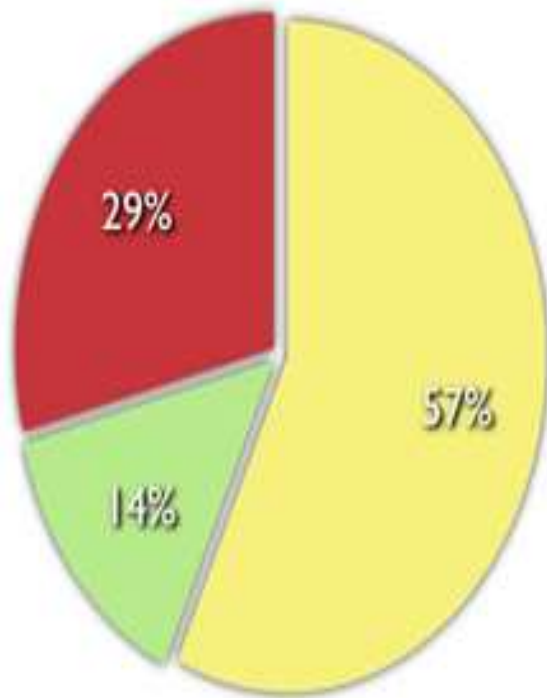




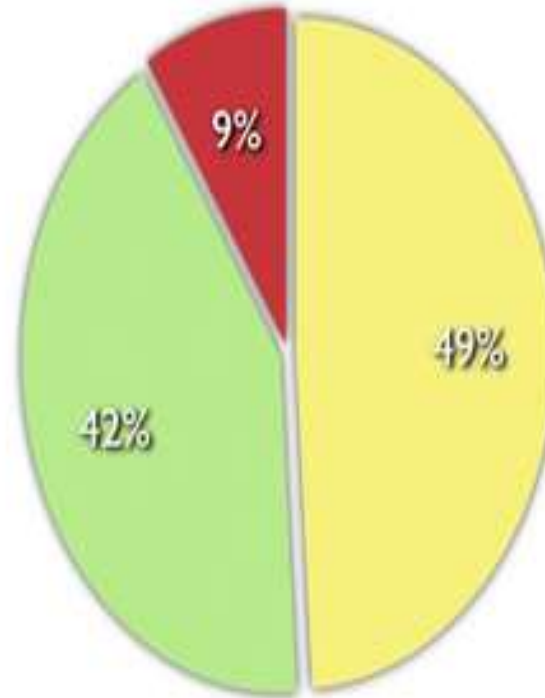
Chapitre 3: Méthodes Agiles



Waterfall



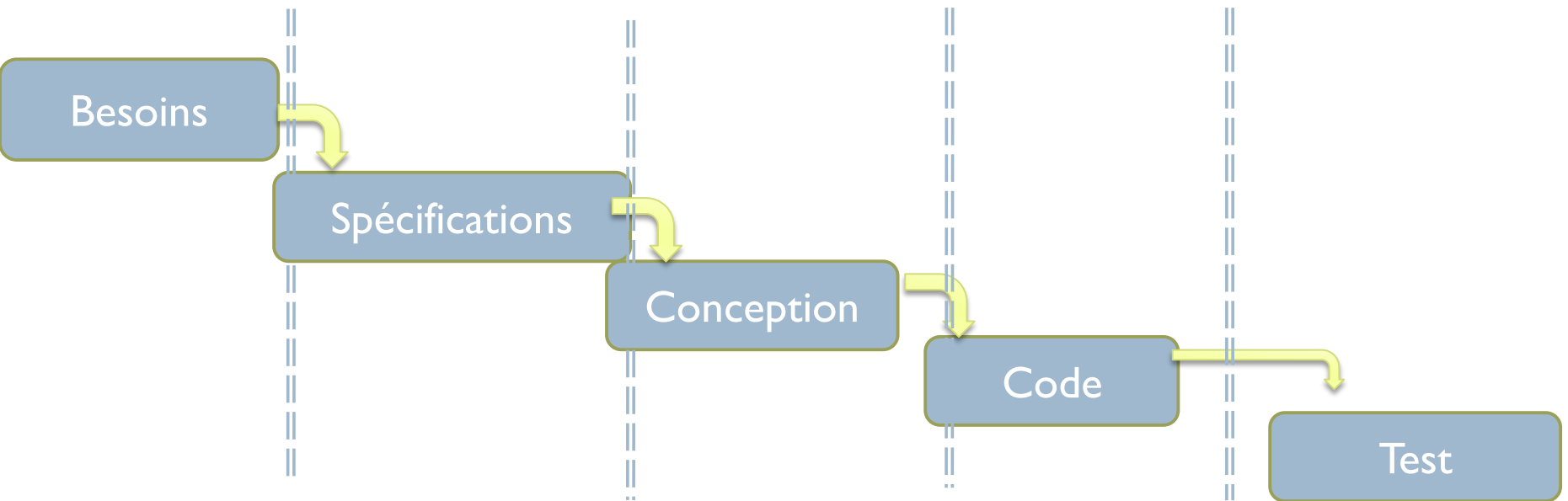
Agile



Source: The CHAOS Manifesto, The Standish Group, 2012.

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.
 - ▶ On n'a pas un résultat concret à évaluer.
 - **Visibilité tardive du produit :**
 - preuve tardive de bon fonctionnement
 - identification tardive des problèmes
 - Cadre **non transparent** au client.
-



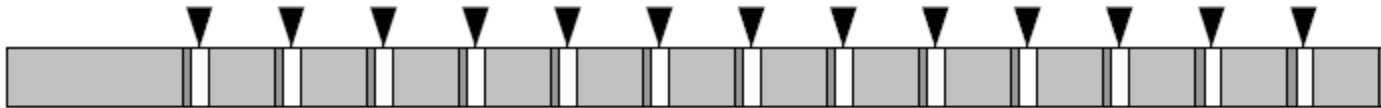
Problèmes linéaires

- ▶ Les méthodes **prédictives** fonctionnent bien, à condition d'avoir:
 - ▶ **Stabilité** et prévisibilité des besoins
 - ▶ Communication et compréhension parfaite
 - ▶ Choix parfaits dès le départ
- ▶ **Ce n'est pas toujours le cas**



Une solution : les méthodes agiles

- ▶ Principe:
- ▶ Diffuser **le processus de décision** tout au long du projet.
- ▶ Enchaînement **de cycles itératifs** courts.



