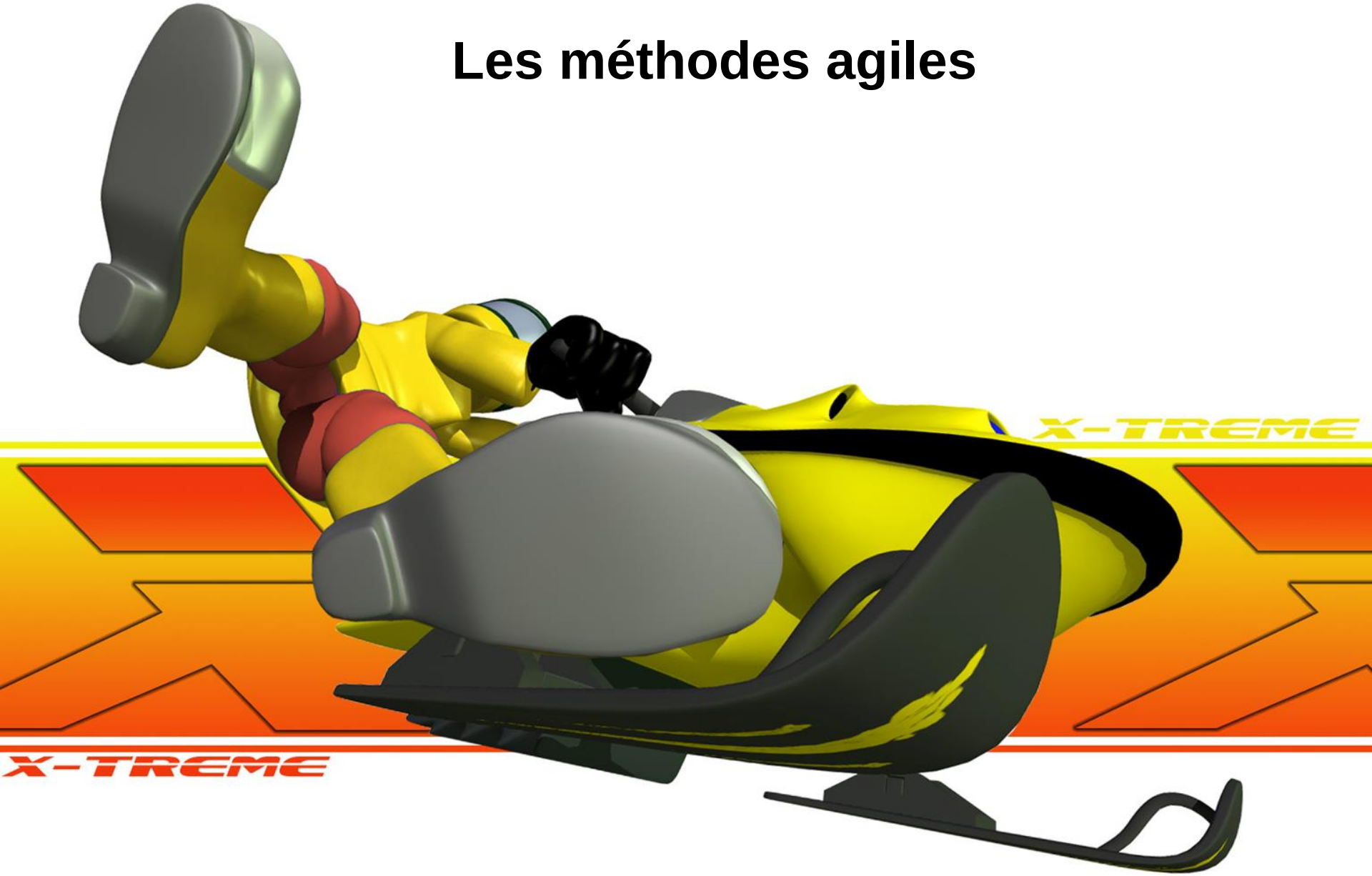
Les méthodes agiles



Présentation

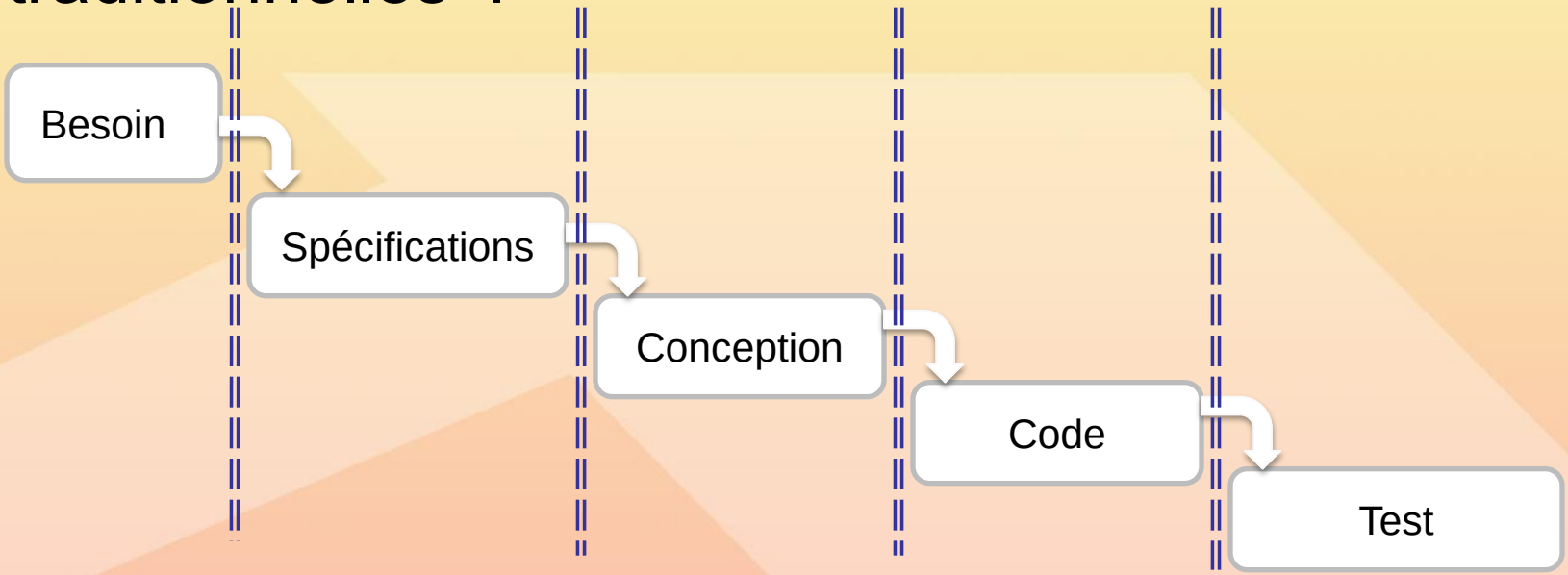
- Approche réactive et itérative d’organisation de travail.
- Focalisée sur la fonctionnalité et satisfaction client.
- Construit en adéquation avec les capacités et limites humaines.

Définition

- Une méthode agile est une approche **itérative** et **incrémentale**, qui est menée dans un **esprit collaboratif** avec juste ce qu'il faut de formalisme…

Pourquoi Agile ?

- En réaction des problèmes avec des approches ‘traditionnelles’ :

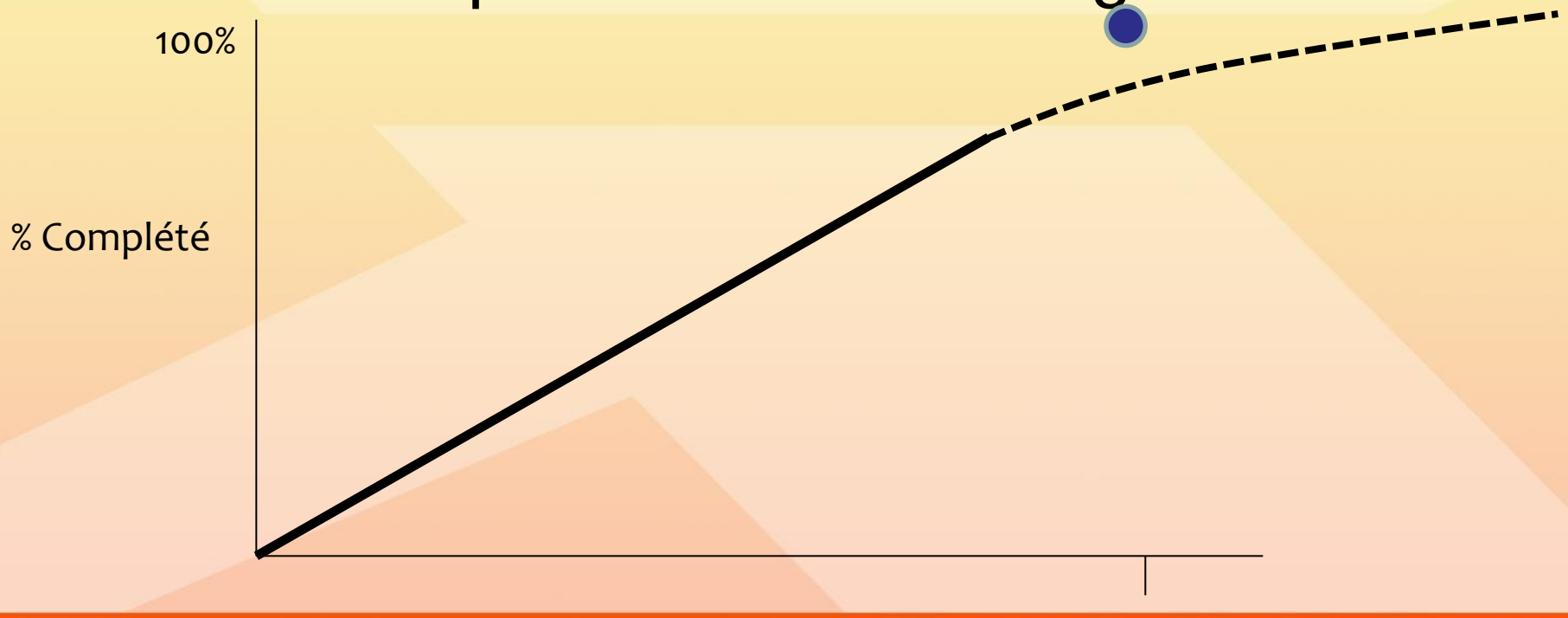


Les constats

- Les meilleures idées ne viennent pas forcément au début du projet :
  - Il est plus facile de construire par étape que tout imaginer dès le début.
- Les besoins peuvent évoluer pendant le projet.
- Le formalisme n’est pas naturel.
- Chiffrages et Reste à Faire sont difficiles à évaluer.

