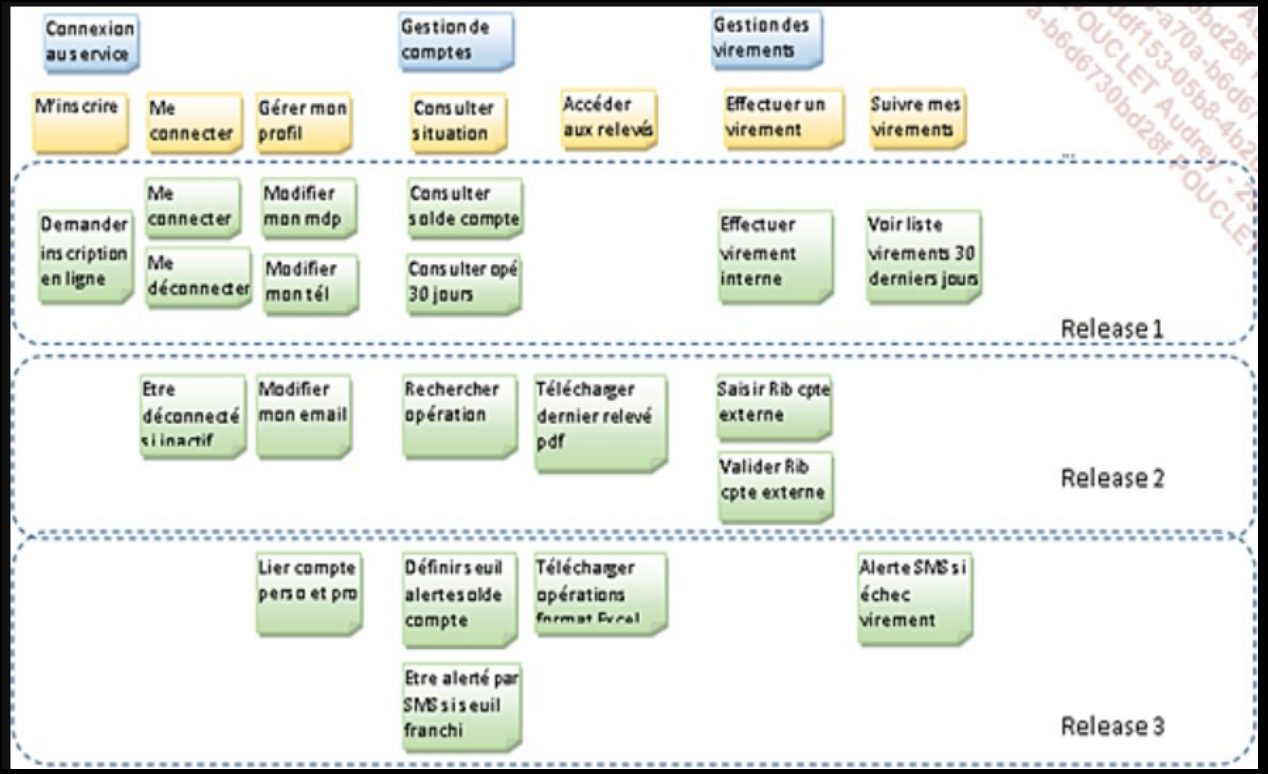
EXEMPLE GESTION DES COMPTES BANCAIRES EN LIGNE

- On peut ensuite le découper en US



- « En tant que client de la banque inscrit au service de gestion de comptes en ligne, je veux pouvoir me connecter au service en saisissant mon identifiant d'utilisateur et mot de passe afin de gérer mes comptes en ligne »
- « En tant que client de la banque inscrit au service de gestion de comptes en ligne, je veux pouvoir être déconnecté automatiquement du service au bout de 1 minute d'inactivité afin de ne pas laisser ma session ouverte »
- etc.

