



# AGILE

## Approche sur la Methode de conduite de projet



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | <b>Accueil des Stagiaires</b>     | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Tour de Table



## Présentation de l'animateur :

- Formation animée par un consultant Séniior habitué à délivrer des formations
  - Culture générale :
    - Connaissance des SI
    - Connaissance des modèles de cycle de développement : Cycle en « V », Agile, ..
    - Connaissance des cadres de références standards :
      - CMMi, TMMi, ITIL, COBIT, REQB, ISO, Six sigma ..
  - Principaux rôles:
    - Formation
    - Conseil, Accompagnement
    - Business développement
    - Organisation d'équipe, pilotage
    - Pilotage de projets
    - Audit...

# Accueil des Stagiaires

## Présentation des stagiaires (*tour de table*)

- Guide de questions :
  - Qui êtes vous (nom, prénom) ?
  - Quel est votre parcours en termes de formations ?
  - Que faites vous : Société, Alternance, Rôle, .. ?
  - Quelle est votre expérience : Gestion de projet, UML ?

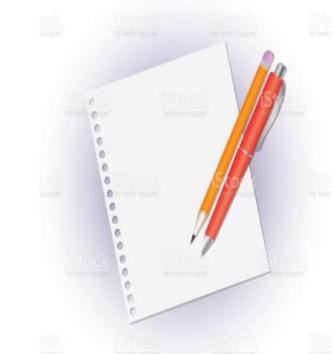


# Accueil des Stagiaires



## Présentation des informations pratiques

- Thème de la formation : « **Approche Méthodologie Agile** »
- Durée de la formation : 3 journées
  - Horaires de la formation : 9h30 à 17h30
- Pauses
  - Café (2 fois par jour) : 10 minutes
  - Pause repas : 1h15 heure (environ)
- Sont utiles pour cette formation
  - PC
  - Feuille, Stylos,...



# Accueil des Stagiaires



## Agenda

- Thèmes et agenda proposés
  - Prise en compte de l'avancement de vos projets
  - Partage sur vos réalisation

|          | Matin             | Après-midi        |
|----------|-------------------|-------------------|
| Lundi    | Présentation,     | Présentation + TP |
| Mardi    | Présentation + TP | Présentation + TP |
| Mercredi | Présentation + TP | Présentation + TP |
|          |                   |                   |
|          |                   |                   |

# AGILE : Approche sur la Méthode de conduite de projet



## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | <b>Cycle de vie Projet</b>        | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Cycle de vie d'un Projet informatique



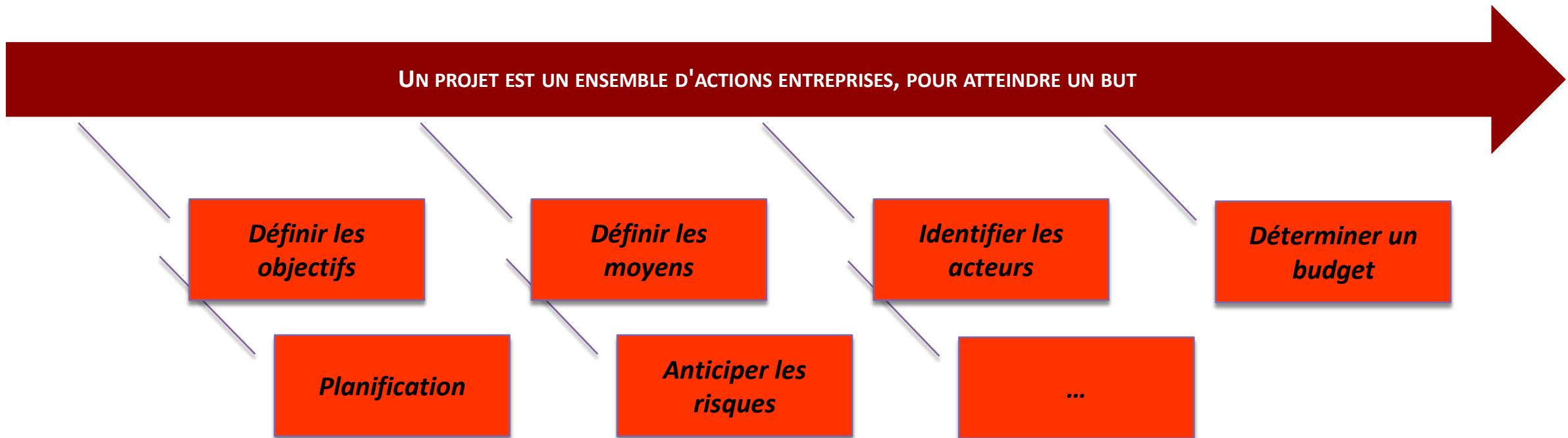
## Notions

- Le cycle de vie d'un projet informatique est la description d'un processus couvrant les phases majeures de :
  - Création d'un produit
  - Distribution sur un marché
  - Disparition
- L'origine de ce découpage provient du constat que
  - les erreurs ont un coût d'autant plus élevé qu'elles sont détectées tardivement dans le processus de réalisation
  - Le cycle de vie permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel
- Le but de ce découpage est principalement de
  - Maîtriser les risques
  - Maîtriser au mieux les délais et les coûts
  - Obtenir une qualité conforme aux exigences

# Cycle de vie d'un Projet informatique

## Définition du projet informatique

- Projet
  - Ensemble unique d'activités, contrôlées et coordonnées, avec des dates de début et de fin, effectuées avec pour objectif de conformité à des exigences spécifiques, incluant des contraintes de temps, de coût et de ressources.



# Cycle de vie d'un Projet informatique



## SDLC : System Development Life Cycle

- Le SDLC est le cycle de vie d'un processus de développement logiciel qui est utilisé par les DSI pour désigner toutes les étapes du développement d'un logiciel, de sa conception à sa disparition.
  - Vise à produire un logiciel de qualité qui respecte les attentes des clients, en termes de délais, de coût et de qualité.
  - Est un cadre définissant les tâches effectuées à chaque étape du processus de développement logiciel
  - Définit une méthode pour améliorer la qualité des logiciels et l'ensemble du processus de développement
- L'objectif d'un tel découpage est de définir des jalons intermédiaires permettant
  - la validation du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés,
  - la vérification du processus de développement, c'est-à-dire l'adéquation des méthodes mises en œuvre
- Le cycle de vie permet de détecter les erreurs au plus tôt et ainsi de maîtriser la qualité du logiciel, les délais de sa réalisation et les coûts associés

# Cycle de vie d'un Projet informatique

## SDLC : Processus



- Phase 1 : Planification et analyse des besoins :
  - L'analyse des besoins est effectuée par les acteurs dédiés avec la contribution des parties prenantes (client, utilisateur final, experts de domaine métier, etc..) . Les informations recueillies permettent de planifier l'approche globale du projet et de mener une étude de faisabilité du produit en accord avec les enjeux économiques, opérationnels et techniques
- Phase 2 : Définition des exigences :
  - Une fois l'analyse des besoins effectuée, cette étape consiste à définir et à documenter clairement les exigences du produit et les faire approuver par les parties prenantes
- Phase 3 : Conception du produit :
  - Sur la base des exigences spécifiées, une approche de conception du produit est documentée et est examinée par l'ensemble des parties prenantes du projet. Différentes informations comme l'évaluation des risques, le périmètre du produit, les choix de conception, les contraintes budgétaires et temporelles, les contraintes d'architecture (internes et externes) du produit sont clairement définies, formalisées et validées.

# Cycle de vie d'un Projet informatique

## SDLC : Processus



- Phase 4 : Développement du produit :
  - Le développement réel peut alors commencer, sur la base du langage de programmation défini. Les développeurs suivent les directives de codification et utilisent les outils de programmation tels que des compilateurs, des interprètes et des débogueurs. Le langage de programmation est choisi par rapport au type de logiciel en cours d'élaboration
- Phase 5 : Testing du produit :
  - Cette étape est généralement un sous-ensemble des activités de Testing qui sont mises en œuvre dans toutes les étapes du SDLC. Cette étape se réfère à l'exécution dynamique des tests, où les anomalies du produit sont signalées, suivies, fixées et re-testées, jusqu'à ce que le produit atteigne les normes de qualité définies et attendues
- Phase 6 : Déploiement et maintenance :
  - Une fois que le produit est testé, il est déployé formellement dans les environnements appropriés. De plus jusqu'à la fin de vie du produit toutes les évolutions ou corrections du produit peuvent être mises en œuvre en respect du SDLC.

# Cycle de vie d'un Projet informatique

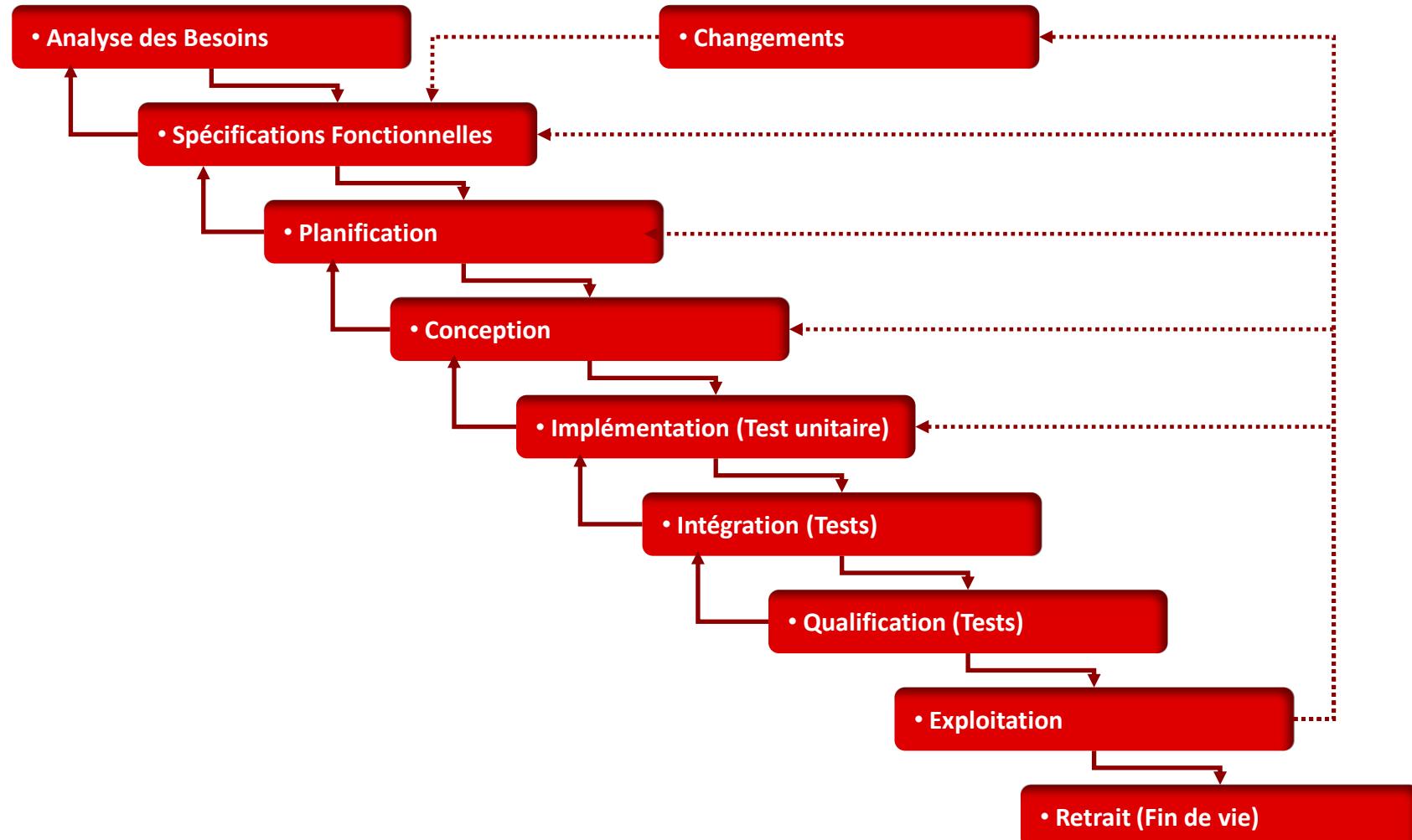


## SDLC : Processus

- Différents SDLC
  - Séquentiel : V, Cascade
  - En spirale
  - Agile : Scrum, RAD (Rapid Application Development)
- S'il y a plusieurs modèles, c'est que :
  - Aucun n'est parfait ni même meilleur que les autres
  - Tous ont des qualités et des défauts selon les contextes d'utilisation
- Chaque entreprise a défini ses propres SDLC et sa propre méthode de conduite de projet basés sur les SDLC "standards"
- A chaque début de « Projet », il est important de comprendre le contexte dans lequel vous évoluez :
  - Phases du processus SDLC et ses objectifs
  - Projet de Conception & de Développement
  - Points d'accroche avec le SDLC et les autres entités

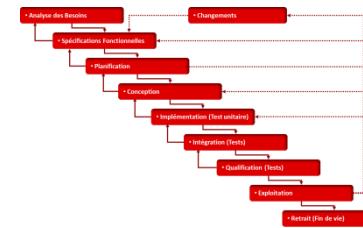
# Cycle de vie d'un Projet informatique

## SDLC : Modèle Cycle en cascade



# Cycle de vie d'un Projet informatique

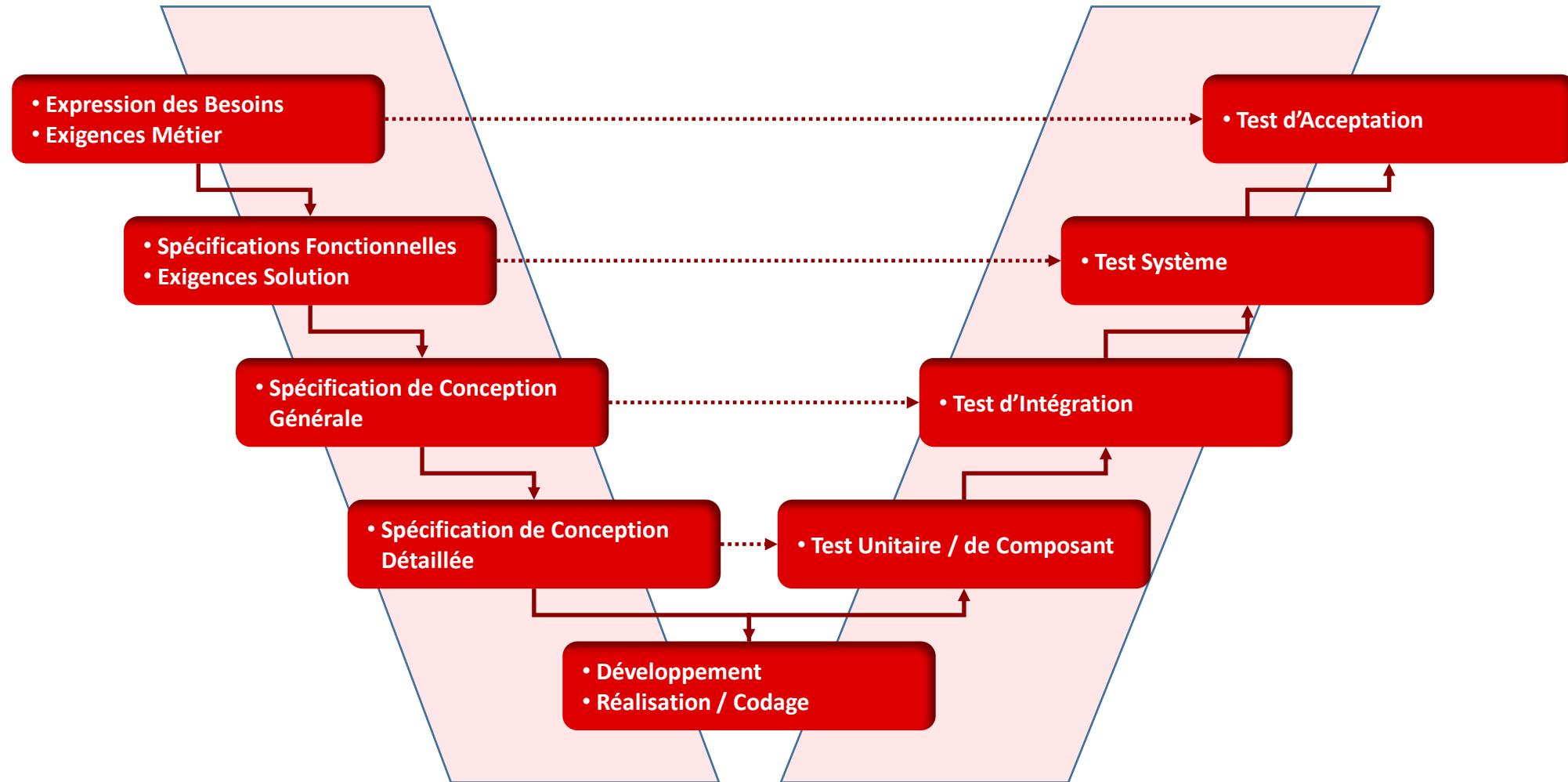
## SDLC : Modèle Cycle en cascade



- Le modèle de cycle de vie en cascade a été mis au point dès 1966, puis formalisé aux alentours de 1970.
- Ce modèle repose sur les principes suivants :
  - Chacune phases se termine à une date précise par la production de certains documents ou logiciels
  - Les résultats sont définis sur la base des interactions entre étapes et sont soumis à une revue approfondie
  - On ne passe à la phase suivante que s'ils sont jugés satisfaisants
- Le modèle original ne comportait pas de possibilité de retour en arrière. Celle-ci a été rajoutée ultérieurement sur la base qu'une étape ne remet en cause que l'étape précédente, ce qui, dans la pratique, s'avère insuffisant.
- L'inconvénient majeur du modèle de cycle de vie en cascade est que la vérification du bon fonctionnement du système est réalisée trop tardivement : lors de la phase d'intégration, ou pire, lors de la mise en production.

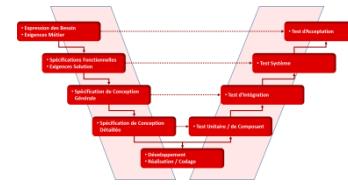
# Cycle de vie d'un Projet informatique

## SDLC : Modèle Cycle en V



# Cycle de vie d'un Projet informatique

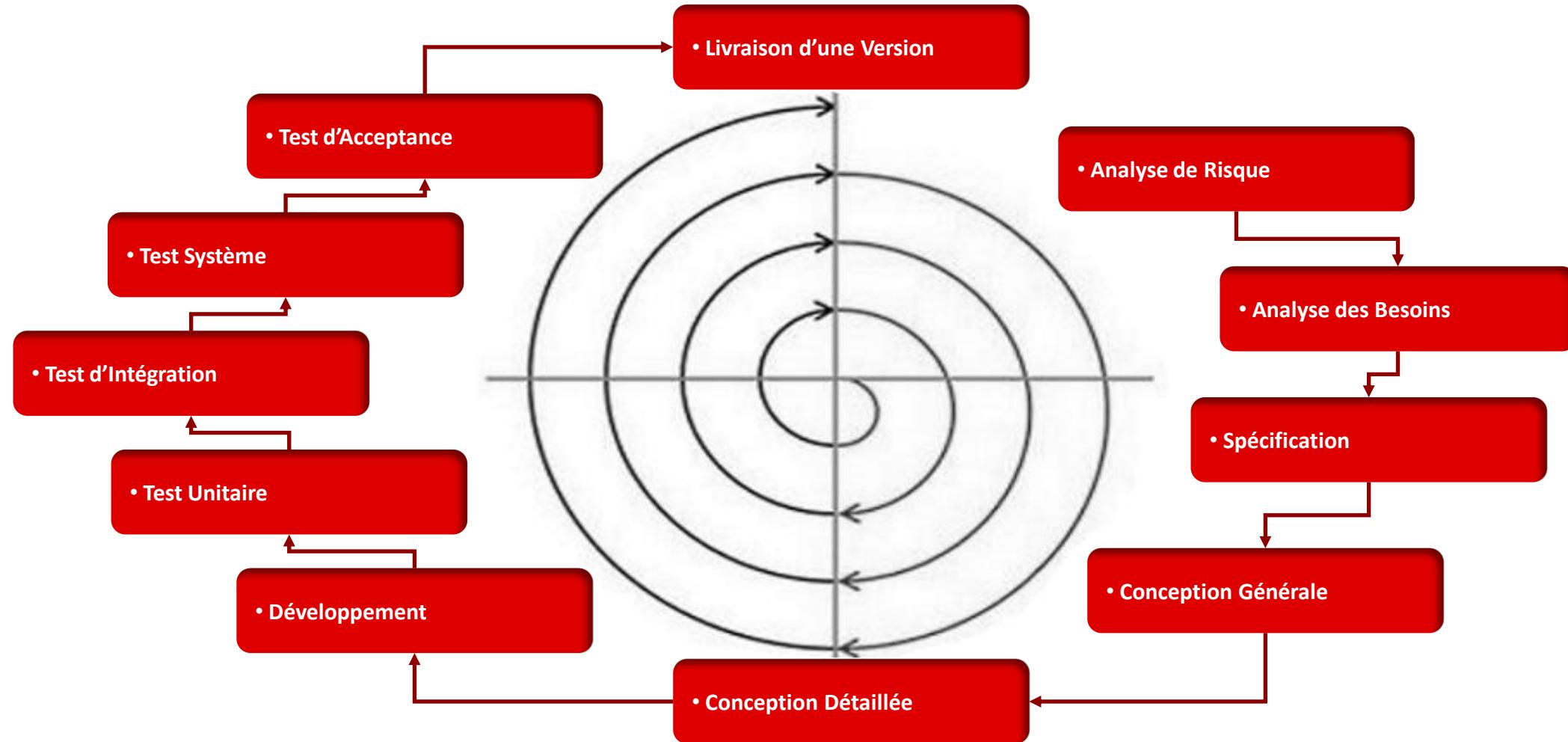
## SDLC : Modèle Cycle en V



- Le modèle en V demeure actuellement le cycle de vie le plus connu et certainement le plus utilisé. Il s'agit d'un modèle en cascade dans lequel le développement des tests et du logiciel sont effectués de manière synchrone.
- Le principe de ce modèle est qu'avec toute décomposition doit être décrite la recomposition et que toute description d'un composant est accompagnée de tests qui permettront de s'assurer qu'il correspond à sa description.
- Ceci rend explicite la préparation des dernières phases (validation-vérification) par les premières (construction du logiciel), et permet ainsi d'éviter un écueil bien connu de la spécification du logiciel : énoncer une propriété qu'il est impossible de vérifier objectivement après la réalisation.
- Cependant, ce modèle souffre toujours du problème de la vérification trop tardive du bon fonctionnement du système

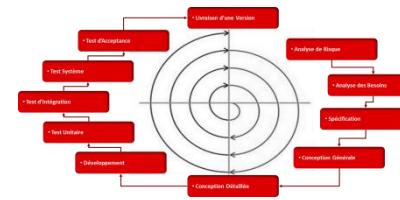
# Cycle de vie d'un Projet informatique

## SDLC : Modèle Cycle en Spirale



# Cycle de vie d'un Projet informatique

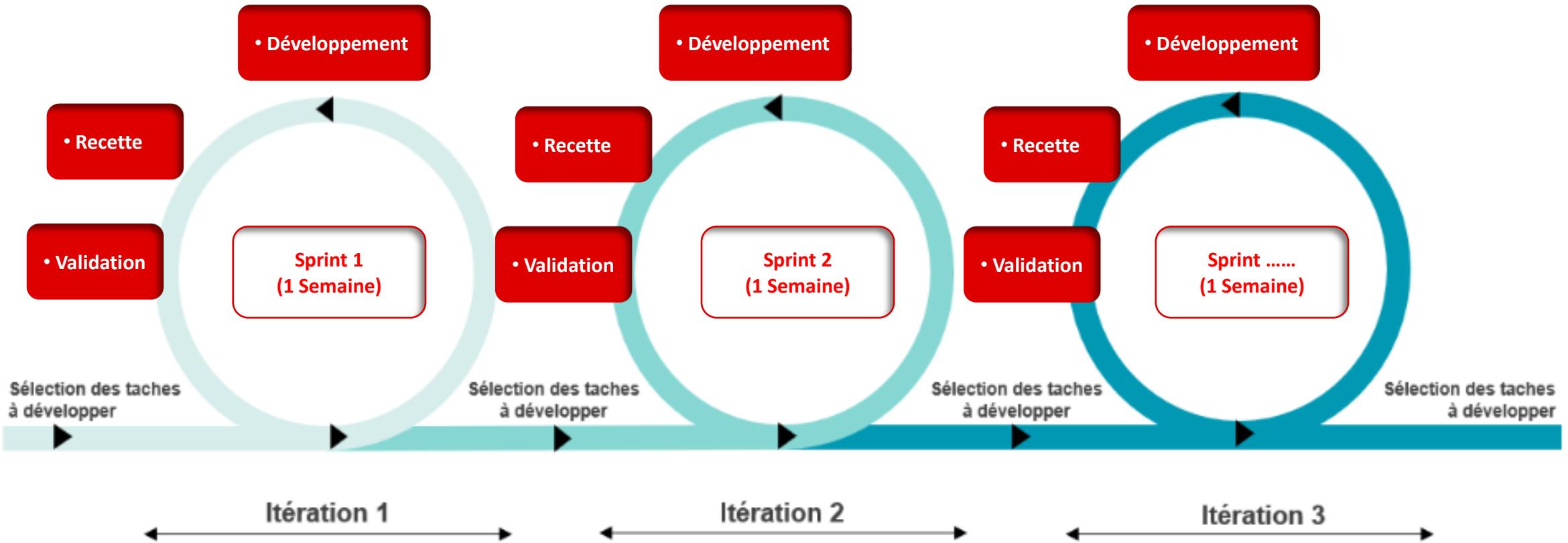
## SDLC : Modèle Cycle en Spirale



- Ce modèle est beaucoup plus général. Il met l'accent sur l'activité d'analyse des risques : chaque cycle de la spirale se déroule en quatre phases :
  - Détermination, à partir des résultats des cycles précédents, ou de l'analyse préliminaire des besoins, des objectifs du cycle, des alternatives pour les atteindre et des contraintes
  - Analyse des risques, évaluation des alternatives et, éventuellement maquettage
  - Développement et vérification de la solution retenue, un modèle « classique » (cascade ou en V) peut être utilisé ici
  - Revue des résultats et vérification du cycle suivant.
- L'analyse préliminaire est affinée au cours des premiers cycles. Le modèle utilise des maquettes exploratoires pour guider la phase de conception du cycle suivant. Le dernier cycle se termine par un processus de développement classique.

# Cycle de vie d'un Projet informatique

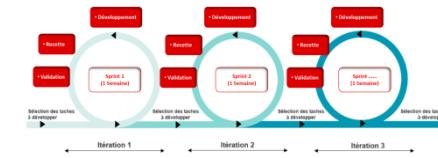
## SDLC : Modèle Cycle Agile



# Cycle de vie d'un Projet informatique

## SDLC : Modèle Cycle Agile

- La méthode Scrum (« mêlée ») est la méthode Agile la plus connue et la plus utilisée dans l'écosystème actuel. Elle est considérée comme une méthode de management de projet structurée et rapidement déployable.
- La méthode Scrum offre la possibilité de piloter les avancées d'un projet sous la forme d'itérations (plus communément appelées « Sprints »).
  - Les Sprints durent en moyenne 1 à 3 semaines et reposent sur un objectif précis : délivrer continuellement de la valeur aux parties prenantes (équipes métier, clients, utilisateurs) toutes associées à un même produit/service.
- La sélection des tâches à développer repose sur la priorisation de l'ensemble des actions à mener
  - Toutes les tâches sont importantes, le Product Owner intervient et est Garant du Backlog (liste complète des tâches à développer), il priorise pour chaque sprint les tâches prioritaires à adresser selon la capacité de production de l'équipe de développement.
- Le développement et la recette s'appuient sur une collaboration permanente entre l'équipe de développement, le Product Owner et le Scrum Master. L'objectif est de minimiser les frictions et les événements bloquants afin de garantir la livraison des tâches associées au Sprint.
- La validation est souvent assurée par le Product Owner et les parties-prenantes externes si le besoin est exprimé



## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | <b>Approche Méthode AGILE</b>     | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Approche : Méthodes Agiles



## Cycle en « V » : Les points de « douleurs »

- L'immuabilité des spécifications et des besoins fonctionnels
  - Ce qui sera délivré sera en partie obsolète
- L'effet tunnel !
  - Feedback trop tardif : toute modification va coûter très cher
- La sectorisation des intervenants
  - Les équipes s'incriminent entre elles et Insatisfaction des équipes
- On oublie où se trouve la valeur du projet et on se concentre sur le respect à la lettre des spécifications (or elles sont souvent mal comprises, et surtout elles changent !) et de la documentation
- On ne délivre pas assez de valeur au client
- Nécessité de documenter de façon détaillée tous les éléments du projet

# Approche : Méthodes Agiles

## Introduction à l'Agilité

- Pour répondre à ce problème est créé en 2001 le manifeste Agile.
- 17 experts qui se dressent contre l'échec des méthodes traditionnelles, rédigent et proposent 4 valeurs fondamentales, et 12 principes



# Approche : Méthodes Agiles

## Les 4 valeurs de l'Agilité

1. La réactivité face au changement plutôt que le suivi d'un plan
2. l'interaction avec les personnes plutôt que les processus et les outils
3. un produit opérationnel plutôt qu'une documentation pléthorique
4. La collaboration avec le client plutôt que la négociation de contrat



# Approche : Méthodes Agiles



## Les 12 principes de l'Agilité (1/2)

1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins, même tard dans le projet. Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
5. Réalisez les projets avec des personnes motivées.  
Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
6. La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.

# Approche : Méthodes Agiles



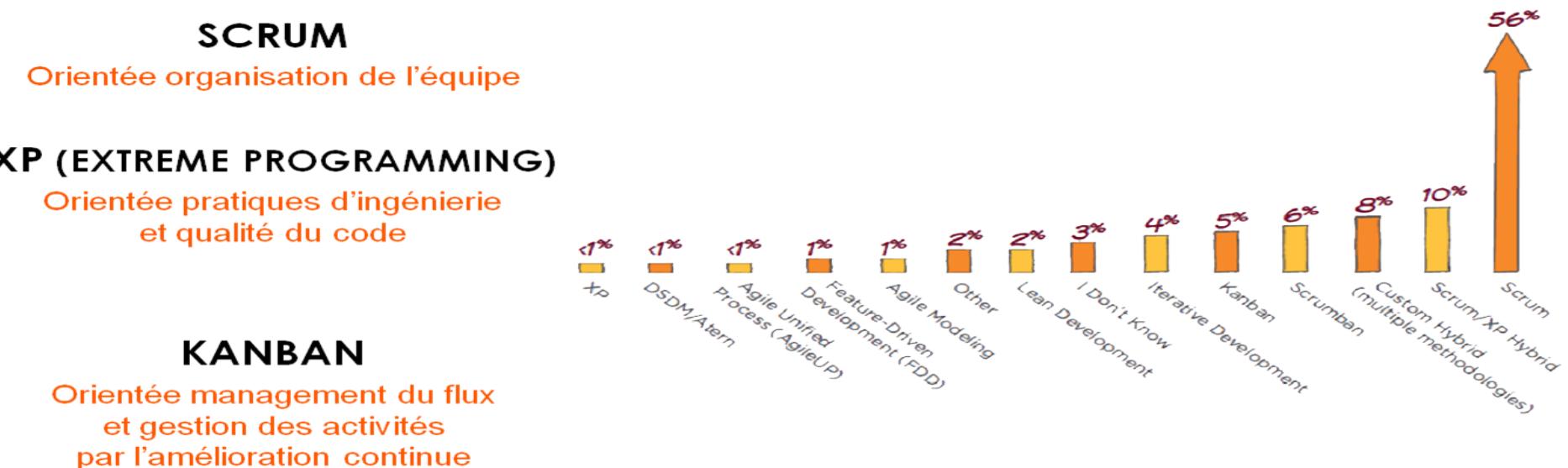
## Les 12 principes de l'Agilité (2/2)

7. Un logiciel opérationnel est la principale mesure d'avancement.
8. Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
9. Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d'équipes auto organisées.
12. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

# Les Méthodes Agiles

## Méthodes Agiles

- Les méthodes agiles caractérisent un mode de gestion des projets informatiques privilégiant le dialogue entre toutes les parties prenantes, clients, utilisateurs, développeurs et autres professionnels du projet, la souplesse en cours de réalisation, la capacité à modifier les plans et la rapidité de livraison.
- Les méthodes Agiles trouvent ainsi plusieurs déclinaison.



# Les Méthodes Agiles



## Les méthodes Agiles : les plus connues

| SCRUM   | KANBAN  | RAD<br>(Rapid Application Development)   | XP<br>(eXtreme Programming)  | DSM Dynamic (Systems Development Method)   |
|---|---|--|--|--|
| <ul style="list-style-type: none"><li>Méthodologie de gestion de projet issue du terme « mélée » (Rugby) en anglais faisant référence au fait d'avancer ensemble vers un but commun</li><li>Pas fondamentalement liée au développement logiciel</li></ul> | <ul style="list-style-type: none"><li>KANBAN est un terme japonais qui signifie « étiquette ». Elle a été créée par Taiichi Ōno pour Toyota en 1950 dans le but d'optimiser sa capacité de production afin d'être compétitive face aux entreprises américaines, puis formalisée en 2010 par Davis Anderson.</li><li>La méthode Kanban se base sur l'approche Lean, c'est-à-dire sur l'amélioration continue des processus de production afin de permettre une gestion de la production sans gaspillage.</li><li>Son principe de base est la limitation du nombre de travaux à faire (TAF) afin d'éviter le gaspillage. L'équipe définit les limites du TAF pour chaque étape du processus et le flux du travail est contrôlé visuellement ; il s'agit d'un flux tiré. L'équipe peut suspendre le processus afin de résoudre un problème bloquant.</li></ul> | <ul style="list-style-type: none"><li>Première méthode de développement de logiciels où le cycle de développement est en rupture fondamentale à celui des méthodes antérieures dites « en cascade »</li><li>Ce nouveau cycle qualifié d'itératif, d'incrémental et d'adaptatif, se retrouvera dans toutes les méthodes dites « agiles » publiées par la suite.</li></ul> | <ul style="list-style-type: none"><li>Crée par Kent Beck entre 1996 et 1999, lorsqu'il travaillait sur un projet pour Chrysler.</li><li>Elle fonctionne pour tous types de projets, de toutes tailles et de tous secteurs confondus, partout dans le monde.</li><li>Cette méthodologie est idéale pour de petites et moyennes équipes, c'est-à-dire pas plus d'une vingtaine de personnes.</li></ul> | <ul style="list-style-type: none"><li>Cette méthode a été développée en Grande-Bretagne à partir de 1994.</li><li>Méthode agile mais pas forcément IT</li><li>Une méthode agile beaucoup moins connue que le Scrum car elle est moins appréciée par les coachs agiles français</li></ul> |

## Méthodes Agiles : Valeurs ajoutées

Pourquoi cela  
marche

- Priorité à ce qui a le plus de valeur, à ce qui est le plus important
  - Démarche itérative, incrémentale et adaptative
  - Des interactions et de la communication
  - Un outillage compact et rapidement assimilable



- De la visibilité
- De la motivation et de la satisfaction dans les équipes
- Un produit opérationnel très tôt
- Une réactivité face au changement

Mais ce n'est  
pas

- L'absence de règle :
  - La discipline est indispensable
  - Les processus sont indispensables
- L'absence de documentation
  - Il faut des spécifications si elles ont de la valeur
- Magique
  - Cela ne fonctionne pas dans tous les contextes
  - Il n'y a pas de checklist Scrum !

## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                                   | N° | Thèmes                              |
|----|--|----|-------------------------------------|
| 01 | Accueil des Stagiaires                   | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet                      | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE                   | 12 | Le KanBan                           |
| 04 | <b>Approche Ingénierie des Exigences</b> | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                                  | 14 | CMMi                                |
| 06 | La méthode SCRUM                         | 15 | Approche TDD & BDD                  |
| 07 | User Stories                             | 16 | SAFe                                |
| 08 | Le Product Backlog                       | 17 | Quizz Scrum                         |
| 09 | Le Sprint                                |    |                                     |

# Approche sur la notion d'Exigences

## Cadrer son besoin / Fixer les exigences

- Une fois le projet acté, l'équipe métier aura en charge de rédiger une première version de la note de cadrage du projet.
- Ce document aura aussi pour objet de fixer l'ensemble des exigences attendues pour répondre aux objectifs du projet.
  - Il précisera, entre-autres, les exigences fonctionnelles et non fonctionnelles.
- Exigences : Une stipulation qui énonce des critères à satisfaire.
- Exigence fonctionnelle : Une exigence qui spécifie une fonction qu'un composant ou système doit être capable de remplir.
- Exigence non-fonctionnelle : Une exigence qui spécifie comment les fonctions peuvent être utilisées facilement et efficacement. Elle décrit comment le système se comporte.

Une EXIGENCE est [d'après IEEE 610] :

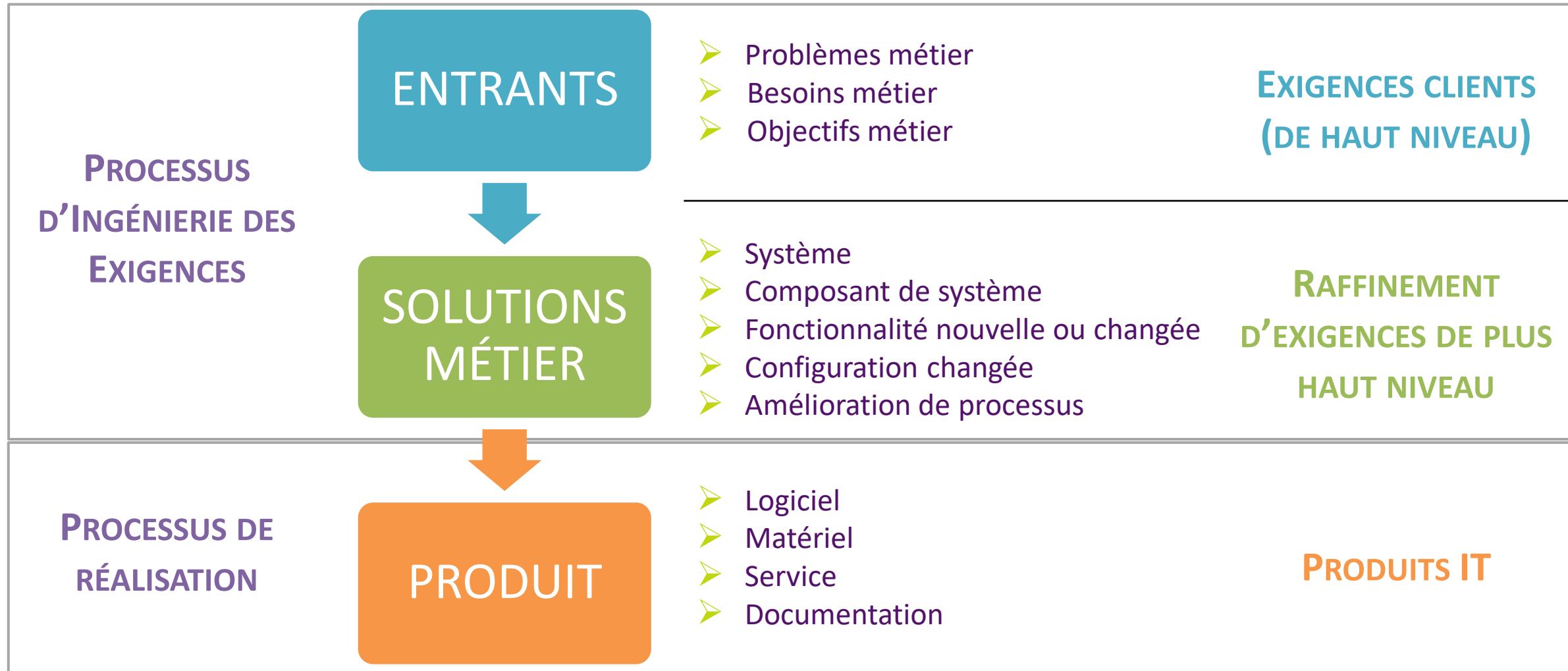
Une condition ou capacité requise par un utilisateur pour résoudre un problème ou atteindre un objectif.

Une condition ou capacité qui doit être tenue ou possédée par un système ou composant pour satisfaire à un contrat, standard, spécification ou autre document imposé formellement.

Document représentant une condition ou une aptitude comme décrit ci-dessus.

# Approche sur la notion d'Exigences

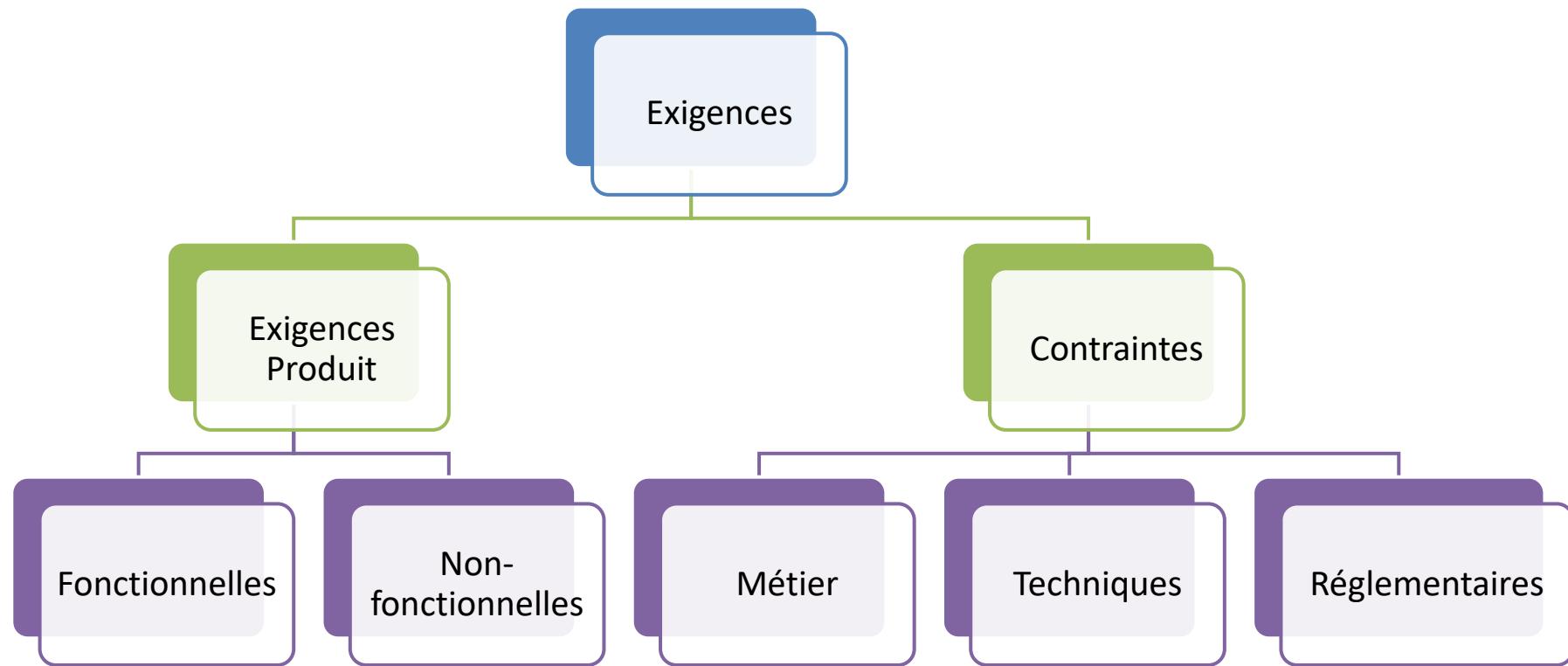
Concepts Problème, Solution, Produit



# Approche sur la notion d'Exigences

## Les catégories des exigences

- Les exigences peuvent être classées en Types suivant qu'elles traitent les différents aspects du produit (Exigences Produit) ou de son processus de développement (Contraintes)



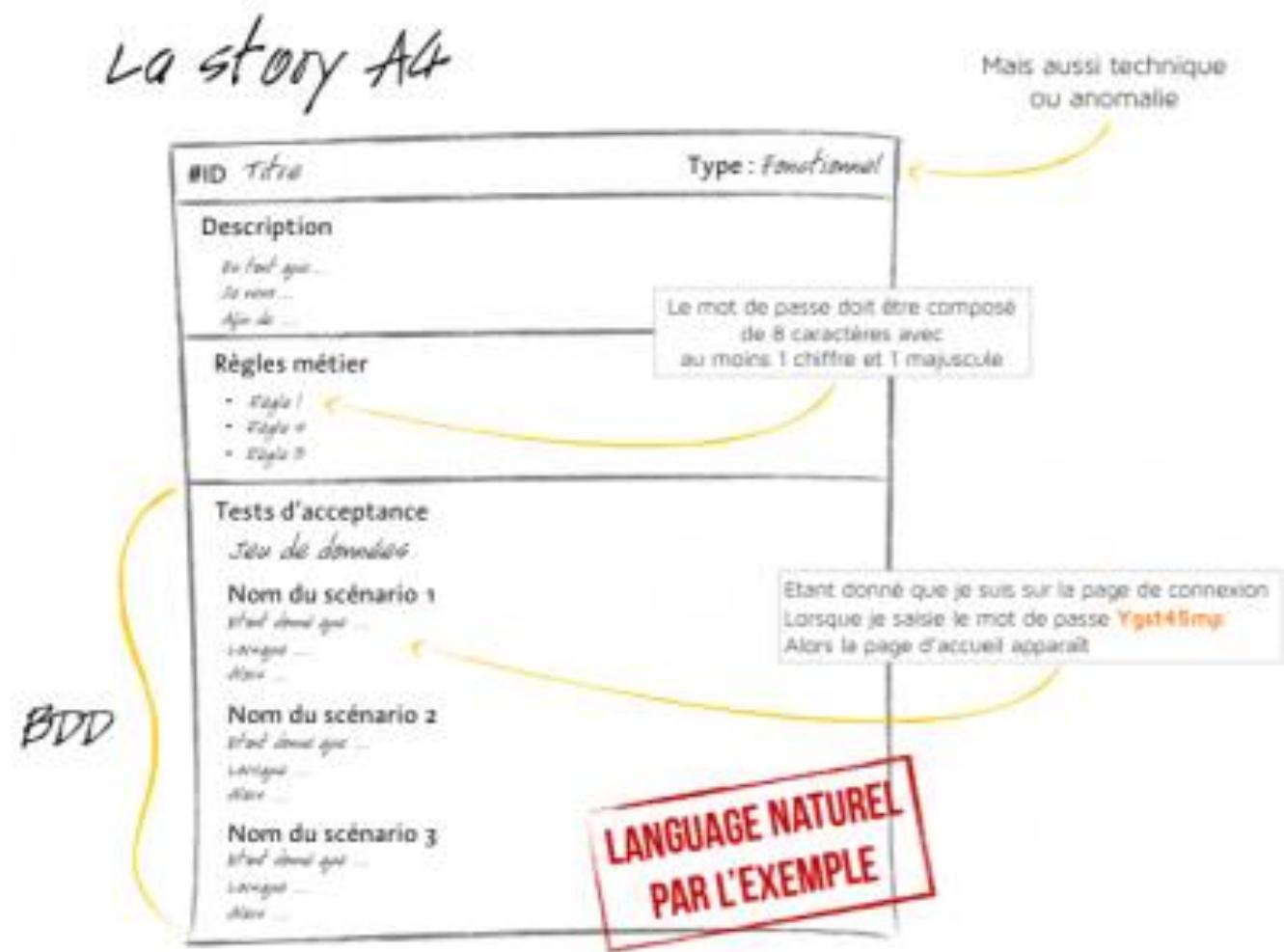
# Approche sur la notion d'Exigences

## Attributs d'une exigence

| Engagement   | Priorité   | Criticité   |
|--|--|---|
| <ul style="list-style-type: none"> <li>• Est le degré d'obligation pour satisfaire à l'exigence.</li> <li>• Mots-clés avant l'accord: « devrait », « pourrait », « peut »...</li> <li>• Après l'accord: « fera », « doit »...</li> <li>• Notation MOSCOW : Must have, Should have, Could have, Won't have</li> </ul> | <ul style="list-style-type: none"> <li>• Est l'importance / urgence d'une exigence.</li> <li>• Plus la priorité est haute, plus l'exigence est essentielle pour atteindre les objectifs globaux du produit.</li> <li>• À rapprocher du coût de développement.</li> </ul> | <ul style="list-style-type: none"> <li>• Est le résultat d'une évaluation du risque des dommages qui se produiraient si cette exigence n'est pas satisfaite.</li> <li>• Plus le niveau est élevé, plus les conséquences sont graves.</li> </ul> |

# Approche sur la notion d'Exigences

## Une User Story au format papier



# Approche sur la notion d'Exigences

## Les exigences fonctionnelles

- Les exigences fonctionnelles :
  - Spécifier ce que le produit doit faire
  - Décrire la fonctionnalité attendue du produit
  - Les exigences fonctionnelles décrivent ce QUE la solution fait.
- Les exigences fonctionnelles précisent :
  - Les fonctions du système telles qu'elles sont perçues par l'utilisateur final.
  - Les services offerts par le système.
  - Les déclencheurs du processus tels que les actions des utilisateurs ou les entrées / sorties de données qui font démarrer le processus métier.
  - Le comportement dans des cas exceptionnels.
  - Ce que le système ne doit pas faire.

Caractéristiques qualité [ISO / IEC

25000] :

- ✓ Pertinence
- ✓ Précision
- ✓ Interopérabilité
- ✓ Fonctionnalité
- ✓ Conformité
- ✓ Sécurité

# Approche sur la notion d'Exigences

## Les exigences non-fonctionnelles

- Déterminent comment les fonctions peuvent être utilisées facilement et efficacement.
- Décrire les attributs qualité attendus du système
- Décrire comment le système se comporte
- Décrire comment la solution exécute ses fonctionnalités
- Elles précisent :
  - Les attributs qualité de l'ensemble du système ou de ses composants et fonctions spécifiques.
  - Elles peuvent limiter la solution (critères d'efficacité spécifiques).
  - Elles peuvent décrire les différents aspects de la performance de la solution.

Caractéristiques qualité [ISO / IEC 25000] :

- ✓ Fiabilité
- ✓ Utilisabilité
- ✓ Efficacité
- ✓ Maintenabilité
- ✓ Portabilité

# Approche sur la notion d'Exigences

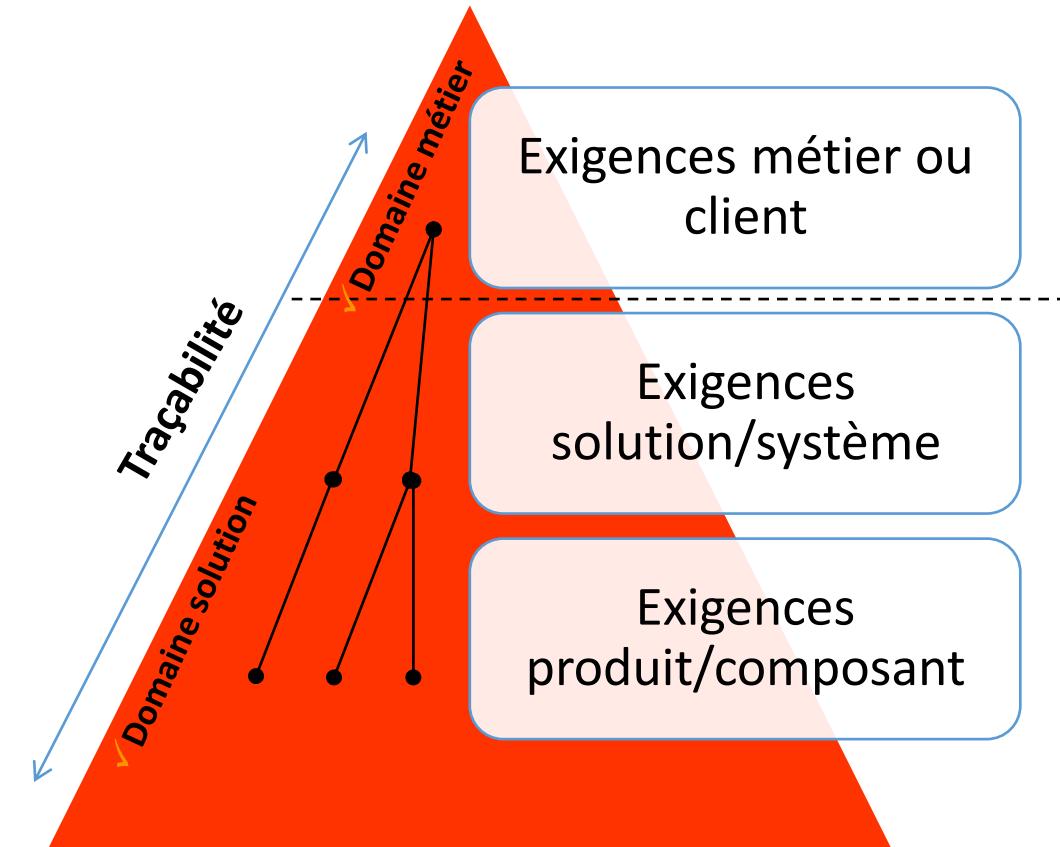


## Les exigences qualité

- Les exigences qualité doivent être documentées explicitement. Les aspects suivants doivent être considérés :
  - performance : Comportement dans la durée, utilisation des ressources, capacité, ...
  - sécurité : Confidentialité, Intégrité, Non-répudiation, Imputabilité, Authenticité, ...
  - fiabilité : Maturité, Disponibilité, Tolérance aux défaillances, Capacité de récupération, ...
  - facilité d'utilisation : Identification de la pertinence, Facilité d'apprentissage, Facilité d'opération, Protection contre les erreurs de l'utilisateur, Esthétique de l'interface utilisateur, Accessibilité, ...
  - maintenabilité : Modularité, Capacité de réutilisation, Capacité d'analyse, Facilité de modification, Testabilité, ...
  - portabilité : Facilité d'adaptation, Facilité d'installation, Facilité de remplacement, ...

## Niveaux d'abstraction des exigences

- Exigences de haut niveau : ce que le métier veut réaliser mais pas comment le mettre en œuvre.
- Raffinement des exigences métier décrivant la solution : expression des exigences métier en termes plus techniques.
- Fonctions et caractéristiques de la solution : spécification complète d'un composant (considéré comme une partie de la conception de la solution).



## Ingénierie des Exigences

- L'ingénierie des Exigences est une sous-discipline de l'ingénierie des systèmes
- Deux concepts clés

### GESTION DES EXIGENCES

- Gère la référence d'un ensemble d'exigences de la solution retenue.
- Assure l'alignement entre ces exigences avec les plans projet et les réalisations.
- Fournit des interfaces aux autres processus de développement et managériaux.
- Un cadre de travail pour
  - L'Ingénierie des Exigences,
  - Les processus de support du processus de développement des exigences.

### DÉVELOPPEMENT DES EXIGENCES

- Ensemble d'activités, de tâches, de techniques et d'outils pour identifier, analyser, documenter et valider les différents niveaux d'abstraction des exigences.
- Comprend :
  - Le processus de transformation des besoins en exigences,
  - Le développement (raffinement des exigences de haut niveau) d'une solution pour ces exigences.

## Comprendre la notion d'exigence

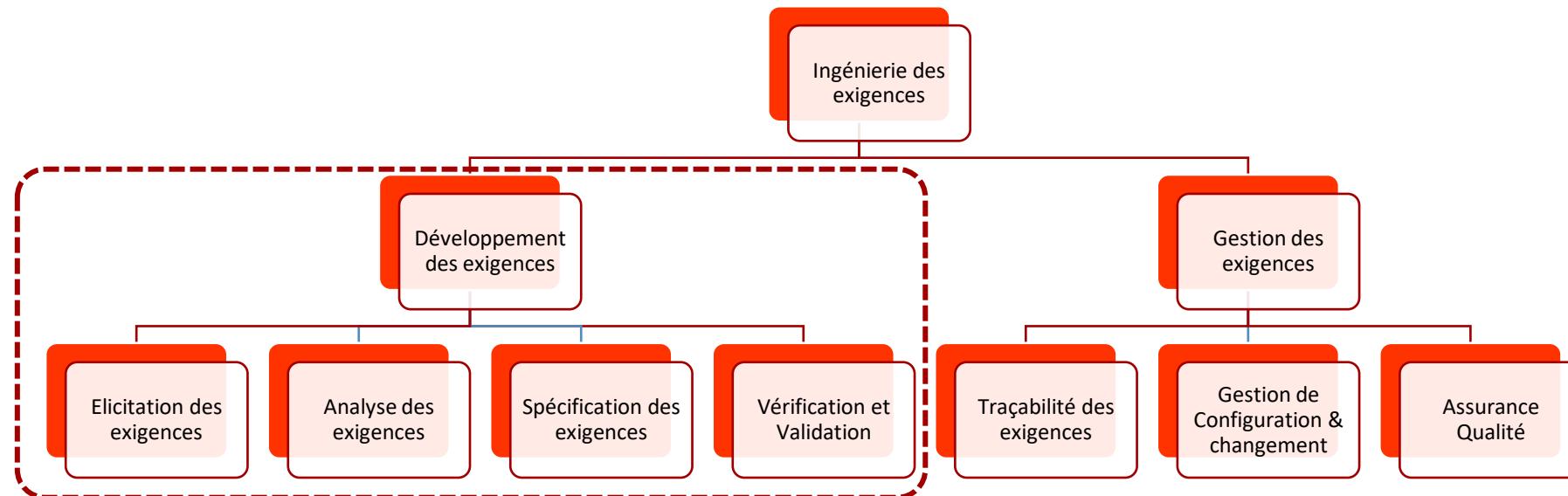
- Les rôles les plus importants affectant ou affectés par l'ingénierie des exigences sont le client et le fournisseur

| CLIENT  | FOURNISSEUR (ou VENDEUR)   |
|---|--|
| <ul style="list-style-type: none"><li>Est une personne, un groupe ou une organisation exigeant une solution.</li><li>Formule ses besoins et fournit les besoins et les attentes métier initiales qui sont généralement fournis en même temps que la demande d'offre / de service.</li></ul> | <ul style="list-style-type: none"><li>Est une personne, un groupe ou une organisation fournissant la solution.</li><li>Est responsable de la compréhension de ces besoins et de l'extraction des exigences à partir de ces données.</li><li>Son principal objectif est de fournir des solutions pour répondre aux besoins du client.</li></ul> |

# Ingénierie des d'exigence

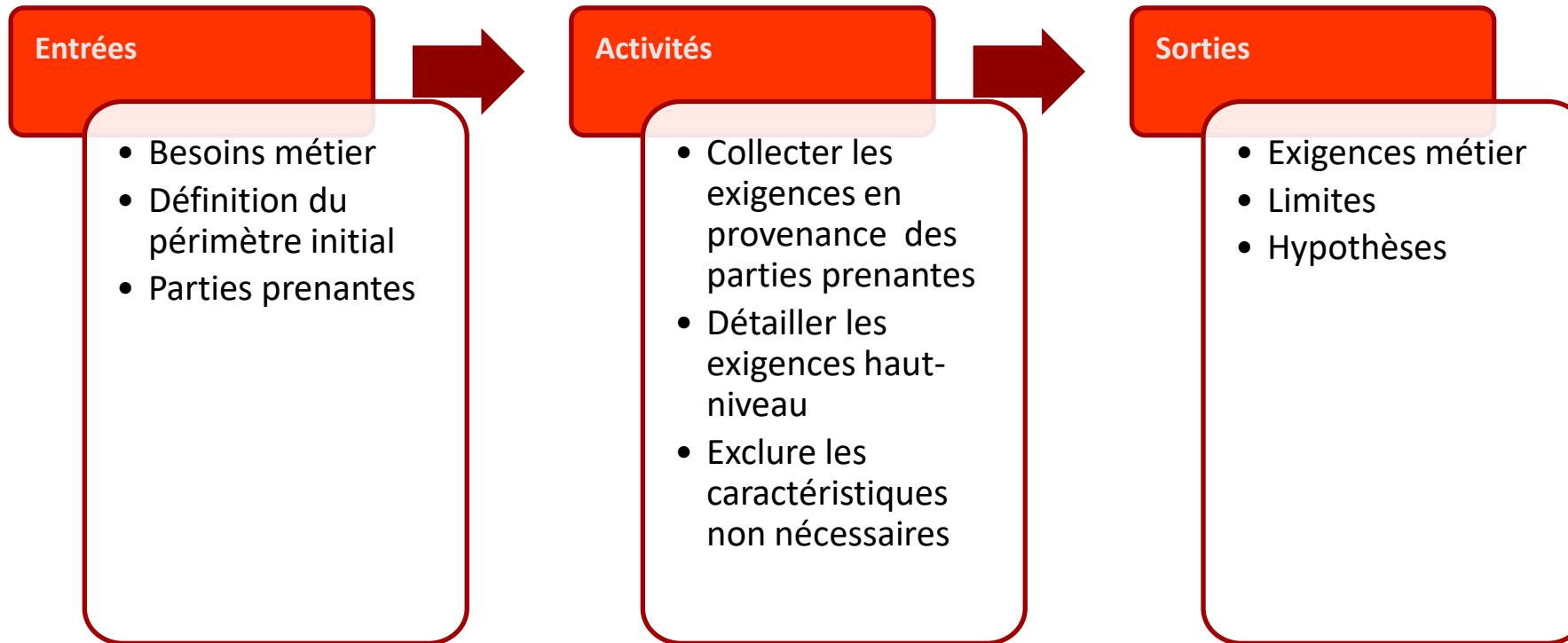
## Processus Générique d'Ingénierie des Exigences

- L'Ingénierie des Exigences est une discipline qui inclut les processus nécessaires
- pour l'Identification, la structuration et la gestion des Exigences.