Réalité

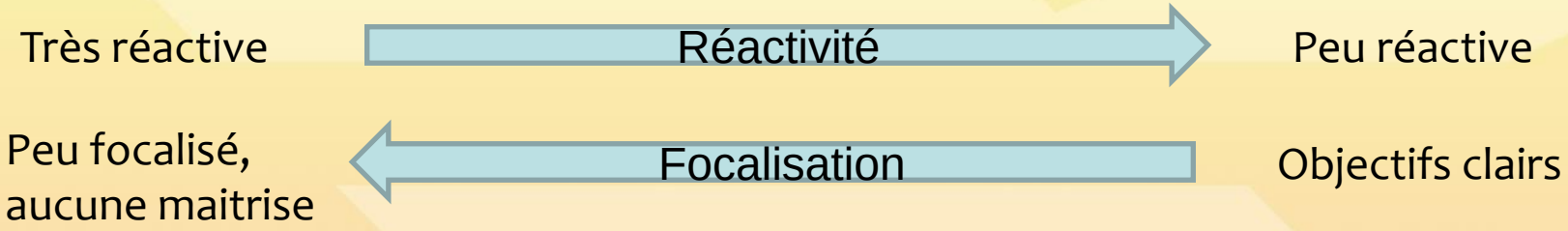
- On ne sait pas estimer la charge restante.



Réalité (2)

- Les méthodes prédictives fonctionnent bien, à condition d'avoir :
  - Stabilité et prévisibilité.
  - Communication et compréhension parfaite.
  - Choix parfaits dès le départ.

Agile : le juste milieu



Absence de méthode

Méthodes prédictives

Agile : une catégorie de méthodes

- ‘Agile’ regroupe plusieurs méthodologies :
  - Scrum.
  - Extreme Programming (XP).
  - …
- Notion officialisée en 2001 avec le Manifeste Agile.

Les 4 principes du manifeste Agile

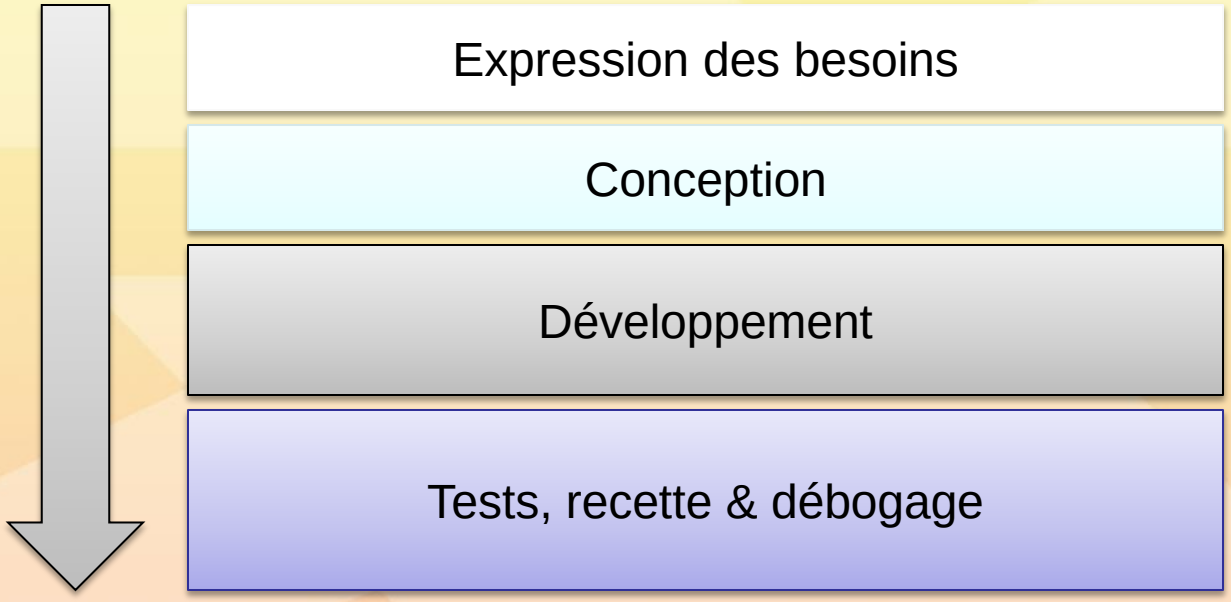
Personnes et **interactions**	Plutôt que	Processus et outils
Un produit opérationnel	Plutôt que	Documentation exhaustive
Collaboration avec le client	Plutôt que	Négociation d'un contrat
Adaptation au changement	Plutôt que	Suivi d'un plan

Le manifeste Agile (2)

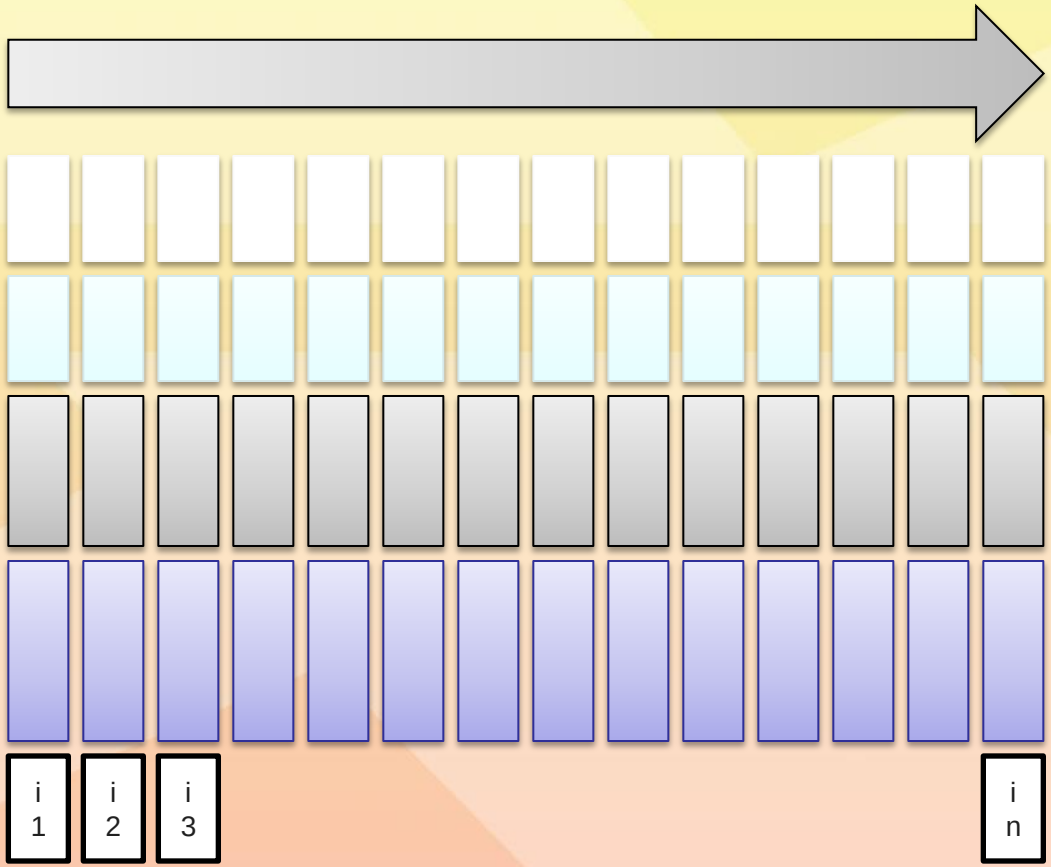
- Libérer le génie humain pour l'auto-organisation dans un contexte qu'il peut maîtriser :
  - La taille de l'équipe est limitée.
  - Le domaine du problème est limité.
  - Avancement par itérations.

Le manifeste Agile (3)

Avant…



Le manifeste Agile (4)

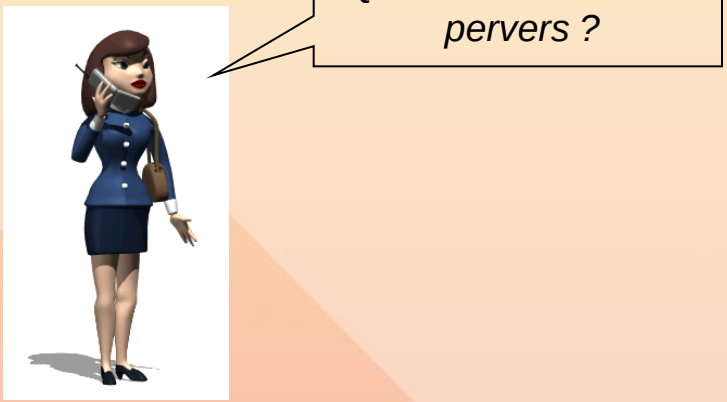


Le manifeste Agile (5)

- Toujours focalisé sur le produit final :
  - Une vision commune pour l'équipe : la satisfaction du client.
  - Découper le produit autrement : par fonctionnalité.
  - Organiser en cycles de développement réduits : itérations.

Le manifeste Agile (6)

- Collaboration avec le client sans contrat :
  - Instaurer la confiance autrement.
  - Eviter les effets pervers d'un contrat.

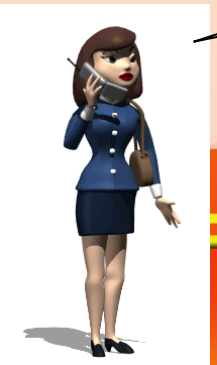


Le manifeste Agile (7)

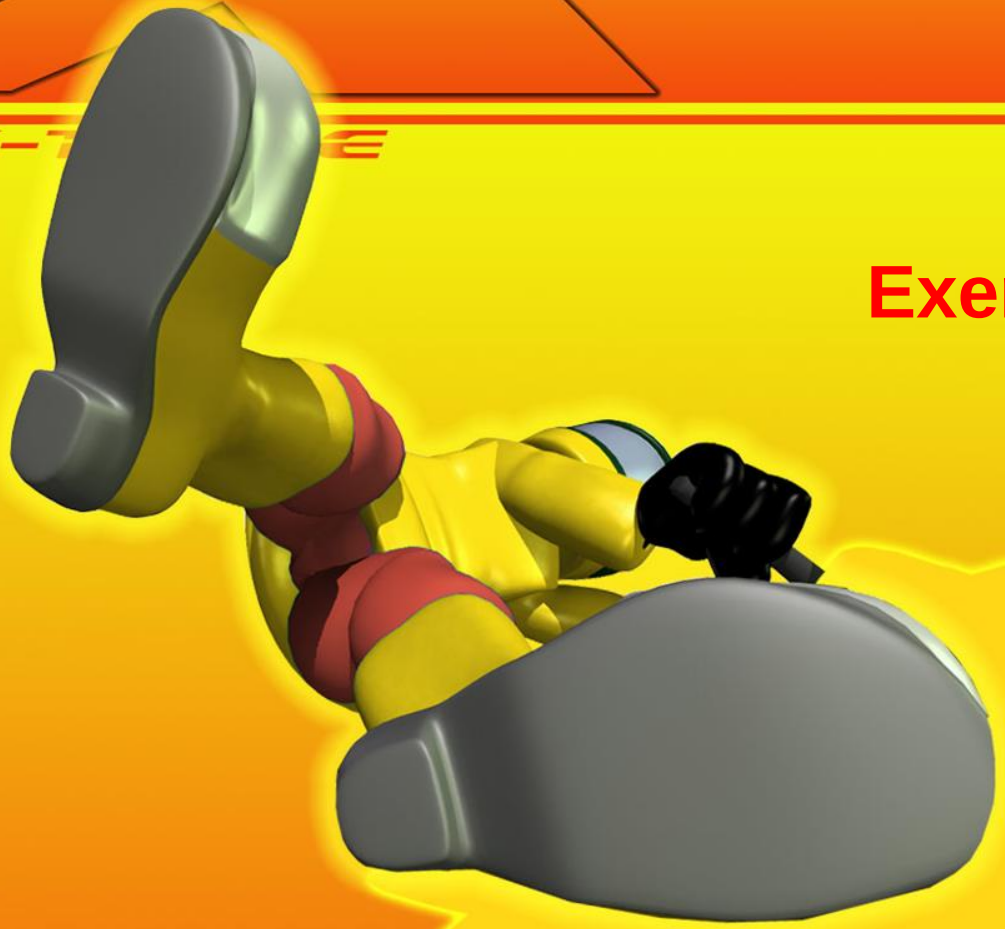
- Des solutions agiles adaptables :
  - Réactives aux nouveaux besoins.
  - Réceptives aux nouvelles solutions :
    - Prendre des décisions définitives le plus tard possible.
    - De courtes itérations permettent de changer de direction rapidement sans perte de temps.