Agile : Une catégorie de méthodes

- ▶ « Agile » regroupe plusieurs méthodologies :
 - ▶ Scrum
 - ▶ Extreme Programming (XP)
 - ▶ DSDM
 - ▶ Crystal
 - ▶ RUP
 - ▶ ...
- ▶ **Notion officialisée en 2001 avec le Manifeste Agile**



Le 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

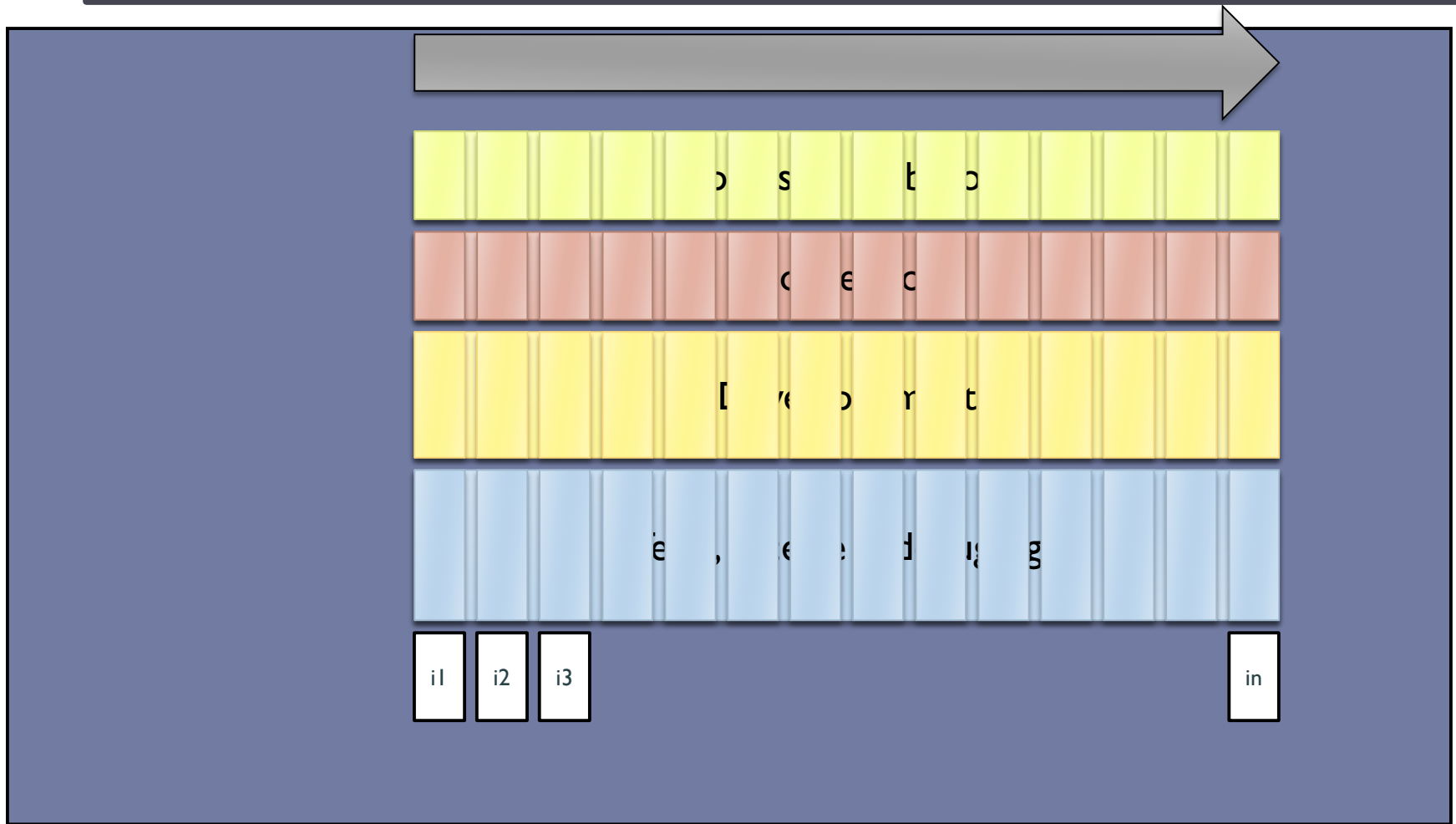
Adaptation au
changement

Plutôt que

Suivi d'un plan



Les solutions Agiles

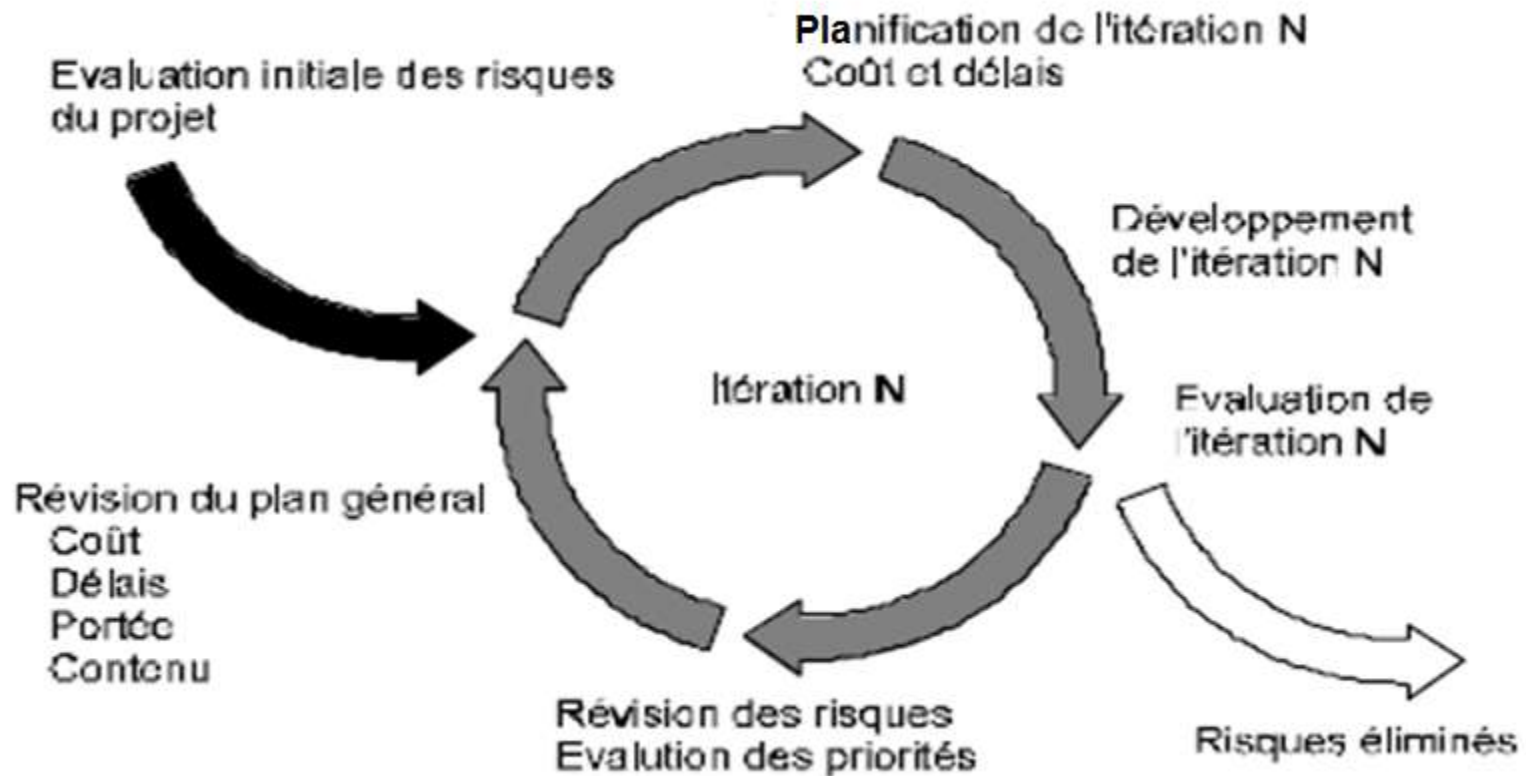


Les solutions Agiles

- Organiser en cycles de développement réduits
 - **Itérations**
- Adaptables
 - **Réactives aux nouveaux besoins**
 - **Réceptives aux nouvelles solutions**
- **Gestion du Risque:**
 - **Au début du projet**
 - **Tout au long du projet**




Une meilleure maîtrise du processus de développement