EXEMPLE GESTION DES COMPTES BANCAIRES EN LIGNE



- On va découper chaque activité en US puis on va les regrouper par ordre de nécessité et ranger celles qui sont les plus nécessaires en haut
- On les regroupe ensuite pour obtenir une version cohérente du système qui peut être mise en production (une release)

COMMENT PRIORISER LE BACKLOG?LES FACTEURS DE PRIORISATION

- La valeur financière : par exemple par l'acquisition de nouveaux clients ou marchés, ou par des réductions de coûts…
- Le désir du client/de l'utilisateur final pour une fonctionnalité.
- Le facteur de risque
- Le coût de développement.
- Le gain de connaissances : l'effet peut être par exemple d'améliorer la compréhension du besoin (du point de vue des utilisateurs ou du PO) ou d'affiner la validation de telle ou telle solution technique.
- L'amélioration de la qualité : si la User Story permet d'augmenter significativement la qualité du livrable, il peut être important de la développer rapidement.
- La dépendance entre Stories : si une user story est nécessaire à toutes les autres, il faut qu'elle soit développer en premier (par exemple pour les comptes pouvoir s'identifier)

BEAUCOUP DE METHODES DE PRIORISATION

- la priorisation par les thèmes (screening, notation, poids relatifs),
- la priorisation via le modèle de Kano,
- le « Priority Poker »,
- l’analyse financière,
- l’analyse hiérachique,
- la méthode MoSCoW,
- la méthode WSJF,
- ...

LE MODÈLE KANO (DU NOM DE L'INVENTEUR)

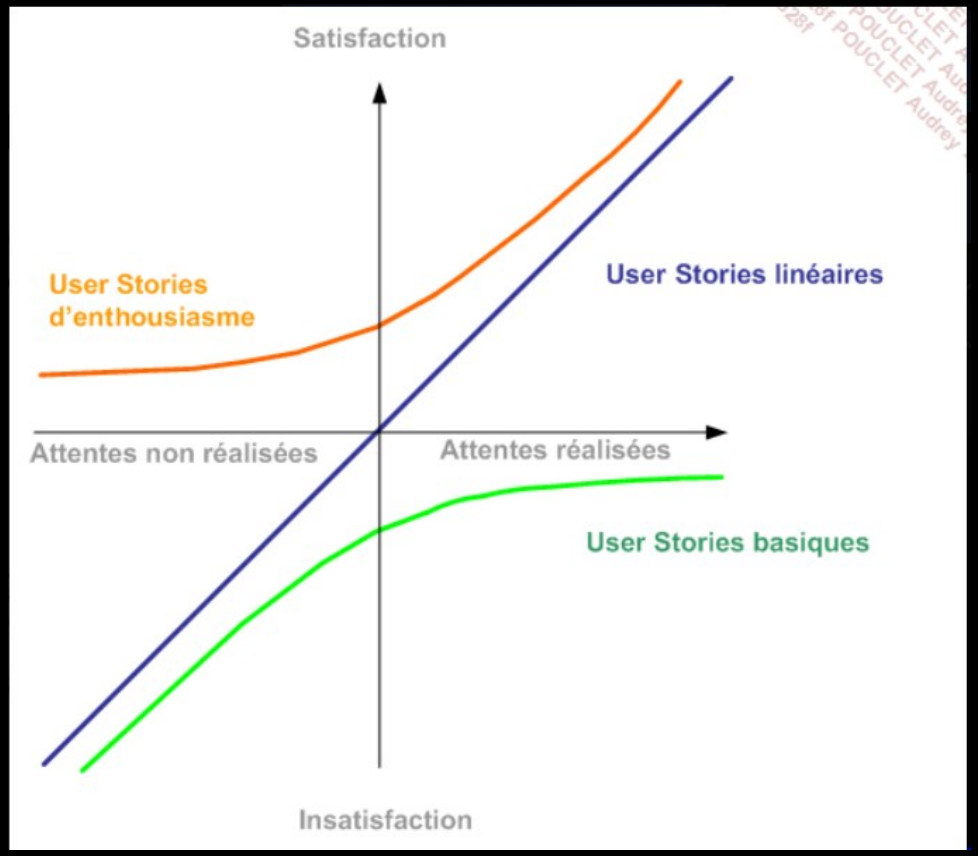
- Il repose sur le constat que la satisfaction et l'insatisfaction résultant chez un observateur de la perception d'un produit ou d'un service ne sont pas des concepts de valeur symétrique.
- Il s'agit de classer les US en trois types :

→Les US basiques (obligatoires) elles expriment **un besoin fondamental dont l'application ne peut pas se passer.**

→Les US linéaires elles expriment un besoin de qualité et parfois le petit plus qui fera la différence

→Les US d'enthousiasme Le besoin n'est pas forcément présent, mais si le produit le permet cela serait un réel avantage (le waouh ça serait super!!)