Le manifeste Agile (8)

- L’estimation de charge est difficile, mais les courtes itérations aident dans la planification.
- On est plus précis sur les petites tâches.
- Feedback très rapide.
- Plus facile à s’adapter face aux dérives.



Exemple de méthode Agile SCRUM



Caractéristiques

- Produire le maximum de valeur pour le minimum de coût.
- Besoins capturés dans un *BackLog de produit* priorisé par une personne.
- Cycles de développement de 2 à 4 semaines *(Sprints) ;* équipes autogérées.
- Mêlée quotidienne.

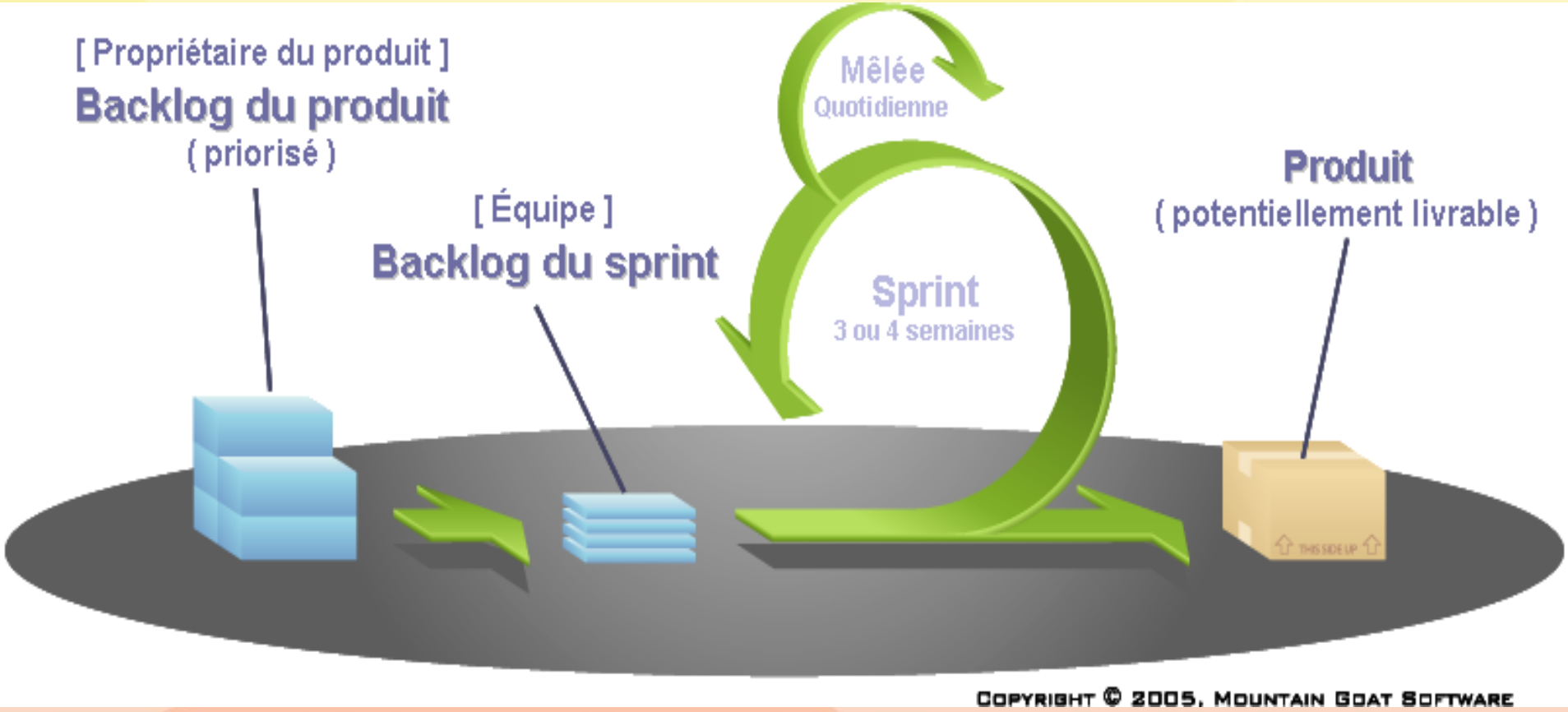
Caractéristiques (2)

- Equipe responsable, en auto-organisation.
- Avancement du produit par une série de « Sprints » d’un mois ou moins.
- Exigences définies comme des éléments d’une liste appelée « BackLog de produit ».
- Un des « processus agiles ».

Caractéristiques (3)

- Pas de prescription de pratiques d’ingénierie.
- Utilisation de règles génériques permettant de créer un environnement agile pour un projet.

Le processus



Les Sprints

- Le projet Scrum progresse par une série de Sprints équivalents aux itérations d'Extreme Programming.
- La durée d’un Sprint est de 2 à 4 semaines.
- Une durée constante apporte un meilleur rythme.
- Le produit (partiel) est conçu, codé et testé pendant le Sprint.

Le cadre

Rôles

- Product Owner
- ScrumMaster
- Equipe

Cérémonial

- Planification du Sprint
- Revue du Sprint
- Rétrospective
- Scrum quotidien

Artefacts

- Backlog de produit
- Liste des tâches
- Burndowns

Le cadre (2)

Rôles

- Product Owner
- ScrumMaster
- Equipe

onial

ation du Sprint
- Revue du Sprint
- Rétrospective
- Scrum quotidien

roduit
- Liste des tâches
- Burndowns

Product Owner

- Définit les fonctionnalités du produit.
- Choisit la date et le contenu de la release.
- Responsable du retour sur investissement.
- Définit les priorités dans le BackLog en fonction de la valeur « métier ».
- Ajuste les fonctionnalités et les priorités à chaque Sprint si nécessaire.
- Accepte ou rejette les résultats.

ScrumMaster

- Représente le management du projet.
- Responsabilités proches de celle d’un Chef de Projet.
- Responsable de faire appliquer par l’équipe les valeurs et les pratiques de Scrum.
- Élimine les obstacles.
- S'assure que l'équipe est fonctionnelle et productive.
- Facilite une coopération entre tous les rôles et fonctions.
- Protège l'équipe des interférences extérieures.

L’équipe

- De 5 à 10 personnes regroupant tous les rôles.
- Architecte, concepteur, développeur, spécialiste IHM, testeur, etc.
- A plein temps sur le projet, de préférence (exceptions possibles (administrateur, …))
- L’équipe s’organise par elle-même.
- La composition de l’équipe ne doit pas changer pendant un Sprint.

Le cadre (3)

Rôles

- Product Owner
- ScrumMaster
- Equipe

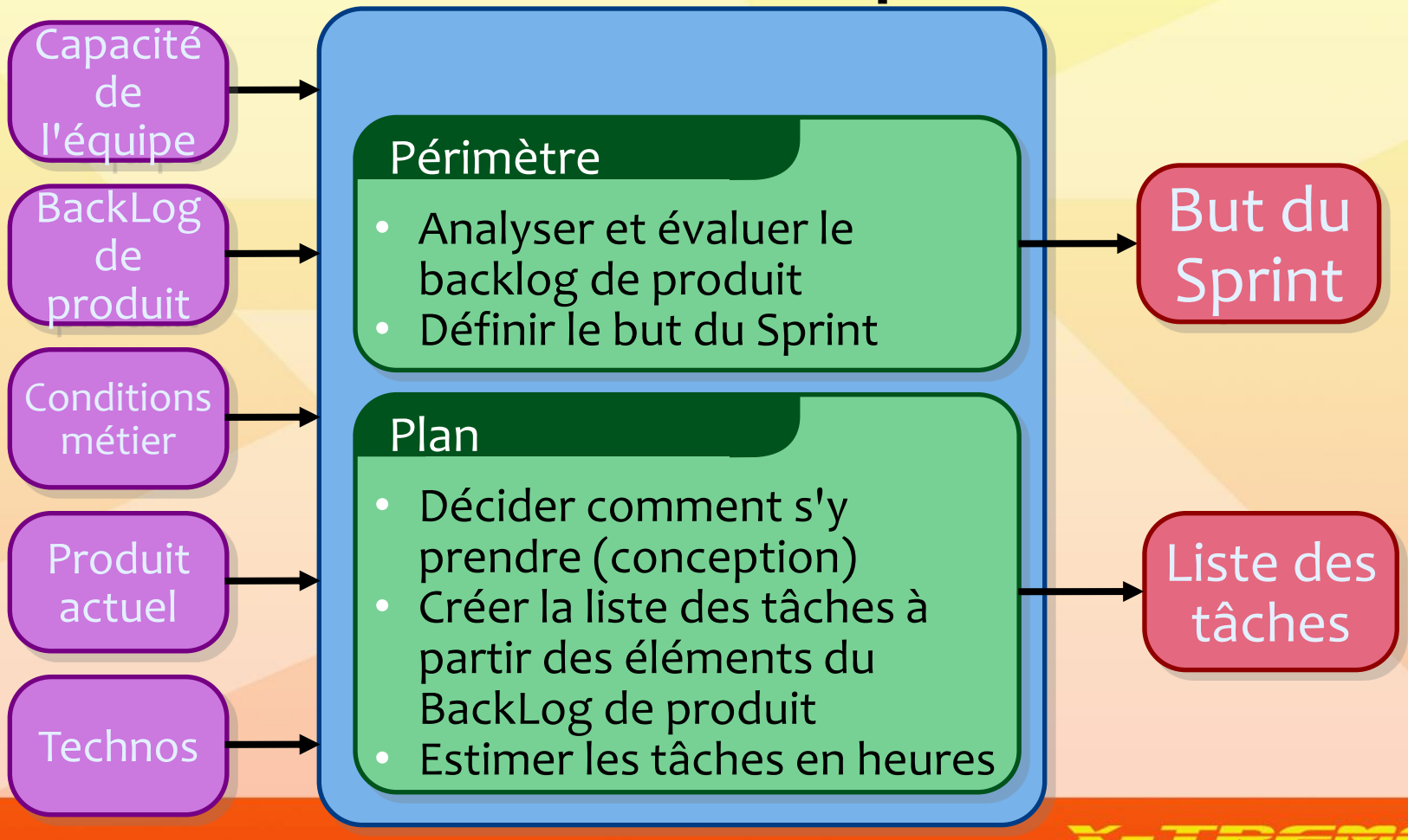
Cérémonial

- Planification du Sprint
- Revue du Sprint
- Rétrospective
- Scrum quotidien

Backlog de produit

- Liste des tâches
- Burndowns

Planification du Sprint

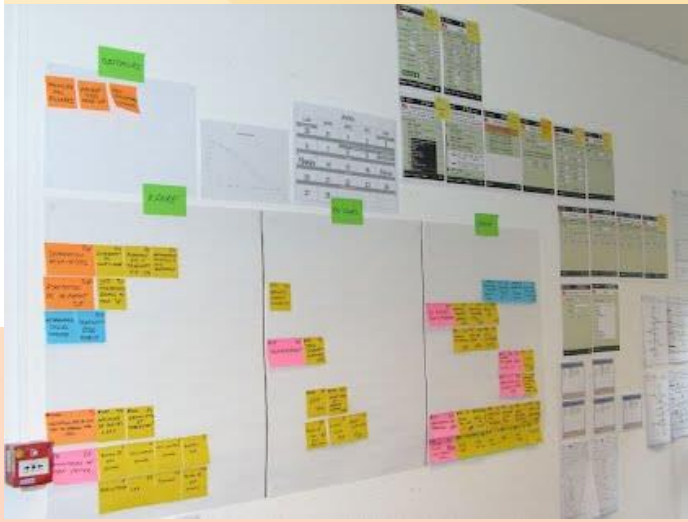


Planification du sprint (2)

- L'équipe choisit, à partir du BackLog de produit, les éléments qu'elle s'engage à finir.
- La liste des tâches est créée (stable dans le Sprint).
- Les tâches sont identifiées et estimées (1-16 heures) collectivement, pas seulement par le ScrumMaster.
- La conception de haut niveau est abordée.