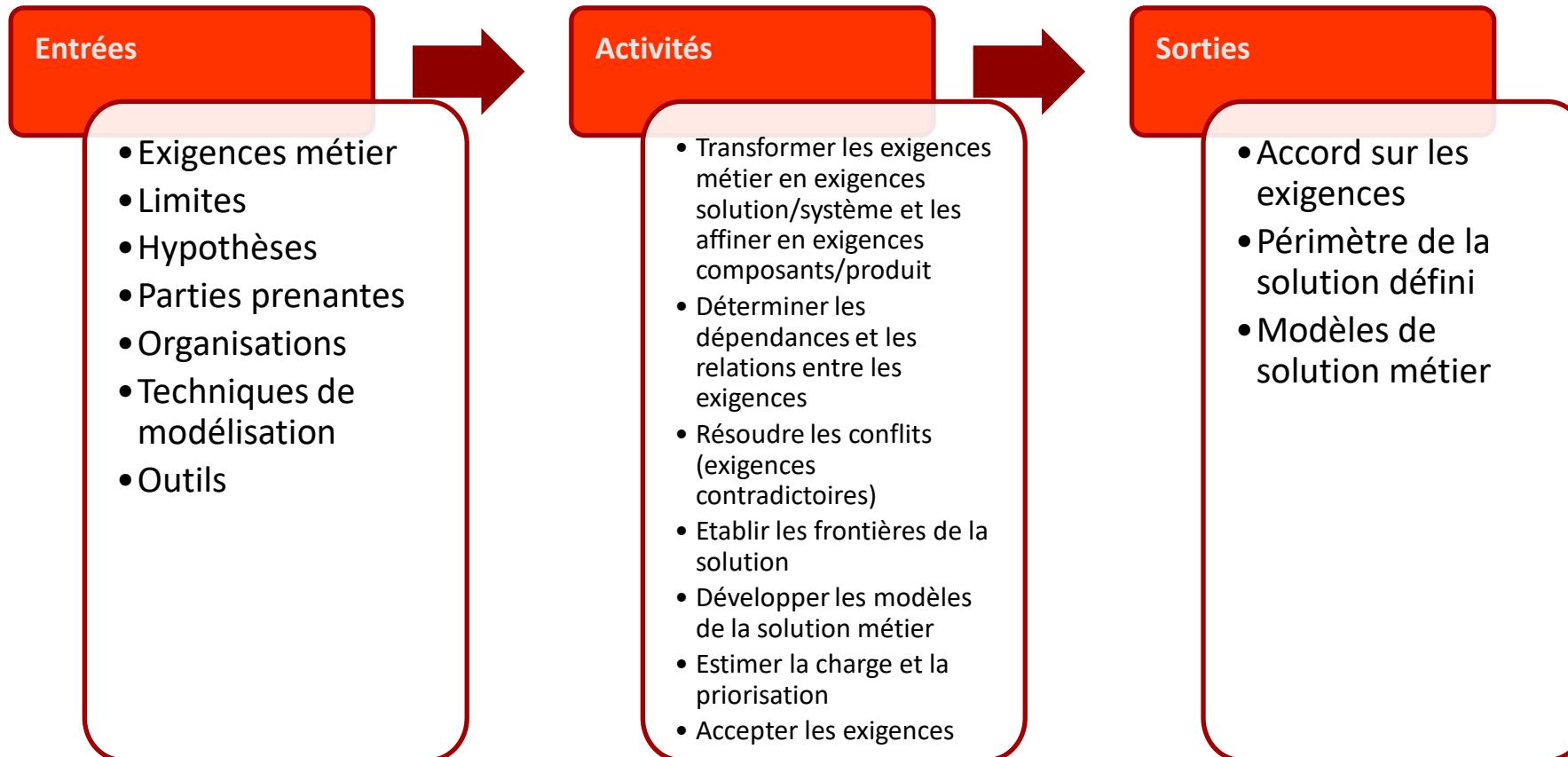
# Ingénierie des d'exigence

## Identification (ou Élicitation) des exigences

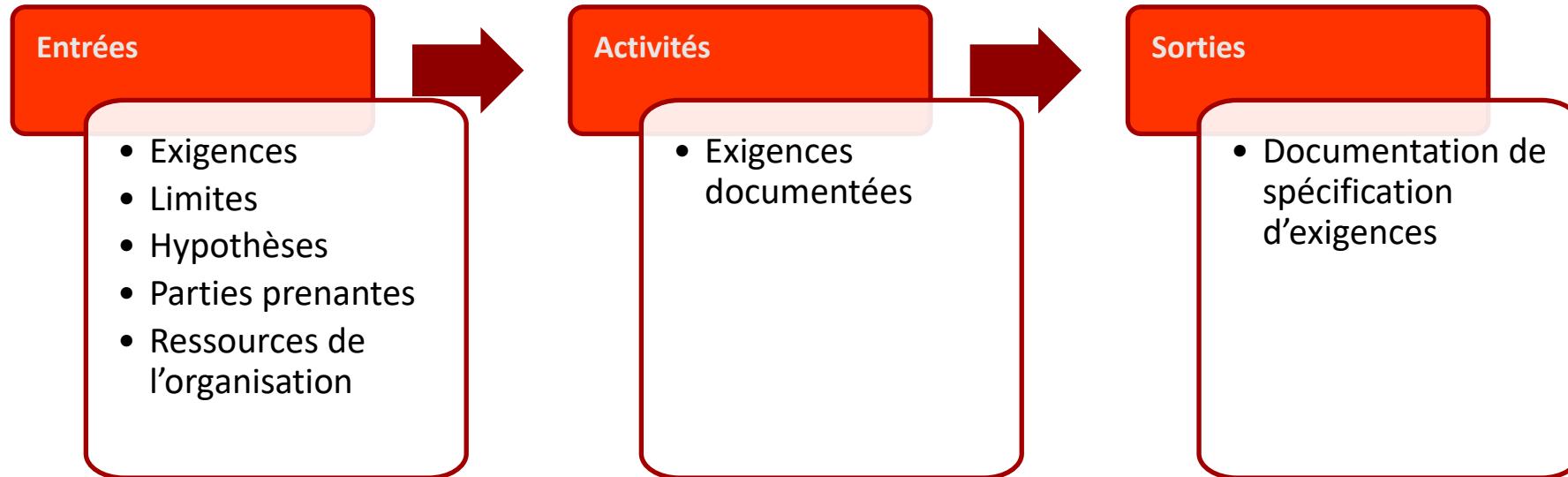


# Ingénierie des d'exigence

## Analyse des exigences

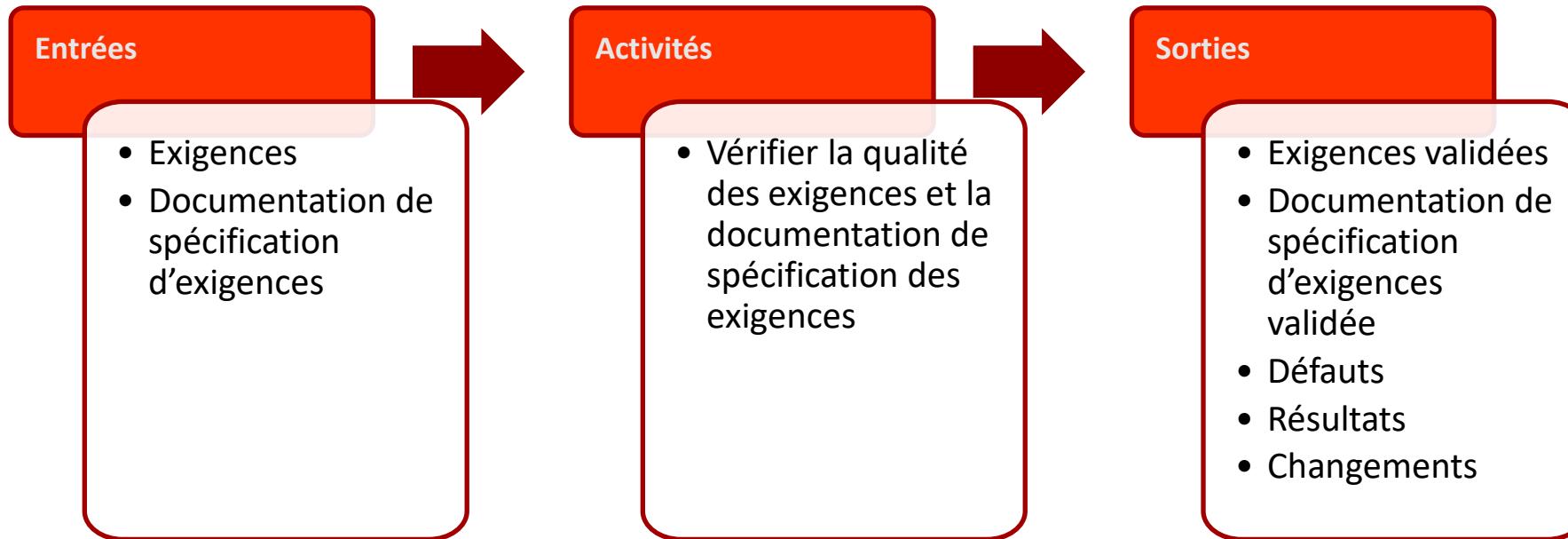


## Spécification des exigences

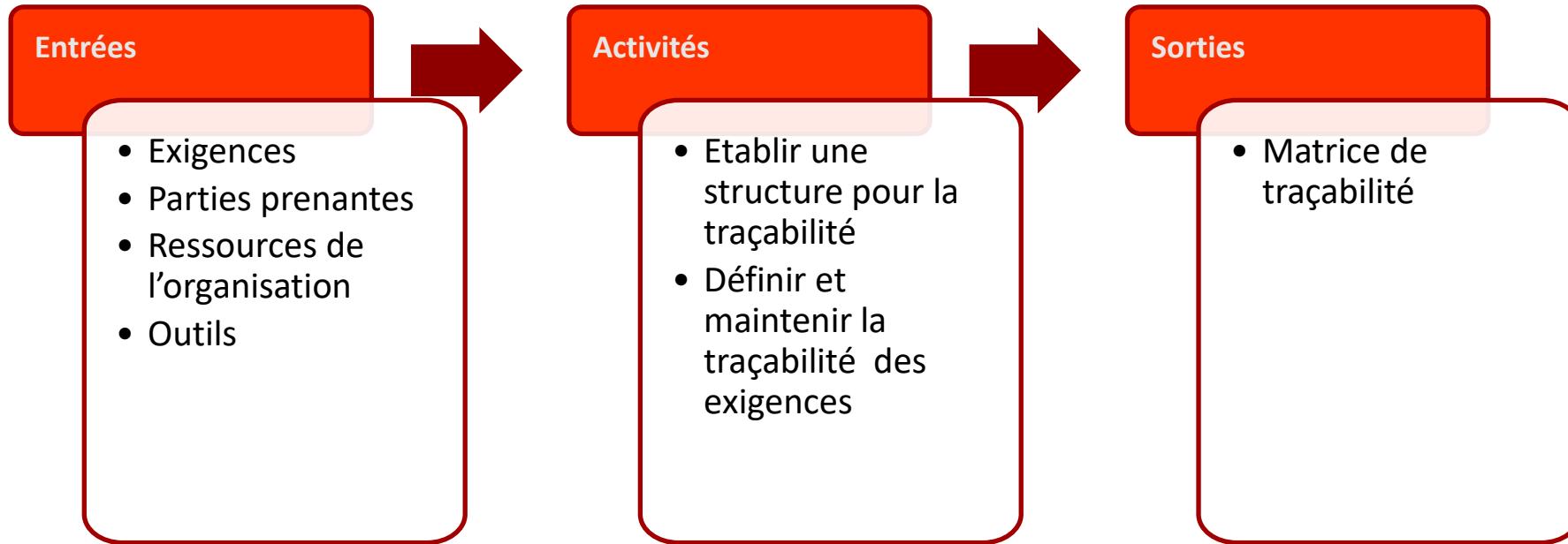


# Ingénierie des d'exigence

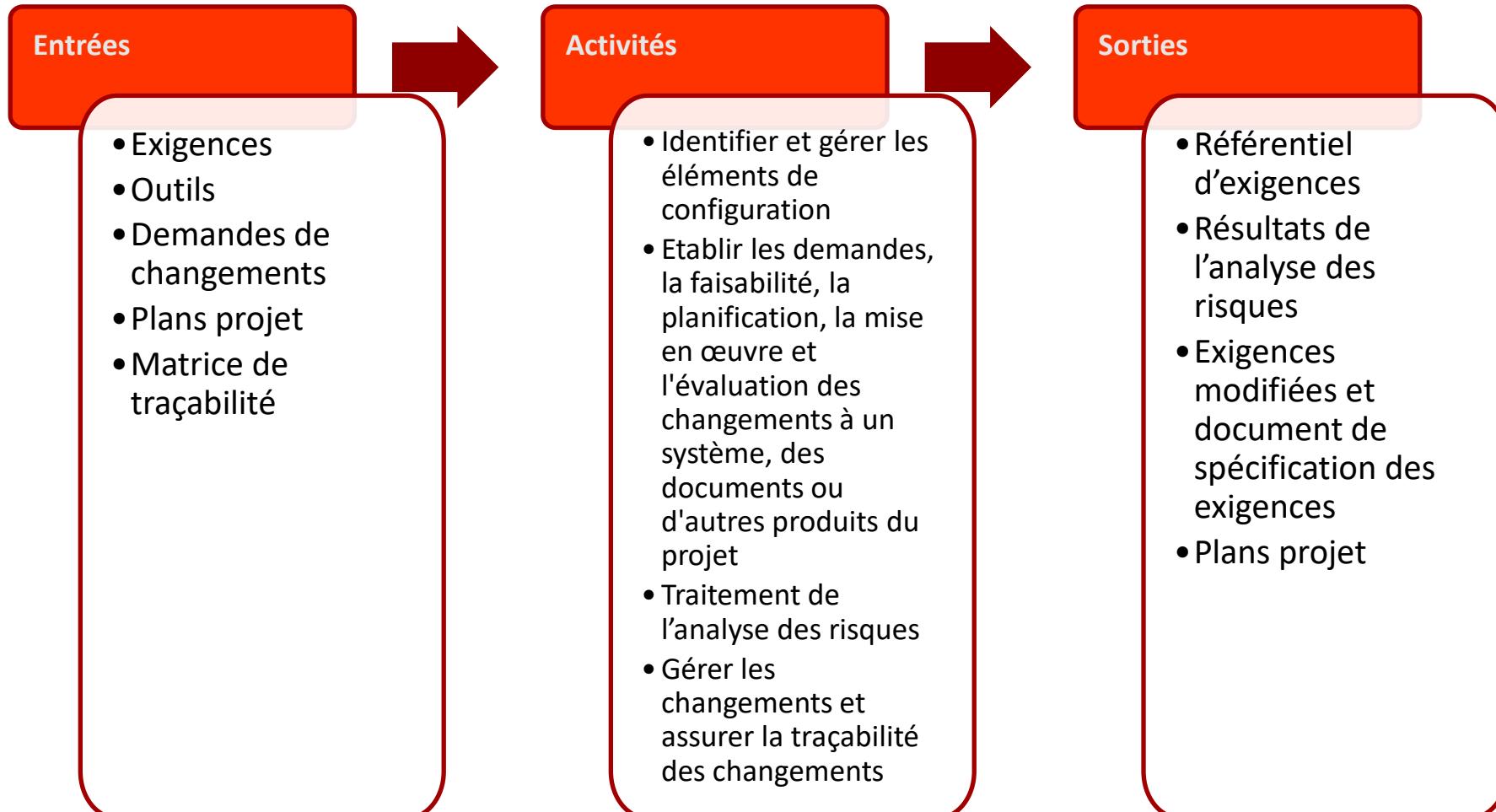
## Vérification et validation des exigences



## Traçabilité des exigences

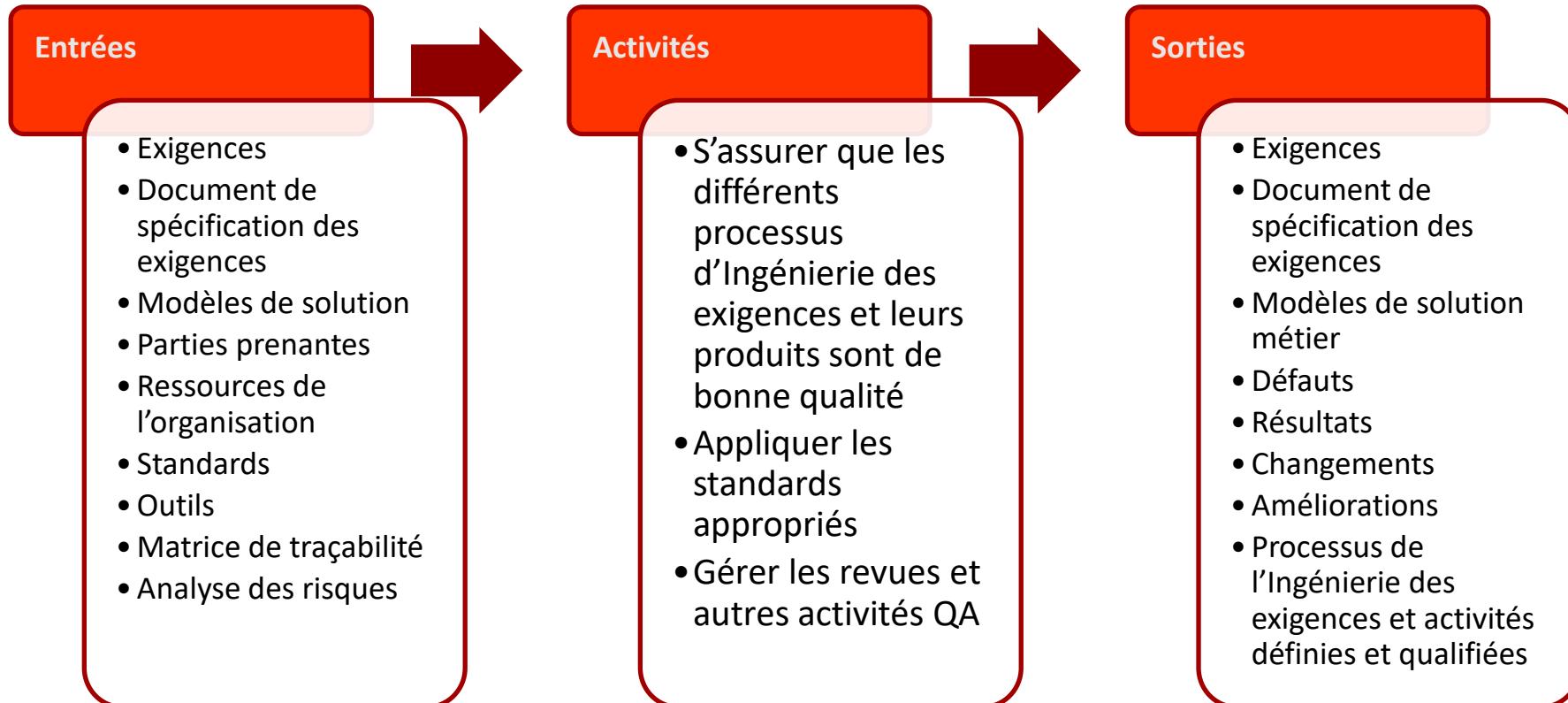


## Gestion de la configuration et du changement



# Ingénierie des exigences

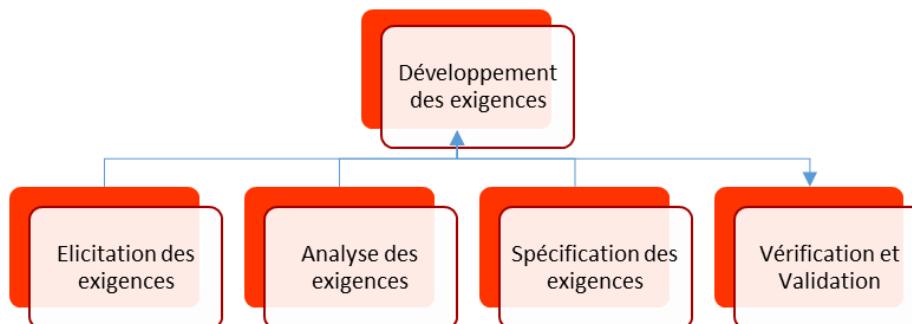
## Assurance Qualité



# Phases de l'ingénierie des Exigences

## L'élucidation des exigences

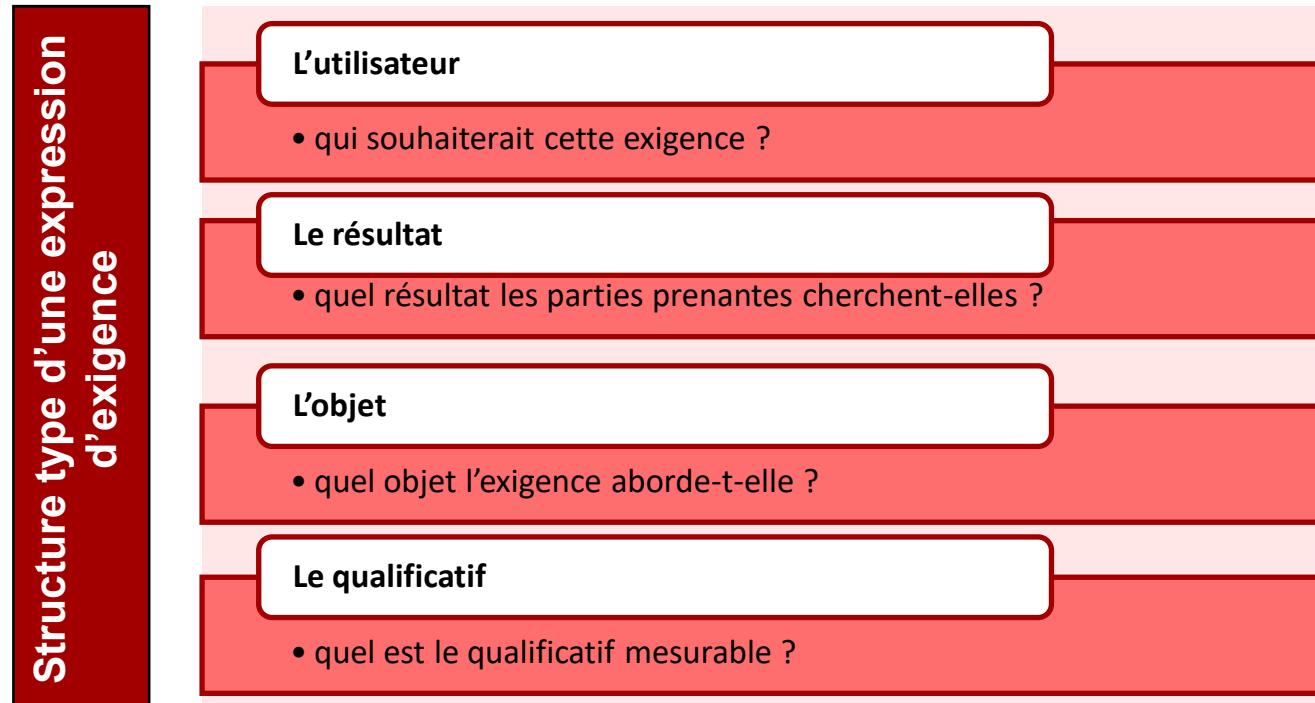
- L'élucidation des exigences couvre les pratiques de découverte, de compréhension du contexte, d'articulation et de documentation des exigences d'un système.
- Les objectifs de l'activité d'élicitation des exigences sont :
  - Collecter les exigences de toutes les parties prenantes possibles (utilisateurs, sponsors, équipes projet, marché, autres sources externes).
  - Identifier toutes les fonctions, caractéristiques, limitations et attentes
  - Orienter les exigences vers la vision du projet
  - Détailer les exigences de haut niveau et décrire les fonctions et services clairement
  - Exclure les fonctions et caractéristiques dont le client ne veut pas.



# Phases de l'ingénierie des Exigences

## L'élucidation des exigences

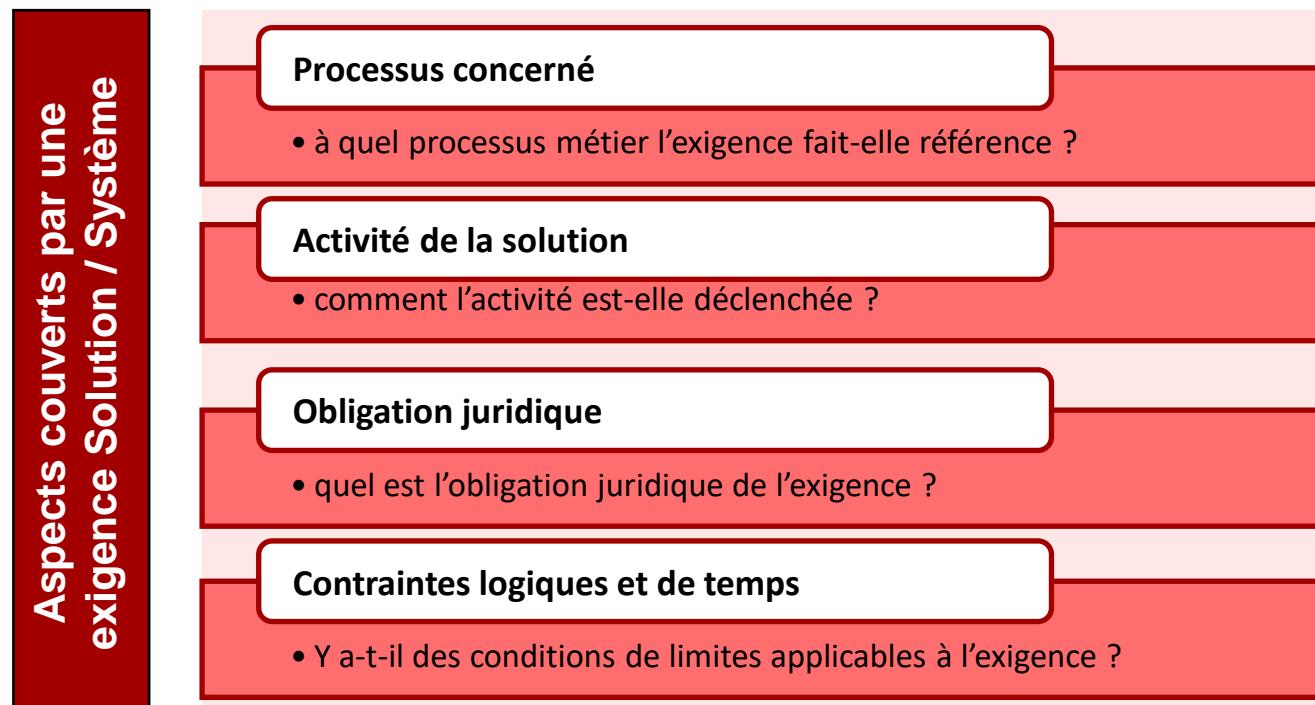
- Description des exigences métier
  - Les exigences peuvent être décrites de façon plus ou moins détaillée.
  - Les exigences métier peuvent être écrites sous la forme de cas d'utilisation de haut niveau, ou des User Stories (approches agiles).



# Phases de l'ingénierie des Exigences

## L'élucidation des exigences

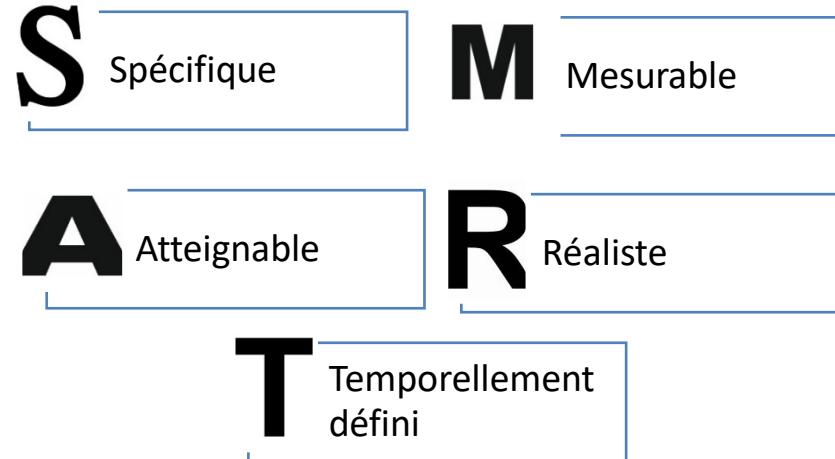
- Description des exigences solution / système
  - Les exigences solution/système (exigences plus précises) peuvent être décrites sous forme de cas d'utilisation du système, souvent avec des scénarios ou sous forme de User Stories (approches agiles).



# Phases de l'ingénierie des Exigences

## L'élucidation des exigences

- Objectifs Métier
  - Sont les résultats attendus du projet d'un point de vue métier.
  - Décrivent « ce qui » doit être accompli, pas « comment ».
  - Les objectifs doivent être quantitatifs et précis.



Lors de l'Élicitation des Exigences, il faut s'assurer que les exigences satisferont à la fois à la vision projet et aux objectifs S.M.A.R.T. définis.

# Phases de l'ingénierie des Exigences

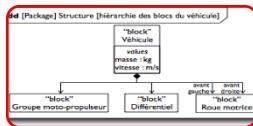
## Documenter les exigences

- Les différentes formes documentaires :



- Spécification NON-FORMELLE

- Les spécifications non-formelles peuvent être utilisées quand les lecteurs n'ont aucune expérience avec les notations formelles ou les langages techniques de spécification et auraient des difficultés dans la compréhension du contenu.



- Spécification SEMI-FORMELLE

- Les spécifications semi-formelles peuvent exprimer des exigences sous la forme de modèles et elles utilisent un langage naturel formalisé.

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

- Spécification FORMELLE

- Une spécification formelle décrit ce que le produit devrait faire et ne décrit pas comment il devrait être fait.

# Les Méthodes Agiles

## Etude de Cas

- Précision des exigences



# Les Méthodes Agiles



## 01 Exercice : Précision des exigences

- Logistiques:
  - Minimum deux groupes (composés d'environ 6 personnes chacun)
  - Objectif : Faire un dessin (en silence) suivant une exigence (à distribuer)
  - Temps : Une minute pour terminer le dessin.
  - Je leur ai fourni un certain nombre de crayons en rouge, vert et bleu et une grande feuille de papier chacun.
  - Remettre les exigences à chaque groupe
  - Commencer un compte à rebours visible de 60 secondes

# Les Méthodes Agiles



## 01 Exercice : Précision des exigences

- Exigence 01:
- Dessinez une belle prairie d'été avec :
  - des fleurs bleues et rouges dans l'herbe verte, des vaches et des oiseaux sous un soleil brillant.
- Exigence 02:
- Dessinez une belle prairie d'été avec :
  - 10 fleurs bleues à 5 pétales chacune
  - 5 fleurs bleues à 6 pétales chacune
  - 13 fleurs rouges à 6 pétales chacune
  - 2 vaches avec 3 points noirs
  - 1 vache avec 5 points noirs
  - 2 vaches avec 4 points noirs
  - 2 oiseaux à résider dans le coin supérieur gauche
  - 3 oiseaux au milieu
  - un soleil à droite avec 5 rayons de soleil

# Les Méthodes Agiles

## 01 Correction: Précision des exigences

- Le **dessin 01** a été fait par le groupe qui a obtenu les exigences ouvertes
- Le **dessin 02** a été fait par le groupe avec beaucoup de détails de spécification.
  - Et ce dessin n'est même pas conforme à l'exigence de base; une prairie d'été.
  - Et les vaches sont, bien que riches en détails, sont dans les « nuages »
  - Le groupe derrière le bon dessin était tellement concentré sur la mise en œuvre de chaque détail des exigences qu'ils ont oublié le but principal de l '«assignemnt», dessiner un pré.
- Points d'attention :
  - Difficulté de mettre en œuvre des exigences sur-spécifiées.
  - Nécessité de faire preuve de créativité
  - Risque de faire preuve de trop de créativité, de manière à trouver des solutions non alignées sur les besoins

Dessin 01



Dessin 02



## Questions Ouvertes

**Des questions ?**



# Sommaire



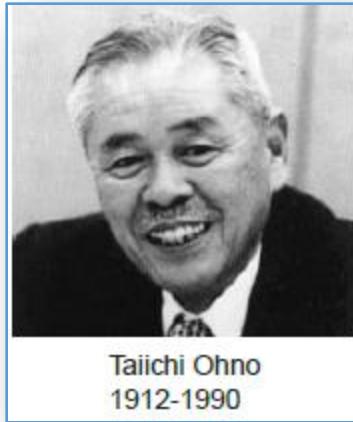
| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | <b>Le LEAN</b>                    | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Les Méthodes Agiles

## Le Lean Management

- **Le modèle de production Toyota**

- Amélioration continue
- Respect des personnes
- Remettre tout en cause
- Embrasser le changement



- **Une philosophie**

- Respect des employés
- Une utilisation de toutes les compétences des employés
- Donner des responsabilités et avoir confiance dans les employés

# Les Méthodes Agiles



## Le Lean Management

- **LEAN Management: Idée maîtresse**
- Maximiser la « valeur-client » en minimisant les gaspillages.
  - Le but ultime du LEAN est d'apporter le maximum de valeur au client avec le minimum de ressources (faire plus avec moins).
- La «pensée» LEAN implique un changement des modes de management et l'adaptation de l'organisation.
  - Passer d'une vision d'organisation en «silo» à une «organisation orientée processus», c'est-à-dire transversale, destinée à se concentrer sur la valeur à apporter au client.
- Le LEAN n'est pas exclusivement destiné au secteur industriel.
  - Il touche aux processus et pratiques de travail, donc, potentiellement à tous les secteurs d'activité.
- Le LEAN est une philosophie qui a pour ambition l'amélioration continue des performances de l'organisation dans le but de mieux satisfaire le client.
  - Il ne se focalise pas sur la réduction des coûts

Viser la perfection en s'appuyant sur la volonté de faire mieux aujourd'hui qu'hier et demain, mieux qu'aujourd'hui (amélioration continue)

# Les Méthodes Agiles



## Le Lean Management

- **Les Principes du Lean**

- Optimiser le système dans son ensemble
  - Approche à long terme.
- Travailler en mode flux
  - Cycle court
  - Livrer rapidement
  - Visibilité régulière
- Décider le plus tard possible
  - Reporter la décision "Last Responsible Moment"
- Lisser le travail
- Maîtriser les standards
- Management visuel simple
- Technologie éprouvée
- Responsabiliser l'équipe
  - Construire de la qualité intrinsèque
- Résoudre les problèmes
- Eliminer les gaspillages
- Montrer le plus tôt possible

# Les Méthodes Agiles



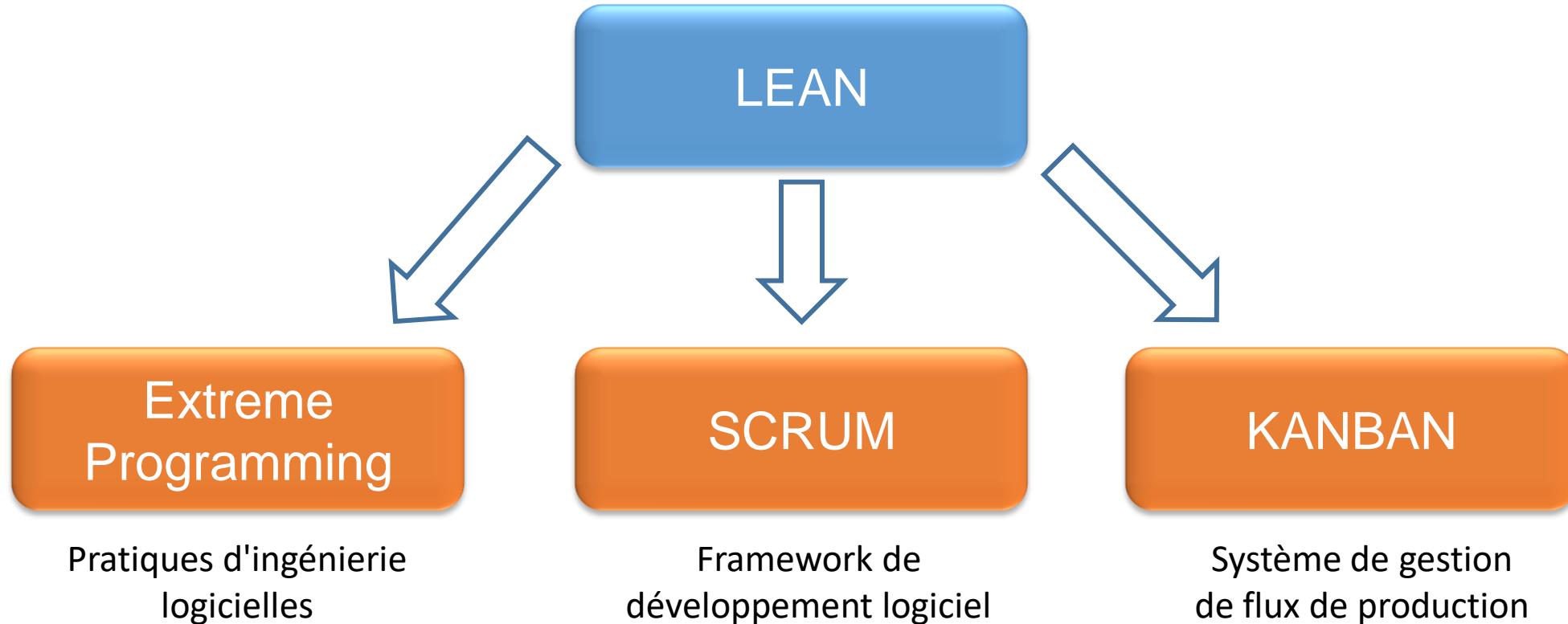
## Le Lean Management

- **Le Lean Management s'appuie sur quatre principes fondamentaux :**
  - La compréhension des besoins clients
  - La réduction du temps de production
  - L'analyse, la compréhension et la résolution des problématiques
  - La fédération et la sensibilisation des collaborateurs
- **Les outils du Lean**
  - Kaizen : amélioration continue
  - Gemba : Là où se trouve la réalité
  - Muda : forme de gaspillage
  - Kanban : fiche, étiquette
  - Yokoten : Diffusion horizontale des connaissances
  - Jidoka : automatisation avec une touche humaine
  - Obeya : grande salle (war room des projets agiles...)