Chaque itération est construite comme un petit projet, dédié à l'élimination d'une partie des risques identifiés lors de la définition du projet.

Réduire le risque: Développement itératif

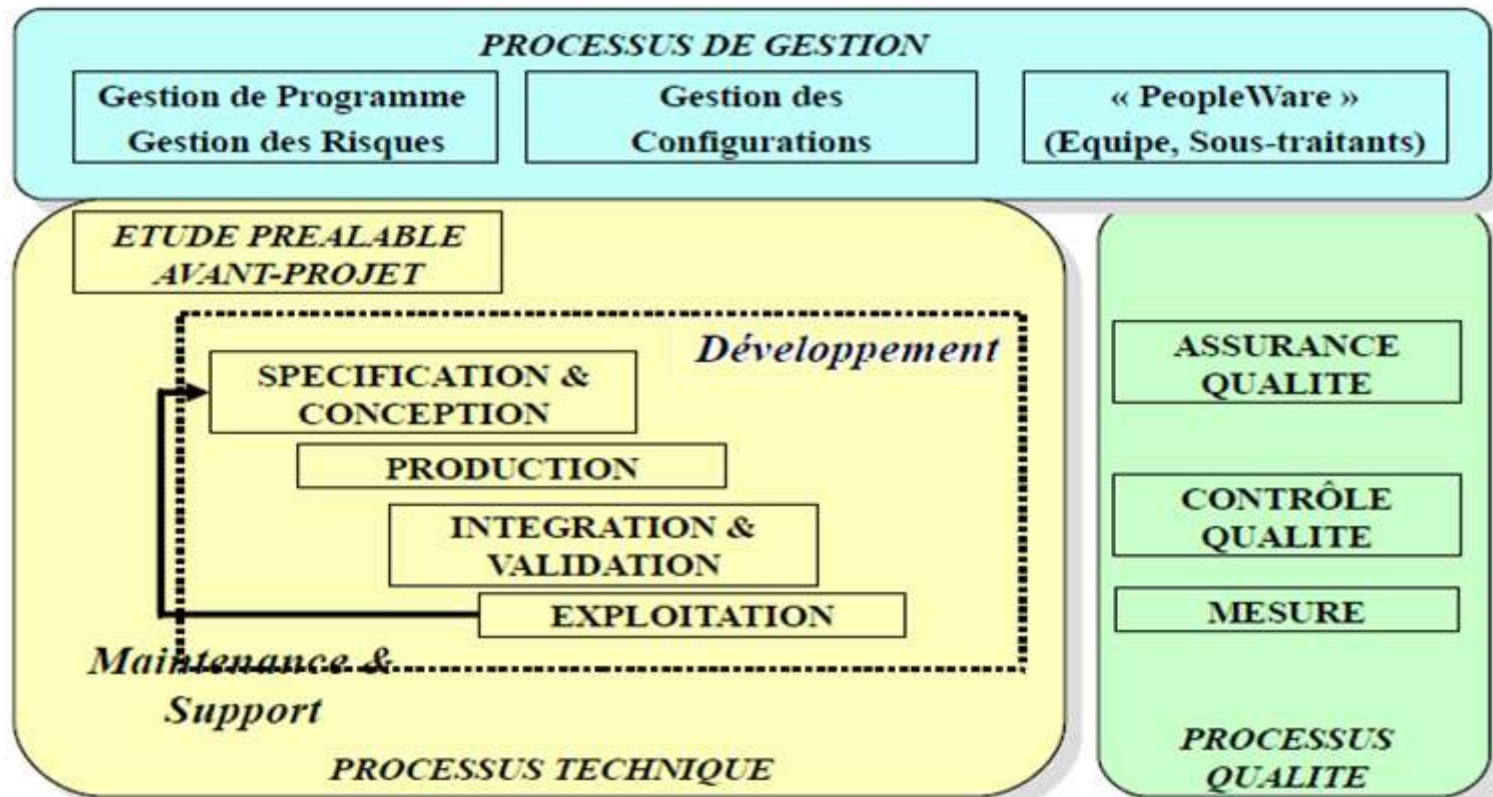
- ▶ **Livraisons fréquentes:** permettant des Tests fréquents
 - ▶ Tests fonctionnels ou de recette (à la fin de l'itération)
 - ▶ Tests unitaires (au cours de l'itération)
 - ▶ Evaluation du produit par le client
 - ▶ **Planification dynamique:**
 - ▶ introduire de nouveaux besoins,
 - ▶ replanifier les scénarios qui présentent des erreurs
 - ▶ **Evaluation des itérations:** Charge, Coût, Délai , Contenu
- 



Scrum



- ❑ **Projet** : ensemble d'activités organisées permettant de créer un produit ou un service avec une qualité définie dans le cadre d'un budget fixé.
- ❑ Un projet est régi par plusieurs Processus.