Exemple de liste de tâches

Story	To Do	In Process	To Verify	Done
As a user, I... 8 points	Code the... 9 · Test the... 8 · Code the... 2 · Code the... 8 · Test the... 8 · Test the... 4	Code the... DC 4 · Test the... SC 8	Test the... SC 6	Code the... · Test the... · Test the... · Test the... SC 6
As a user, I... 5 points	Code the... 8 · Test the... 8 · Code the... 4 · Code the... 6	Code the... DC 8		Test the... · Test the... · Test the... SC 6



Planification du sprint (3)
Exemple

En tant que touriste potentiel dans la région, je veux voir les photos des hôtels

- Coder la couche de persistance (8 heures)
- Coder l'IHM (4)
- Ecrire les test fixtures (4)
- Coder la classe foo (6)
- Maj les tests de performance (4)

Mêlée quotidienne

- Paramètres :
  - Tous les jours.
  - 15 minutes.
  - Debout.
- Pas fait pour résoudre les problèmes.
- Tout le monde est invité mais seuls les membres de l'équipe peuvent parler.
- Permet d'éviter l'organisation d'autres réunions.

Déroulement du scrum quotidien

- Chacun répond à 3 questions :

1. Qu'as-tu fait hier ?
2. Que vas-tu faire aujourd'hui ?
3. Y a t-il un obstacle qui te freine ?

Revue du Sprint

- L'équipe présente ce qu'elle a fait pendant le Sprint.
- Se fait avec une démo de l'architecture ou des nouvelles fonctionnalités.
- Informelle.
- Préparation < 2h.
- Pas de slides.
- Toute l'équipe participe et on invite du monde.

Rétrospective du Sprint

- Réfléchir régulièrement à ce qui marche et ce qui ne marche pas.
- Dure en général de 15 à 30 minutes.
- Fait à la fin de chaque Sprint.
- Toute l'équipe participe :
  - ScrumMaster.
  - Product Owner.
  - Equipe.
  - Eventuellement clients et autres intervenants.

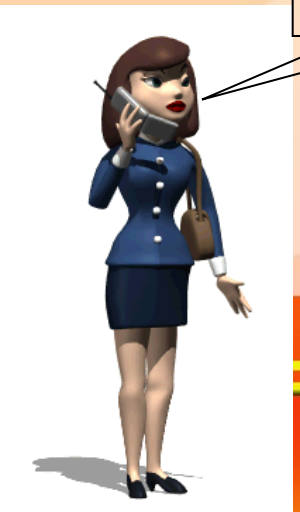
Retour sur les pratiques

- Toute l'équipe collecte du feedback et discute sur ce qu'elle aimerait :

Commencer à faire

Arrêter de faire

Continuer à faire



Scrum Framework

Rôles

- Product
- ScrumM
- Equipe

Cérémonial

- Planification de sprint
- Revue de sprint
- Rétrospectiv
- Scrum quotic

Artefacts

- Backlog de produit
- Liste des tâches
- Burndowns

BackLog du produit

- Les exigences.
- Une liste de tout ce qui va entraîner du travail pour l'équipe.
- Exprimé de telle façon que chaque élément apporte de la valeur aux utilisateurs ou clients du produit.
- Les priorités sont définies par le Product Owner.
- Les priorités sont revues à chaque Sprint.

BackLog du produit (2)
Exemple

Elément de backlog	Estimation
Un invité peut faire une réservation	3
En tant qu'invité, j'annule une réservation	5
En tant qu'invité, je change les dates d'une réservation.	3
En tant qu'employé de l'hôtel, je produis les rapports de revenu par chambre	8
Améliorer la gestion des exceptions	8
…	30
…	50

Mise en place d'un Sprint

- Un bref énoncé de sur quoi va porter l'essentiel du travail pendant le Sprint.

Application BD

Faire tourner l'application sur une base MySQL en plus d'Oracle.

Services financiers

Offrir plus d'indicateurs que le produit ABC sur les données de streaming .

La liste des tâches

- Chacun s'engage sur du travail qu'il choisit, le travail n'est jamais attribué par un autre.
- L'estimation du reste à faire est ajustée tous les jours.
- N'importe qui peut ajouter, supprimer ou changer la liste des tâches.

La liste des tâches (2)

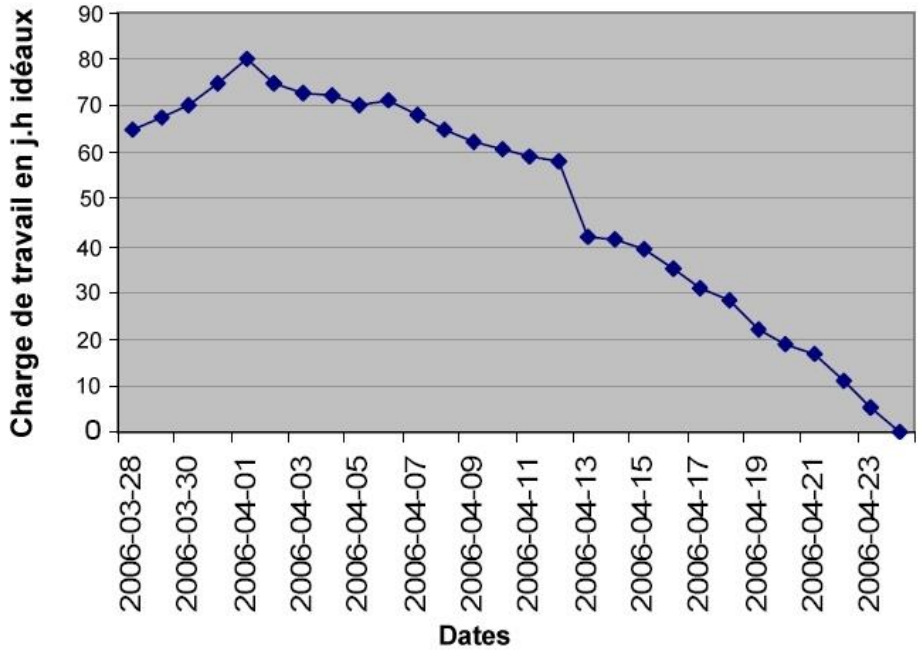
- Le travail du Sprint émerge progressivement.
- Si un travail n'est pas clair, définir une tâche avec plus de temps et la décomposer après.
- Mise à jour du travail restant quand il est mieux connu.

La liste des tâches (3)
Exemple

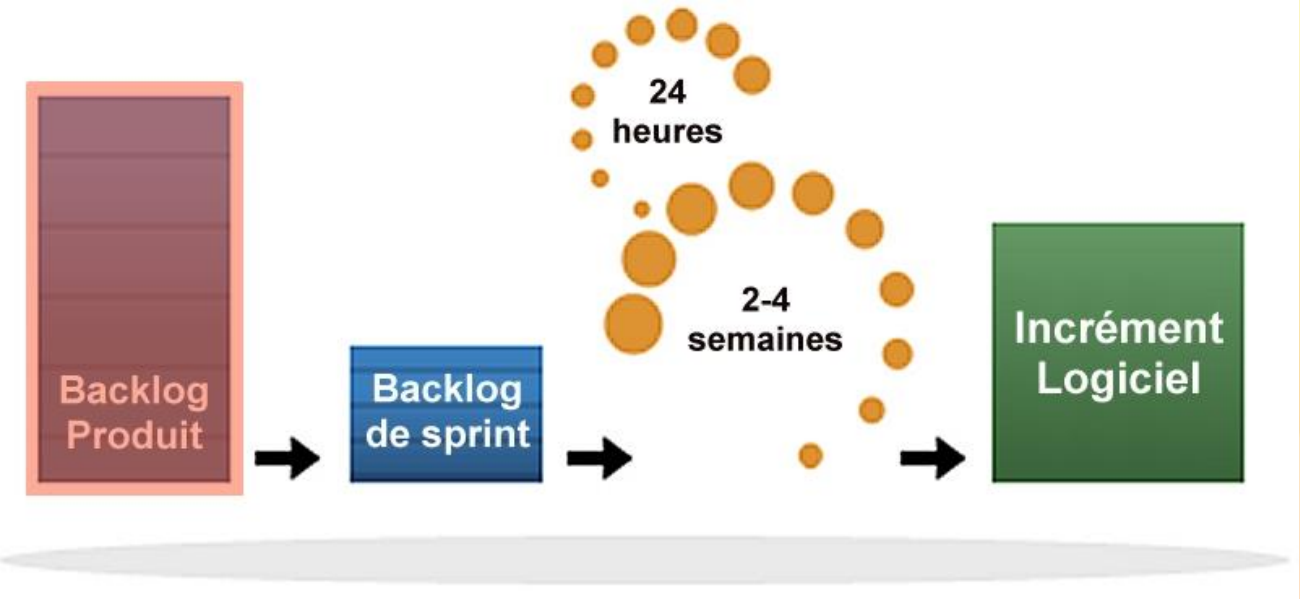
Tâches	Lun	Mar	Mer	Jeu	Ven
Coder l'IHM	8	4	8		
Coder couche métier	16	12	10	4	
Tester l'intégration	8	16	16	11	8
Ecrire l'aide en ligne	12				
Ecrire la classe foo	8	8	8	8	8
Tracer les erreurs			8	4	

BurnDown

- Ce qui reste à faire.
- Ajusté à la fin de chaque Sprint.