# Les Méthodes Agiles

## Le Lean Management

- LEAN, XP, SCRUM, KANBAN



## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | <b>La méthode SCRUM</b>           | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

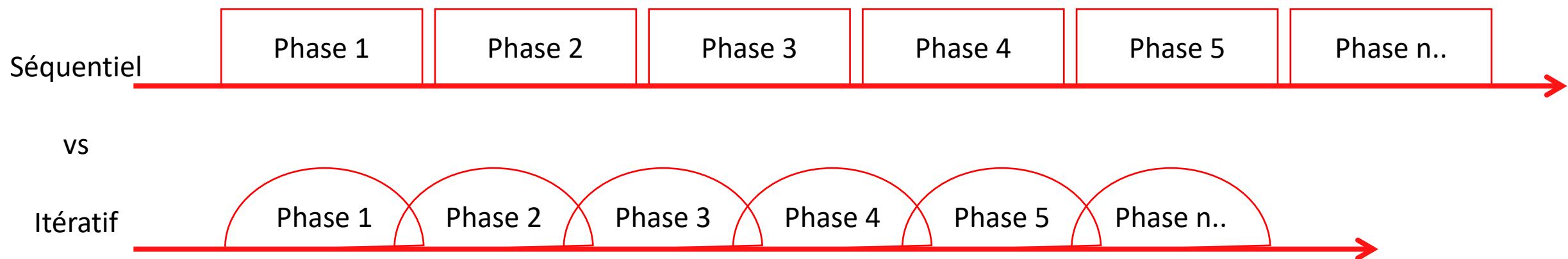
# Les Méthodes Agiles

## Les grands principes de la méthode SCRUM

- Les valeurs du SCRUM
  - Transparence
  - Inspection
  - Adaptation
- Le développement itératif SCRUM



Atelier Coin Toss



# Les Méthodes Agiles



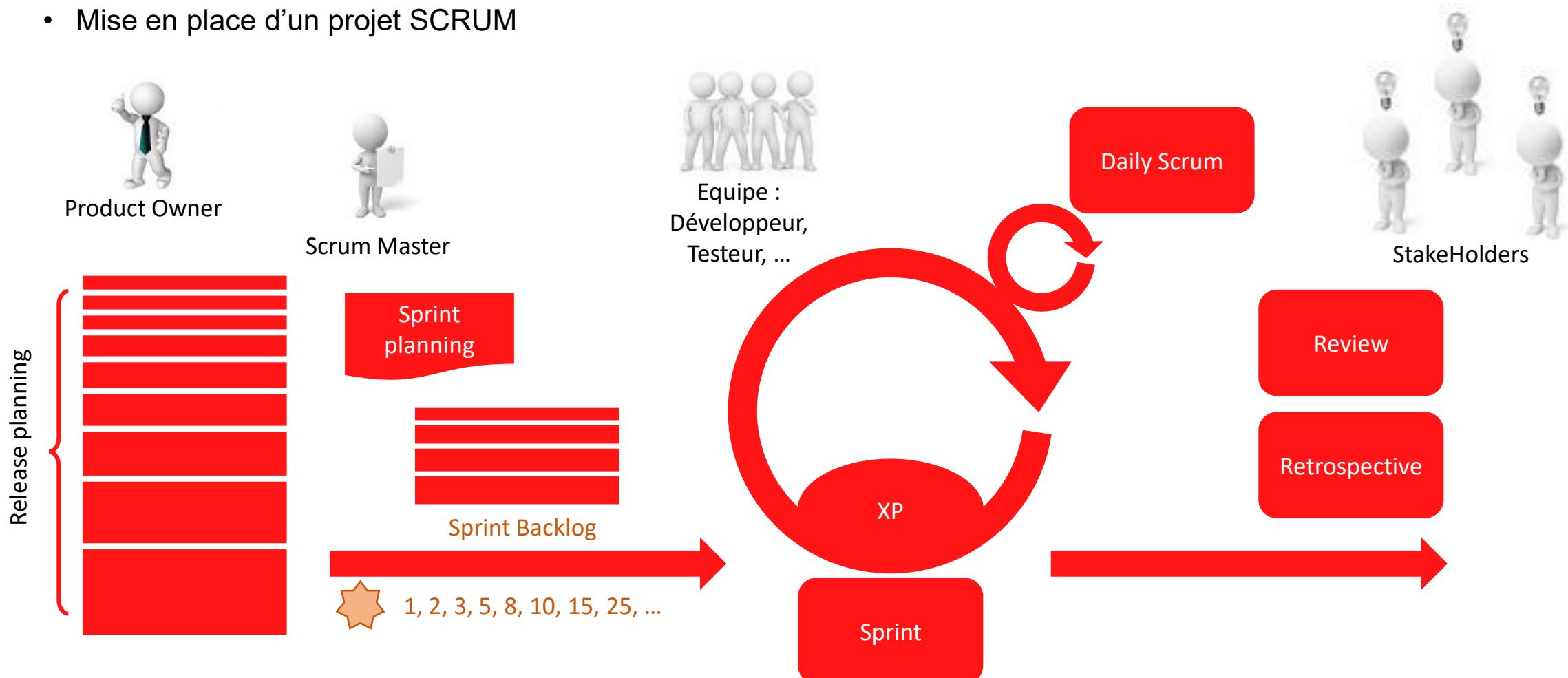
## Les grands principes de la méthode SCRUM

- Les Time-Boxes dans SCRUM
  - Sprint
    - Habituellement entre 2 et 4 semaines.
  - Sprint Planning Meeting
    - Pour 2 à 3 semaines de sprint : généralement 1 journée, dont la moitié pour définir le contenu du sprint, et l'autre moitié pour définir comment les fonctionnalités sélectionnées vont être développées.
  - Sprint review
    - Généralement 4h< (la moitié de la durée du Sprint Planning Meeting). Plutôt 1 à 2h.
  - Sprint retrospective
    - Généralement 4h< (la moitié de la durée du Sprint Planning Meeting). Plutôt 1 à 2h.
  - Daily Scrum
    - 15Mn

# Les Méthodes Agiles

## Les grands principes de la méthode SCRUM

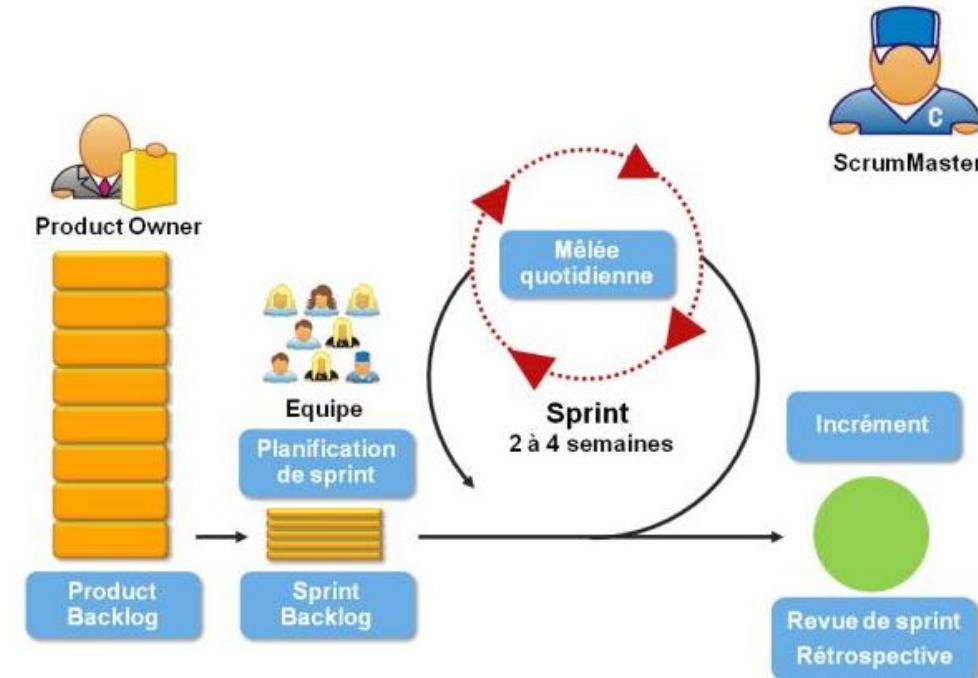
- Mise en place d'un projet SCRUM



# Les Méthodes Agiles

## Les grands principes de la méthode SCRUM

- Scrum répond à la complexité en s'appuyant sur trois piliers permettant une approche empirique : transparence, inspection et adaptation. Ces trois piliers sont incarnés en cérémoniaux rythmant les sprints, ces itérations d'une à quatre semaines au cours desquelles nous retrouvons :
  - Le daily-meeting : chaque matin, les membres de l'équipe se retrouvent pour quinze minutes et abordent à tour de rôle les tâches traitées la veille, les problèmes rencontrés et les tâches qu'ils traiteront dans la journée. Chaque membre de l'équipe se doit d'être transparent sur les problèmes qu'il rencontre pour mieux les résoudre collectivement.
  - Le sprint planning : en début de sprint, la Scrum Team définit un objectif de Sprint et dresse une liste de tâches à traiter durant l'itération. Il en résulte un Sprint backlog composé de tâches bien spécifiées et priorisées.
  - La revue : en fin de sprint, la Scrum Team (Product Owner, développeurs et Scrum Master) se réunit pour examiner ce qui a été développé (l'incrément) et récolter un maximum de retours qui seront qualifiés en tâches pour les prochaines itérations.
  - La rétrospective : la revue est au « quoi » ce que la rétrospective est au « comment » : l'équipe de développement recense et capitalise sur les problèmes rencontrés durant le sprint afin d'améliorer son organisation et être plus efficace. La rétrospective est un élément-clé de l'amélioration continue et de la quête du « zéro-déchet ».



# Les Méthodes Agiles

## Les grands principes de la méthode SCRUM

- Le Product Owner

- Décide des fonctionnalités du produit
- Décide des dates de disponibilité des versions et de leurs contenus
- Responsable de la valeur du produit
- Priorise les fonctionnalités du produit en fonction de leurs valeurs ajoutées, du retour sur investissement, de leurs valeurs
- Accepte ou rejette les résultats produits par l'équipe
- Il doit être légitime et avoir assez de pouvoir pour pouvoir trancher quand cela est nécessaire (ce n'est pas un comité, c'est une seule personne)
- Il doit être disponible et impliqué sur le projet (présence lors du sprint planning meeting, de la démo, de la rétrospective, des daily scrum)



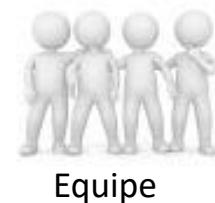
Product Owner

# Les Méthodes Agiles

## Les grands principes de la méthode SCRUM

- L'Equipe

- Généralement entre 5 et 9 membres (3 et 4 c'est bien, mais l'équipe doit être assez autonome (crossfunctionnality))
- Hétérogène (elle mêle architecte, développeur, testeur, analystes, graphistes, etc.)
- Donne son accord aux objectifs de chaque itération
- Spécifie les résultats à réaliser, elle s'engage et est responsable
- Fonctionne de manière empirique et a le droit de faire tout ce qu'elle souhaite pour atteindre l'objectif de l'itération
- Autonome, elle fonctionne en autogestion
- Stable
- Présente les résultats du travail de chaque itération au Product Owner Il s'agit de résultats d'équipe



# Les Méthodes Agiles

## Les grands principes de la méthode SCRUM

- Le Scrum Master

- S'assure que Scrum est bien appliqué
- S'assure que l'équipe est opérationnelle et productive (et en bonne santé)
- Facilite la coopération entre les différents acteurs (de l'équipe scrum ou des parties prenantes)
- S'assure que rien ne gène l'avancée du travail, le cas échéant essaye de trouver la solution pour lever les blocages, obstacles visibles ou invisibles.
- Réduit au maximum les perturbations extérieures avec l'équipe Scrum
- Communique avec le management
- S'assure que la qualité du produit n'est jamais remise en cause



Scrum Master

# Les Méthodes Agiles

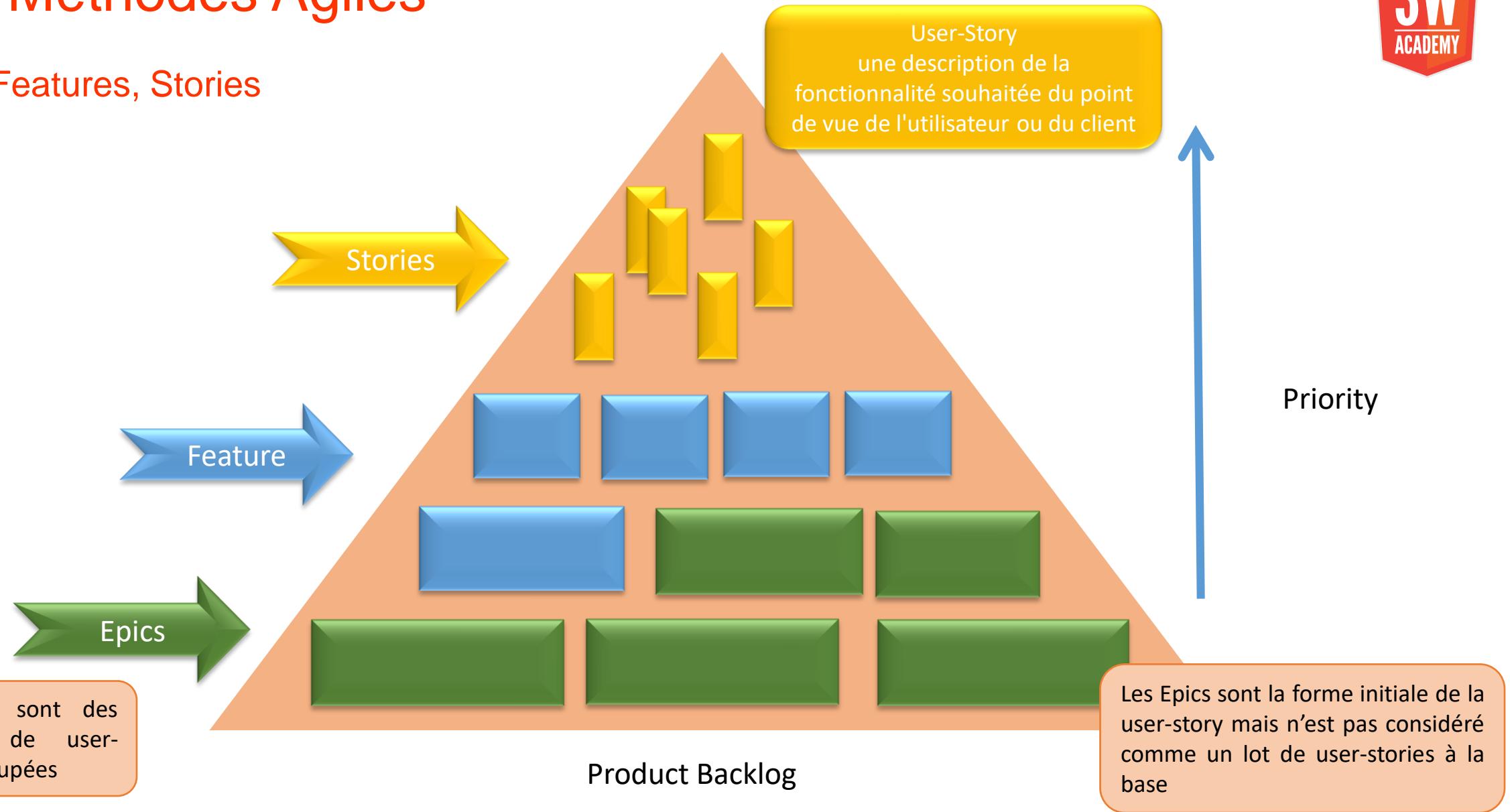
## Les grands principes de la méthode SCRUM

- Délivrer de la Valeur
  - La force de Scrum et de l'agile est de délivrer d'abord et avant toute chose ce qui a le plus de valeur (retour sur investissement). Cette valeur peut être dans certains cas estimée (en monnaie).
  - Le Product Owner définit dans un premier temps une vision, une épopée concernant le produit. Puis il décline features/thèmes, cas d'utilisation ou histoires d'utilisateurs (User Story).



# Les Méthodes Agiles

## Epics, Features, Stories

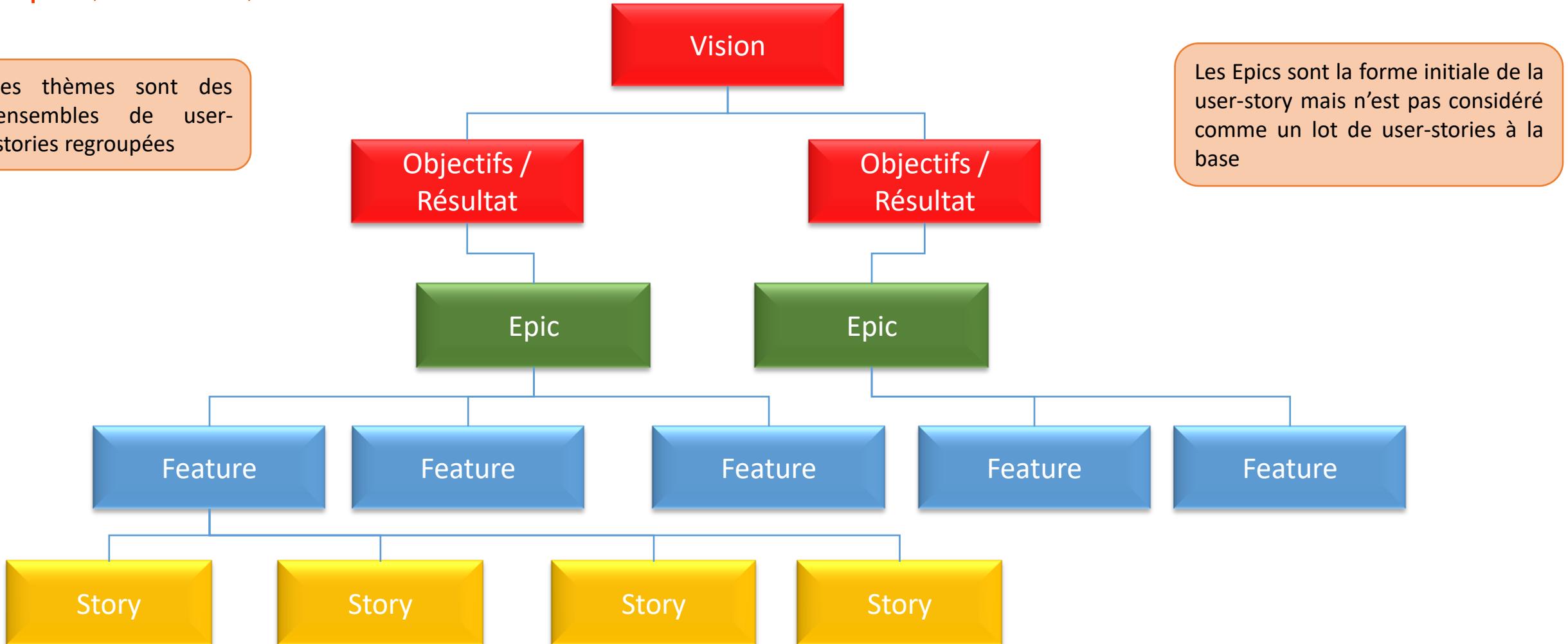


# Les Méthodes Agiles

## Epics, Features, Stories

les thèmes sont des ensembles de user-stories regroupées

Les Epics sont la forme initiale de la user-story mais n'est pas considéré comme un lot de user-stories à la base



# Les Méthodes Agiles



## Epics, Features, Stories

- **Epics**

- Les épopées « Epiques » sont des fonctionnalités ou des activités de haut niveau qui couvrent les Sprints, voire les Releases. Ex :
  - Ajouter un centre client pour le libre-service.
  - Améliorez le temps de réponse de la base de données de 50%.
- Logistique
  - Le Product Owner travaille avec les parties prenantes et l'équipe pour créer des épopées qui répondent aux objectifs souhaités.
  - Les épopées sont souvent définies avant la planification des versions. Souvent des mois d'efforts.
- Une épique sera donc découpée en de petites histoires
  - Si vous avez 4 histoires (Stories) représentant ce qu'était votre épique, l'épique disparaît, remplacée par 4 histoires.

# Les Méthodes Agiles



## Epics, Features, Stories

- **Epics**

- Un Epic peut ressembler beaucoup à une User Story.
- C'est l'expression d'un besoin métier, il est également neutre en termes de solution, mais il est important et devra généralement être décomposé pour pouvoir y répondre, que ce soit pour des raisons de temps ou de complexité.
  - Exemple un Epic lié à la User Story «En tant que client, je veux pouvoir acheter des produits en ligne».
  - Cela devra sûrement être décomposé en un certain nombre de Stories d'utilisateurs.
- Les Epics ont deux objectifs Lors du processus d'élicitation des exigences, en particulier lorsque vous travaillez en méthode Agile
  - nous commençons par des exigences de haut niveau (Epics)
  - Les Epics sont ensuite affinés et décompés en exigences plus détaillées (User Stories).
  - Ils nous aident à structurer et regrouper nos besoins et ainsi à gérer notre backlog.

# Les Méthodes Agiles



## Epics, Features, Stories

- **Feature, Themes, Fonctionnalités**

- Les caractéristiques sont des expressions tangibles de fonctionnalité, mais encore trop volumineux à construire. Ex :
  - En tant que client bancaire, trouver une agence pour que je puisse déposer chèques.
  - En tant qu'acheteur, configuez un portefeuille mobile pour que je puisse payer achats via Near Field Communications.
- Logistique
  - Créé par le Product Owner avec la contribution de l'équipe
  - Souvent défini avant la planification des versions
  - Décomposé au fil du temps en histoires plus petites
  - Généralement des semaines d'effort
- Un thème apporte une façon commode d'indiquer que des histoires ont quelque chose en commun, comme à faire partie de la même fonctionnalité.

# Les Méthodes Agiles



## Epics, Features, Stories

- **User Stories**

- Les User Stories sont prêtes pour que l'équipe puisse les développer.
  - En tant que client bancaire, rechercher une succursale à proximité d'une adresse afin que je puisse minimiser les déplacements.
  - En tant qu'acheteur, ajouter un compte à mon portefeuille mobile afin que je puisse le financer.
- Logistique
  - Affiné dans les sessions de préparation de l'arriéré par PO et représentants de l'équipe
  - Les histoires doivent être bien définies avant le sprint Planification
  - Généralement environ 1 à 3 jours d'effort

## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Les Méthodes Agiles



## User Stories

- Les histoires utilisateurs (user story) ne sont pas des cas d'utilisations (use case)
- « Une user story » (histoire utilisateur) est une exigence logicielle formulée en une ou plusieurs phrases dans le langage de tous les jours ou celui lié au métier de l'utilisateur.
- Les user stories sont utilisées dans le développement logiciel dit Agile comme spécifications (en même temps que les tests d'acceptance). Le formalisme est limité à une « étiquette » de type post-it.
- Les User Story incitent à ne pas entrer dans le détail tant que cela n'est pas nécessaire.



# Les Méthodes Agiles



## User Stories

- Ron Jeffries propose une règle des « **3C** » pour gérer les User Story
- **Card « Etiquettes »**
  - L'histoire est écrite sur une carte de taille assez réduite.
  - Ces fiches peuvent être annotées (estimation, etc.)
- **Conversation « Echange »**
  - Les détails de la user story seront exprimés lors de partages avec le Product Owner
- **Confirmation « Validation »**
  - Des tests d'acceptation sont définis avec la user story pour garantir leurs réalisations, valider l'alignement et la cohérence avec la user story
- le format adopté dans la plupart des équipes Scrum aujourd'hui.
  - AS .....
  - I want to ...
  - So that ...

# Les Méthodes Agiles



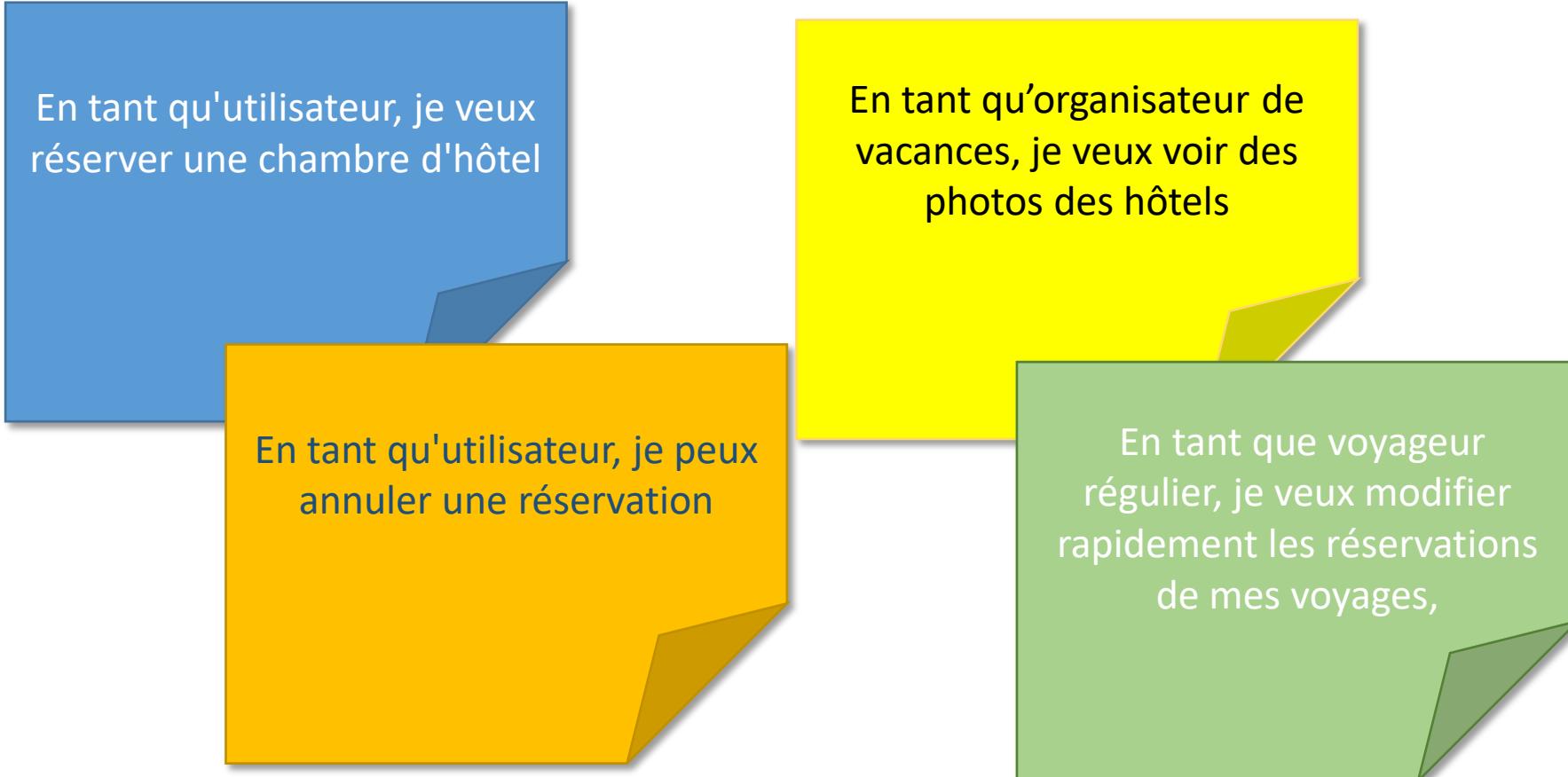
## User Stories

- Une User Story se base sur la syntaxe suivante
  - **En tant que <rôle>,**
  - **je peux <besoin>**
  - **de façon à <bénéfice>**
- Elle sera comprise correctement par toute la population qui gravite autour et dans le projet : parties prenantes et acteurs clés.
- Une bonne user story doit respecter : **L'ACRONYME "INVEST"**
  - **I** : Independant (indépendante)
  - **N** : Negotiable(négociable)
  - **V** : Valuable (avec de la valeur)
  - **E** : Estimable
  - **S** : Sized to fit (assez petite)
  - **T** : Testable

# Les Méthodes Agiles

## User Stories

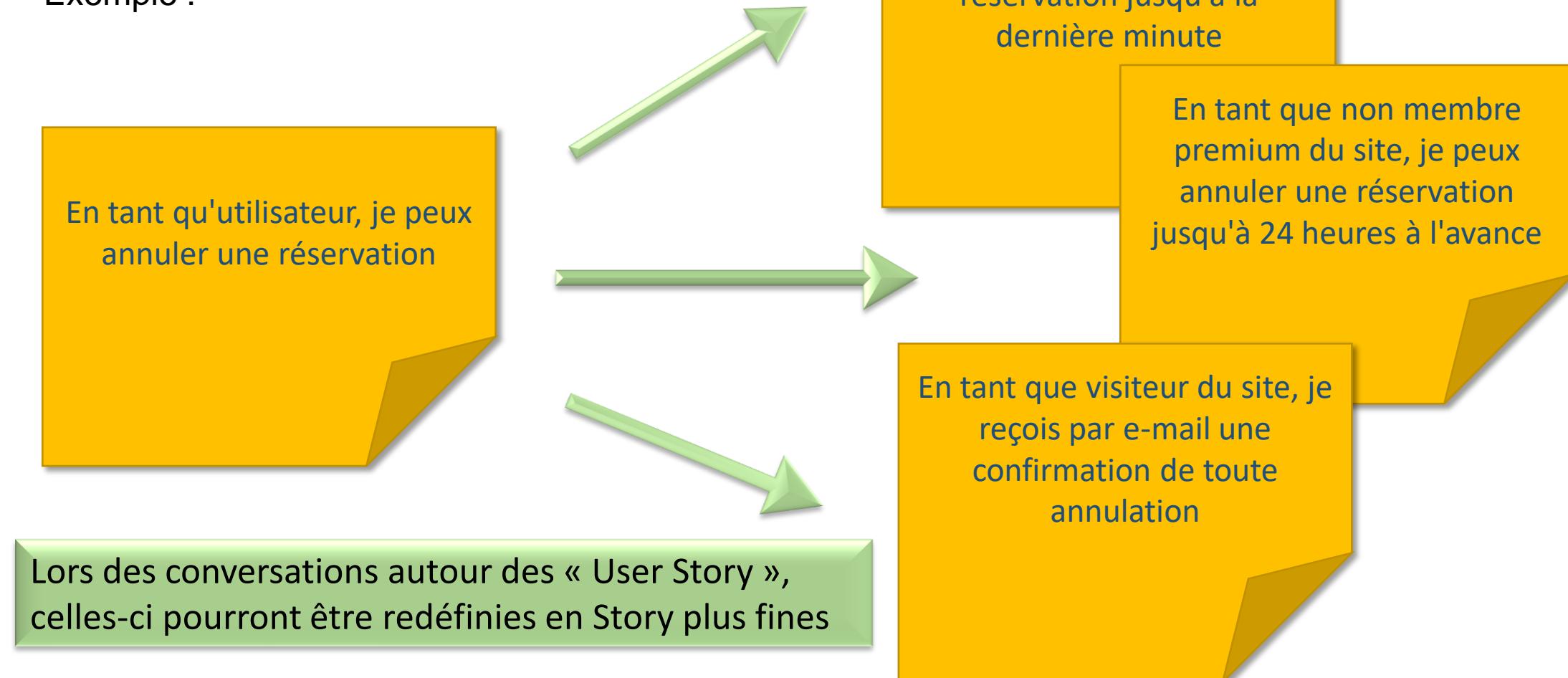
- Exemple :



# Les Méthodes Agiles

## User Stories

- Exemple :



# Les Méthodes Agiles



## User Stories

- Découper les user stories :
  - Workflow steps (par étape)
  - Business rule variations (par règle métiers)
  - Major effort (par taille d'effort)
  - Simple / Complex (approche du simple au complexe)
  - Variations in data (variations dans les données)
  - Data entry methods (façon d'intégrer les données)
  - Defer performance (retarde la question des performances)
  - Operations (CRUD) (par opérations)
  - Break out a spike (analyser une piste)

# Les Méthodes Agiles



## User Stories

- User stories et tests d'Acceptation :
  - Lors des conversations autour des « User Story » il faudra préciser les tests d'acceptation qui permettront de valider en partie que la Story a bien été réalisée

En tant qu'utilisateur, je peux annuler une réservation

- Vérifier qu'un membre premium peut annuler le même jour sans frais
- Vérifier qu'un membre non premium est facturé 10% pour une annulation le jour même
- vérifier qu'un e-mail de confirmation est envoyé
- vérifier que l'hôtel est informé de toute annulation

# Les Méthodes Agiles



## User Stories

- User stories et tests d'Acceptation :
- Ecrire des Acceptance Test Driven Développement : **ATDD**
  - Le Gherkin est un langage très simple pour formaliser et unifier l'écriture de vos tests d'acceptation.
  - Il pourra à terme être utilisé par des outils qui vont permettre d'automatiser ces tests, ou du moins fournir facilement une « enveloppe » de tests (specflow, cucumber, jbehave, etc.)

Etant donné la page d'accueil du site avec un bouton « enregistrez vous »

Quand je clique sur le bouton « enregistrez vous »

Alors j'arrive sur le formulaire d'enregistrement

Etant donné le formulaire d'enregistrement

Quand je m'enregistre

Alors je reçois un email de demande de confirmation

Et cet email contient un lien de confirmation

Etant donné un lien dynamique sur une page de confirmation d'enregistrement

Quand j'accède à la page mon enregistrement est confirmé

Alors je reçois un email de confirmation d'enregistrement

Et je peux désormais me connecter au site

# Les Méthodes Agiles

## User Stories

- User stories et Personas :
- Il est souvent bienvenu de définir et de discuter des rôles (on dit des "personas" / personnages) du projet.
- Certains poussent le « vice » à les nommer, fabriquer des photos, décrire leur caractère.
- L'utilisateur devient réel : on commence à concevoir le produit comme une réponse à un vrai besoin, à une vraie nécessité.
- Pour cela ne dites donc plus systématiquement « l'utilisateur » mais le « responsable des ventes », « la personne qui voyage beaucoup », etc.

Avant même d'écrire ses user stories, il apparaît donc indispensable d'avoir en amont des personae (soit nos utilisateurs types).



# Les Méthodes Agiles

## User Stories

- Personas : Exemple

| Elisabeth : La gestionnaire  |   |
|--|---|
|    | <p>Contexte</p> <ul style="list-style-type: none"><li>• Gérer ses comptes est devenu une activité quasi quotidienne</li><li>• Jongle entre son travail, sa vie de famille, et ses loisirs</li><li>• Sites préféré XXXX</li><li>• .....</li><li>➔ On veut la fidéliser</li><li>➔ On veut la pousser à épargner</li><li>➔ On veut améliorer son expérience client</li></ul> |
| Buts et comportements  | Ce que cela implique  |
| <ul style="list-style-type: none"><li>• Utilise beaucoup internet dans le cadre de son travail</li><li>• Vient 3 à 4 fois par semaine en soirée</li><li>• Veut consulter ses compte</li><li>• Veut faire des virement : nationaux et internationaux</li><li>• Veut échanger avec un conseiller</li><li>• .....</li></ul> | <ul style="list-style-type: none"><li>• Un accès rapide à ses comptes</li><li>• Des raccourcis intuitifs pour ses actions simples et fréquentes</li><li>• Un tableau de bord simplifié basé sur la synthèse des comptes</li><li>• .....</li></ul>   |

# Les Méthodes Agiles

## Etude de Cas

- 01 - Prune the tree



# Les Méthodes Agiles



## 01 Exercice : Prune the Tree

- Préparation de l'atelier Prune the Product Tree
- Les participants auront 20 minutes pour définir chaque pré-requis, fonctionnalités clés et fonctionnalités complémentaires / différenciantes du produit.
  - Ils noteront une idée par post'it et poseront ceux-ci au fur et à mesure sur l'arbre.
  - Le but du facilitateur lors de cet exercice sera de challenger l'ensemble des participants sur le produit afin de ne pas oublier les pré-requis mais aussi le besoin d'innovation.
- Quand les 20 minutes sont écoulées, le facilitateur va enlever les doublons et demandera aux participants de voter pour définir les idées les plus importantes (si il y a trop de post'it sur l'arbre).
  - Le facilitateur amènera une grande feuille de papier de type paperboard et des feutres de couleurs. L'idéal sera d'avoir cette feuille collée au mur ou sur un board blanc.
  - Celui-ci mettra également des post'it de couleurs claires et des stylos/feutres pour chacun des participants.

# Les Méthodes Agiles



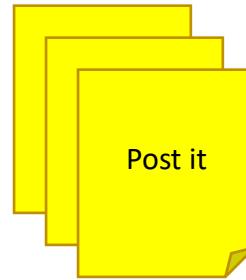
## 01 Exercice : Prune the Tree

- Besoins Client
  - L'application (le système) devra permettre à l'utilisateur de faire le choix d'un parking.
  - Ce choix devra être réalisable par détection automatique de la situation géographique ou par choix personnel.
  - Le système devra permettre à l'entreprise de facturer le client par l'utilisation de sa place de parking.