LE MODÈLE KANO



La règle est généralement :

- Prise en compte de **toutes** les User Stories basiques.
- Prise en compte d'**une partie** des User Stories linéaires.
- Prise en compte de **quelques** User Stories d'enthousiasme.

MODÈLE KANO PRISE EN COMPTE DE L'AVIS DES UTILISATEURS

- Que ressentez-vous si cette fonctionnalité est développée en premier lieu ? (question fonctionnelle)
- Que ressentez-vous si cette fonctionnalité n'est pas développée tout de suite ? (question dysfonctionnelle)

Le ressenti d'un utilisateur pourra s'exprimer de la manière suivante :

- "Je suis neutre" (Pas d'avis en particulier).
- "J'aime" (Je trouve cette idée sympa).
- "Je n'aime pas" (Je n'en vois pas l'intérêt).

		Question dysfonctionnelle		
		Aime	Neutre	Aime pas
Question fonctionnelle	Aime	?	**Enthousiasme**	**Linéaire**
	Neutre	Contradictoire	Indifférent	**Basique**
	Aime pas	Contradictoire	Contradictoire	?

LA METHODE MOSCOW

Les besoins sont positionnés en 4 catégories:

- **M** : « Must have this » en anglais, c'est-à-dire « doit être fait ». Il s'agit d'un besoin vital qui sera nécessairement en haut de Backlog (donc à faire avec le plus haut niveau de priorité).
- **S** : « Should have this if at all possible », c'est-à-dire « devrait être fait dans la mesure du possible ». Il s'agit de besoins essentiels mais qui passent après la première catégorie.
- **C** : « Could have this if it does not affect anything else », c'est-à-dire « pourrait être fait dans la mesure où cela n'a pas d'impact sur la réalisation des autres sujets ». On parle ici plutôt de User Stories « de confort », c'est-à-dire qu'elles ne sont pas indispensables et qu'on n'en priorisera la réalisation que s'il n'y a pas d'autre besoin des catégories M et S à traiter.
- **W** : « Won't have this time but would like in the future », c'est-à-dire « ne sera pas fait cette fois (peut-être plus tard) ».

LA METHODE WSJF (WEIGHTED SHORTEST JOB FIRST)

- Dans la priorisation du Backlog, on a intérêt à traiter en premier les Stories dont le rapport Bénéfice/Effort est le plus grand possible.
- WSJF= (valeur métier + réduction du risque + impact du délai)/ effort

→La valeur métier : on donne un score qui reflète la valeur apportée aux utilisateurs/clients.

→Risque de ne pas faire : on donnera d'autant plus de points qu'il y a un niveau de risque élevé si on ne traite pas le sujet.

→Impact du délai : en gros, est-ce qu'il faut faire vite ou pas ? Plus la réponse est oui, plus le nombre de points sera important. On peut l'évaluer aussi en se posant des questions du type : y a-t-il une date limite ou la valeur diminue-t-elle avec le temps ?

→Effort : il s'agit de l'effort de réalisation, idéalement exprimé avec une échelle de type « Story Points »

LA METHODE WSJF EXEMPLE

Trois Stories relatives à un site de vente en ligne et priorisation avec la valeur métier seulement :

Story	Valeur métier (de 0 à 10)
Ajouter un écran de consultation des demandes faites au support client	5
Mettre à jour les mentions légales du site web	3
Corriger un petit bug agaçant : libellés coupés à l'affichage d'un tableau	1

LA METHODE WSJF EXEMPLE

- les mêmes Stories avec l'effort et classement des demandes selon le ratio Valeur/Effort :

Story	Valeur métier	Effort	Valeur/Effort
Corriger un bug agaçant : libellés coupés à l'affichage d'un tableau	1	0,5	2
Mettre à jour les mentions légales du site web	3	3	1
Ajouter un écran de consultation des demandes faites au support client	5	8	0,625

LE CLASSEMENT EST INVERSE

LA METHODE WSJF EXEMPLE

- Une nouvelle loi peut mettre le site dans l'illégalité

Story	Valeur métier	Réduction du risque	Effort	CoD/Effort
Mettre à jour les mentions légales du site web	3	10 *	3	4,33
Corriger un bug agaçant : libellés coupés à l'affichage d'un tableau	1	0	0,5	2
Ajouter un écran de consultation des demandes faites au support client	5	2	8	0,875