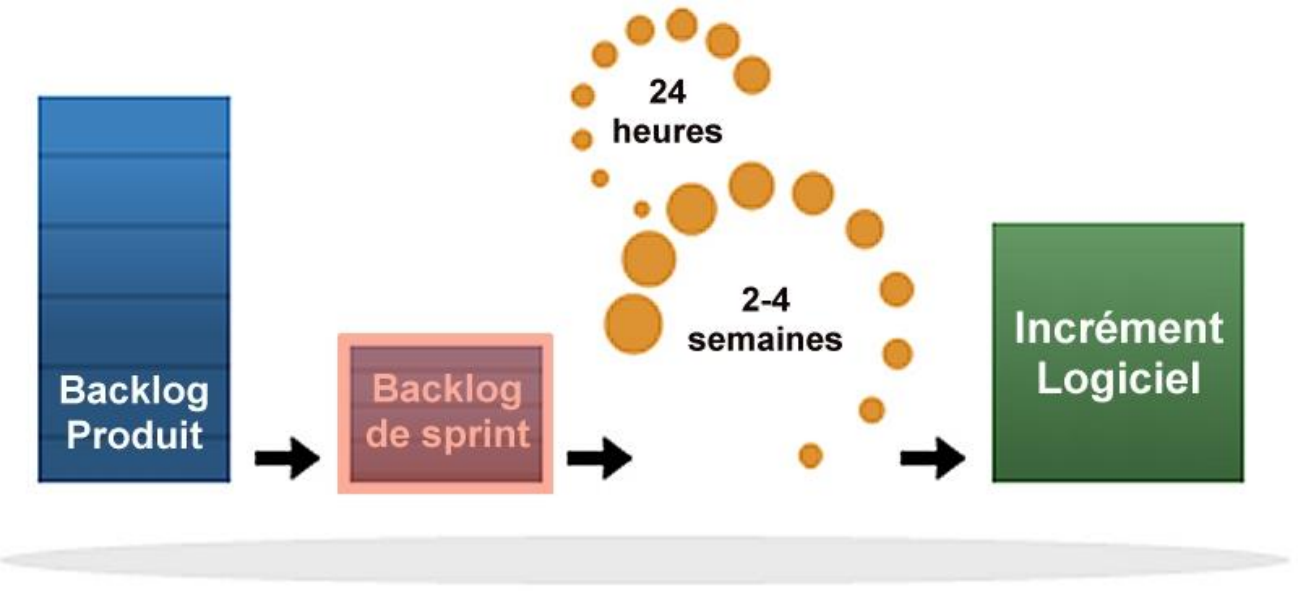
Résumé



Source : www.scrumalliance.org

1. **BackLog produit** (ou catalogue des besoins) :
   - Besoins priorisés par le Product Owner.
   - Besoins évalués par l'équipe.

Résumé (2)

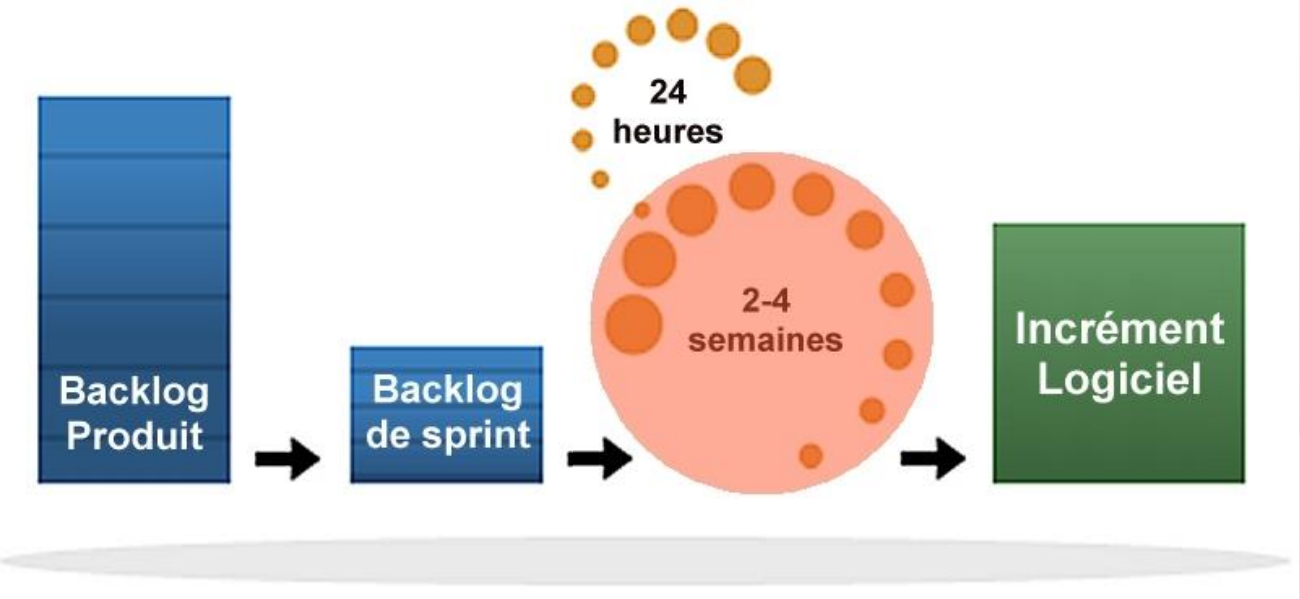


Source : www.scrumalliance.org

2. Backlog de sprint :

- Extrait du BackLog produit.
- Besoins éclatés en tâches.

Résumé (3)

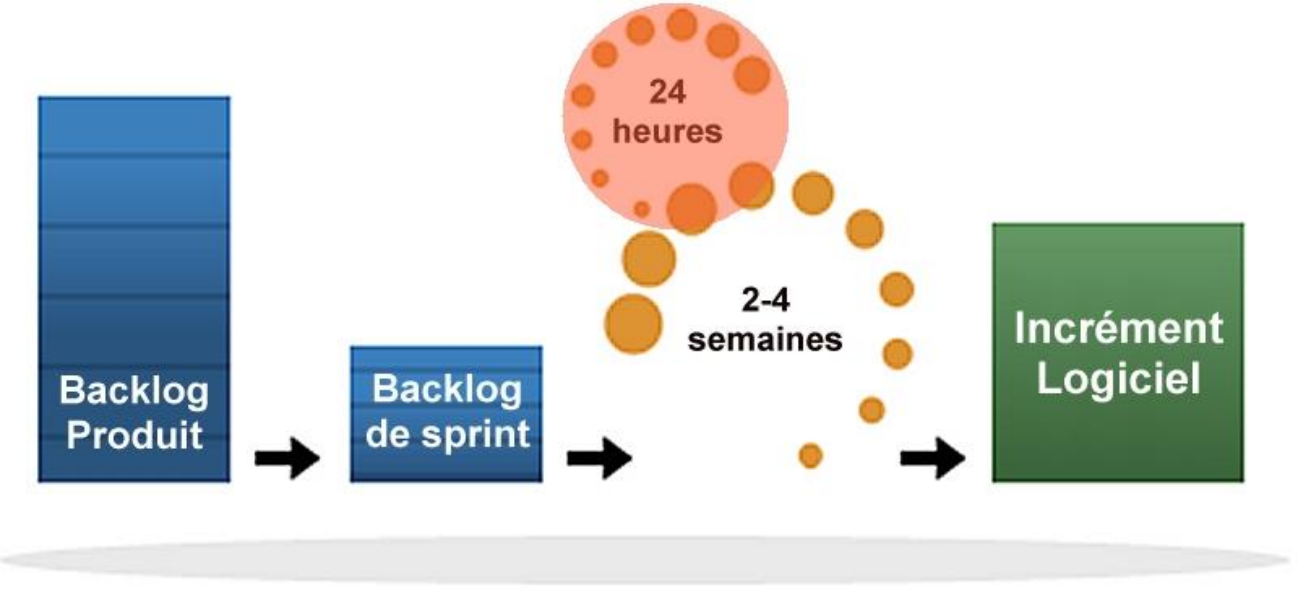


Source : www.scrumalliance.org

3. Sprint :

- Développement des fonctionnalités du BackLog de Sprint.
- Aucune modification du BackLog de Sprint possible.

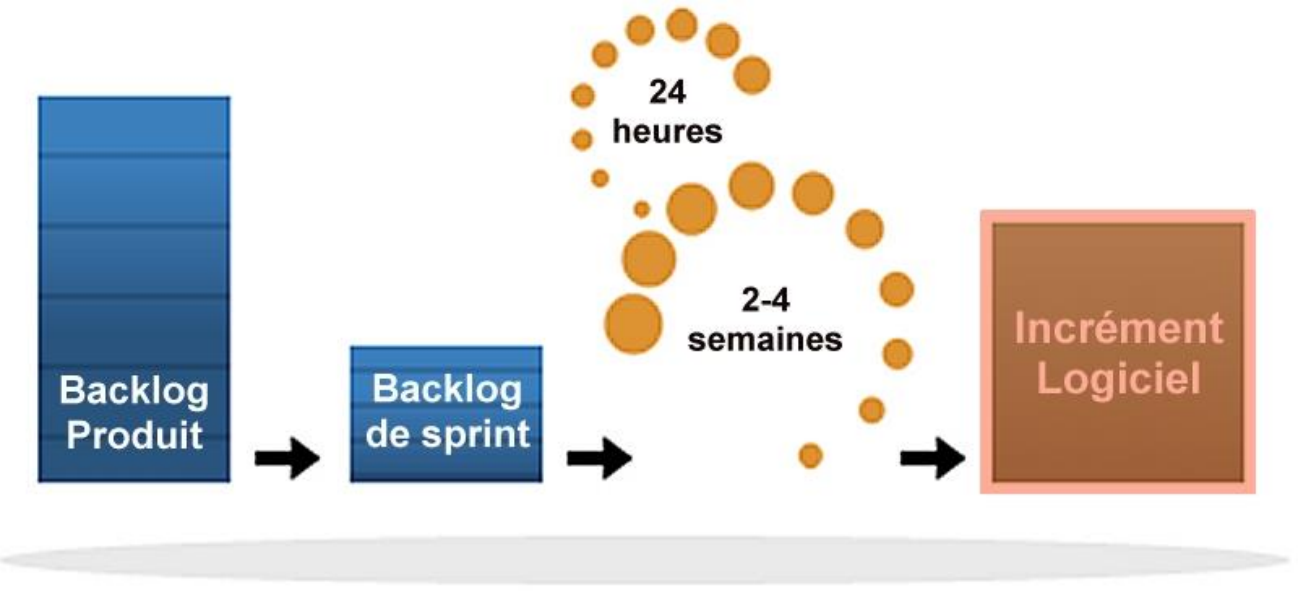
Résumé (4)



Source : www.scrumalliance.org

4. Mêlée quotidienne = point de contrôle quotidien de l'équipe.

Résumé (5)



Source : www.scrumalliance.org

5. **Incrément logiciel** : livré au Product Owner à la fin du sprint.

SCRUM à grande échelle

- Une équipe typique : 7 ± 2 personnes.
- Le changement d'échelle se fait par la collaboration de plusieurs équipes :
  - Facteurs dans la scalabilité.
  - Type d'application.
  - Taille de l'équipe.
  - Répartition géographique des équipes.
  - Durée du projet.