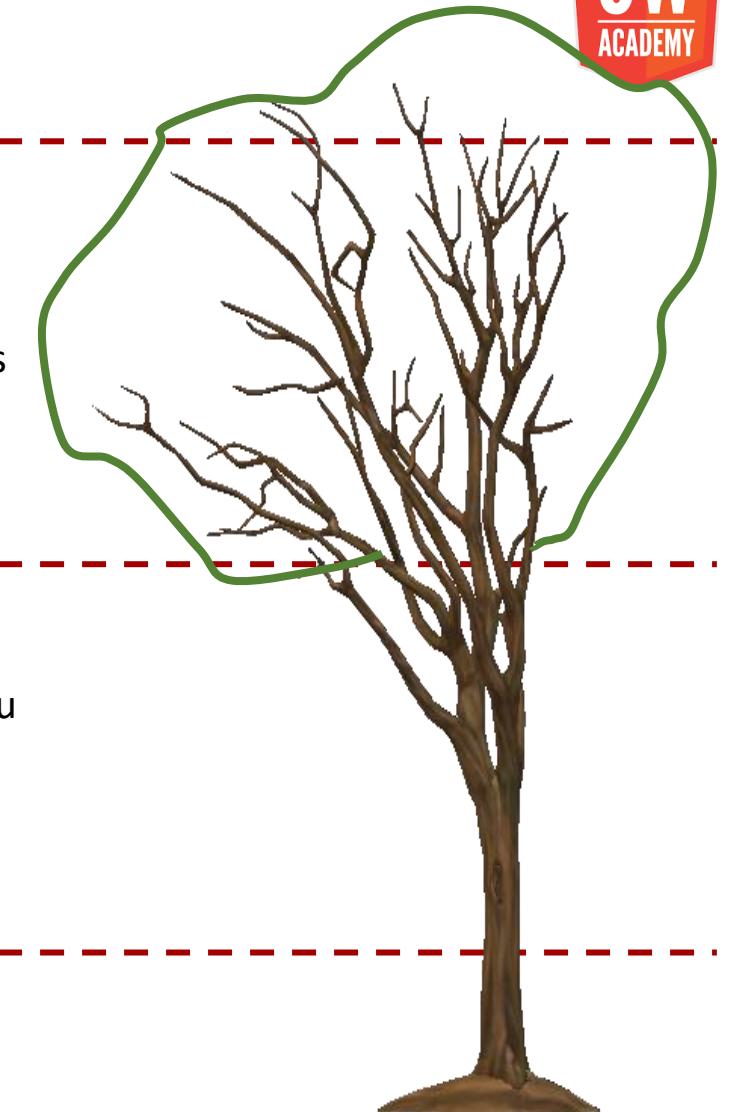
# Les Méthodes Agiles

## 01 Exercice : Prune the Tree

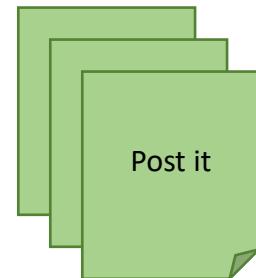
Fonctionnalités supplémentaires / Différenciantes



les branches pour décrire les fonctionnalités complémentaires au produit voire différenciantes

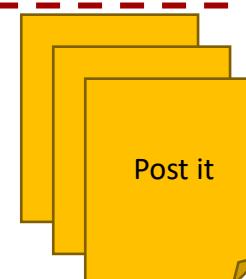


Fonctionnalités clés



le tronc pour les fonctionnalités indispensables au produit/service (connexion utilisateur...)

Pré-requis



les racines pour les pré-requis du produit/service (développeur, infrastructure, prestataire...)

# Les Méthodes Agiles

## 01 Corrigé : Prune the Tree

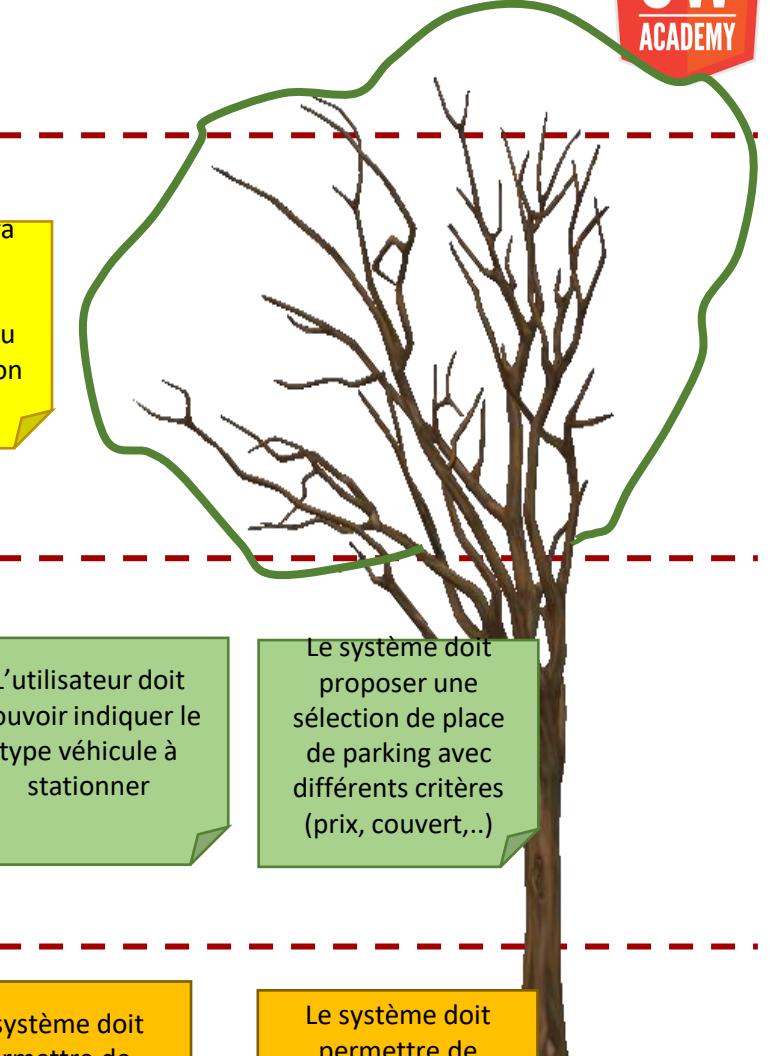
Fonctionnalités supplémentaires / Différenciantes

La sélection peut être basée sur la géolocalisation

Le système devra proposer l'itinéraire d'accès au parking

Le système devra accepter plusieurs moyen de paiement

Le système devra adresser un message régulièrement du temps d'utilisation du parking



Fonctionnalités clés

L'applicatif devra permettre à l'utilisateur de sélectionner un parking

L'applicatif doit enregistrer l'utilisateur afin qu'il puisse accéder au système

L'applicatif doit permettre la facturation sur la base de du temps d'occupation du parking

L'utilisateur doit pouvoir indiquer le type véhicule à stationner

Le système doit proposer une sélection de place de parking avec différents critères (prix, couvert,...)

Pré-requis

Le système doit être multi plates-formes

Le système doit permettre la reconnaissance unique de l'utilisateur

Le système doit permettre d'identifier toutes les places disponibles

Le système doit permettre de mettre à jour la liste des parking éligibles

Le système doit permettre de contrôler l'arrivée et le départ de la place de parking

# Les Méthodes Agiles



## 01 Corrigé : Prune the Tree

### Mots-clés

- Appli mobile
  - Utilisateur
    - Choix parking
      - Automatique (géolocalisation)
      - Personnel
    - Facturation
      - Paiement

### Les exigences

- E1 : Le système doit être multi plateforme mobile
- E2 : Le système doit permettre la reconnaissance unique de l'utilisateur
- E3 : L'applicatif devra permettre à l'utilisateur de sélectionner un parking
- E4 : Sélection / géolocalisation
- E5 : Sélection personnel /de convenance
- E6 : L'utilisateur ayant libéré sa place de parking devra être facturé
- E7: Permet à l'utilisateur de payer sa facture en ligne

# Les Méthodes Agiles



## 02 Exercice : Formalisation des Epics, Features, Stories

- Besoins Client

Un site e-commerce : Ce commerce électronique concerne le processus d'achat/vente de livres sur Internet.

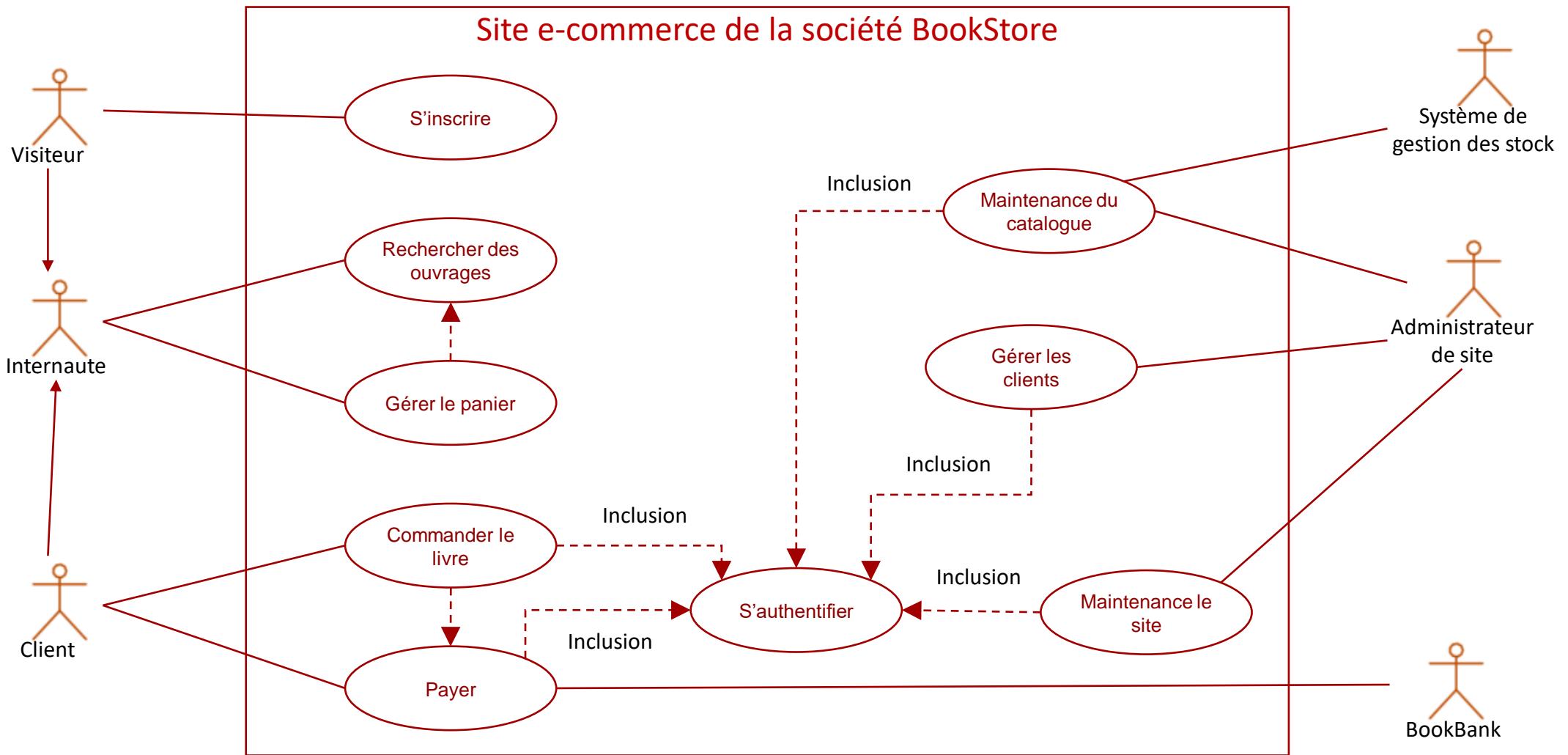
➔ Actuellement, La société BookStore souhaite réaliser un site e-commerce pour gérer seulement la vente de ses livres en ligne, gérer son catalogue de livre et sa base de données de clients. L'objectif fondamental du futur site BookStore est de permettre aux internautes de rechercher des ouvrages par thème, auteur, mot-clé, etc., de se constituer un panier virtuel, puis de pouvoir les commander et les payer directement sur le Web.

- La société BookStore est spécialisée dans la vente de livres de tout type (science, littérature, Art, etc...) et sur tout support (livre papier, numérique, DVD, etc..).
- BookStore reçoit ses commandes par fax et les chèques par courrier puis envoie le bon de commande au client. Une fois le chèque encaissé par la banque partenaire BookBank, elle utilise la société de transport BookEx pour acheminer les livres vers leurs clients.
- La société BookStore possède un système « Gestion des stocks » qui met à jour les données concernant le prix et l'état du stock des livres du catalogue. Ce système est automatiquement chargé dans la base de données de façon périodique.

➔ Faire l'analyse et Identifier les Epics, Features, et rédiger les Stories

# UML : Etude de cas

## 02 Corrigé: Formalisation des Epics, Features, Stories - Prune the Tree



## 02 Corrigé: Formalisation des Epics, Features, Stories - Prune the Tree

- Les Epics sont :
- En tant que client du site e-commerce je veux pouvoir commander des livres en lignes
  - L'application (le système) devra permettre à l'utilisateur de s'inscrire
  - L'application (le système) devra permettre à l'utilisateur de s'authentifier
  - L'application (le système) devra permettre à l'utilisateur de rechercher des ouvrages
  - L'application (le système) devra permettre à l'utilisateur de rechercher de gérer le panier
  - L'application (le système) devra permettre à l'utilisateur de rechercher de commander le livre
  - L'application (le système) devra permettre à l'utilisateur de rechercher de payer
  - L'application (le système) devra permettre au gestionnaire de gérer les clients
  - L'application (le système) devra permettre au gestionnaire de maintenir le catalogue
  - L'application (le système) devra permettre au gestionnaire de maintenir le site

# Les Méthodes Agiles

## 02 Corrigé: Formalisation des Epics, Features, Stories - Prune the Tree

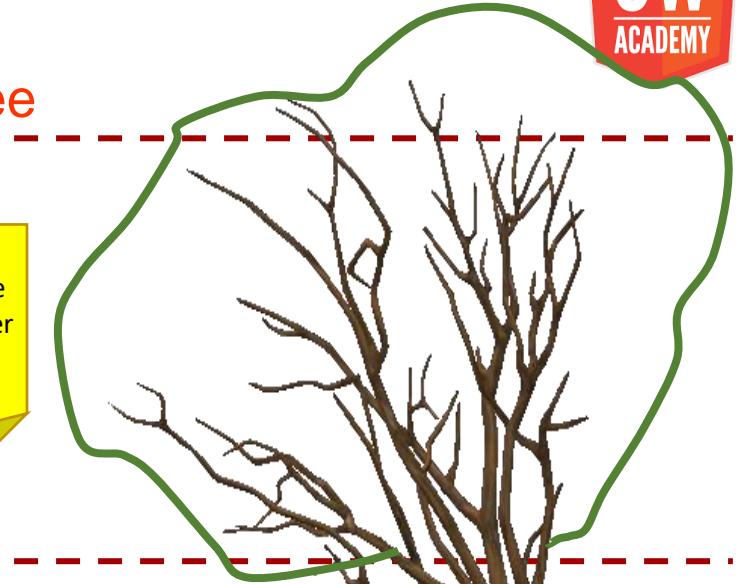
### Fonctionnalités supplémentaires / Différenciantes

En tant que client je veux que le système m'adresse des propositions d'ouvrage

En tant que client je veux pouvoir suivre le statut de ma commande

En tant que client je veux pouvoir échanger avec un support de clientèle (réclamation, ...)

En tant que client je veux pouvoir annuler ma commande



### Fonctionnalités clés

En tant que visiteur je dois pouvoir m'inscrire

En tant qu'internaute je dois pouvoir visionner le catalogue (critères de choix)

En tant qu'internaute je dois pouvoir sélectionner des ouvrages (critères de choix)

En tant que client je dois pouvoir payer ma sélection d'ouvrage

En tant que internaute je dois pouvoir gérer ma sélection d'ouvrage de mon panier

En tant que client je veux pouvoir choisir un mode de livraison (société, points relais, adresse,...)

En tant que client je veux pouvoir choisir un mode de règlement

### Pré-requis

Le système doit être multi plates-formes

L'applicatif doit permettre au gestionnaire de gérer le stock

L'applicatif doit permettre au gestionnaire de gérer le catalogue (ouvrage, prix,...)

L'applicatif doit permettre à l'administrateur de gérer les clients (nouveaux, internaute, client,...)

L'applicatif doit permettre de référencer des sociétés (modalités, prix, modes de livraison,...)

## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | <b>Le Product Backlog</b>         | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Les Méthodes Agiles



## Le Product Backlog

- **Backlog**

- Le Backlog produit liste toutes les fonctionnalités, les fonctions, les exigences, les améliorations et les corrections qui constituent des modifications à apporter au produit dans les versions futures.
- Les items du Backlog produit ont les attributs d'une description, d'un ordre, d'une estimation et d'une valeur.
- Les items du Backlog produit incluent souvent des descriptions de test qui prouveront leur complétude lorsqu'ils sont « Finis »

# Les Méthodes Agiles

## Le Product Backlog

- La liste de tous les besoins exprimés que le Projet a pour cible
  - Une rencontre entre les besoins métiers et le point de vue technique
  - Exprimé de façon à mettre en évidence la valeur apportée par chaque composant (User Story)

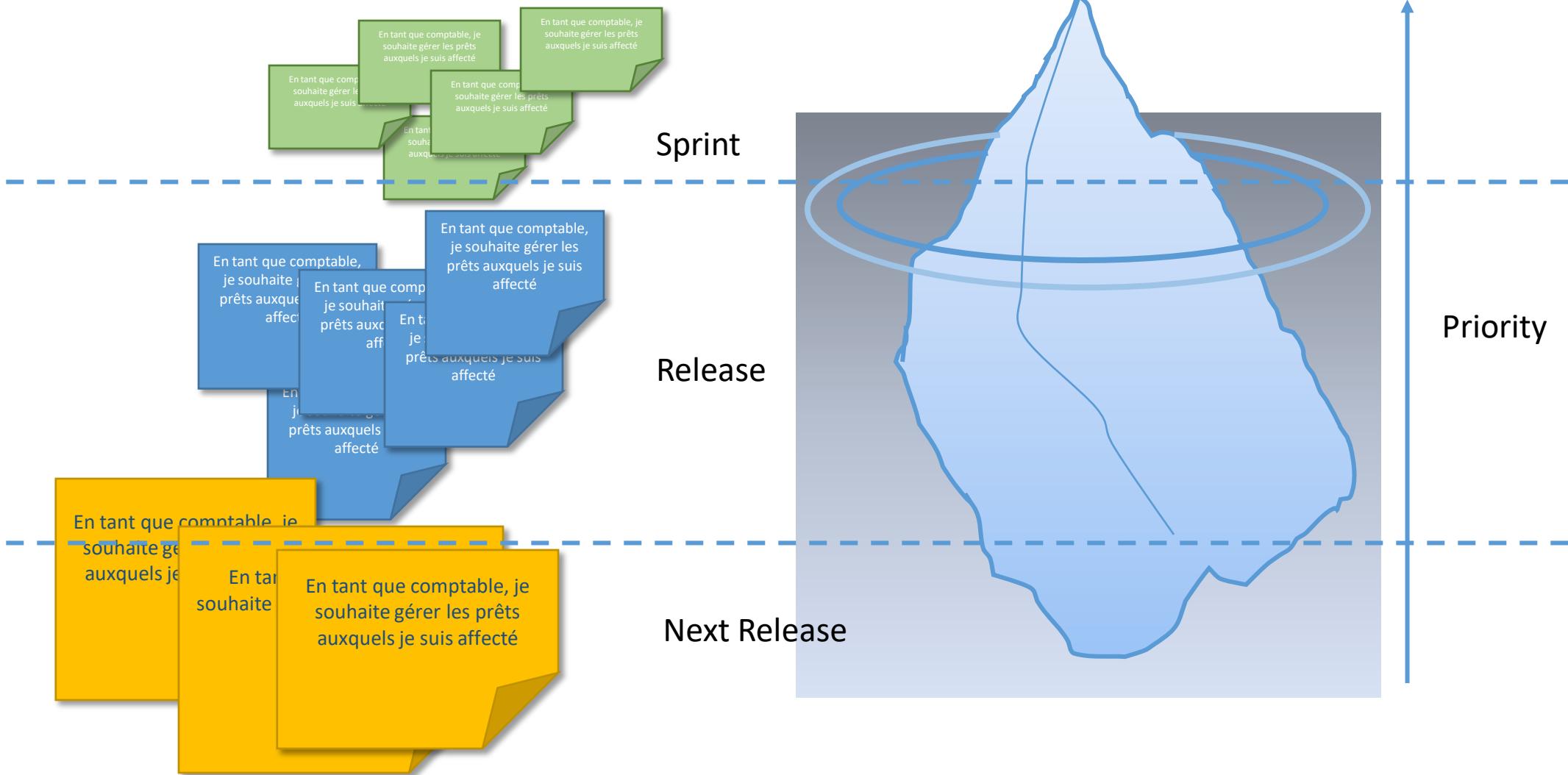


Le Product Owner

| Product Backlog      |          |
|----------------------|----------|
| Demandes             | Priorité |
| Demande A : xxxxxxxx | 1        |
| Demande B            | 2        |
| Demande C            | 3        |
| Demande D            | 4        |
| Demande E            | 5        |
| Demande F            | ..       |
| Demande ...          |          |

# Les Méthodes Agiles

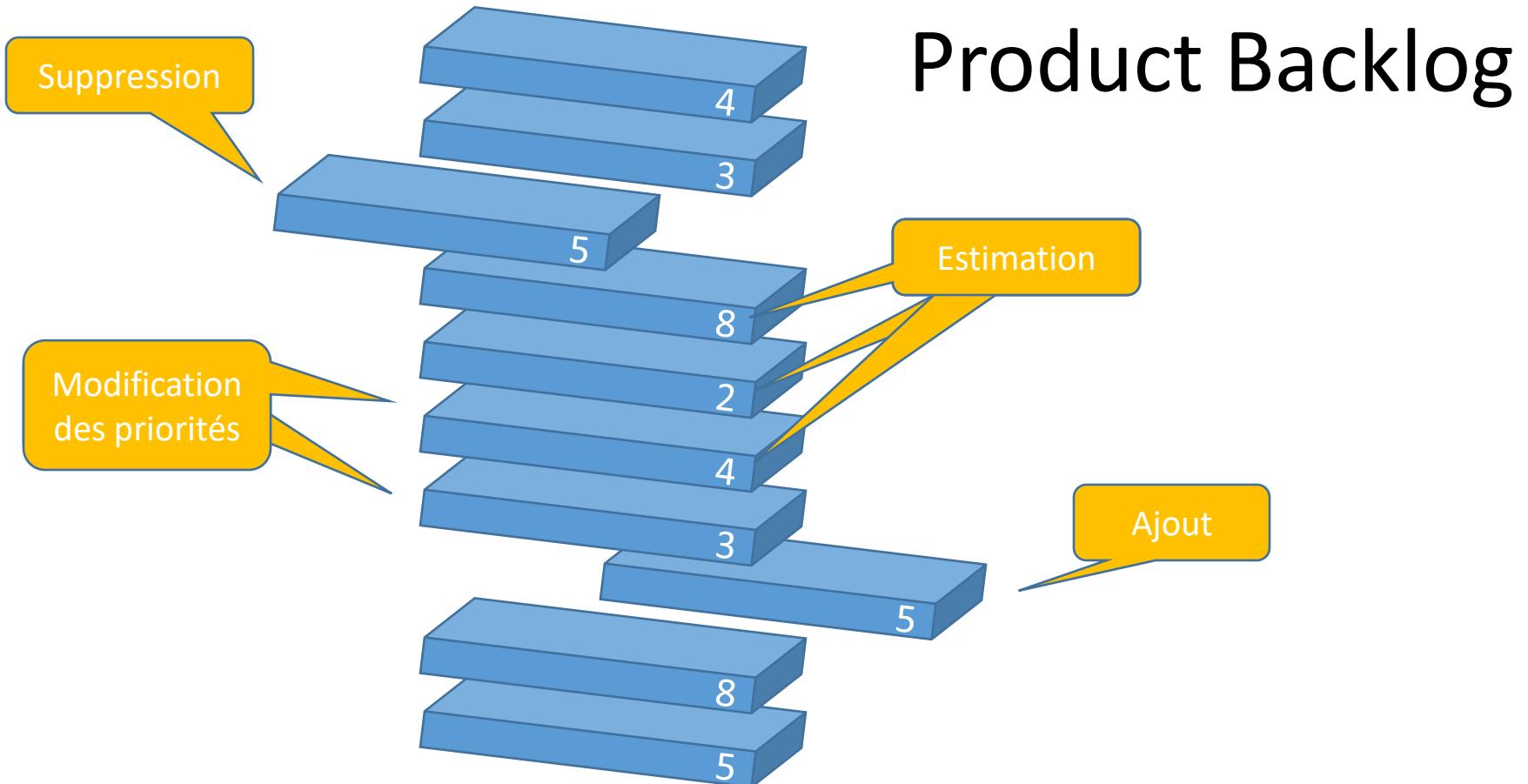
## Le Product Backlog



# Les Méthodes Agiles

## Le Product Backlog

- Priorisé par le Product Owner, repriorisé en début de chaque sprint, ou à tout moment



# Les Méthodes Agiles



## Le Product Backlog

- Définir un Backlog



Atelier Backlog

| Atelier Backlog |   |   |                |            |
|-----------------|---|---|----------------|------------|
| En tant que ... | Je veux ....  | Afin que (je puisse)                        | Business value | Estimation |
| Responsable     | Valider toutes les demandes                           | Les procédures de demande continue          | 80             | 20         |
| Chef de gestion | Établir les tableau de synthèse de toutes les données | ...   | 80             | 40         |
| Visiteur        | Pouvoir imprimer les informations de mes demandes     | Pour avoir un trace papier de mes démarches | 50             | 8          |
| Client          | Effectuer des demandes                                | ...   | 50             | 12         |
| Administrateur  | Pouvoir administrer toutes les données                | Pour effectuer les sauvegardes              | 30             | 20         |
| ...             |   |   | ...            | ...        |
|                 |   |   |                | ...        |

# Les Méthodes Agiles



## Notion de fini « Done »

- Done
  - Cette notion a pour objectif de fixer le niveau de qualité attendu, et ce dans tous les cas de figure. C'est une condition pour dire qu'une user story est finie.
  - Cette notion doit être partagée et connue de tous
  - Généralement elle est partagée entre les différents projets, au niveau de l'entreprise
  - Cette notion est très liée à la notion de test (unitaire et d'acceptation)
  - La notion de « done » implique plusieurs couches : technique, métiers, IHM, documentation, etc
- Définissez les rôles d'un projet
  - Quelle serait la notion de « fini » pour chacun de ces rôles ?
  - Quelle serait la notion de « fini » sur le projet Scrum de cette équipe ?

la définition de « Done » par l'équipe Equipe XXX

- Toutes les tâches sont terminées
- Tous les tests sont exécutés et réussis
- Les Procédures (workflow) manuelle sont testées sur tous les navigateurs définis
- Les impacts des opérations d'implémentation sont étudiés et mesurés
- Des tests de performance sont validés
- toutes les anomalies sont fermées

## Questions Ouvertes

**Des questions ?**



# Sommaire

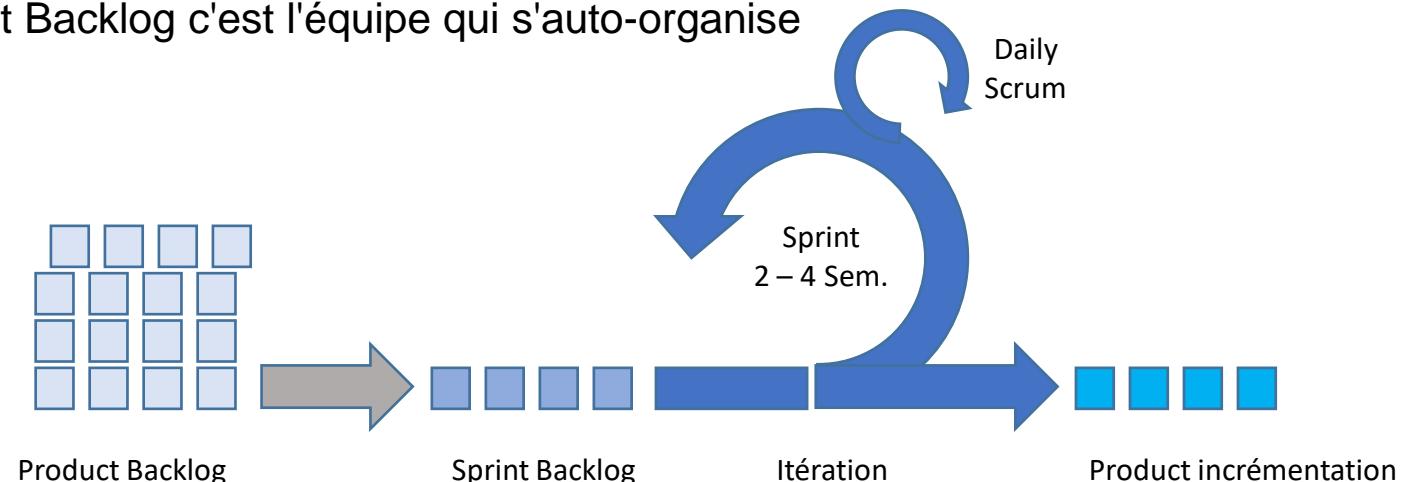


| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Les Méthodes Agiles

## Le Sprint

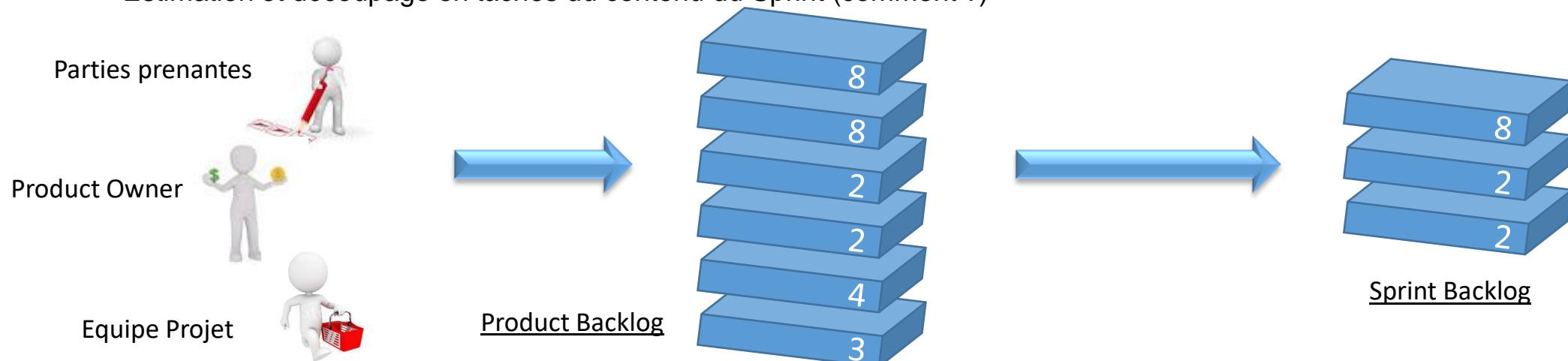
- On parlera de Sprint ou d'itération
  - Il dure entre 2 semaines ou 1 mois selon les équipes, mais sa durée ne varie plus dans un même projet.
  - Une durée de 2 semaines facilite la prise en main de Scrum car elle donne de la visibilité à intervalle resserré.
  - Il faut décomposer les tâches de façon à ce qu'elles durent un jour maximum pour assurer de la visibilité sur l'avancement
  - Si en cours de Sprint l'équipe souhaite changer la priorité (sortir une story, en intégrer une nouvelle en provenance du Product Backlog) c'est toujours le Product Owner qui aura le dernier mot
  - Concernant le Sprint Backlog c'est l'équipe qui s'auto-organise



# Les Méthodes Agiles

## Le Sprint

- Le Sprint Planning
  - Le Sprint Planning permet de définir le Sprint Backlog
  - Le Sprint Backlog est un sous ensemble du Product Backlog
  - Le Sprint Backlog correspond à un accord mutuel entre le Product Owner et l'équipe mais c'est le Product Owner qui a le dernier mot.
  - Le Sprint Planning se découpe en deux étapes
    - Définition du contenu du Sprint (quoi ?)
    - Estimation et découpage en tâches du contenu du Sprint (comment ?)