Introduction à Scrum

- ▶ Sprint = Course de vitesse sur une courte distance
- ▶ Scrum= mêlée
- ▶ *Terme emprunté au rugby: quand une équipe essaie d'avancer en restant unie,*
- ▶ **Méthode itérative et incrémentale**
- ▶ Equipes de 5 à 9 personnes.
- ▶ **Méthode agile la plus utilisée avec eXtreme Programming**
- ▶ **Méthode de gestion de projet.**
- **2001** : K. Schwaber et M. Beedle publient « Agile software development with Scrum ».



Scrum

- ▶ Conforme au manifeste de l'agilité
 - ▶ Manifeste de l'agilité publié en **2001**
 - ▶ **4** valeurs :
 1. Les personnes et les interactions plutôt que les outils et les processus.
 2. Le logiciel fonctionnel plutôt que de la documentation exhaustive
 3. La collaboration avec le client plutôt que la négociation de contrat
 4. L'adaptation au changement plutôt que le respect d'un plan pré-établi
-

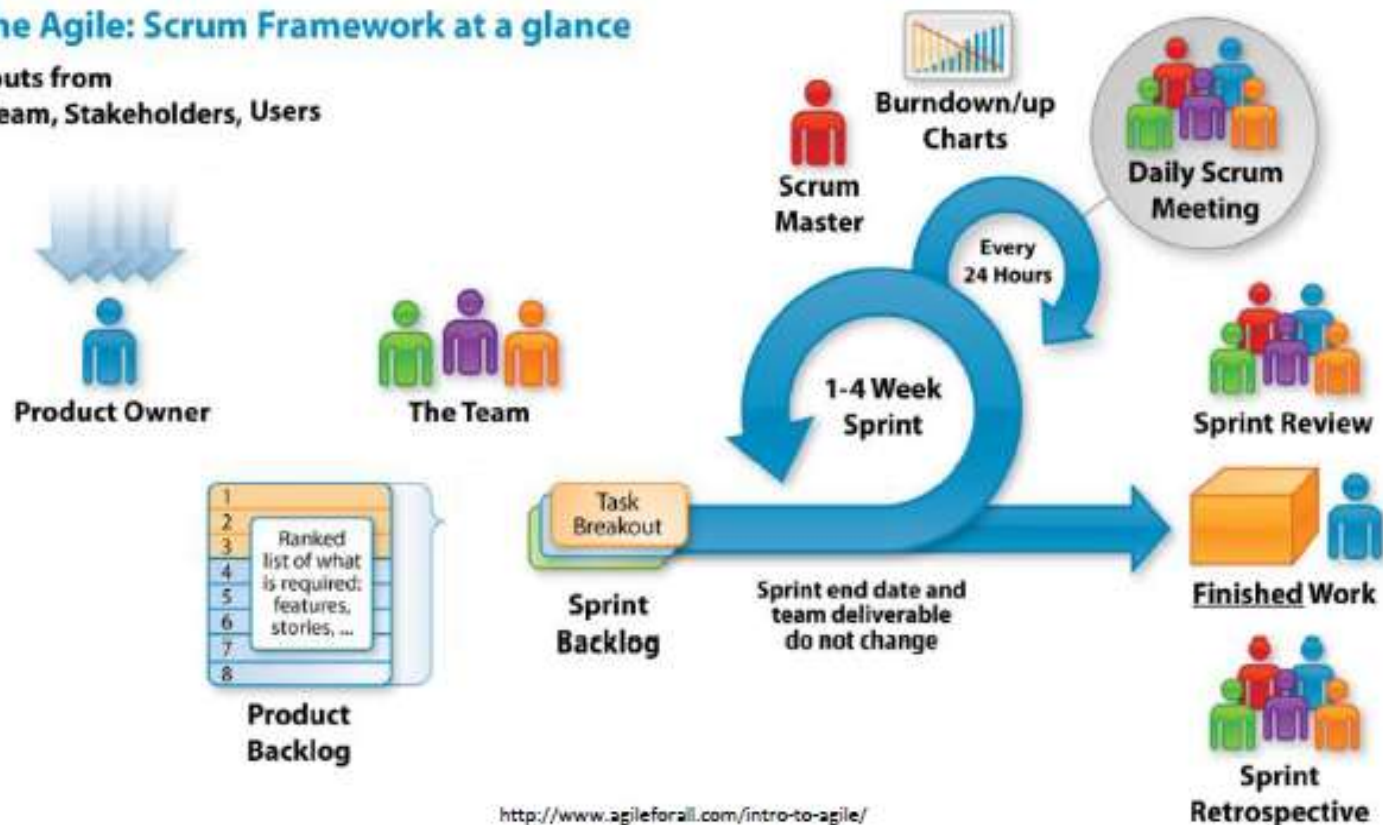


« Agile Development » : méthode

• SCRUM

The Agile: Scrum Framework at a glance

Inputs from
Team, Stakeholders, Users

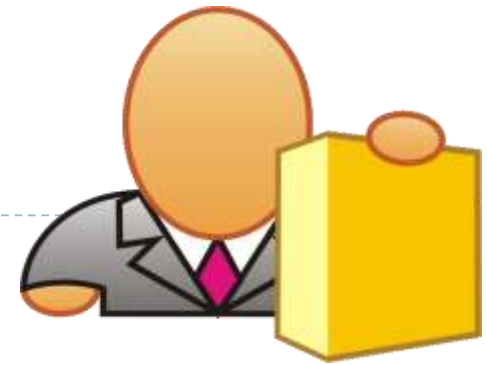


<http://www.agileforall.com/intro-to-agile/>

Scrum – Les rôles

- ▶ **Le product owner:** Le Propriétaire du produit (Product Owner) est le représentant des clients et des utilisateurs.
- ▶ **Le scrummaster: est responsable de la méthode.** Il doit s'assurer que celle-ci est comprise, et bien mise en application. Il joue aussi le rôle de facilitateur.
- ▶ Ex: Apprendre au propriétaire du produit à rédiger les composantes du carnet de produit (product backlog).
- ▶ **L'équipe:**
 - ▶ Une équipe auto-organisée choisit la façon d'accomplir son travail, sans que ce soit imposé par une personne externe.
 - ▶ **Il n'y a pas non plus de notion de hiérarchie interne :** toutes les décisions sont prises ensemble.