Les outils

- **Outils traditionnels :**
  - Tableau blanc et post-its.
  - Excel – BackLog produit et BackLog de Sprint.
- **Outils dédiés :**
  - Outils commerciaux / Open source.
  - Absence de PERT / Gantt.
  - Intégration avec : IDE, contrôle de sources, gestion des tests, bug tracking, intégration continue.
- **Autres outils :**
  - Connexion large bande.
  - Wiki, webcams, messagerie instantanée…

Conclusion

- Pas d'évolution, peu de critiques.
- Défauts à palier :
  - Absence de dépendance entre les tâches.
  - Polyvalence des programmeurs.
  - Productivité équivalente supposée.
  - Grande **maturité** nécessaire.
- Contrats à adapter.
- **Stratégie d'introduction de Scrum en entreprise.**

Conclusion (2)

- Scrum est une méthode de gestion de projet.
- Doit être complétée par des techniques d’ingénierie logicielle.
- Complémentaire avec Extreme Programming.

Exemple de technique Agile
eXtreme Programming (XP)

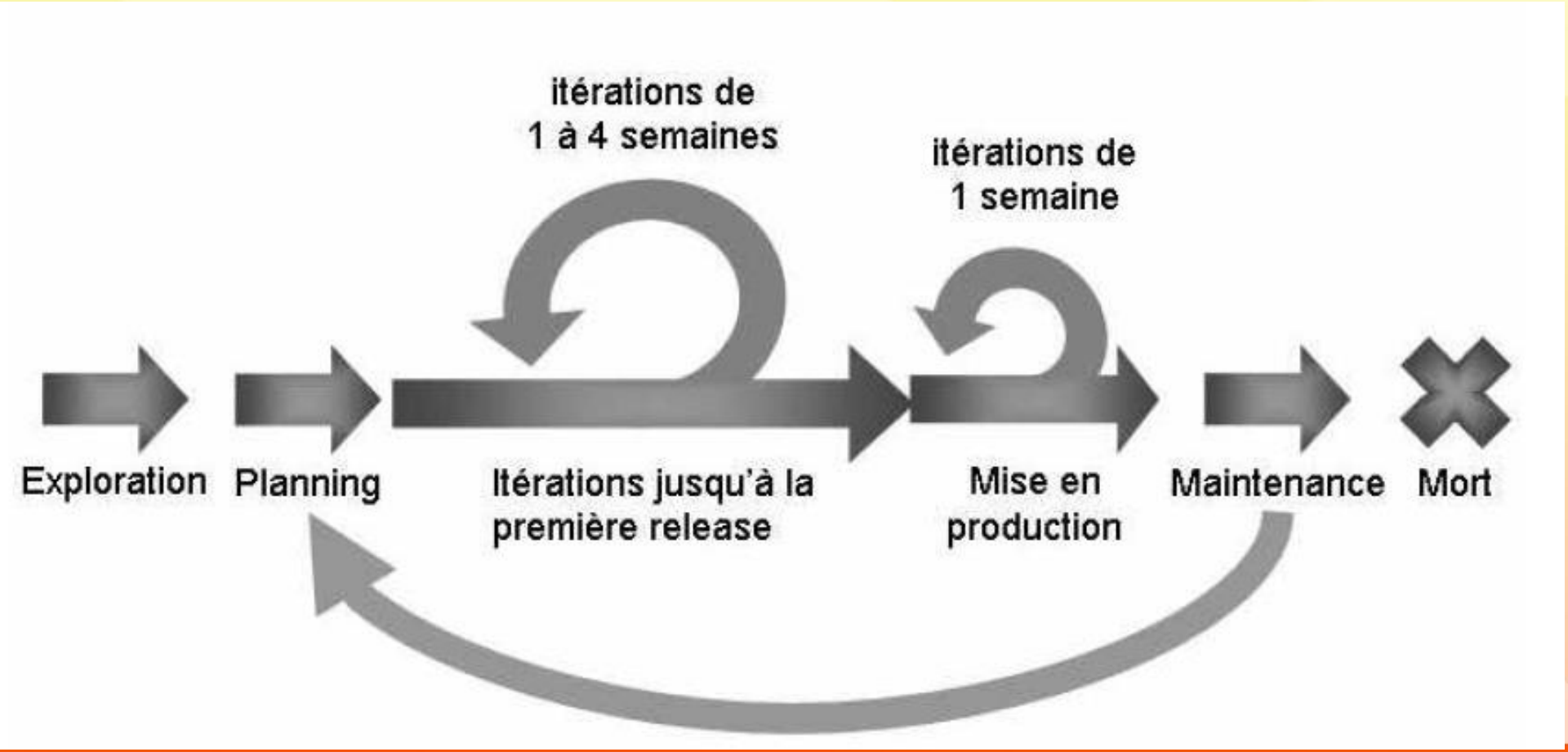
Présentation

- Méthode de développement.
- Rassemblement de bonnes pratiques déjà connues et utilisées.
- Une des méthodes agiles.
- Récente : 1999.
- Bilan des premiers retours : fin 2004.
- Prend son essor début 2000 (Web).

Principes

- Feedback rapide et constant.
- Compréhension partagée.
- Bien-être de l’équipe.
- Processus fluide et continu en binôme.

Principes (2)


itérations de
1 à 4 semaines
itérations de
1 semaine
Exploration
Planning
Itérations jusqu'à la
première release
Mise en
production
Maintenance
Mort

Aspects humains et collaboratifs

- Dimension humaine = souvent la plus déterminante pour la réussite de tout projet.
- Aspects importants :
  - Le comité de pilotage.
  - Les différents profils et le rôle de chaque profil.
  - L’unification des savoirs : « la formation ».
  - Les règles de collaboration.

Le comité de pilotage

- Comité de pilotage :
  - dirige les aspects stratégiques du projet.
  - définit les objectifs et alloue les moyens nécessaires.

Le comité de pilotage (2)

- Client interlocuteur :
  - le plus à même de connaître le besoin et de l'exprimer.
  - « Client interlocuteur » = appui du « sponsor principal ».
  - directeur des systèmes d'information.
  - porte-parole officiel de l'entreprise disposant d'une autorité suffisante pour légitimer le projet et incarner l'engagement de l'entreprise.

Le comité de pilotage (3)

- Gère la phase de préparation, il faut s’assurer de :
  - la clarté des objectifs du projet.
  - leur compréhension par toutes les parties prenantes.
  - Accord de l’entreprise cliente.

Le comité de pilotage (4)

- Assure le bien-être de l'équipe :
  - Une élimination progressive des risques.
  - Établir un rythme de travail régulier, sans heures supplémentaires .
  - But : Éliminer
    - Le stress.
    - Tensions à l'intérieur du groupe.
    - Tensions entre le groupe et les donneurs d'ordres.
    - Les risques de départ ou d'absentéisme.

Les différents profils

- Profils intervenants dans une équipe XP :
  - Le client.
  - Le testeur.
  - Le manager.
  - Le coach.
  - Le tracker.
  - Le programmeur.

Le rôle des profils

- Le **client** :
  - Membre à part entière de l'équipe.
  - Présence physique imposée dans l'équipe tout au long du développement.
  - Spécifie les fonctionnalités à implémenter et tests fonctionnels.
  - Rôle pouvant être tenu par une ou plusieurs personnes.

Le rôle des profils (2)

- Le **testeur** :
  - Assistant du client = un programmeur.
  - Implémente (code) les tests de **recette** le plus tôt possible, valide le fonctionnement du code et vérifie la non **régression.**

Vocabulaire

- **Recette (ou test d'acceptation)** est une des phases de développement des projets.
- Les différents acteurs du projet se rencontrent afin de vérifier que le produit est conforme aux attentes formulées.

La Recette

- On distingue :
  - La recette Usine.
  - La recette Utilisateur.

La recette USINE

- Comprend tous les tests réalisés chez le fournisseur, avant la livraison :
  - Tests unitaires.
  - Tests de validation.
  - Tests d'intégration.

Les tests unitaires

- Procédé permettant de s'assurer du fonctionnement correct d'une partie déterminée d'un logiciel ou d'une portion d'un programme (appelée unité ou module)
- Outil .Net : NUnit

Les tests de validation

- Vérifient si toutes les exigences client sont respectées.

- Aspect purement fonctionnel.

Les tests d’intégration

- Vérifient que la cohésion des différents modules est assurée.

Les tests de non régression

- Vérifie qu'un correctif de bug n'engendre pas un nouveau dysfonctionnement.

La recette UTILISATEUR

- Le client réalise deux catégories de tests :
  - Recette technique : vérifier que le produit livré est techniquement conforme sur toute la chaîne de processus.
  - Recette fonctionnelle : contrôler l'aspect fonctionnel du produit.

Le rôle des profils (3)

- Le **manager** :
  - Responsable de l'infrastructure dans laquelle l'équipe travaille.
  - S'assure la non existence de problèmes étrangers au projet ou logistiques (espace de travail, outillage, documentation, …).

Le rôle des profils (4)

- Le **coach** :
  - S’ assurer la bonne compréhension de la méthode XP et son application correcte par les différents acteurs du projet.
  - Généralement « expert méthode » et bon communicateur doublé d’un technicien crédible et respecté.

Le rôle des profils (5)

- Le **tracker** :
  - Contrôle l'avancement des tâches à l'intérieur d'une itération.
  - S'entretient fréquemment avec chaque programmeur pour s'enquérir des difficultés rencontrées et du travail déjà effectué.
  - Construit régulièrement une vision fiable de l'avancement des tâches afin de détecter le plus tôt possible les dérives et de lancer une nouvelle phase si nécessaire

Le rôle des profils (6)

- Le **programmeur** :
  - Estime la charge nécessaire à l'implémentation d'un scénario dans le cadre du jeu de la planification.
  - Implémente les scénarios en s'appuyant sur l'écriture de tests unitaires.
  - Est aussi analyste, concepteur.

Unification des savoirs par la formation

- **Mise à niveau des ressources avant le lancement du projet :**
  - Phase d'apprentissage théorique par un formateur professionnel.
  - Mise en pratique dans le cadre du projet pilote.

Unification des savoirs par la formation (2)

- **Projet partagé en 2 phases :**
  - Coaching, peu productive. Objectif :
    - Réaliser en situation le potentiel de chaque membre.
    - Gérer les parcours individualisés avec le soutien pédagogique de moniteurs (les coachs).
  - Fonctionnement productif du mode coaching lors de la première phase du vrai projet.

Unification des savoirs par la formation (3)

- **La montée en compétence :**
  - Maximiser la rentabilité de l'investissement pour la montée en compétences des ressources.
  - Éviter l'écueil 80/20 : l'acquisition des 20% de connaissances les plus pointues requiert 80% des efforts (et de l'investissement) en formation.

→ Faire confiance au temps pour l'uniformisation des compétences collectives.

Les règles de collaboration

- Exigence de qualité du code produit
  - Conventions de codage :
    - Homogénéité du niveau et style de codage des développeurs.
    - Utilisation d'outils de mise en forme automatique du code basés sur des conventions de codage paramétrables.
    - Intégration continue.

Les règles de collaboration (2)

- Programmation en binôme pour :
  - Se contrôler mutuellement
  - Diminuer les erreurs de conception
  - Meilleure concentration sur le travail
  - Éliminer le risque de dépendance à un développeur
  - Lisser les inégalités d'expériences en associant confirmé & débutant

Le Planning Game ou jeu de planification

- Choix d'un rythme de livraison (2-3 mois).
- Choix du rythme des itérations (1 à 3 semaines).
- **Étapes :**

1. Le client décrit ses besoins : user stories

Histoires simples, dans le langage du client décrivant les fonctionnalités demandées.