# Les Méthodes Agiles

## Le Sprint

- Le Sprint Planning : Quoi ?
  - En fonction de la dernière revue de sprint, de la dernière rétrospective, d'autres informations, le Scrum Master aura pris soin de sensibiliser le Product Owner à valider ou à reprioriser le Product Backlog avant cette réunion
  - Le Product Owner présente le Product Backlog à l'équipe
  - L'équipe estime quelles « user story » (dans l'ordre des priorités) sont susceptibles d'intégrer le Product Backlog (la vitesse est une indication forte). Elle s'engagera là-dessus
  - L'équipe peut proposer et justifier un ordonnancement du Product Backlog différent mais c'est le Product Owner qui aura le dernier mot.
  - On peut dans cette phase estimer les User Story qui ne le sont pas encore (on va travailler généralement sur 60/70% du Product Backlog)
  - On n'assigne jamais une User Story à l'équipe, c'est l'équipe qui définit son propre potentiel et qui s'auto-organisera quand à l'affectation des tâches



# Les Méthodes Agiles

## Le Sprint

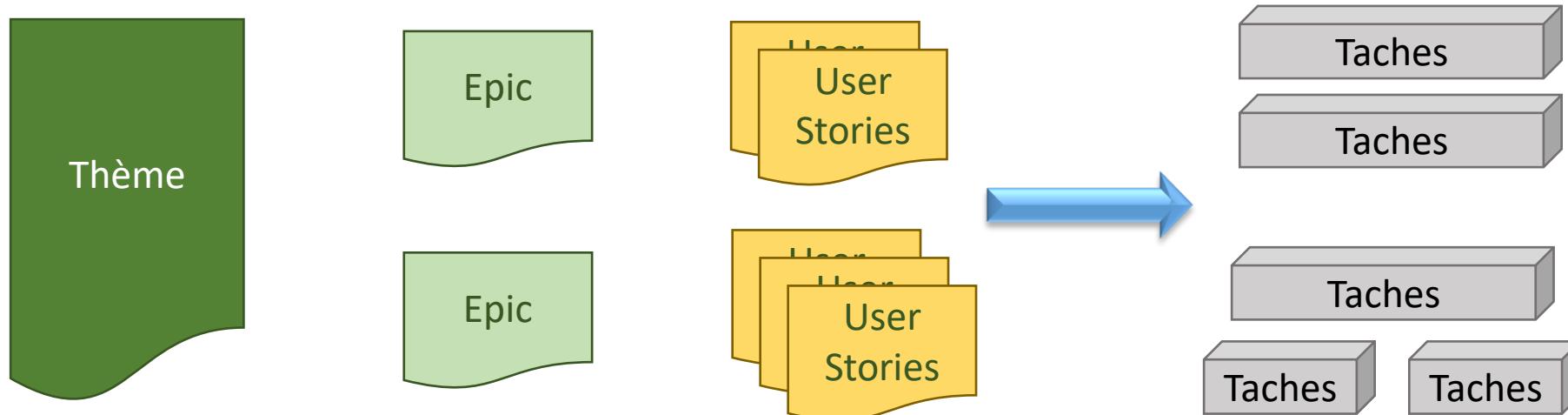
- Le Sprint Planning : Quoi ?
- Objectif
  - Cette première partie de « Sprint Planning » est aussi le moment où l'on définit le but du Sprint, son « Leitmotiv », son objectif, sa cible.
  - Par exemple :
    - « Mise en œuvre du système de réservation en ligne des croisières »
    - « Réalisation des échanges entre le site web et l'ERP concernant les réservations en lignes »
  - Cet objectif doit être l'élément indispensable à atteindre pour que l'on puisse considérer que le sprint est une réussite.
  - Si des arbitrages doivent avoir lieu durant le Sprint, c'est souvent l'objectif du Sprint qui permettra de faire pencher la balance.
- Engagement
  - La fin de cette première partie de Sprint Planning se conclue avec l'engagement de l'équipe sur un périmètre.



# Les Méthodes Agiles

## Le Sprint

- Le Sprint Planning : Comment ?
  - La seconde partie du Sprint Planning consiste à découper les User Story en tâches.
  - Il est nécessaire de discuter du contenu de chaque Story et de l'intérêt de chaque tâche
  - Il ne faut pas oublier les tâches de spécifications, de documentation, de test, etc. Tout ce qui est nécessaire pour que la Story soit achevée convenablement (voir la notion de « done » (fini)).
  - On peut ajouter des tâches sans Story (bugs, etc.) mais essayez plutôt d'ajouter des Story de type « process » ou « system » (afin que la vitesse reste au plus juste)



# Les Méthodes Agiles

## Le Sprint

- Le Sprint Planning : Comment ?
  - Les tâches sont estimées (ou pas) en heures selon la maturité de l'équipe (on peut se passer de l'estimation).
  - On peut aussi estimer les tâches en point (avec un nouveau planning poker, mais c'est relativement rare et très chronophage)
  - L'estimation d'une tâche en heure ne devrait pas dépasser 1 jour de façon à pouvoir rapidement détecter les dérives le cas échéant

| Story                               | A Faire   | En Traitement   | En Validation                      | Fini  |
|-------------------------------------|---|---|------------------------------------|---|
| <p>En tant que ...<br/>8 points</p> | <p>Code le ...<br/>8 points</p> <p>Code le ...<br/>2 points</p> <p>Test le ...<br/>8 points</p> <p>Test le ...<br/>7 points</p> <p>Code le ...<br/>5 points</p> <p>Test le ...<br/>4 points</p> | <p>Code le ... DC<br/>8 points</p> <p>Code le ... SC<br/>2 points</p> | <p>Test le ... SC<br/>7 points</p> | <p>Test le ... DC<br/>Test le ... SC<br/>Code le ...<br/>Code le ... CS<br/>Code le ... DC<br/>7 points</p> |
| <p>En tant que ...<br/>8 points</p> | <p>Code le ...<br/>4 points</p> <p>Code le ...<br/>5 points</p> <p>Test le ...<br/>8 points</p> <p>Test le ...<br/>3 points</p>   | <p>Code le ... DC<br/>4 points</p>                                    |                                    | <p>Code le ... DC<br/>Test le ... SC<br/>Test le ... SC<br/>7 points</p>                                    |

# Les Méthodes Agiles



## Le Sprint : Fin exceptionnelle ? , De Stabilisation ? , Sprint 0

- Le Sprint peut être annulé
  - Il n'a plus de valeur !
  - Il est impossible de le réussir
- Seul le Product Owner peut annoncer l'annulation et la fin du Sprint (mais l'équipe ou le management peuvent essayer de le convaincre)
- Les « user stories » estimées « finies » sont auditées
- Tous les « user stories » non achevées sont replacées dans le Product Backlog
- Existe des « sprint de stabilisation » ?
- Qu'est ce qu'un Sprint 0 ?

# Les Méthodes Agiles



## Le Sprint : Fin exceptionnelle ?, De Stabilisation ?, Sprint 0

- Existe des « sprint de stabilisation » ?
  - Dernier sprint pour garantir la complétude du projet et son exploitabilité par l'utilisateur final : niveau qualité et attendu
- Qu'est ce qu'un Sprint 0 ?
  - L'objectif est de mettre en place des fondations techniques en début de projet ; cela dans le but de réaliser les futures premières user-stories sur un socle technique à peu près en place.
- Le Sprint
  - Un Sprint peut être considéré comme un mini projet avec :
    - un objectif clair et immuable (le Sprint Goal),
    - une durée fixe (maximum un mois),
    - une qualité irréprochable (rendue transparente dans la définition de « Fini »)

Par conséquent, un Sprint n'a pas d'adjectifs : le Sprint 0, le Sprint de stabilisation, le Sprint de test, le Sprint de conception, etc. n'existent pas en Scrum

## Questions Ouvertes

**Des questions ?**



# Sommaire

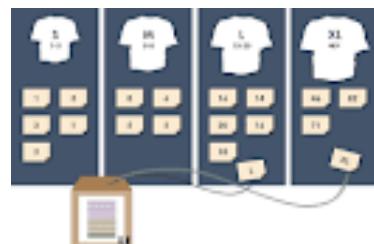


| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | <b>Pratiques d'estimation</b>       |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Les Méthodes Agiles

## Pratiques d'Estimation

- L'estimation est collective, c'est l'équipe qui la réalise mais en conversant avec le Product Owner et le Scrummaster
- L'estimation est exprimée en « story point » à partir d'un planning poker, soit exprimés en « jour idéaux ». Je préconise le planning poker car toute notion de durée (jour) vient troubler la réflexion. Des fois on utilise même des « points animaux » ou des tailles de tshirt...
- L'estimation est relative : une « story » par rapport aux autres (triangulation)
- Attention de bien prendre en compte tout ce qui a été prévu par « done »
- En fonction de l'estimation réalisée sur des story il se peut que le Product Owner repriorise son Product Backlog
- Estimation de la complexité, de la durée des tâches à accomplir, de la prise de risque, etc.



# Les Méthodes Agiles



## Pratiques d'Estimation

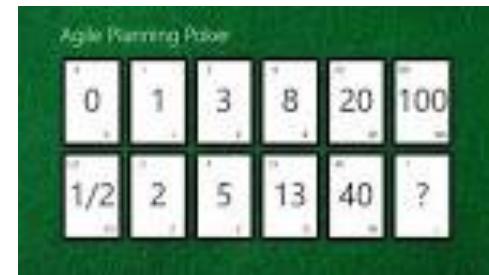
- **Point de Story**
  - Mesure un effort
  - Relatifs jamais absous
  - Difficile à faire comprendre au management
- **Jours idéaux**
  - Mesure une durée
  - Devraient être relatifs
  - Plus aisés à expliquer aux personnes externes
  - On les raccroche trop à une notion de calendrier

# Les Méthodes Agiles

## Pratiques d'Estimation

- **LE "PLANNING POKER"**

- Chaque personne possède un jeu de planning poker
- Le Product Owner décrit la Story devant être estimée
- On discute de la story
- Tout le monde propose sa mise simultanément
- Si toutes les estimations convergent, c'est ok
- Si les estimations ne convergent pas, les discussions reprennent, notamment entre les deux personnes ayant les plus grands écarts de point.



# Les Méthodes Agiles

## Pratiques d'Estimation

- **LE « PLANNING POKER » : Ça Marche**

- Ce sont les gens qui réaliseront la tâche qui l'estiment !
- Le planning poker nécessite que l'on justifie les points que l'on attribue
- Propose un étalonnage assez réduit qui limite les erreurs (si une Story est trop grosse, on la découpe en plusieurs Story)
- Les valeurs sont relatives
- Les valeurs sont prédéfinies (on ne chipote pas...)
- Tout le monde a son mot à dire (parmi l'équipe)
- C'est rigolo



# Les Méthodes Agiles



## Pratiques d'Estimation

### • LE « PLANNING POKER » : Déroulement

- Chaque développeur reçoit un paquet de cartes
- Pour chaque « User Story » ou activité à évaluer
  - Le modérateur lit la description
  - Le product owner réponds aux éventuelles questions
  - Chaque développeur choisit ensuite une carte pour cette estimation , la carte reste cachée
- Ensuite, les cartes sont retournées
  - Les estimations vont différer , la plus grande et la plus petite explique leur point de vue
  - On discute
  - Et ainsi de suite
- Le product owner utilise les résultats pour fixer les priorités
- A la fin de l'itération, on compare l'estimation aux nombres de jours réel
  - On obtient la vitesse de l'équipe
- Au début il faut bien choisir la 1ère Story qui va servir à l'estimation
  - Trop grande on va se retrouver avec des fractions
  - Trop petite : l'estimation sera trop facile

### • LE « PLANNING POKER » : Pourquoi ça marche

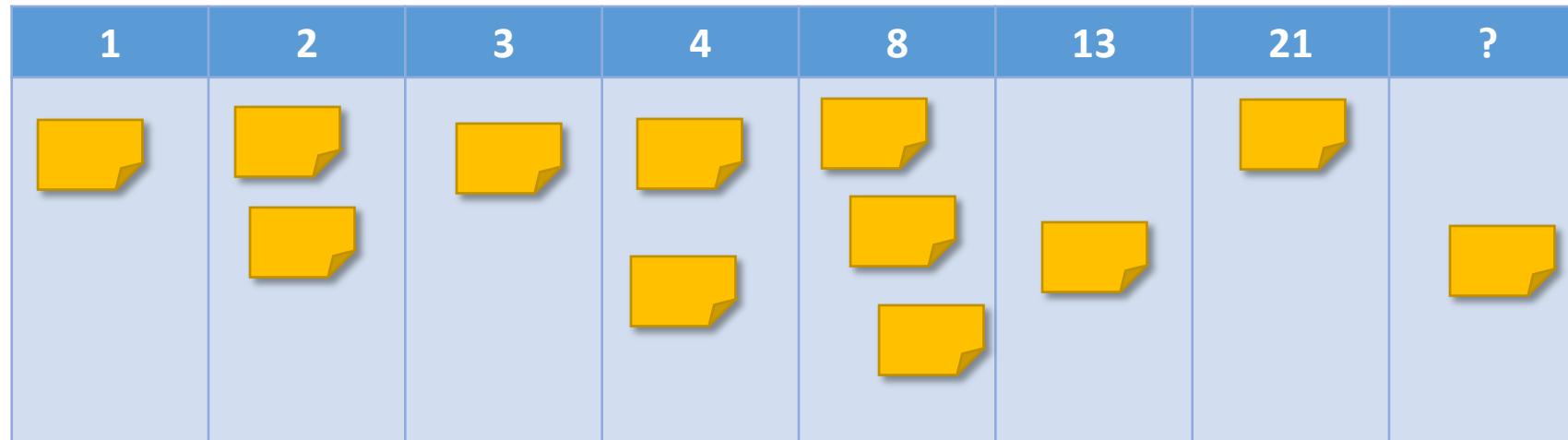
- Plusieurs experts donnent leurs opinions sans s'influencer
  - On ne parle pas
  - On n'influence pas par le langage du corps
- Cela améliore la qualité de l'estimation
  - On doit justifier ses estimations
  - Moyenner les estimations des personnes donne de meilleurs résultats
- L'utilisation des nombres de **FIBONACCI**
  - 1 pour une tâche extrêmement simple, comme la correction d'un libellé par exemple,
  - 2, 3, 5 pour une tâche légèrement plus complexe, comme la création d'un formulaire de saisie simple,
  - 8, 13, 21, 34, 55, 89, 144 si l'on ne dispose pas de suffisamment d'informations pour estimer correctement cette tâche.

# Les Méthodes Agiles

## Pratiques d'Estimation

- **Une alternative LE « WALL PLANNING POKER »**

- On positionne les stories par comparaison en colonne sur un mur
- On affecte les points ensuite
- C'est très rapide et on oublie toute valeur numérique !
- Mais l'investissement n'est pas forcément égal entre tous les participants
- Ma recommandation : faire des wall planning poker hors des sprints planning pour faire des estimations assez loin dans le temps ou initier un projet, utiliser le planning poker lors des sprint planning.

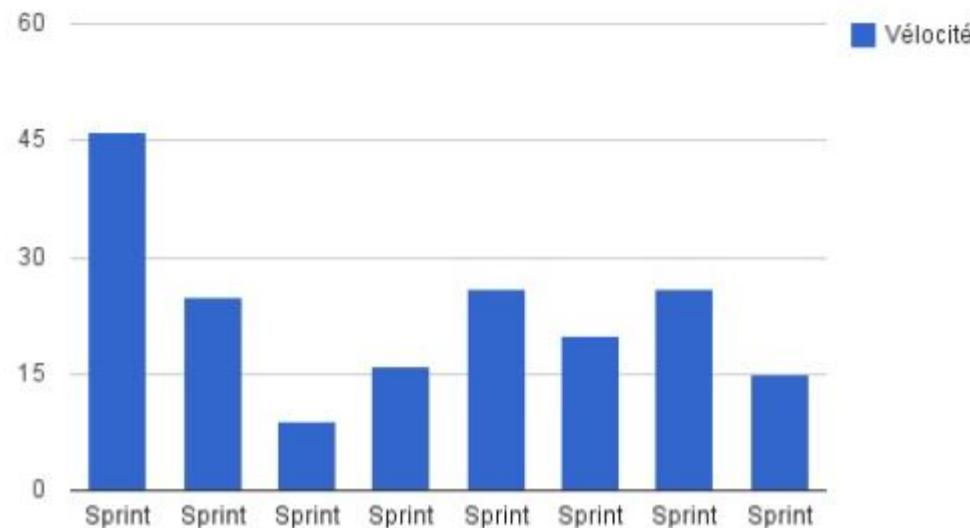


# Les Méthodes Agiles

## Pratiques de Planification

- **PRATIQUE DE PLANIFICATION : LA "VÉLOCITÉ"**

- Est calculée à la fin du sprint (itération)
- C'est la somme des points des « Story » qui ont été acceptées pour la démo
- La régularité de la courbe de la vélocité est signe du respect de la qualité, ou de la stabilité de l'environnement (et la stabilité de l'environnement mène au respect de la qualité)
- Si on décide de rompre avec la qualité pour améliorer la vélocité (fin des tests, etc.) on va améliorer celle-ci temporairement puis décliner vers un « noyau foutu ». C'est la notion de dette technique.



# Les Méthodes Agiles



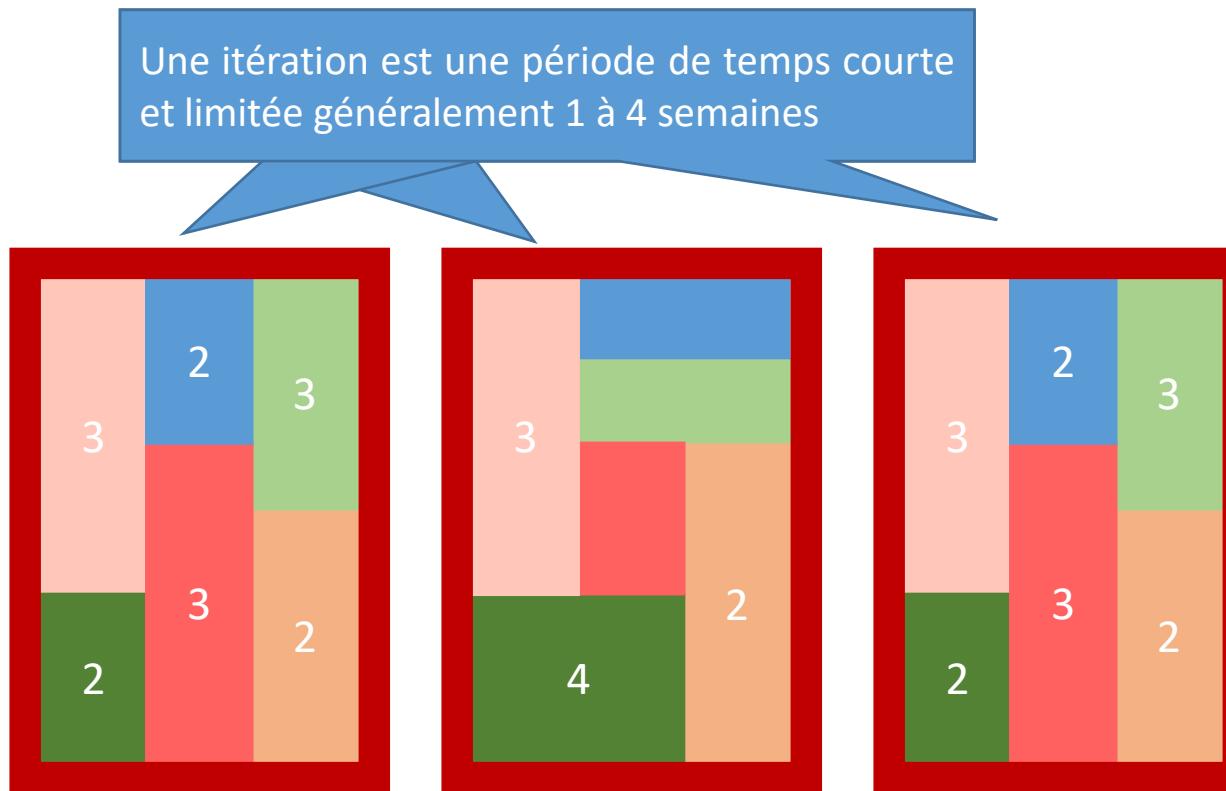
## Pratiques de Planification

- **PLANIFICATION de RELEASE**

- Une version de produit a une « release » et a un product backlog
- Une « release » contient plusieurs sprints (itérations)
- Chaque sprint possède une vélocité
- La vélocité est une valeur (non relative à une durée comme jour ou heure) que l'on déduit en additionnant la vélocité de chaque story (nous verrons comment nous calculons la vélocité plus loin)
- Chaque sprint est « timeboxé » (disons 3 semaines)
- L'équipe est sensée être pérenne et donc la vélocité devrait donc peu ou prou rester la même par sprint

# Les Méthodes Agiles

## Pratiques de Planification



Une version comprend généralement plus d'une itération

La « Velocity » est la quantité de travail planifiée ou réalisée est une itération

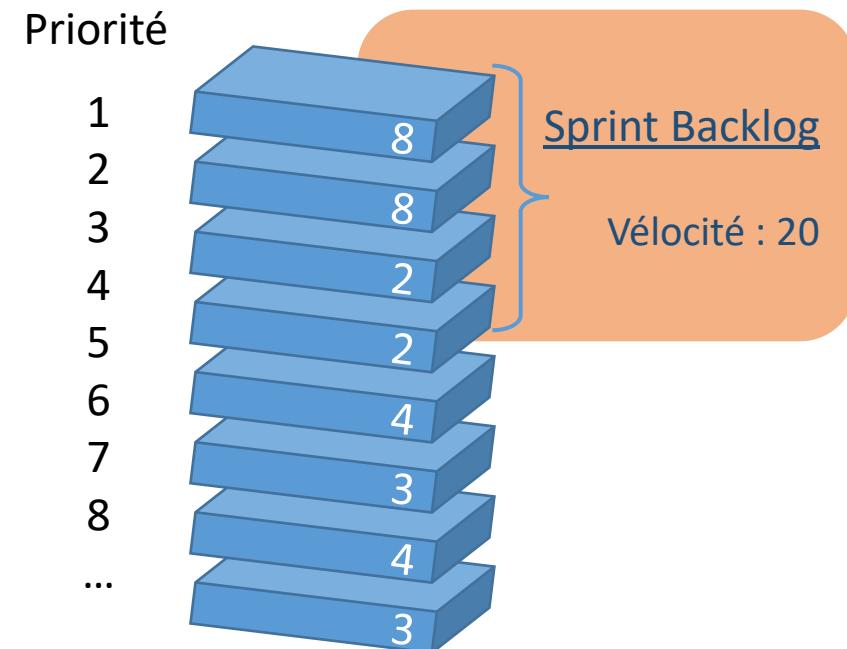
La vélocité est un indicateur utilisé sur des projets gérés à l'aide d'une méthode agile, comme Scrum par exemple. La vélocité agile permet de déterminer l'effort qu'est capable de fournir une équipe de développement pour la réalisation des tâches programmées dans un sprint. Elle est exprimée en nombre de points.

# Les Méthodes Agiles

## Pratiques de Planification

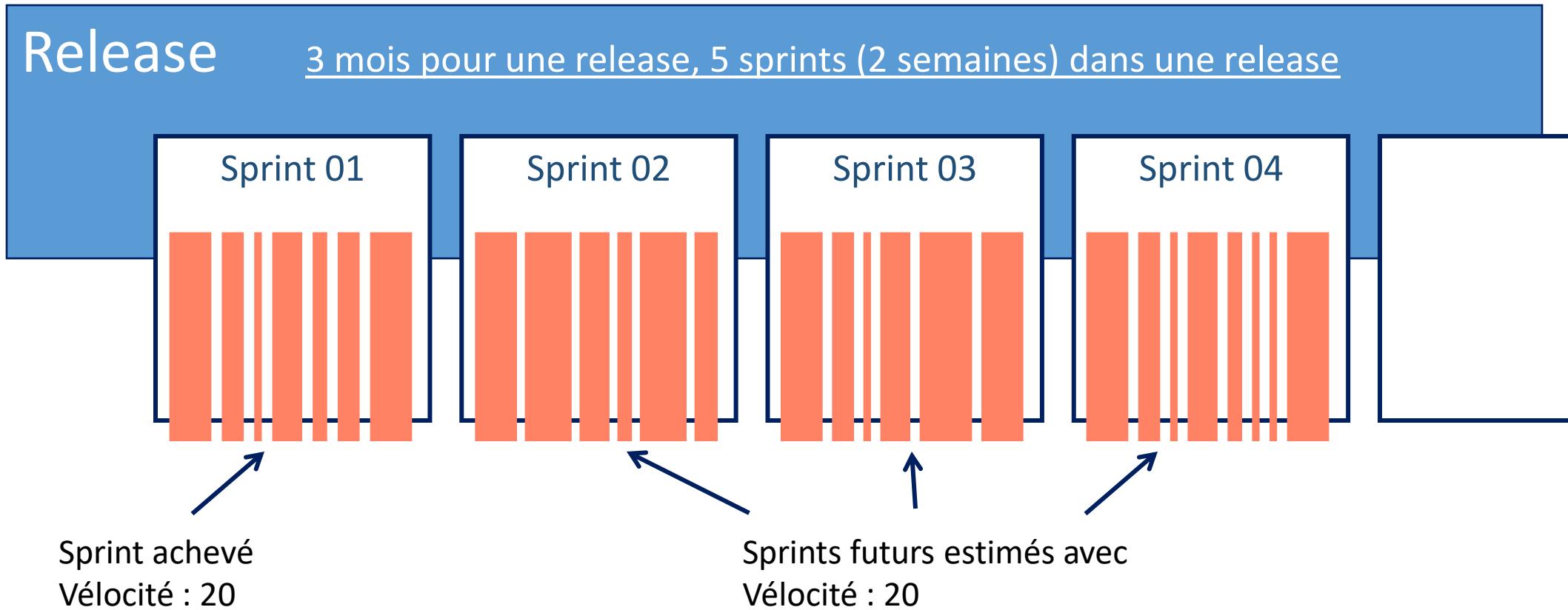
- **PLANIFICATION de RELEASE**

- On sait donc calculer le nombre de sprints nécessaires pour atteindre la fin de la release.
- La fin de la release cela peut être
  - Le Product Backlog est vide (rarement le cas...)
  - Une date prédéfinie
  - La fin du budget
  - Assez de valeur produite
  - Etc.



# Les Méthodes Agiles

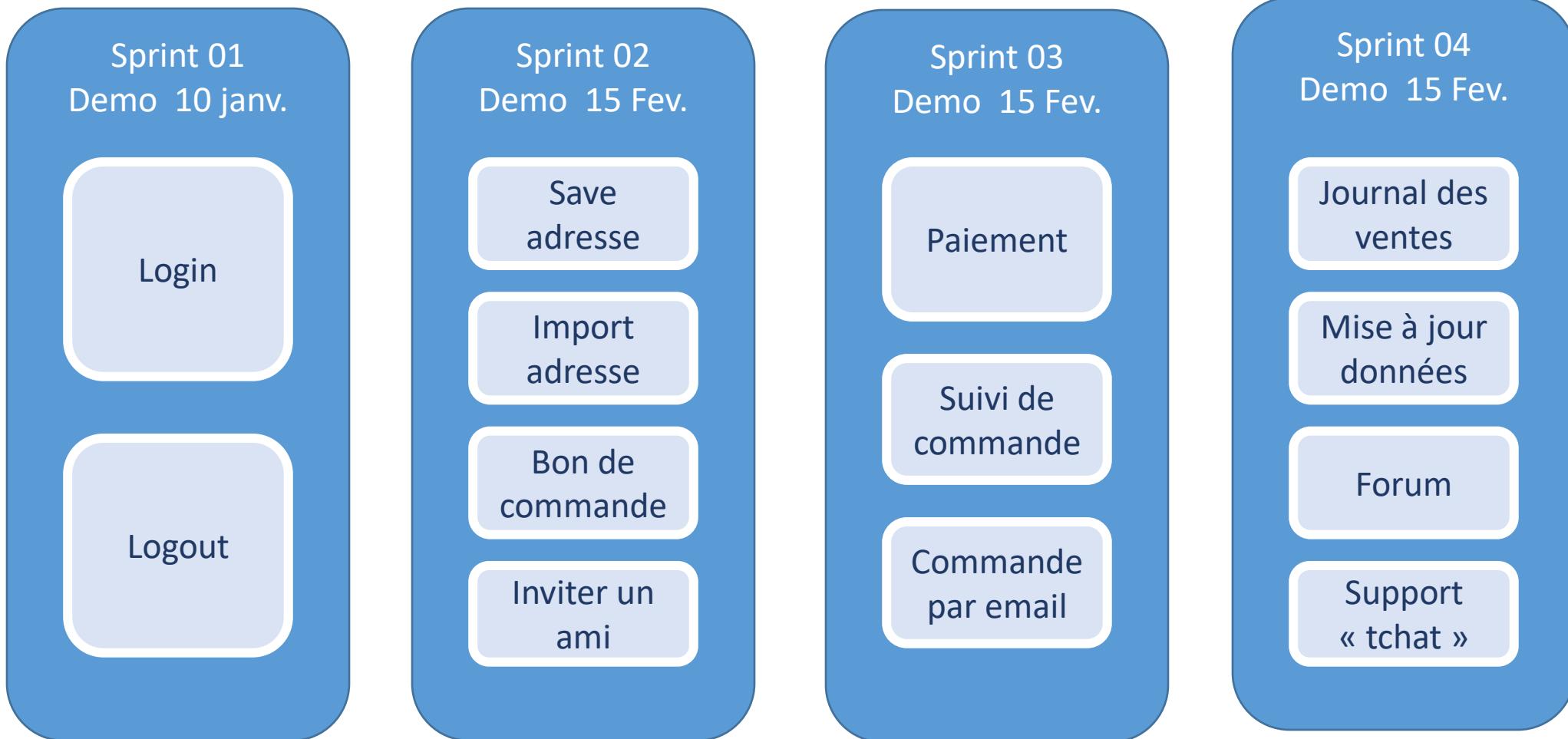
## Pratiques de Planification



# Les Méthodes Agiles



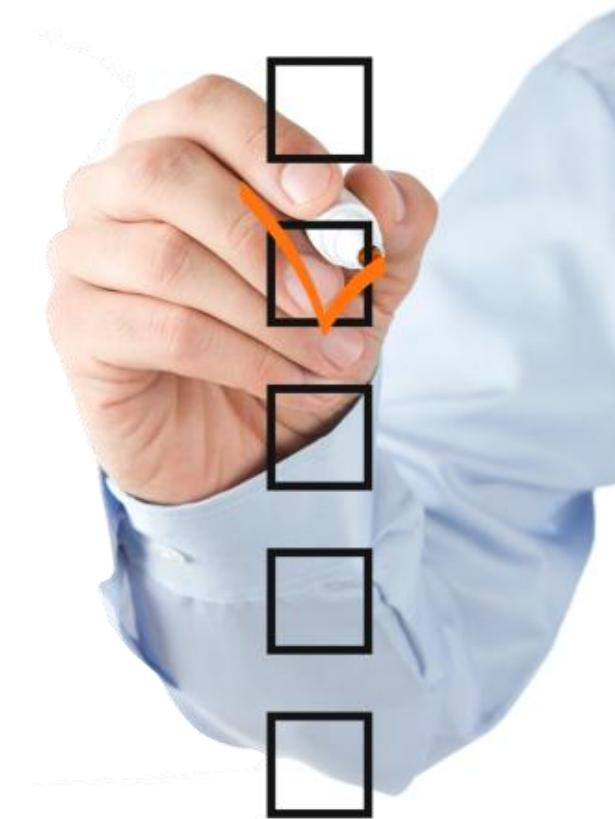
## Pratiques de Planification



# Les Méthodes Agiles

## Etude de Cas

- 01 – Estimation Doggy Planning



# Les Méthodes Agiles



## Pratiques d'Estimation

- Estimation Doggy Planning
  - Pour chaque chien ci-dessous, estimatez l'effort de travail (taille des Stories) requis pour toileter le chien.
  - Vous supposez que vous ayez les outils et l'expérience nécessaires pour toileter les chiens.
  - Vous discutez avec vos coéquipiers de ce que signifie le toilettage
- Dans le cadre de mes expériences, le toilettage comprend
  - le lavage total
  - le séchage
  - le brossage
  - la coupe des griffes
  - la coupe des poils
  - Autres ....

# Les Méthodes Agiles

## Pratiques d'Estimation

|   |  |   |   |
|---|--|---|---|
| Golden retriever : 70 – 100 cm, 60 - 90 kg  | Teckel a poil court : 6-9 kg   | Caniche : 38-45 cm, 18 – 38 kg  | Bouvier bernoisDog : 63-71 cm, 30-55 kg   |
|    |    |    |    |
| German Shepherd : 58-66 cm, 22-40 kg  | Yorkshire terrier : 12 cm, <5 kg   | Beagle : 33-40 cm, 8-15 kg  | Boxer : 66-78 cm, 25-50 kg  |
|    |    |    |    |
| Bulldog : 20-55 kg  | Labrador Retriever : 53-63 cm, 140-300 kg  | Danois : 70-90 cm, 54-90 kg   | Komondor (25-32 in, 90-130 lbs)   |
|  |  |  |  |

# Les Méthodes Agiles



## Pratiques d'Estimation

- Estimation Doggy Planning
- Quelles sont certaines des questions à résoudre lors de l'estimation du toilettage de chacun de ces chiens?
  - Vous êtes-vous demandé comment couper les poils du Komondor?
  - Quelle a été la réponse de votre Product Owner?
  - Avez-vous réfléchi à la taille d'un ou deux boxers?, Le tempérament du Yorkie importait-il à votre équipe?
- Le moyen le plus simple de dimensionner tous ces chiens
  - Sélectionnez un chien simple comme point de départ de la discussion.
  - Déterminez si les chiens suivants sont plus grands ou plus petits
  - Lorsque le groupe a obtenu un consensus, collez-le sur le mur à gauche ou à droite du chien initial
  - Continuez avec le chien suivant en comparant chaque fois plus petit / plus grand
- C'est l'activité du dimensionnement relatif

# Les Méthodes Agiles

## Pratiques d'Estimation

- **Estimation Doggy Planning (1 point → 10 mn)**

- On positionne les stories par comparaison en colonne sur un mur
- On affecte les points ensuite

| 1  | 2 | 3 | 5   | 8  | 13  | 21  | 34  | 55  | 89   | 144   |   |
|--|---|---|---|--|---|---|---|---|--|---|---|
|  |   |   |  |  |  |  |  |  |  |  |  |

## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | <b>Suivi &amp; Reporting</b>        |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

## Suivi & Reporting

- **Radiateurs d'information**

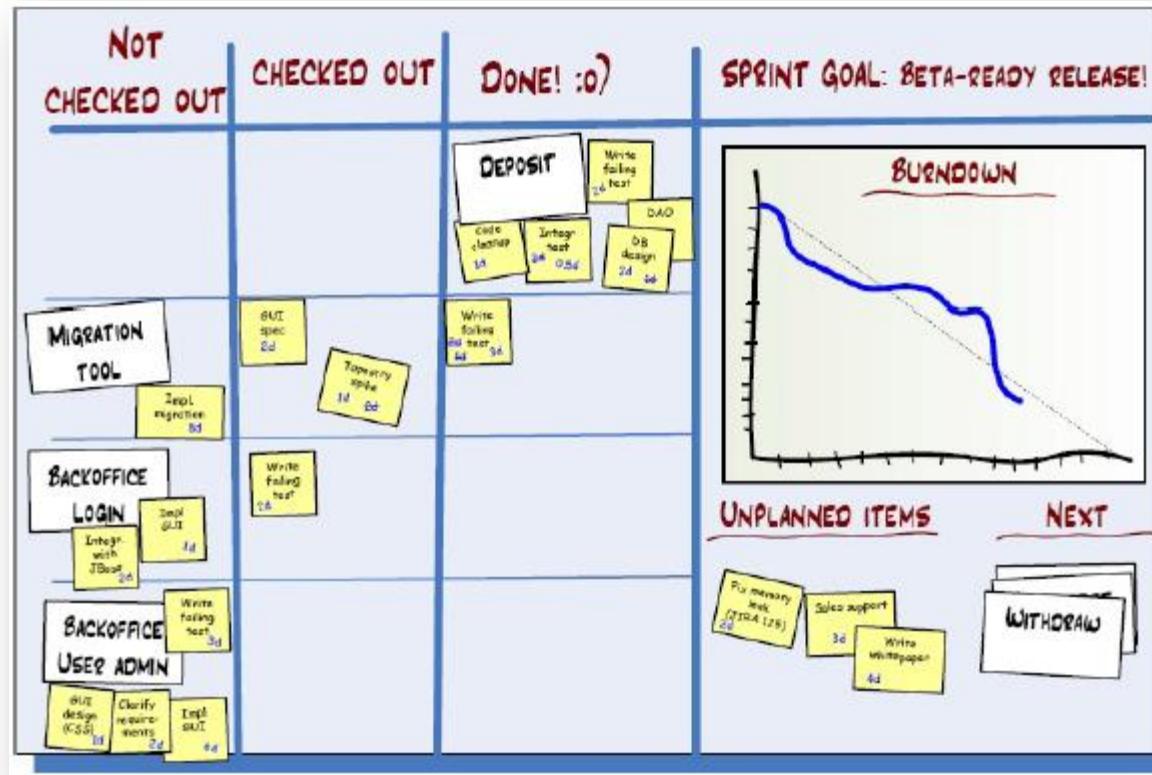
- Ils permettent de suivre de façon collective, interactive, et simple l'avancée du sprint, et du projet
  - Ils engagent et motivent l'équipe
  - La mise à jour est quasi en temps réel
  - Ils doivent être visibles par tous (autant par l'équipe que par les parties prenantes).
  - Toutes les informations clefs sont rassemblées en un seul lieu
- 
- Ps : attention aux coups de vent, aux ménages le soir, aux post-it défectueux que l'on retrouve au sol...
  - Astuce : prendre en photo le radiateur au jour le jour

"Radiateur d'information" est le terme générique désignant un affichage mural (passif, par exemple une feuille de papier, ou actif, par exemple un écran affichant des résultats d'intégration continue) réalisé par une équipe pour diffuser publiquement une information jugée pertinente: nombre total de tests, vélocité, nombre d'incidents en exploitation, etc.