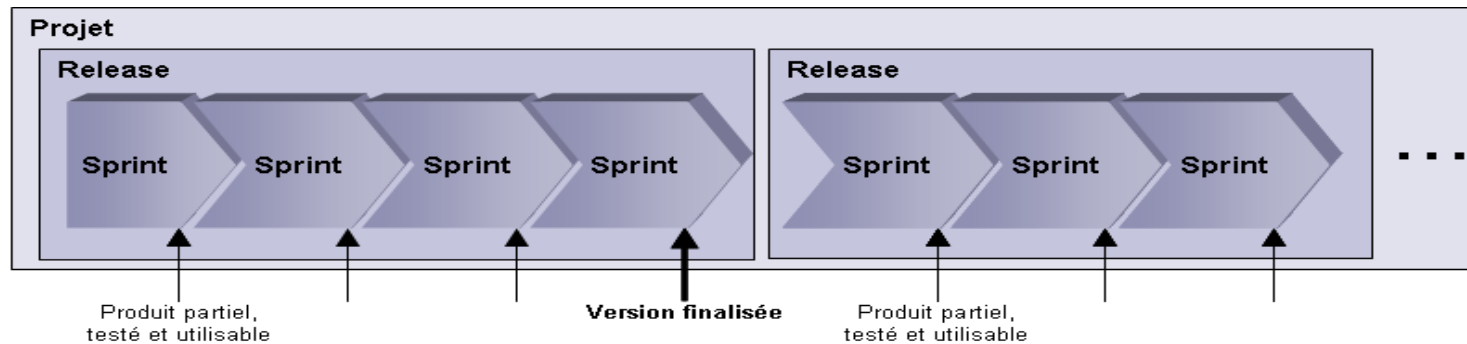
Product Owner



- **Définit les fonctionnalités du produit**
- Choisit la date et le contenu de la release
- Définit les priorités dans le product backlog en fonction de la valeur « métier »
- Ajuste les fonctionnalités et les priorités à chaque sprint si nécessaire
- **Accepte ou rejette les résultats**

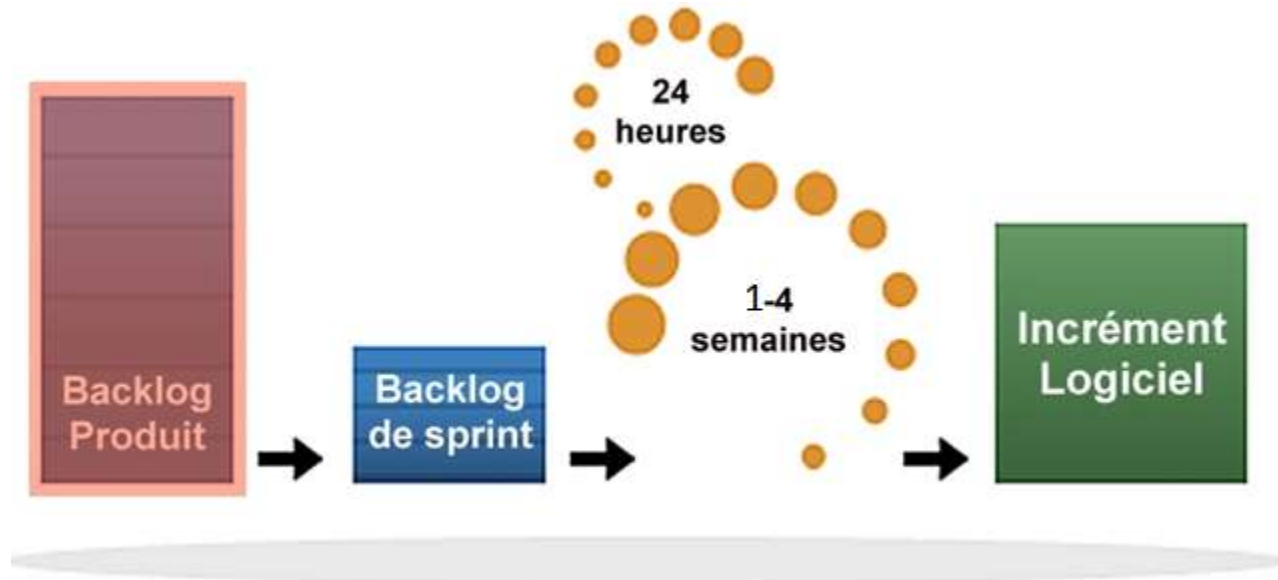


Scrum – Principes clés



- ▶ Répartition en **sprints** et en **releases**.
- ▶ Itération d'une durée fixe (de 1 à 4 semaines) = **sprint**
- ▶ Livraison d'un produit partiel fonctionnel par **sprint**
- ▶ Définition **des fonctionnalités prioritaires**: Constitution du **backlog produit** avec la participation du product owner.

Scrum – Organisation



Source : www.scrumalliance.org

1. Besoins capturés dans un **Backlog produit** (ou catalogue des besoins)
 - Besoins priorisés par le product owner
 - Besoins évalués par l'équipe (charge et risque), **vélocité** déterminée.
 - **Vélocité**: nombre de points par sprint.