Un projet peut comporter une centaine de user stories.

Le Planning Game ou jeu de planification (2)

2. Les développeurs estiment le coût d'implémentation

En équipe avec la collaboration du client.

On note en "semaines idéales".

Une histoire trop grande (>3 points) doit être partagée. Si on ne sait pas estimer, on explore le sujet en faisant un essai ("spike").

Le Planning Game ou jeu de planification (3)

3. Le client choisit les histoires à implémenter

Le client trie les histoires par priorité.

Les développeurs déclarent leur vélocité (performance réelle mesurée à chaque itération).

Calcul du nombre de points que l'équipe peut traiter en une itération.

Le client choisit les histoires à implémenter dans la prochaine livraison.

Le Planning Game ou jeu de planification (4)

4. Les développeurs implémentent les histoires :

Planning d'une itération : pour chaque histoire prévue dans la livraison, les développeurs déterminent les tâches concrètes et les estiment en "jours idéaux".

Les développeurs acceptent les tâches et les implémentent.

Le Planning Game ou jeu de planification (5)

- **Avantages**
  - Confrontation et homogénéisation des connaissances métier par les développeurs.
  - La priorisation des scénarios permet d'avoir rapidement un noyau fonctionnel.
- **Limites**
  - Une inconnue technique risque de limiter la fiabilité de l'évaluation de la charge donnée par les développeurs (lors de la phase d'exploration).

Frequent Releases ou livraisons fréquentes

- **Principe**
  - Livrer en fin de chaque itération, une toutes les quelques semaines.
- **Avantages**
  - Donner au client le plus fréquemment possible, un aperçu sur l'état d'avancement de l'application.
  - La possibilité de corriger rapidement d'éventuelles erreur ou mauvaises interprétations.
  - Diminuer la quantité de travail à valider.

Frequent Releases ou livraisons fréquentes (2)

- **Limites**
  - Techniquement, les livraisons fréquentes demandent d'avoir un mécanisme d'intégration fiable :
    - Performant (compatible avec la fréquence de livraison).
    - Automatisé (reproduire une livraison antérieure).

Client sur site

- **Avantages**
  - Fiabiliser la communication des besoins aux développeurs et accélérer le processus d'une manière générale.
  - Élimination de la charge de production des documents de spécifications.
  - Meilleure visibilité sur l'avancement du projet pour le client.
  - Tout problème qui pourrait nécessiter une remise en question des priorités (lot de scénarios à implémenter) peut être traité sans délai.

Client sur site (2)

- **Limites**
  - Une des exigences de XP les plus complexes à réaliser dans la pratique.
  - En pratique, il arrive que le client présent sur le site du projet ne dispose pas de :
    - La connaissance pour répondre aux questions des développeurs.
    - Le pouvoir et l'autorité nécessaires à la prise de décision.
    - Avoir une personne consacrée au rôle de client XP est un luxe !

Architecture

- XP propose une démarche basée sur l'enchaînement suivant :
  - Ecriture d'un test.
  - Conception.
  - Programmation.
  - Refactoring (reconception du code).
- XP a une préférence pour la technique des CRC Cards (Class Collaborators Responsabilities).

CRC Cards

Class Name	
Superclasses	
Subclasses	
Responsibilities	Collaborators

La classe apporte un service aux classes décrites

Les classes dont la classe se sert pour assurer ses responsabilités

Règles de programmation

- Une classe = une seule et unique personne mais le code de l'application appartient à toute l'équipe (consultable par tous).
- Développement en binôme.
- Le code créé doit être composé de petites classes et de méthodes courtes.
- Programmation pilotée par les tests.
- Aucune duplication de code ne doit exister.
- Présence obligatoire de commentaires.

Nommage et langage

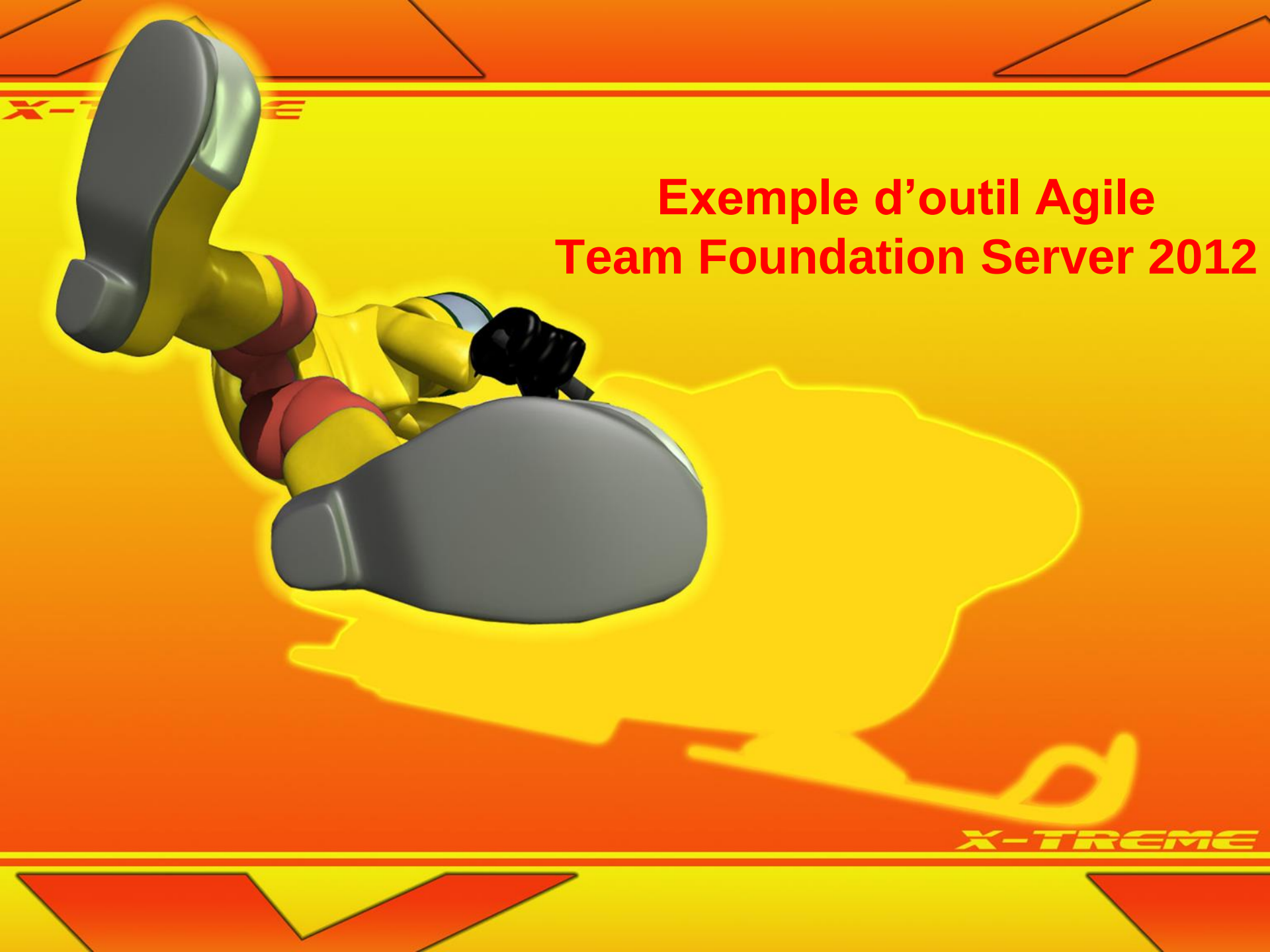
- Respecter des normes de nommage pour les variables, méthodes, objets, classes, fichiers.
- Langage idéal :
  - orienté objet (.Net, Java).
  - des cycles de développement courts.
  - permettant la mise en œuvre rapide de tests.

Les freins d'XP

- Méthodologie lightweight : blocage culturel.
- Des exigences fortes :
  - développeur : compétence suffisante.
  - équipe: communication & collaboration efficace.
- Une très forte implication :
  - responsabilité du coach.
  - le client XP.

Conclusion

- Encore à une phase de lancement.
- Une nouvelle vague de méthodes :
  - Pragmatisme.
  - Orientation humaine .
- Bien adapté au monde du Web et du jeu vidéo.
- Argument commercial.


Exemple d’outil Agile
Team Foundation Server 2012
X-TREME

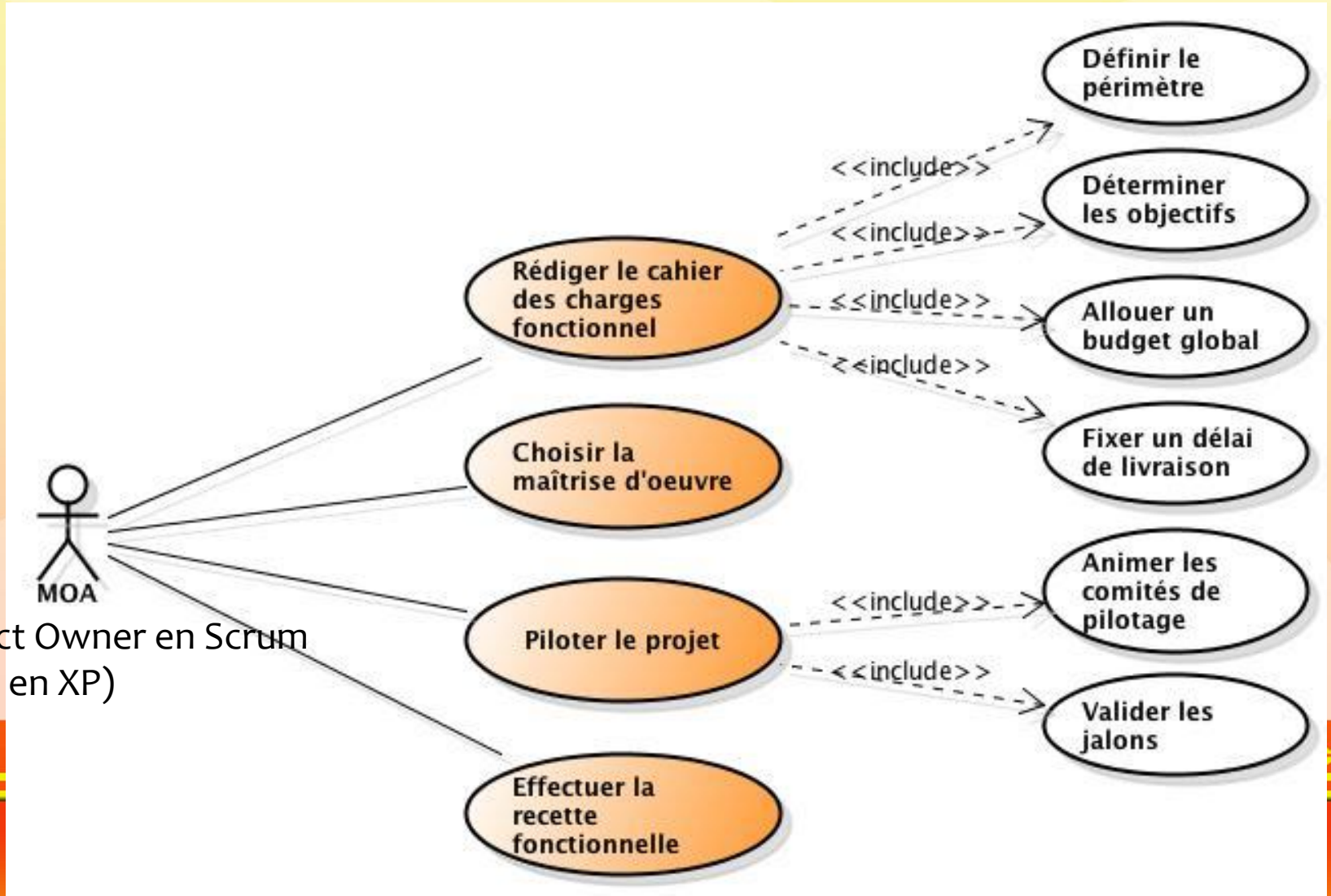
Introduction

- Aujourd'hui seuls ~35% des projets aboutissent.
- Problèmes divers :
  - Barrières organisationnelles.
  - Mauvaise formalisation du besoin.
  - Manque de connaissance du domaine fonctionnel (focus technique au lieu de focus sur l'usage).
  - Manque de communication dans l'équipe.
  - Mauvaise approche du changement.
- Projets réussis = exploits individuels et non d'équipes.