# Les Méthodes Agiles

## Suivi & Reporting

- Radiateurs d'information : Exemple



# Les Méthodes Agiles



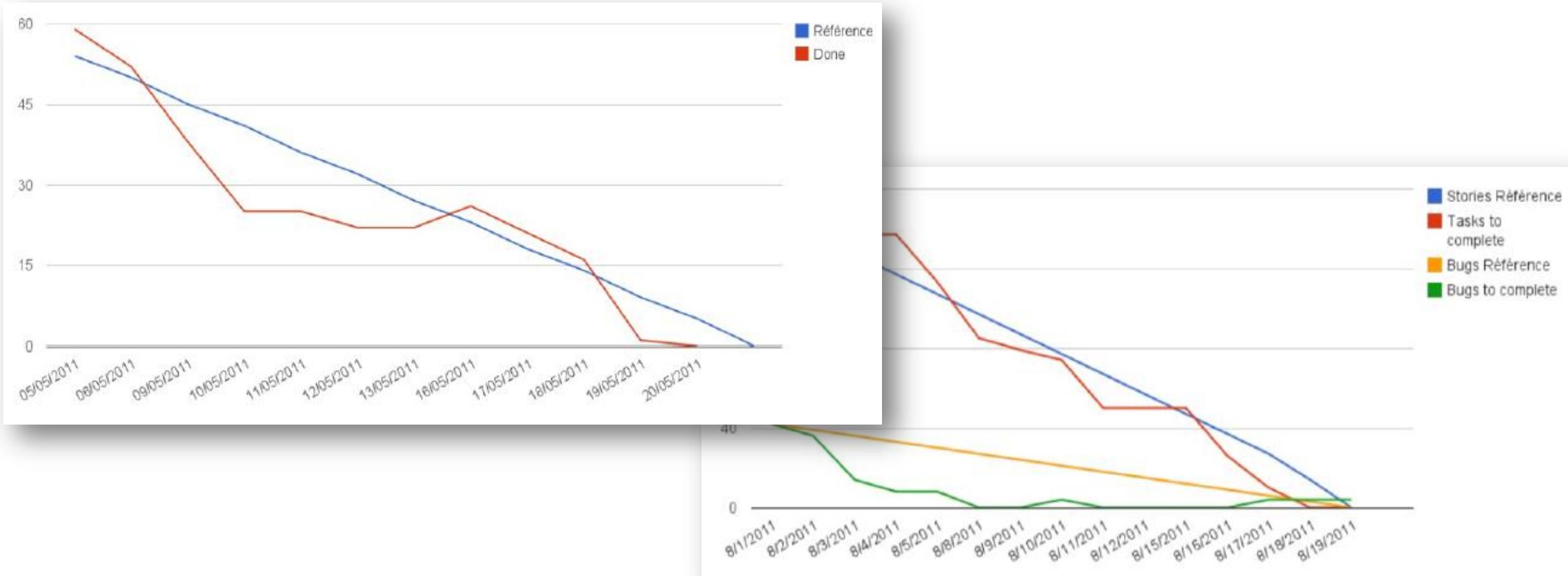
## Suivi & Reporting

- **Burndown Chart**
  - Les burndown charts montrent l'avancement du projet
- Il existe 2 types de burndown chart : celui de la release, celui du sprint
  - Celui de la release se base sur les points de story
  - Celui du sprint peut se baser :
    - Sur les heures
    - Sur les points affectés aux tâches
    - Sur le nb de tâches restantes
- L'avancée du projet est collective
- Les burndown charts doivent être visibles par tous (autant par l'équipe que par les parties prenantes).

# Les Méthodes Agiles

## Suivi & Reporting

- **Burndown Chart : Exemple**



# Les Méthodes Agiles

## Suivi & Reporting

- **Pratique Quotidienne : Le « Daily Scrum »**

- Il dure ~ 15 mn, il a lieu tous les jours, tout le monde est debout (standup meeting), chacun y répond à 3 questions
  - Qu'est ce que tu as réalisé hier ?
  - Quelles sont tes tâches aujourd'hui ?
  - Est ce que quelque chose te gène dans la réalisation de tes tâches ?



# Les Méthodes Agiles



## Suivi & Reporting

- **Pratique Quotidienne : Le « Daily Scrum »**

- Il donne de la visibilité à toute l'équipe sur l'ensemble des travaux à intervalles réguliers
- Il devrait se dérouler systématiquement à la même heure et dans le même lieu
- C'est l'outil de l'équipe, le ScrumMaster, le Product Owner sont optionnels. Le ScrumMaster doit cependant s'assurer que le Daily Scrum se déroule comme convenu mais il doit se faire discret lors du StandUp. Le Product Owner peut profiter de ce moment pour suivre l'avancement du projet. Les parties prenantes peuvent assister mais ne doivent pas intervenir. Seuls les membres de l'équipe peuvent parler.
- Il faut trouver un lieu tranquille mais proche des « radiateurs d'informations »
- Le Daily Scrum se déroule impérativement debout.

## Suivi & Reporting

- **Pratique Quotidienne : Le « Daily Scrum »**

- Il améliore la communication et permet d'éviter d'autres réunions (mais son sujet ne doit pas dévier).
- Chacun doit s'exprimer brièvement car le Daily Scrum est « Timeboxé » à 15mn.
- Attention de ne pas dévier sur des conversations annexes (règlement d'un point technique complexe, modification des priorités, etc.), ni de régler lors du point les problèmes, il s'agit de les repérer pour les régler ultérieurement.
- Chacun s'engage, car l'équipe est responsable. On ne fait pas de compte rendu au Product Owner ou au ScrumMaster (d'ailleurs ces deux rôles ne sont peut-être pas présents)
- Ce n'est pas non plus le lieu des règlements de compte (l'équipe doit comprendre qu'elle doit s'unir pour réussir)
- Par moment quand un Daily Scrum dérape le ScrumMaster peut se tourner face au mur et appuyant ses mains contre celui-ci (« spread eagle ») pour indiquer que ce qui se déroule est inutile et n'est pas le but du Daily Scrum.

# Les Méthodes Agiles



## Suivi & Reporting

- **Revue de Sprint**
  - On présente le résultat du Sprint (toutes les User Story achevées)
  - C'est le moment durant lequel le Product Owner et les parties prenantes (stakeholders) inspectent le produit en détails en fonction de l'objectif du Sprint et la vision du produit et prennent les décisions d'adaptation nécessaires à la réussite de l'objectif du projet.
  - Il peut y avoir de nombreux invités à cette occasion
  - Il faut préparer la réunion (scénariser) et rappeler les objectifs du Sprint en premier lieu
  - A la fin de la démo on peut calculer la vitesse effective du Sprint et donc réajuster le plan de release
  - A la fin de la démo le Product Backlog est actualisé (User Story achevées, nouvelles User Story émergentes, anomalies, etc.)
  - **Les BurndownChart sont réinitialisés ou remis à jour**

# Les Méthodes Agiles



## Suivi & Reporting

- **Retrospective**

- Elle est là pour améliorer les processus : (identifier les axes d'amélioration, capitaliser sur les bonnes pratiques), faire un état des lieux du sprint qui vient de s'achever
  - Comment améliorer les choses ?
  - Qu'est ce qui a bien fonctionné ?
  - Qu'est ce qui a mal fonctionné ?
  - Comment résoudre les problèmes que nous voyons apparaître ?
- Il faut préparer une rétrospective (la préparation peut durer aussi longtemps que la rétrospective elle-même)
- C'est le ScrumMaster qui va faciliter et organiser cette réunion
- C'est le ScrumMaster qui devrait savoir si une amélioration est réellement nécessaire ou pas, il ne va pas choisir mais il sera un arbitre déterminant.
- **A la fin de la rétrospective on a un plan d'action**

# Les Méthodes Agiles

## Suivi & Reporting

### • Retrospective : Exemple de Mise en œuvre → Le Speed Boat

#### Le vent

Ce qui nous a poussé et fait avancé

- Qu'est ce qui nous motive ?
- Quels sont nos moteurs ?

#### Le bateau

C'est l'équipe, les collaborateurs  
C'est le sprint

Tous les acteurs du projet Agile écrivent des post-it et les déposent

#### Le Soleil

Qu'est-ce qui à bien été pendant le sprint

- Qu'avons-nous réussi à accomplir ensemble ?
- Qu'ai je réussi à construire moi-même ?

#### L'encre coincée

Ce qui nous freine

- Qu'est-ce qui nous a encombrés ?
- Qu'est-ce qui nous a compliqué la tache ?

Tous les acteurs du projet Agile identifient les axes d'amélioration et les classent par domaine

#### Les Rochers

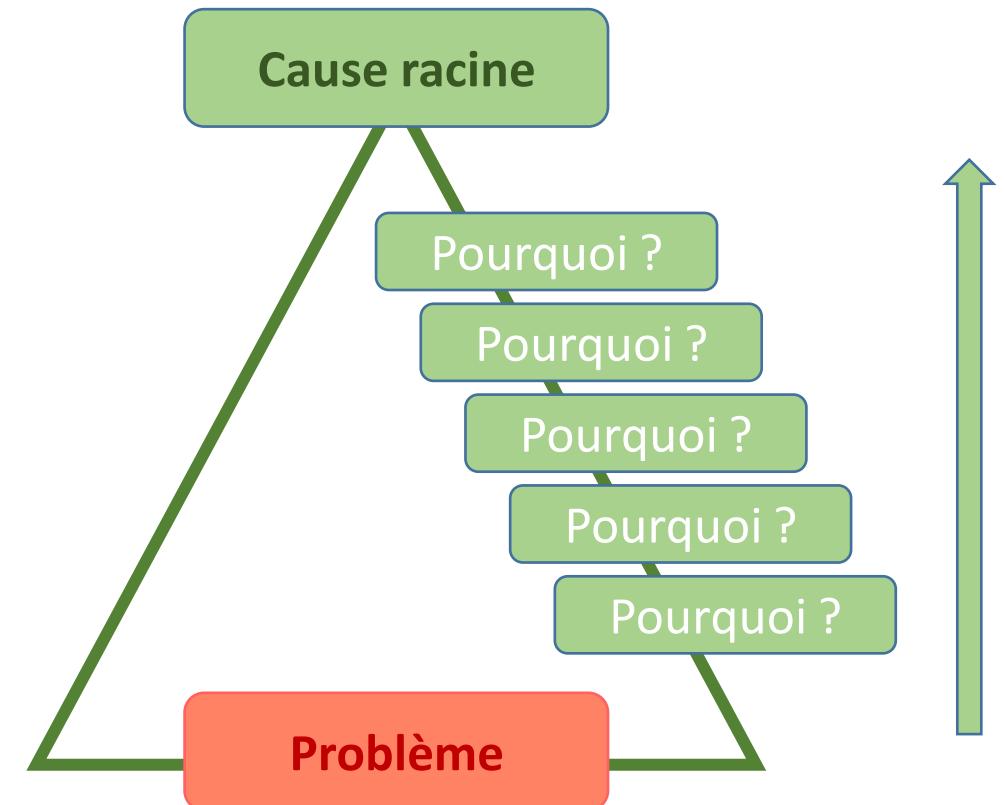
Les dangers, les accueils

- Qu'est-ce qui aurait pu nous faire échouer ?
- Quelles étaient nos craintes ?
- Quels étaient les pièges et les obstacles ?

# Les Méthodes Agiles

## Suivi & Reporting

- **Les 5 pourquoi : L'obstacle n'est pas toujours celui que l'on croit : Concept des 5 pourquoi**
- Partir de l'effet constaté « en l'occurrence le problème » et remonter vers la cause racine par une succession de questions "Pourquoi ?"
- La méthode est à utiliser avec les personnes concernées par le problème ou qui ont pu le constater
  - Rester factuel et clair
  - Ne pas faire de supposition ou de déduction
  - Ne mentionner que les causes sur lesquelles vous avez un contrôle



# Les Méthodes Agiles

## Suivi & Reporting

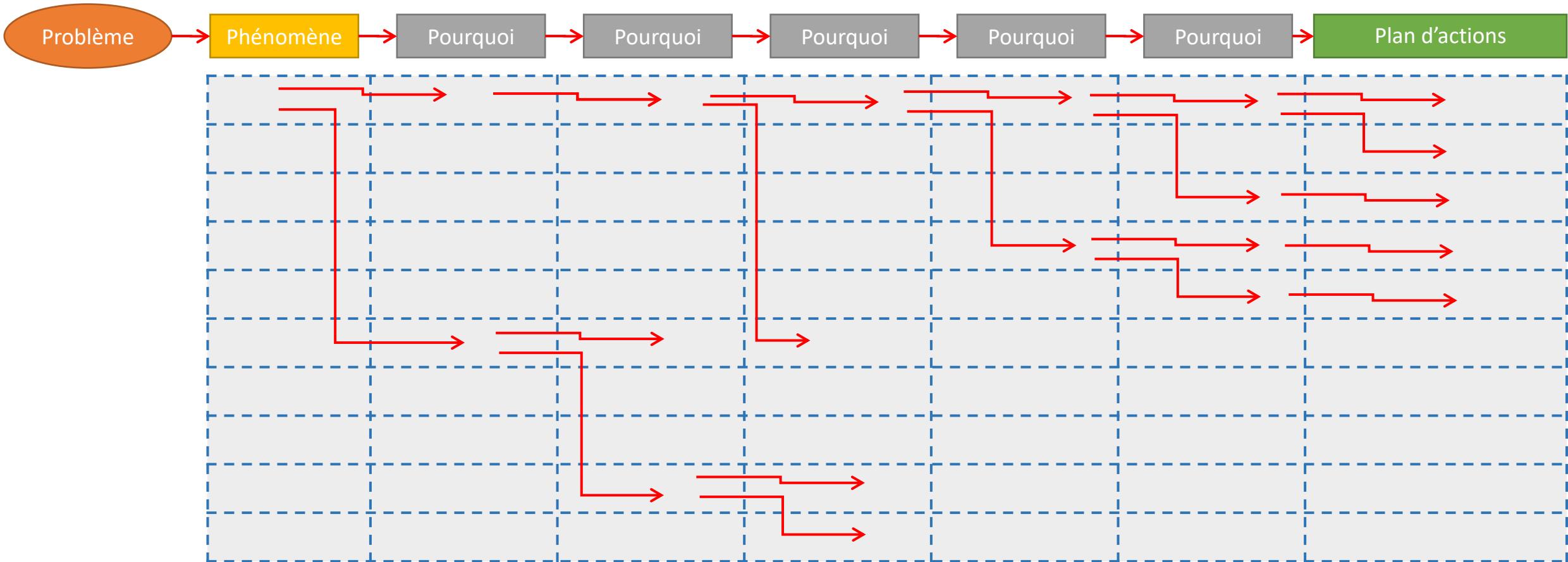
- **Les 5 pourquoi : L'obstacle n'est pas toujours celui que l'on croit : Concept des 5 pourquoi**
- Le « Washington Monument » s'érode et la firme responsable du ciment ne réussit pas à en trouver la cause (« root cause analysis »).
  - Pourquoi le bâtiment se désagrège-t-il ?
    - Parce que l'on y applique trop de produits chimiques
  - Pourquoi applique-t-on trop de produits chimiques ?
    - Pour nettoyer les crottes de pigeons !
  - Pourquoi y a-t-il autant de pigeons ?
    - Car ils mangent les insectes sur le bâtiment !
  - Pourquoi y a-t-il autant d'insectes ?
    - A cause de la lumière !
- **Solution : Réduire les horaires d'éclairages du Monument...**



# Les Méthodes Agiles

## Suivi & Reporting

- **Les 5 pourquoi : Exemple de représentation**



# Les Méthodes Agiles



## Suivi & Reporting

- **Les 5 pourquoi : Exemple d'exercice**
- Il y a souvent trop de recettes à réaliser à la fin de chaque sprint
  - Le product owner n'est pas assez présent avec l'équipe
  - Le daily-stand up n'est jamais achevé
- Il n'y a pas assez de développeurs/testeurs dans l'équipe
  - On dépend trop de développeurs externes à l'équipe
  - On travaille sur trop de projets



Identifier les  
5 pourquoi



Identifier les  
5 pourquoi

## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | <b>Le KanBan</b>                    |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Approche : Méthodes Agiles

## Les grands principes de la méthode KANBAN

- **La Méthode KANBAN**
  - Elle s'est développée au Japon après la Seconde Guerre Mondiale.
  - Elle a été élaborée par M. Ohno dans l'entreprise TOYOTA Motor Company
  - Elle a commencé à bien fonctionner dès 1958.
- 
- M. Ohno, a constaté que « les gens des usines ont toujours tendance à faire de la surproduction »
  - Il a alors recherché le moyen qui permette de produire :
    - Le produit demandé,
    - Au moment où il est demandé,
    - Dans la quantité demandée



*Taïchi Ohno*

# Approche : Méthodes Agiles

## Les grands principes de la méthode KANBAN

- **Qu'est-ce que Kanban ?**

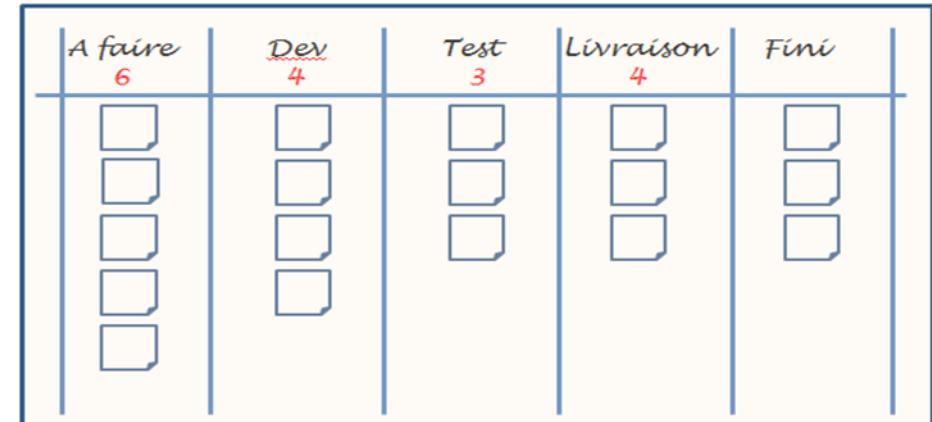
- Est un framework populaire pour implémenter le développement logiciel Agile .
- Repose sur un travail effectué en toute transparence
- Repose sur une communication en temps réel de la capacité.

- Les tâches sont représentées visuellement sur un tableau Kanban. Ainsi, les membres de l'équipe peuvent voir l'état de chaque tâche à tout moment.

- Le nom signifie « panneau » ou « étiquette », et désigne la fiche de commande fixée sur le conteneur destiné au fournisseur, qui déclenche la production et assure le suivi de sa progression.

- Cette méthode de gestion des stocks dite « à flux tiré » permet de minimiser les stocks, et donc diminuer les coûts et le gaspillage.

- Le but est de diminuer le nombre de kanbans en circulation pour diminuer l'en-cours et permettre de produire à la demande.
- Zéro déchet, zéro stock.
- Permet de diminuer l'en-cours, d'augmenter la concentration des équipes, donc la qualité du livrable, et également de livrer rapidement et plus régulièrement.

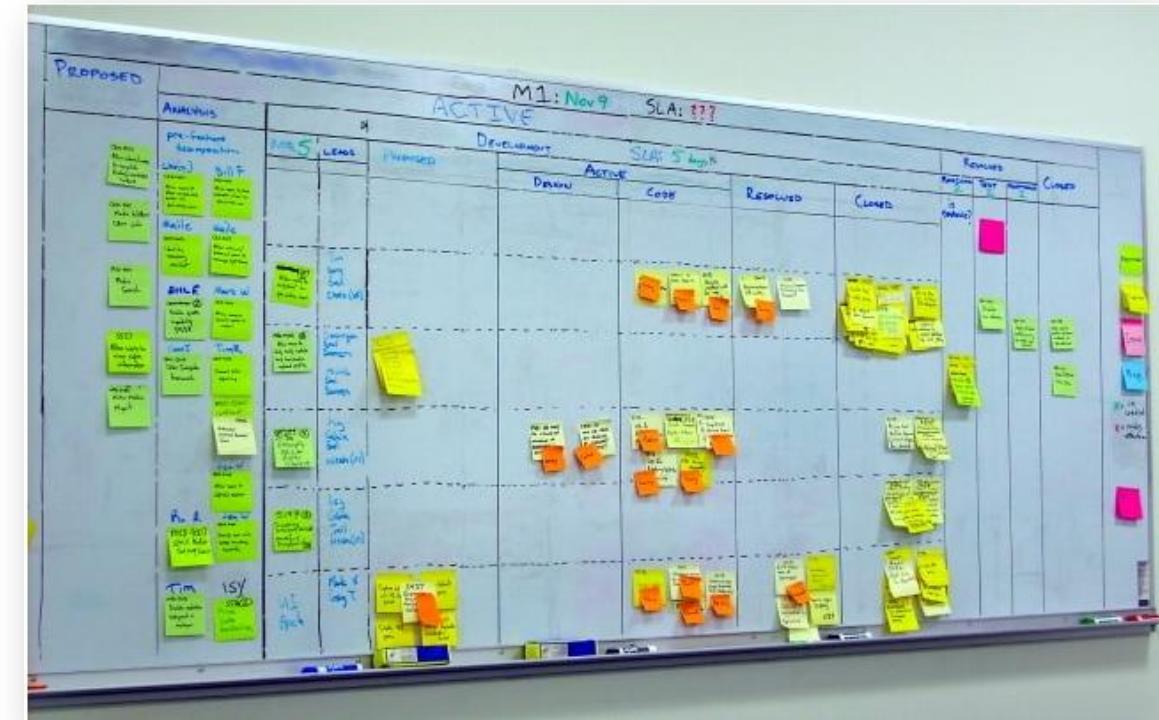


# Approche : Méthodes Agiles

## Les grands principes de la méthode KANBAN

- **En synthèse :**

- Visualiser et suivre le Workflow : Découper le travail, décrire les éléments sur des fiches (KANBAN) et les afficher dans un tableau comportant des colonnes représentant les différentes étapes du workflow.
- Limiter le TAF ou WIP (Work In Progress) : Des indicateurs doivent être mentionnés dans chaque colonne du workflow. Cela sera la limite max à ne pas dépasser.
- Mesurer le temps de cycle pour un élément donné : Temps moyen pour qu'un élément soit complètement traité, on cherchera constamment à optimiser ce temps de cycle.



# Les Méthodes Agiles



## Les grands principes de la méthode KANBAN

- **Pratiques kanban**

- 1 Visualisez le flux de travail:
  - Notre cerveau traite beaucoup mieux l'information visuelle que l'information invisible. Si vous parvenez à rendre visibles le travail, les flux de travail et les risques commerciaux, vous êtes en mesure de mieux les gérer et de le faire en collaboration et par consensus. C'est l'idée du Kanban en un mot. Apprenez donc à définir et à saisir les demandes de travail. Visualisez chaque demande avec une carte sur un tableau Kanban.
- 2 Limiter les travaux en cours:
  - Limitez le nombre de cartes ou de notes autocollantes sur le tableau Kanban et pour chaque colonne afin de vous assurer que l'équipe se concentre sur les résultats et ne se perd pas en travaillant sur trop de tâches à la fois.
- 3. Gérer le flux:
  - Le flux des work items à travers chaque état du flux de travail devrait être surveillé et rapporté. Contrôler la vitesse du mouvement et la fluidité. Un écoulement rapide et régulier est idéal.
- 4. Rendre les politiques explicites:
  - Tant que toutes les personnes concernées ne comprennent pas le processus de travail, il est difficile de discuter des problèmes à un niveau constructif. Assurez-vous d'expliquer les définitions (c.-à-d. quand une tâche est-elle considérée comme accomplie ?) et les règles. Cela permettra aux membres de l'équipe d'améliorer la qualité du travail et de faire des suggestions pour améliorer les processus.

# Les Méthodes Agiles



## Les grands principes de la méthode KANBAN

- **Pratiques kanban**

- 5. Boucles de rétroaction sur la mise en œuvre:
  - Les boucles de rétroaction régulières constituent un élément important du système Kanban pour comparer les résultats escomptés aux résultats réels et apporter des ajustements. Lorsqu'il est mis en œuvre au niveau de la prestation de services, le Kanban utilise quatre pratiques de rétroaction : la réunion de standup, l'examen de la prestation de services, l'examen des opérations et l'examen des risques.
- 6. Améliorez de manière collaborative, évoluez de manière expérimentale (Kaizen):
  - David Anderson encourage les entreprises à utiliser des modèles pour générer une compréhension commune du travail, du flux de travail et des processus. De plus, les modèles permettent de prédire l'effet d'un changement ou d'une intervention. L'observation empirique (résultat) peut ensuite être évaluée à l'aide de la prédition du modèle. Cette approche scientifique peut améliorer l'apprentissage et créer une culture de l'apprentissage dans les entreprises.

# Les Méthodes Agiles



## Les grands principes de la méthode KANBAN

- **Méthode Kanban vs Méthode Scrum**

| Points communs  | Différences  |
|---|--|
| <ul style="list-style-type: none"><li>• Les deux méthodes sont agiles.</li><li>• Le Kanban et la mélée suivent une logique d'attraction (les membres de l'équipe choisissent une tâche à partir d'un arriéré).</li><li>• Les équipes ont plus de responsabilité et de liberté.</li><li>• Les travaux en cours sont limités.</li><li>• Les deux méthodes visent à créer des processus transparents.</li><li>• Les résultats du projet sont obtenus par étapes.</li><li>• Les processus sont évalués régulièrement afin d'optimiser le flux de travail.</li></ul> | <ul style="list-style-type: none"><li>• Scrum est la méthode la plus complète.</li><li>• Kanban, son approche visuelle peut être facilement combinée avec d'autres méthodes.</li></ul> |

# Les Méthodes Agiles



## Les grands principes de la méthode KANBAN

- **Méthode Kanban vs Méthode Scrum**

| Kanban  | Scrum  |
|---|--|
| <ul style="list-style-type: none"><li>• quelques règles rigides</li><li>• aucun rôle prescrit</li><li>• des changements à tout moment</li><li>• Le travail est tiré à travers le système</li><li>• priorisation de l'arriéré facultatif</li><li>• itérations en option / livraison continue</li><li>• à partir d'une certaine taille de projet : risque de confusion/besoin de décomposition en unités séparées</li></ul> | <ul style="list-style-type: none"><li>• ensemble complexe de règles</li><li>• rôles fixes (Scrum master, propriétaire du produit et membre de l'équipe)</li><li>• pour les équipes de trois membres ou plus</li><li>• pas de changements à mi-parcours</li><li>• Le tableau de mélée est rempli de nouveau après chaque sprint.</li><li>• établir l'ordre de priorité de l'arriéré prescrit</li><li>• le travail se fait au sprint</li></ul> |

## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                                     |
|----|-----------------------------------|----|--|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation                     |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                          |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                                  |
| 04 | Approche Ingénierie des Exigences | 13 | <b>Approche sur le Extreme Programming</b> |
| 05 | Le LEAN                           | 14 | CMMi                                       |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                         |
| 07 | User Stories                      | 16 | SAFe                                       |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                                |
| 09 | Le Sprint                         |    |  |

# Méthodes de conception particulières en Agile



## Approche : EXtreme Programming

- **EXtreme Programming : XP**
  - Cette méthode a été créée par Kent Beck entre 1996 et 1999, lorsqu'il travaillait sur un projet pour Chrysler.
  - La méthodologie XP est une méthode de gestion de projet qui applique à l'extrême les principes du développement agile,
    - Se concentrer sur les besoins du clients
    - Mettre en place un développement itératif
    - Mettre en œuvre l'intégration continue
  - L'équipe projet et ses relations avec le client sont au coeur de XP.
- La méthode eXtreme Programming s'appuie sur :
  - une forte réactivité au changement des besoins du client ;
  - un travail d'équipe ;
  - la qualité du travail fourni ;
  - la qualité des tests effectués au plus tôt

# Méthodes de conception particulières en Agile



## Approche : EXtreme Programming

- **EXtreme Programming : 5 valeurs fondamentales**

- La Communication :
  - Chaque membre de l'équipe communique quotidiennement avec les acteurs clés et les parties prenantes du projet. C'est un moyen incontournable pour résoudre les problèmes.
- La Simplicité :
  - la façon la plus simple d'arriver au résultat est privilégiée. L'équipe projet fait ce qui est nécessaire et demandé, rien de plus. Une application simple sera plus facile à faire évoluer ensuite.
- Le Feedback :
  - le retour d'information entre l'équipe projet et les parties prenantes est essentiel.
  - Chaque étape du projet est envoyée aussi rapidement et souvent que possible aux parties prenantes afin qu'ils testent, donnent leurs avis et valident les étapes.
  - Chaque demande de modification est prise en compte immédiatement.
- Le Respect :
  - le respect de chaque membre de l'équipe et de son travail sont primordiaux. Le management, l'équipe projet et le client se respectent mutuellement.
- Le Courage :
  - Il faut du courage pour effectuer certains changements comme essayer une nouvelle technique, recommencer une itération non validée ou revoir l'organisation du projet.
  - Le courage permet de sortir d'une situation inadaptée.

# Méthodes de conception particulières en Agile



## Approche : EXtreme Programming

- **Les cinq valeurs de XP se déclinent en treize pratiques qui se renforcent mutuellement :**

### 1.Client sur site :

- le client doit être représenté sur place pendant toute la durée du projet. Ce représentant doit avoir une vision globale du résultat à obtenir et être disponible pour répondre aux questions de l'équipe.

### 2.Jeu du planning (ou planning poker):

- le planning est réalisé en collaboration avec le client. Ce dernier crée des scénarios pour les fonctionnalités qu'il souhaite obtenir. L'équipe évalue le temps nécessaire pour les mettre en œuvre. Le client sélectionne ensuite les scénarios en fonction des priorités et du temps disponible.

### 3.Intégration continue :

- lorsqu'une tâche est terminée, elle est tout de suite intégrée dans le produit complet. Cela permet d'éviter la surcharge de travail due à l'intégration de tous les éléments avant la livraison. Les tests facilitent cette intégration : quand tous les tests sont positifs, l'itération est terminée.

### 4.Petites livraisons:

- les livraisons doivent être les plus fréquentes possible afin que le client donne son avis et que les modifications soient rapidement prises en compte par l'équipe.

# Méthodes de conception particulières en Agile



## Approche : EXtreme Programming

### 5. Rythme soutenable :

- aucune heure supplémentaire n'est tolérée. S'il y en a, alors le planning doit être revu. Un collaborateur fatigué travaille mal et fait plus d'erreurs.

### 6. Tests fonctionnels :

- à partir des scénarios définis par le client, l'équipe crée des procédures de test qui permettent de vérifier l'avancement du développement. Lorsque tous les tests fonctionnels passent, l'itération est terminée.

### 7. Tests unitaires :

- pour chaque fonctionnalité, un test est écrit afin de vérifier qu'elle fonctionnera comme prévu. Ce test sera conservé jusqu'à la fin du projet, tant que la fonctionnalité est requise. À chaque modification du code, tous les tests sont lancés afin d'identifier immédiatement s'il y a un problème de fonctionnement.

### 8. Conception simple :

- on va droit à l'essentiel en se focalisant uniquement sur les besoins actuels du clients. Plus l'application est simple, plus il sera facile de la faire évoluer lors des prochaines itérations.

# Méthodes de conception particulières en Agile



## Approche : EXtreme Programming

### 9. Utilisation de métaphores:

- les équipes XP utilisent des métaphores pour décrire le système et son fonctionnement afin de clarifier les fonctionnalités à atteindre. Tout le monde parle le même langage.

### 10. Refactoring (ou remaniement du projet):

- le projet est amélioré régulièrement. Le but étant d'avoir de bonnes bases et de meilleures conditions de travail pour l'équipe.

### 11. Appropriation collective du projet :

- la responsabilité du projet est collective. Chaque membre de l'équipe peut modifier toutes les portions du projet, même celles sur lesquelles il n'a pas travaillé. L'objectif est d'être efficace et rapide.