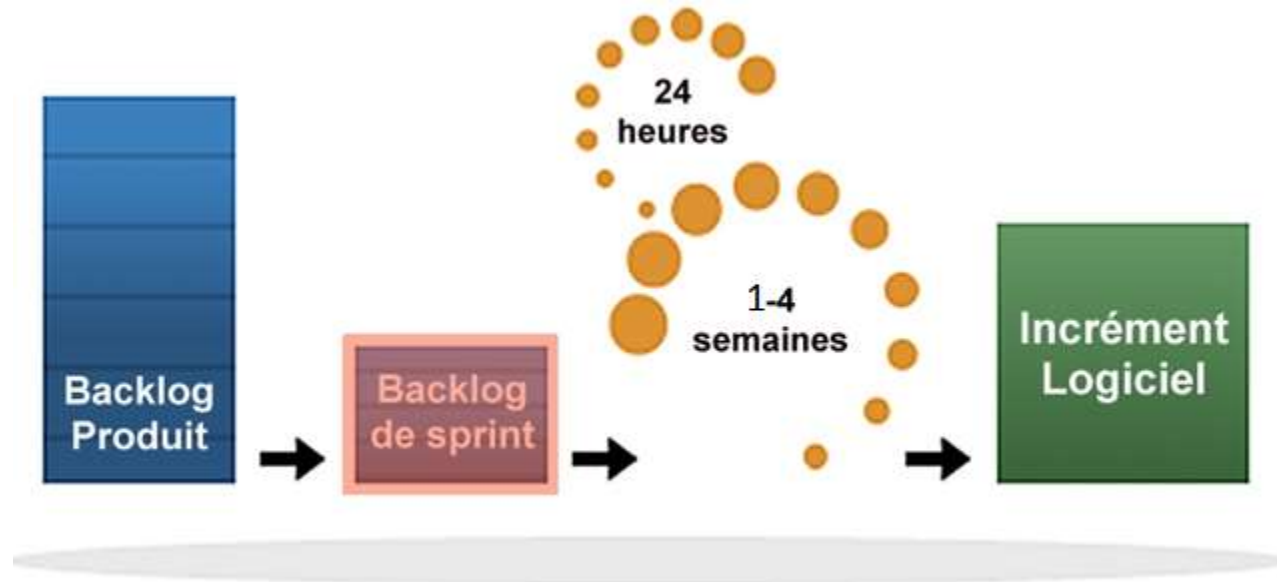
Scrum – Planifier un projet

Backlog produit - Site marchand XY					
ID_Item	Titre	Importance	Estimation	Démonstration de la fonctionnalité	Commentaires
SPRINT 1					
1	Besoin 1	130	12	XXXXXXXXXX	XXXXXXXXXX
2	Besoin 2	120	9	XXXXXXXXXX	XXXXXXXXXX
3	Besoin 3	115	20	XXXXXXXXXX	XXXXXXXXXX
SPRINT 2					
4	Besoin 4	110	8	XXXXXXXXXX	XXXXXXXXXX
5	Besoin 5	100	20	XXXXXXXXXX	XXXXXXXXXX
6	Besoin 6	95	12	XXXXXXXXXX	XXXXXXXXXX
SPRINT 3					
7	Besoin 7	80	10	XXXXXXXXXX	XXXXXXXXXX
8	Besoin 8	70	8	XXXXXXXXXX	XXXXXXXXXX
9	Besoin 9	60	10	XXXXXXXXXX	XXXXXXXXXX
10	Besoin 10	40	14	XXXXXXXXXX	XXXXXXXXXX
SPRINT 4					
11	Besoin 11	35	4	XXXXXXXXXX	XXXXXXXXXX
12	Besoin 12	25	6	XXXXXXXXXX	XXXXXXXXXX
13	Besoin 13	10	7	XXXXXXXXXX	XXXXXXXXXX
14	Besoin 14	10	11	XXXXXXXXXX	XXXXXXXXXX
15	Besoin 15	10	3	XXXXXXXXXX	XXXXXXXXXX

- Constitution du **backlog produit** par le product owner.
- Répartition en **sprints** et en **releases**.