Introduction (2)

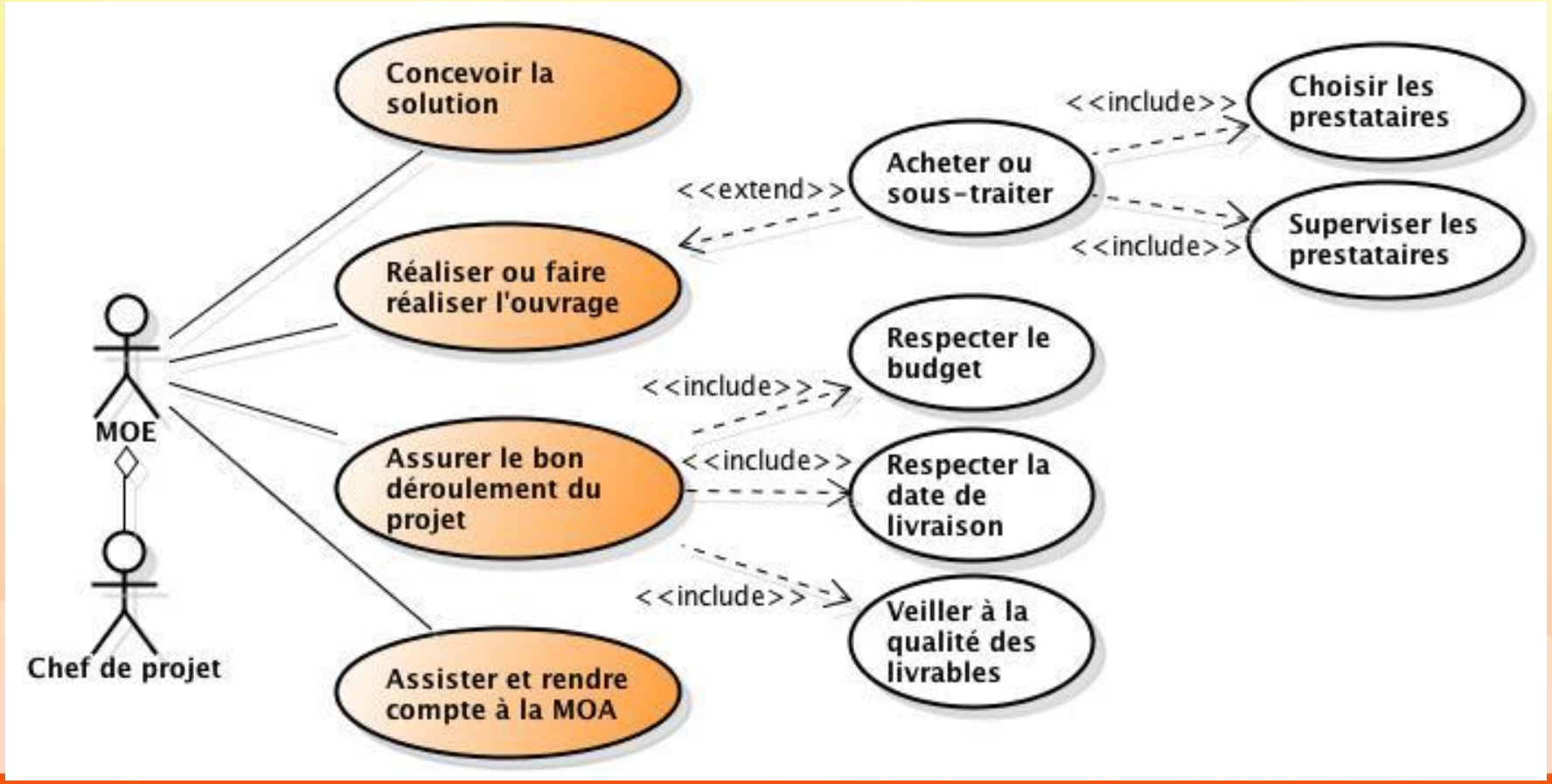
- Reproches classiques :
  - La MOA et la MOE ont du mal à se comprendre !
  - Les projets sous-traités manquent de transparence !
  - Il est difficile de gérer l'avancement des projets !
  - On a besoin de process mais légers et peu intrusifs !
  - On ne fait pas assez de tests !

La maîtrise d'ouvrage…ou client



(Product Owner en Scrum
Client en XP)

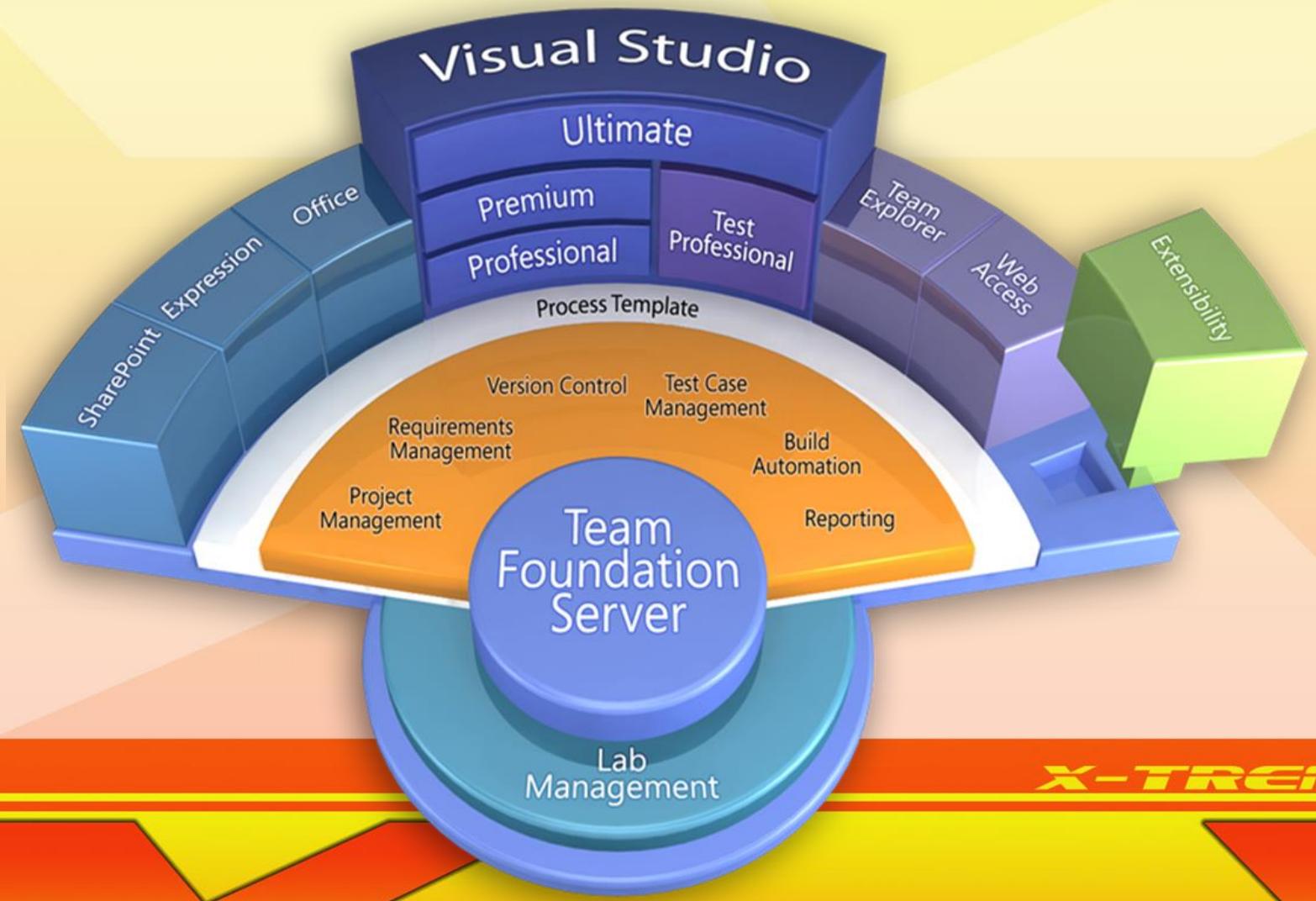
Maître d'œuvre classique



TFS ?

- Plateforme de travail collaboratif : permet de connecter les développeurs, les testeurs, les managers et les analystes en créant un environnement collaboratif pour l'ensemble de l'équipe.
- TFS permet notamment de :
  - Suivre l'évolution des projets, accéder à un ensemble de reports.
  - Partager des informations (Work Items), des documents.
  - Partager les sources des projets.

Architecture générale



Architecture technique

- TFS est basé sur une architecture 3 tiers.
- TFS repose sur :
  - SQL Server 2012 pour stocker les données.
  - Un ensemble de web services permettant d’interagir avec ces données.
  - Windows 2008 server.
  - Sharepoint Services.

Les Works Items

- Elément sur lequel intervient un membre de l'équipe : une story, une tache, un bug, etc.
- Eléments de base permettant d'assurer le suivi du projet.
- Peuvent être entièrement personnalisés pour les besoin de la méthode, du projet.
- Les mises à jour sont sauvegardées.

Les reports

- TFS permet d’avoir des reports sur toutes les données stockées.
- Des reports sont fournis de base.
- Ils sont basés sur la technologie Reporting Services.

Intégration

- **Visual Studio 2012** : permet de gérer les work items, d'accéder aux documents partagés…
- **Excel** : permet de gérer les work items.
- **Project** : permet de gérer certains work items, selon la configuration (exemple : les tâches).

Sharepoint

- TFS créé automatiquement un nouveau site Sharepoint pour chaque projet.
- Ce site permet :
  - D’accéder aux documents.
  - D’accéder aux reports.
  - Plus généralement de partager toute sorte d’informations (contacts, meetings, forums, etc.).

Sécurité

- TFS nécessite une authentification Windows pour se connecter.
- TFS permet d’attribuer des permissions à des utilisateurs ou à des groupes d’utilisateurs.

Et maintenant,
passons à la pratique !