### 12. Standards de langage :

- puisque tout le monde travaille ensemble sur le projet, il est essentiel de faciliter le travail de chacun en utilisant les mêmes termes, le même style et des règles de communication claires.

### 13. Travail en binôme :

- les collaborateurs travaillent en binôme. Le pilote et le copilote changent régulièrement afin d'améliorer la communication et la connaissance collective du projet.

## Approche : EXtreme Programming

- **Cycle standard de l'Extreme Programming**

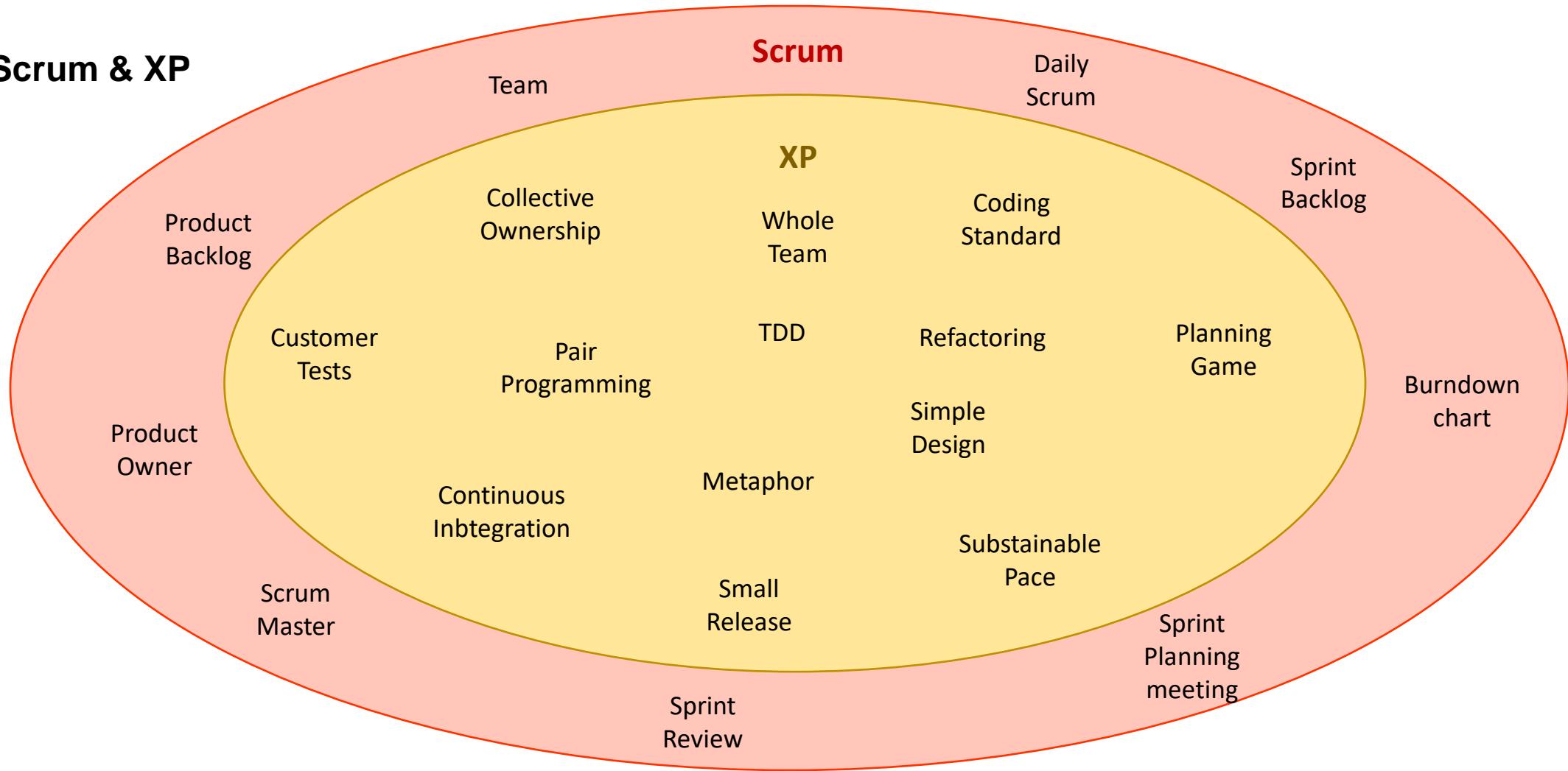
- Le client écrit ses besoins sous forme de scénarios
- Les développeurs évaluent le coût de chaque scénario, en collaboration avec les parties prenantes
- les parties prenantes choisissent les scénarios à intégrer à la prochaine livraison
- Chaque développeur prend la responsabilité d'une tache pour la réalisation d'un scénario
- Le développeur choisi un partenaire
- Le binôme écrit les tests unitaires correspondant au scenario à implémenter
- Le binôme prépare l'implémentation en réorganisant le code existant, puis il procède à l'implémentation
- Le binôme intègre ses développements à la version d'intégration

Les projets gérés par la méthode eXtreme Programming reposent sur des cycles de développement (itérations) courts et rapides qui sont réalisés collectivement par l'équipe projet et les parties prenantes dont l'implication est constante.

# Les Méthodes Agiles

## Approche : EXtreme Programming

- **Scrum & XP**



## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | <b>CMMi</b>                         |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Les Méthodes Agiles



## CMMI : Capability Maturity Model integration



- **Un peu d'histoire :**
  - C'est une demande du DOD (US Department of Defense) pour
    - Évaluer la capacité d'une organisation à faire un logiciel de qualité
    - Définir des niveaux
    - Proposer un cheminement d'amélioration
    - Promouvoir les meilleures pratiques
    - Rester indépendant des processus / cycles de vie
    - Se positionner comme "ensemble d'exigences" sur le processus
  - Le résultat
    - Le CMM® (Capability Maturity Model), version 1.1 en 1993
      - Développé par le SEI (Software Engineering Institute)
      - Une méthode d'évaluation - CBA-IPI / SCE
      - Une méthode de progrès (IDEAL process of the SEI)
    - En 2001, le SEI a proposé une nouvelle version de son modèle
      - CMMI (Capability Maturity Model Integration)

# Les Méthodes Agiles



## CMMI : Capability Maturity Model integration

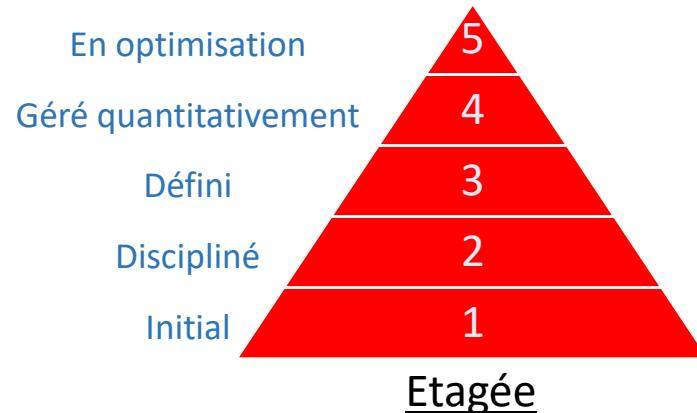


- **Le modèle**

- Regroupe 22 domaines de processus
- Représente plus de 600 bonnes pratiques
- Un domaine de processus = un ensemble de pratiques liées dont l'objectif est de satisfaire un ensemble cohérent d'objectifs d'amélioration

- **Les deux représentations**

- Etagée : mesurée par niveau de maturité sur un ensemble donné de domaines de processus
- Continue : mesurée par niveau de capacité par domaine de processus



Source : SEI, 2007-03

|   |     |     |     |     |  |
|---|-----|-----|-----|-----|--|
| 3 |     |     |     |     |  |
| 2 |     |     |     |     |  |
| 1 |     |     |     |     |  |
| 0 |     |     |     |     |  |
|   | PA1 | PA2 | PA3 | PAn |  |

Continue

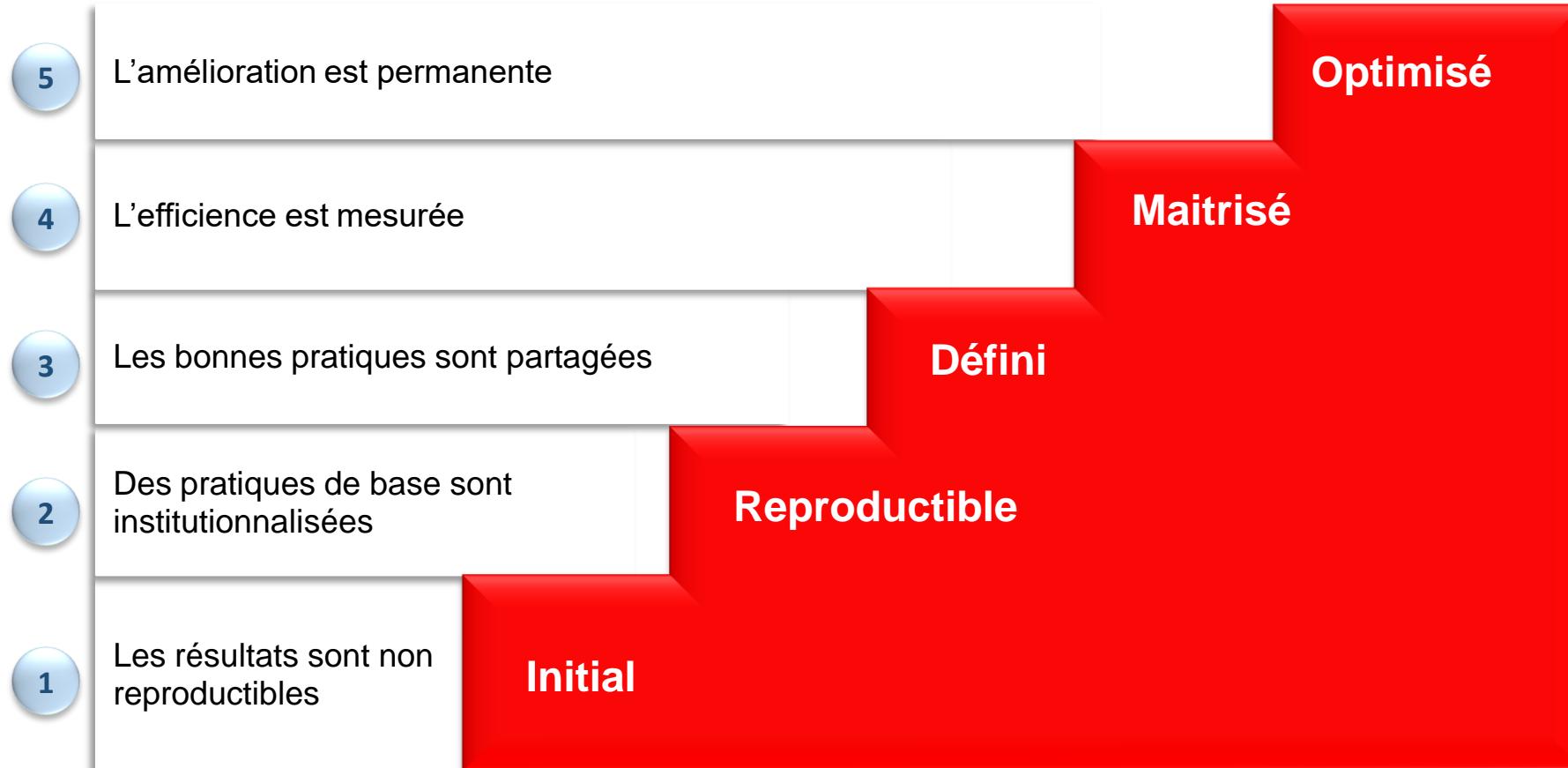
# Les Méthodes Agiles



CMMI : Capability Maturity Model integration



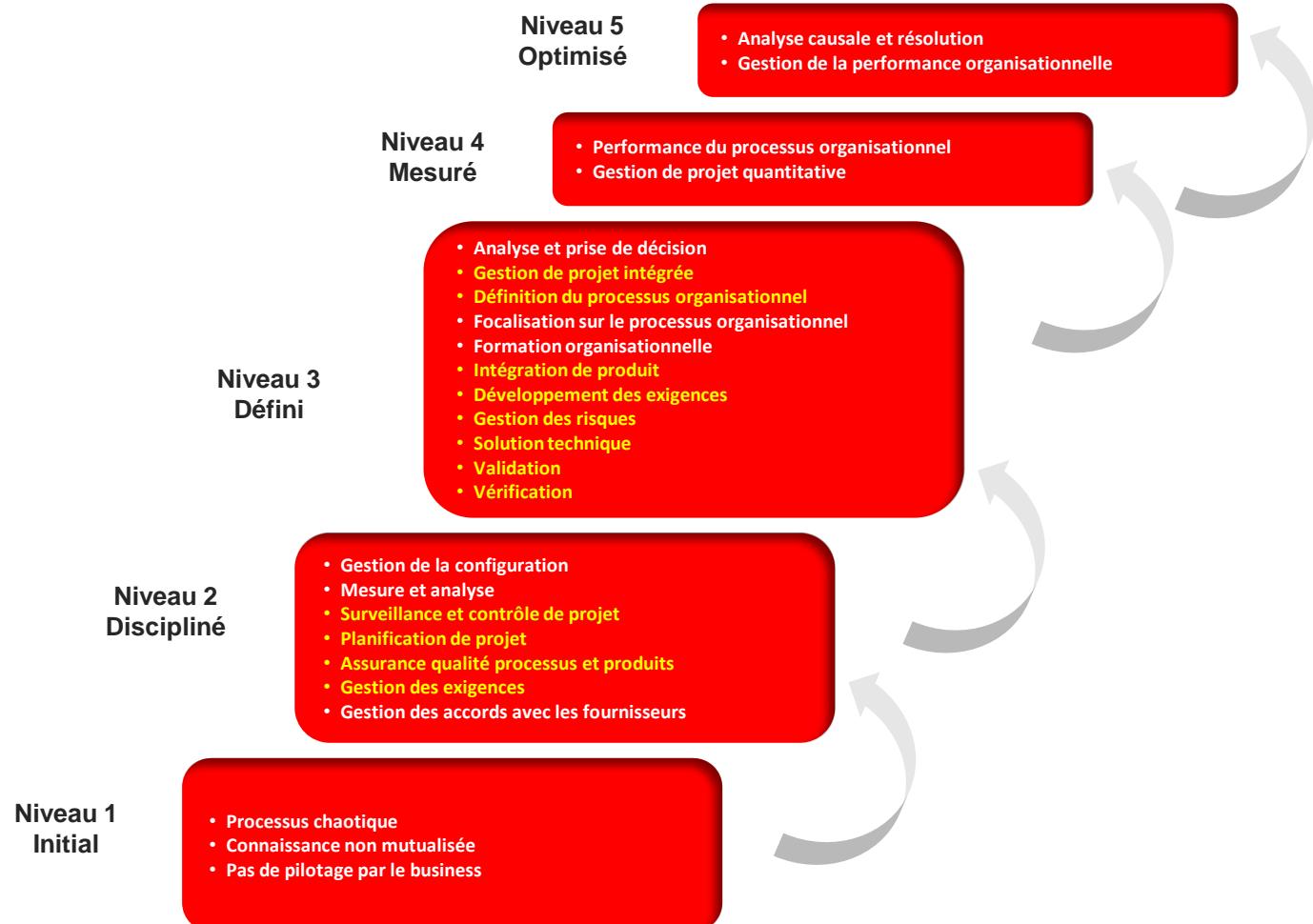
- **Les niveaux de maturité:**



# Les Méthodes Agiles

## CMMI : Capability Maturity Model integration

- Le modèle



## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | <b>Approche TDD &amp; BDD</b>       |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Méthodes de conception particulières en Agile



## Behaviour Driven Development (BDD)

- Le Behavior-Driven Development, ou BDD,
  - Est une méthode de développement logiciel dérivée du Test-Driven Development – TDD.
  - Elle incite à la collaboration des différentes parties prenantes au projet logiciel, équipes de développement, qualification et management en instaurant que le comportement d'une fonctionnalité sera décrit par des phrases basées sur un canevas composé de mots clés du langage courant.
- La communication est ainsi facilitée entre les équipes et évite des incompréhensions inhérentes à des parties d'environnements différents.
- L'accent est mis sur les processus métiers auxquels le logiciel devra apporter des solutions.

# Méthodes de conception particulières en Agile



## Behaviour Driven Development (BDD)

- Le BDD est basé sur des mots que l'on utilise souvent dans la vraie vie comme **Étant donné, Quand, Alors, et Et**, qui vont décrire le comportement de votre fonctionnalité.
  - Étant donné : Given
  - Quand : When
  - Alors : Then
  - Et : And

# Méthodes de conception particulières en Agile



## Behaviour Driven Development (BDD)

- Vous décrivez une situation théorique et ce qui doit se passer en utilisant ces mots.
  - Par exemple :
    - **Étant donné** qu'un utilisateur laisse un commentaire
    - **Quand** le commentaire dépasse 1000 caractères
    - **Alors** le commentaire ne doit pas être sauvegardé
    - **Et** l'utilisateur doit voir un message d'erreur
- Vous pouvez vous demander, mais où est le code ? Il n'y en a pas pour l'instant !

Certains frameworks de test vous permettent même d'utiliser ces mots-clés ("given", "when", "then", "and" en anglais) dans votre tests, et les rendent très lisibles.

# Méthodes de conception particulières en Agile



## Behaviour Driven Development (BDD)

- Les outils et langage

- Des outils comme [Cucumber](#) (open source) ou Hiptest (éditeur Français) permettent de mettre en œuvre cette approche BDD / Agile.
- Le langage utilisé s'appelle le Gherkin :

```
Scenario Outline: Add two numbers.  
Given the input "<input>"  
When the calculator is run  
Then the output should be <output>"  
Examples  
| input      | output |  
| 2+2        | 4      |  
| 98+1       | 99     |  
| 255+390    | 645    |
```

# Méthodes de conception particulières en Agile



## Test Driven Development (TDD)

- Approche :
  - Le TDD est l'étape suivante naturelle pour fermer la boucle de BDD que vous avez commencé dans le dernier chapitre.
  - Voici comment les deux fonctionnent ensemble. Vous allez :
    - Identifier des **comportements** ("behaviors" en anglais) avec BDD.
    - Écrire des **tests** pour ces comportements avec TDD.
- Démarche :
  - En TDD, l'objectif est d'écrire un programme le plus basique possible pour faire passer les tests.
  - Il y a trois règles à respecter, selon Robert Martin (un leader dans le monde de TDD) :
    - Vous devez écrire un test qui échoue avant d'écrire votre code lui-même.
    - Vous ne devez pas écrire un test plus compliqué que nécessaire.
    - Vous ne devez pas écrire plus de code que nécessaire, juste assez pour faire passer le test qui échoue.

# Méthodes de conception particulières en Agile



## Test Driven Development (TDD)

- Red – Green – Refactor :
  - Ces 3 règles fonctionnent ensemble et font partie d'un processus qui s'appelle le cycle **rouge-vert-refactor** ("red-green-refactor" en anglais).
    - **Rouge** : Vous écrivez un test simple qui échouera avant même d'avoir du code qui l'accompagne. Le test échoue, évidemment, sans le code. Du coup, le test est rouge.
    - **Vert** : Vous écrivez le code le plus simple possible pour faire passer le test, même si le code est un peu ridicule ou simplifié. Le test réussit, il est donc vert.
    - **Refactor** : le code que vous avez écrit pour faire passer le test est peut être illogique ou trop simple. Il faut faire un refactor de ce code maintenant si possible.
- Exemple TDD sans code
  - Soit une fonction qui doit effectuer une division du nombre A par le nombre B.
  - Voila ce que je veux tester :
    - **Quel est le résultat si A = 10 et B = 0**
    - **Quel est le résultat si A = 0 et B = 10**
    - **Quel est le résultat si A = 100 et B = 10**

## Questions Ouvertes

**Des questions ?**



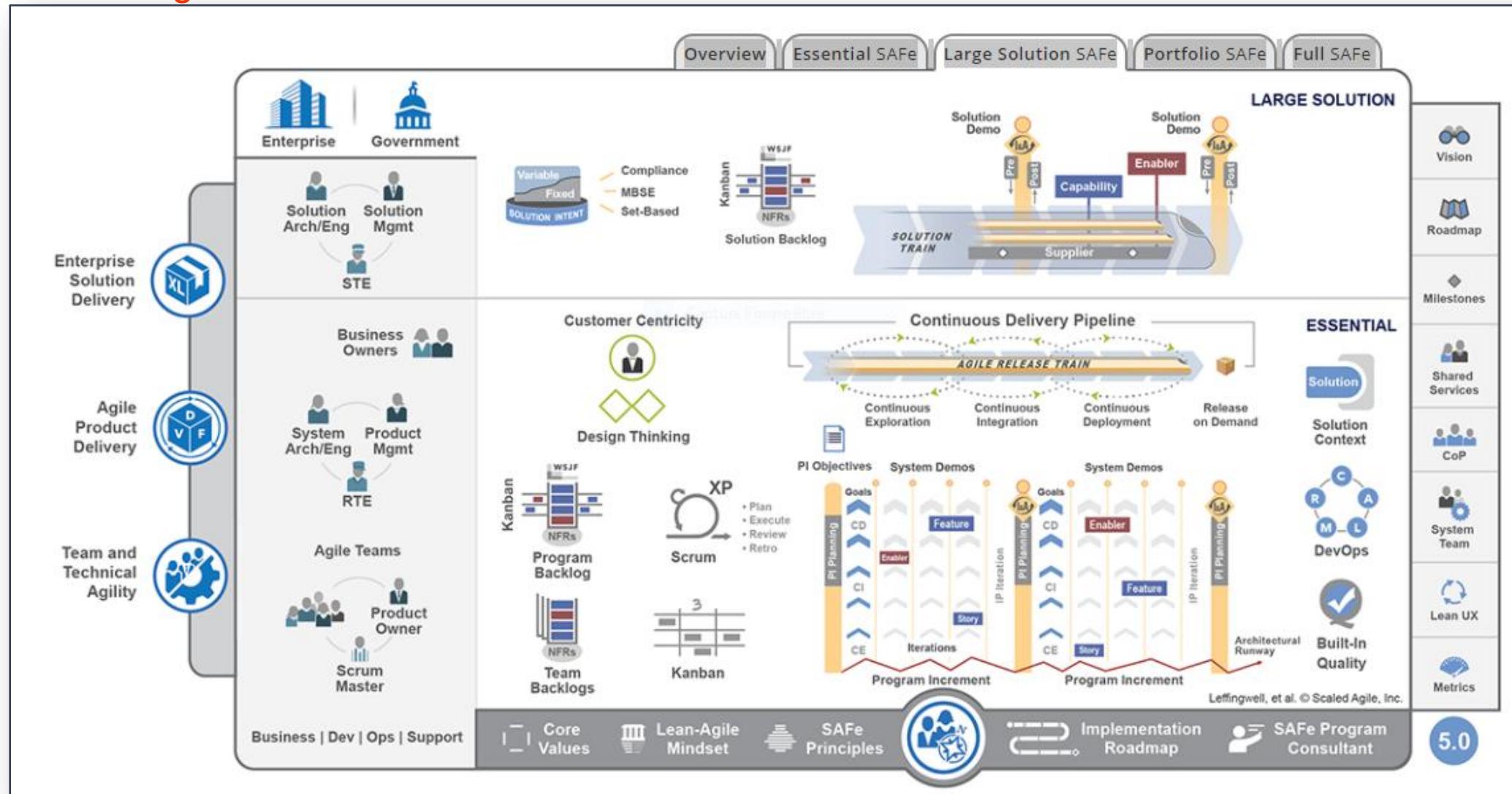
# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Les Méthodes Agiles

# SAFe : Scaled Agile Framework 5.0



# Les Méthodes Agiles



## SAFe : Scaled Agile Framework

- Dean Leffingwell et Drew Jemilo ont lancé SAFe en 2011 pour aider les organisations à concevoir des systèmes et des logiciels plus performants, qui répondent mieux à l'évolution des besoins des clients
  - Mettre en œuvre une agilité à l'échelle au sein des grandes DSI
- Le Scaled Agile Framework® (SAFe®)
  - SAFe est un ensemble de modèles d'organisation et de workflow pour l'implémentation des pratiques Agile à l'échelle de l'entreprise
  - SAFe est un framework lean-agile pour faire travailler plusieurs équipes Scrum ensemble au sein d'une même structure
  - SAFe fait la promotion de l'alignement, la collaboration et la livraison dans un grand nombre d'équipes Agile
- On distingue 3 niveaux d'évolutivité
  - Essential SAFe
  - Large Solution SAFe
  - Portfolio SAFe
  - Full SAFe

# Les Méthodes Agiles



## SAFe : Scaled Agile Framework

- Les valeurs fondamentales SAFe

- Alignement
- Qualité intégrée
- Transparence
- Exécution du programme
- Direction

# Les Méthodes Agiles



## SAFe : Scaled Agile Framework

- Les Principes SAFe
  - Principe n° 1 : Adopter une vision économique
  - Principe n° 2 : Appliquer une pensée systémique
  - Principe n° 3 : Supposer la variabilité, préserver les options
  - Principe n° 4 : Se développer de manière incrémentielle à l'aide de cycles d'apprentissage rapides et intégrés
  - Principe n° 5 : Baser les étapes importantes sur une évaluation objective des systèmes de travail
  - Principe n° 6 : Visualiser et limiter le travail en cours (Work in Process, WIP), réduire la taille des lots et gérer la longueur des files d'attente
  - Principe n° 7 : Appliquer la cadence, synchroniser avec la planification transverse
  - Principe n° 8 : Susciter la motivation intrinsèque des travailleurs du savoir
  - Principe n° 9 : Décentraliser la prise de décision

# Les Méthodes Agiles



## SAFe : Scaled Agile Framework

- Le Fonctionnement : Feuille de route pour l'implémentation de SAFe, par étapes
  1. Atteindre le point de bascule
  2. Former des agents du changement Lean/Agile
  3. Former des directeurs, des responsables et des leaders
  4. Créer un centre d'excellence Lean/Agile
  5. Identifier des flux de valeur et des Agile Release Trains (ART)
  6. Élaborer le plan d'implémentation
  7. Préparer le lancement de l'ART
  8. Former les équipes et lancer l'ART
  9. Préparer le lancement de l'ART
  10. Lancer d'autres ART et flux de valeur
  11. Étendre au portefeuille
  12. Maintenir et améliorer

## Questions Ouvertes

**Des questions ?**



# Sommaire



| N° | Thèmes                            | N° | Thèmes                              |
|----|-----------------------------------|----|-------------------------------------|
| 01 | Accueil des Stagiaires            | 10 | Pratiques d'estimation              |
| 02 | Cycle de vie Projet               | 11 | Suivi & Reporting                   |
| 03 | Approche Méthode AGILE            | 12 | Le KanBan                           |
| 04 | Approche Ingénierie des Exigences | 13 | Approche sur le Extreme Programming |
| 05 | Le LEAN                           | 14 | CMMi                                |
| 06 | La méthode SCRUM                  | 15 | Approche TDD & BDD                  |
| 07 | User Stories                      | 16 | SAFe                                |
| 08 | Le Product Backlog                | 17 | Quizz Scrum                         |
| 09 | Le Sprint                         |    |                                     |

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 01 : En scrum, qui valide les user-stories pour qu'elles passent de « test » à « done » ?
  - Le scrum master
  - Le product owner
  - L'équipe de développement
  - Toute l'équipe

Bien que cela se répand beaucoup car paraît plus logique pour ceux qui viennent des méthodes traditionnelles, ce n'est pas le product owner qui teste (dans le sens vérifier le travail de l'équipe de développement) les user-stories avant de les déclarer « done ». L'équipe de développement est 100% responsable de fournir des user-stories « done » potentiellement livrable.

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 02 : L'équipe de développement peut-être composée :

- ✓ • De (compétences) développeurs front et de développeurs back
- ✓ • De (compétences) développeurs, d'un business analyst et d'un testeur
- ✓ • De (compétences) de testeur et de développeurs
- ✓ • De (compétences) développeurs, d'un PMO et d'un testeur

Toutes les réponses sont bonnes. L'équipe de développement ne veut pas dire équipes de « développeurs » mais équipe de réalisation. Tous ceux qui ne sont pas « product owner » ou « scrum master » et qui participe à la réalisation du produit font parti de l'équipe de développement. Donc même si cela surprend, le Business Analyst ou le PMO si nécessaire (bien que ce rôle est peu aimé) font parti de l'équipe de développement. L'équipe de développement ne reconnaît aucun titre individuel

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 03 : Le product backlog refinement est :



- Le nouveau nom de la backlog grooming
- Une cérémonie scrum
- La pratique d'affinage de user-stories et d'estimation en poker planning
- Ce n'est pas officiel en scrum

Seule la première réponse est vraie. Le terme de grooming a une connotation « pédophile » et a donc été remplacé par « product backlog refinement ».

Le product backlog refinement n'est pas une cérémonie bien que ce soit souvent réalisé de cette façon. C'est à l'équipe de définir comment fonctionnera l'affinage des « items » du product backlog ; cela n'est pas forcément sous forme de cérémonies

Si le product backlog refinement est un moment d'affinage, cela concerne des « items ». Bien que la user-story soit une pratique ultra populaire, le scrum ne parle que d'items. C'est l'équipe qui décide de ce qui est un « item ». De plus l'estimation en poker planning est une pratique qui ne vient pas du scrum. Cette réponse était tendancieuse mais ne peut être cochée car ce n'est pas comme cela qu'elle est définie.

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 04 : Quels artefacts existent en scrum ?

- ✓ • La definition of done
- La definition of ready
- ✓ • Le sprint backlog
- L'increment backlog

Le definition of ready est une pratique répandue mais non reconnue par le scrum guide. Je ne connais pas les raisons de cette absence mais cette pratique amène souvent à rendre le scrum « non agile ».

L'increment backlog n'existe pas ; il ne faut pas confondre avec l'artefact « increment ». L'increment est l'ensemble des éléments en « done » du sprint ajouté à l'incrément du sprint précédent. Il n'y a aucune notion de backlog dans cet artefact.

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 05 : Le product owner est responsable :

- De valider que les items sont bien « done »
- ✓ • De la gestion du product backlog
- ✓ • Que l'équipe de développement comprenne bien les items
- De rendre transparent le sprint backlog

Le product owner est responsable du product backlog et de sa compréhension. En revanche, il n'a aucune responsabilité sur le sprint backlog ; ce dernier est 100% sous la responsabilité de l'équipe de développement. Le product owner n'est pas responsable des développements. Seule l'équipe de développement doit mettre en place des exigences de qualités et vérifier par elle même que ces critères sont respectés à 100% avant de mettre l'item en « done ».

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 06 : Le scrum master doit :

- Avoir des connaissances avancées sur la gestion d'un product backlog
- Faciliter les cérémonies scrum
- ✓ • Travailler avec les autres scrum master de l'entreprise
- ✓ • Coacher l'équipe de développement si nécessaire

Si vous avez coché la première case également, il est vrai que c'est tendancieux (mais c'est faux). Le scrum guide dit :

« S'assurer que le Product Owner sait comment organiser le Backlog produit pour maximiser la valeur ; »

Il dit qu'il doit s'assurer que le product owner sache organiser le backlog mais ne dit pas que c'est lui qui doit lui apprendre. Donc mot pour mot, rien n'interdit au scrum master de faire appel à un autre product owner ou à un coach agile pour aider sur cette partie.

Légende urbaine, le scrum master n'a pas à faciliter les cérémonies. Il ne le fait que si l'équipe fait appel à lui ou que cela est nécessaire.

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 07 : Quelles sont les phrases qui sont vraies autour du sprint review

- ✓ • Le product owner y invite l'équipe scrum et les principales parties prenantes
- ✓ • L'équipe discute des problèmes rencontrés et comment ils ont été résolus
- ✓ • L'équipe y démontre le travail « fini »
- ✓ • La revue des délais et des budgets doit être fait

Toutes les réponses sont bonnes, elles font parties de la liste de la sprint review proposée par le scrum guide.

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 08 : Le scrum :

- Est un cadre de travail
- Est une méthode agile
- Ne s'occupe pas de la maintenance applicative
- Est un cadre léger

Scrum est un cadre de travail léger pour le développement, la livraison et la maintenance de produits complexes. Il n'est en aucun cas une méthode bien que c'est souvent comme cela qu'on le qualifie.

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 09 : Le scrum permet dans des cas exceptionnels d'annuler un sprint. Quelles phrases sont vraies ?

- ✓ • L'équipe de développement peut demander au product owner d'annuler le sprint
- ✓ • Seul le product owner est autorisé à annuler un sprint
- Un sprint recommence la semaine suivante pour ne pas perturber les dates des cérémonies
- Seul le scrum master est autorisé à annuler un sprint

En scrum, seul le product owner est autorisé à prendre la décision d'annuler un sprint (soit d'y mettre fin). Cependant l'équipe de développement peut parfaitement influencer pour aller vers cette annulation. En revanche après un travail de fermeture de ce sprint annulé, l'équipe recommence directement un nouveau sprint.

# Les Méthodes Agiles



## Quizz Scrum

- Réponse 10 : Quelle(s) affirmation(s) sont vraies concernant la sprint planning ?

- Tous les items choisi en sprint planning ne sont pas forcément finalisés.
- Seule l'équipe de développement valide sa capacité de faire (le nombre d'items du sprint)
- Un sprint planning peut durer jusqu'à 8h
- L'équipe de développement doit définir comment elle va réaliser les items

Toutes les réponses sont bonnes. Attention ! En aucun cas, les items doivent être 100% ready pour pouvoir être dans le « todo ». L'équipe de développement peut demander de nouveaux items nécessaires à l'atteinte de l'objectif du sprint ou accepter des items non finalisées qu'elle est sûre de pouvoir faire si ils sont finalisés en cours de sprint par le product owner. Le fait de dire que tous les « items » sont « ready » pour pouvoir être pris en sprint rend votre scrum non agile.

Si un feedback très important ressort de la sprint review, le product owner pourra mettre cet item dès le prochain sprint. Il n'aura probablement pas eu le temps de le finaliser. Il serait bien triste de devoir attendre un sprint supplémentaire pour faire une amélioration ayant beaucoup de valeurs alors que c'est une attente forte du client.

## Questions Ouvertes

**Des questions ?**



Merci Pour votre attention