Scrum – Organisation



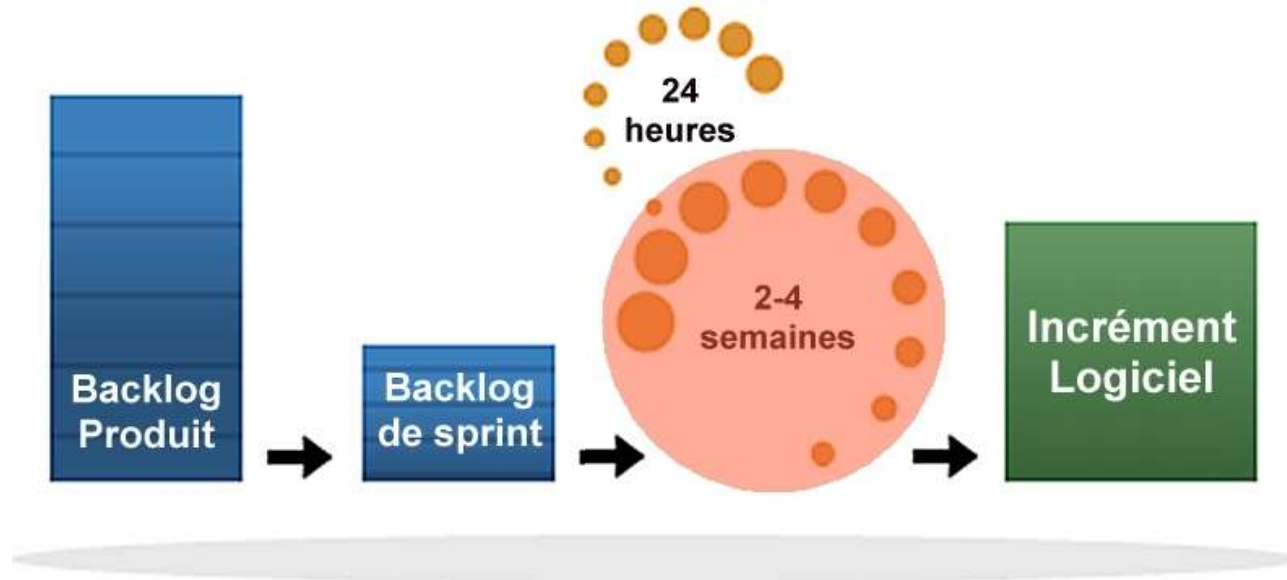
Source : www.scrumalliance.org

2. Backlog de sprint

- Extrait du backlog produit
- Besoins éclatés en tâches



Scrum – Organisation



Source : www.scrumalliance.org

3. Sprint

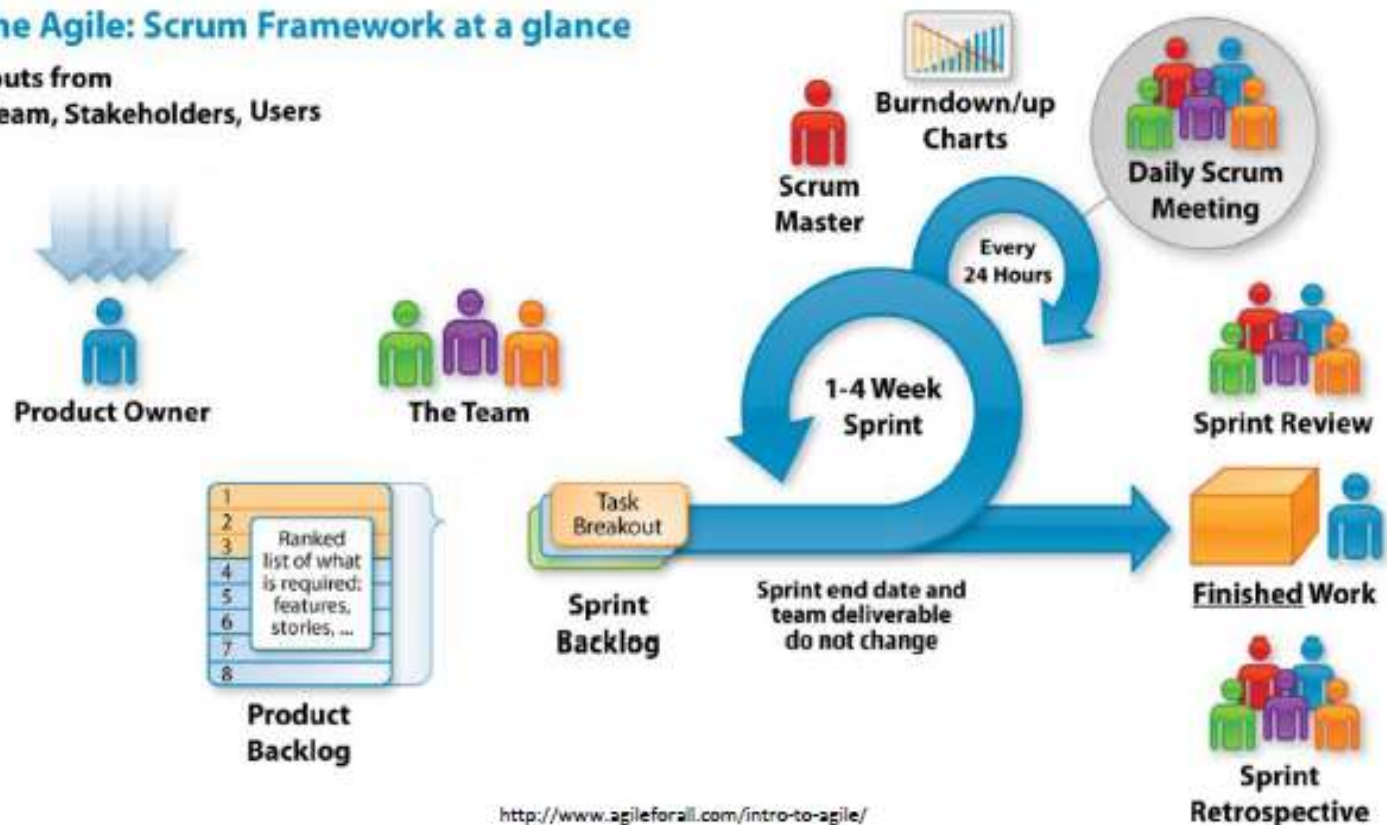
- Développement des fonctionnalités du backlog de sprint
- Il vaut mieux que aucune modification du backlog de sprint ne soit possible.

« Agile Development » : méthode

• SCRUM

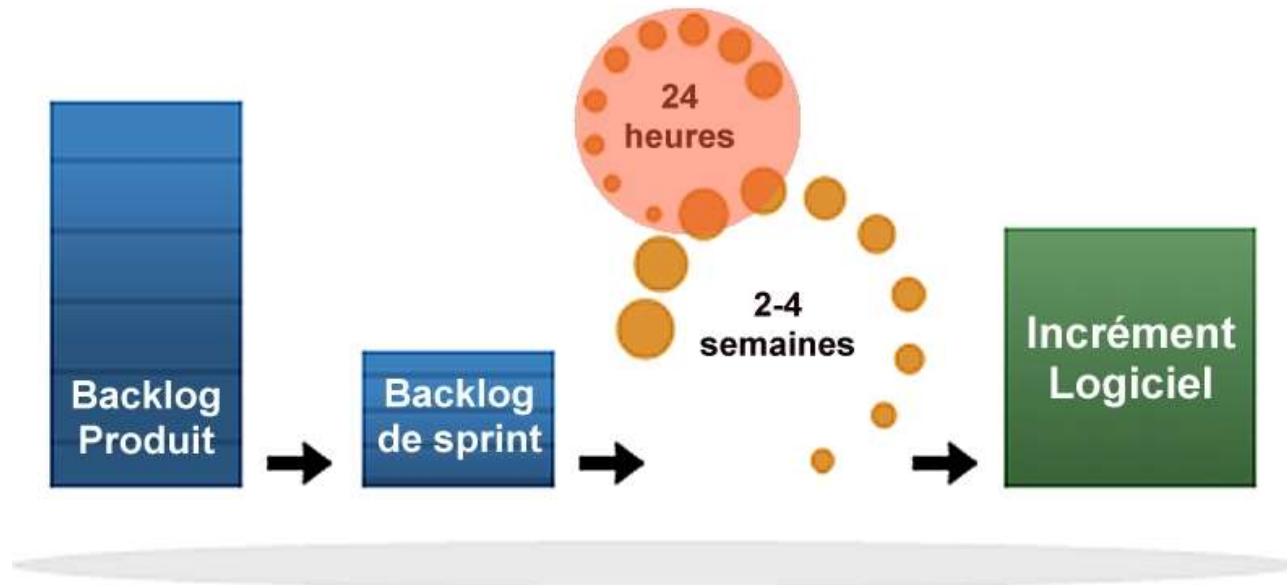
The Agile: Scrum Framework at a glance

Inputs from
Team, Stakeholders, Users



<http://www.agileforall.com/intro-to-agile/>

Scrum – Organisation



Source : www.scrumalliance.org

4. Mêlée quotidienne- daily scrum meeting

- Point de contrôle quotidien de l'équipe
- Interventions régulées – 2 min. par personne

Mêlée quotidienne (Daily scrum)

- 15 minutes, tous les jours
- Trois questions pour chacun
 - Qu'avez-vous fait hier
 - Qu'allez-vous faire aujourd'hui
 - Quels sont vos problèmes
- Le reste à faire pour le Sprint: burndown chart

Remarque: Dans XP, c'est le stand up meeting

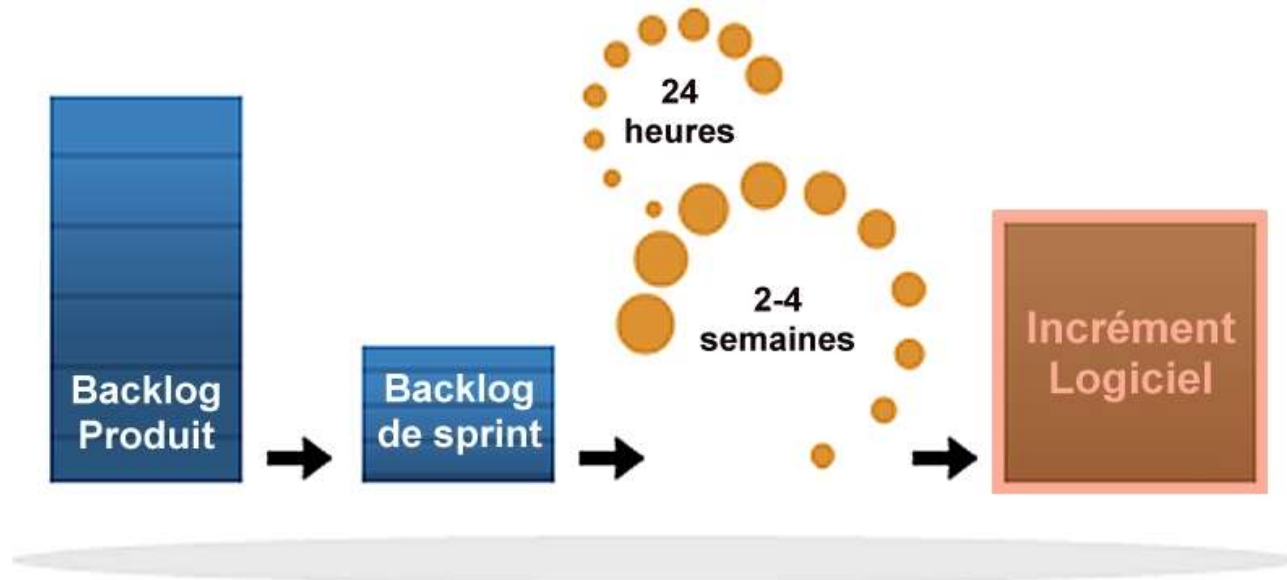


Daily scrum



On peut utiliser des outils de suivi, exemple: trello (To do, done, doing).

Scrum – Organisation

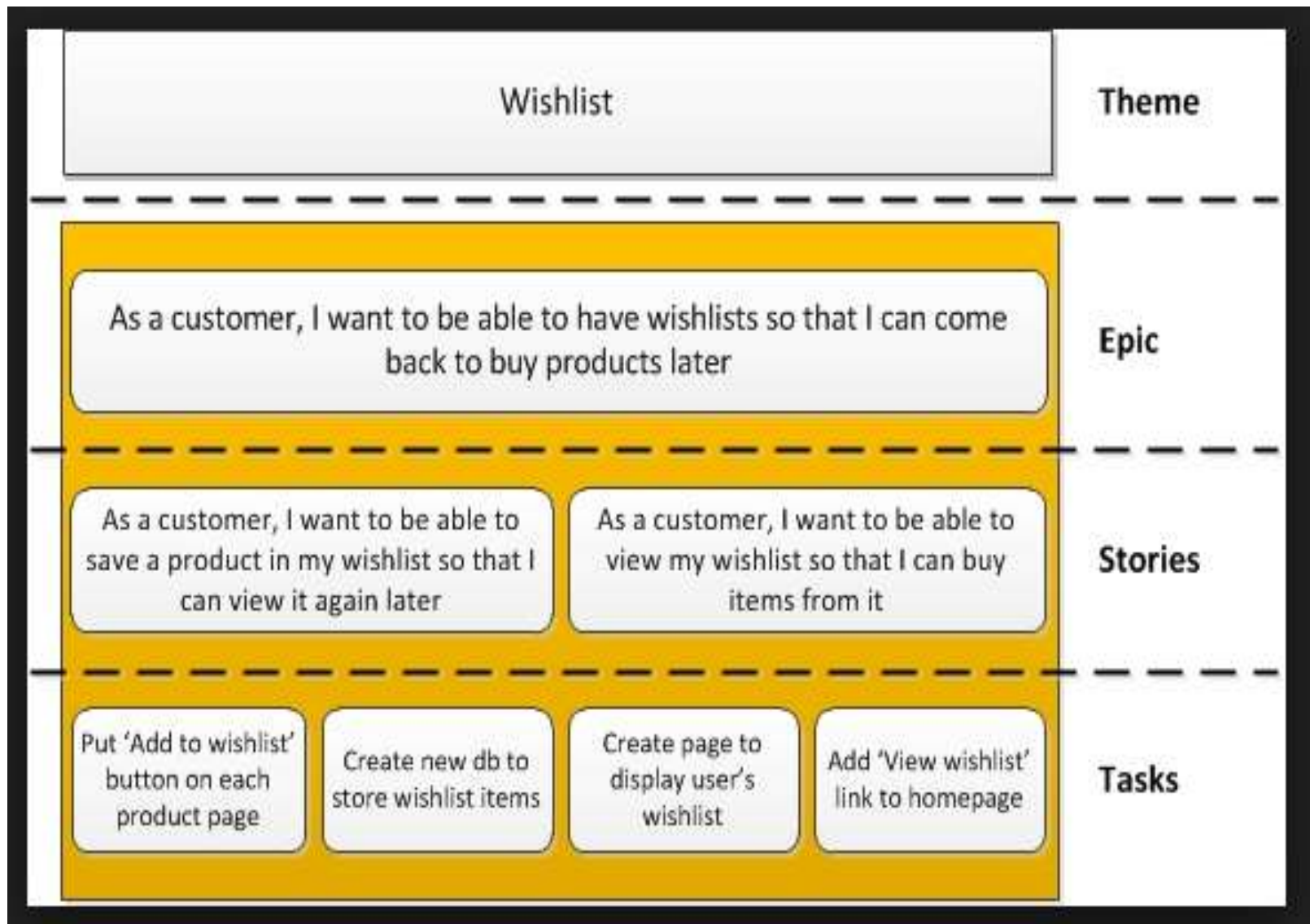


Source : www.scrumalliance.org

5. Incrément logiciel : livré au product owner à la fin du sprint.



Principaux Artéfacts



Theme ou Feature.

Une équipe qualifie **une story d'epic** pour indiquer qu'elle devra être décomposée en plusieurs stories (on ne pourra pas l'intégrer dans un sprint dans sa taille actuelle).

Exemple de User Stories :

En tant qu'utilisateur
Je souhaite pouvoir créditer
mon compte en ligne
Afin de pouvoir parier

Product Backlog

VS

Sprint Backlog

Product Backlog

DEEP

Technique

- * Detailed
- * Estimated
- * Emergent
- * Prioritized

Userstory#1

Userstory#2

Userstory#3

Userstory#4

Userstory#5

Userstory#6

Sprint Backlog

Sprint #1

Userstory#1

Userstory#2

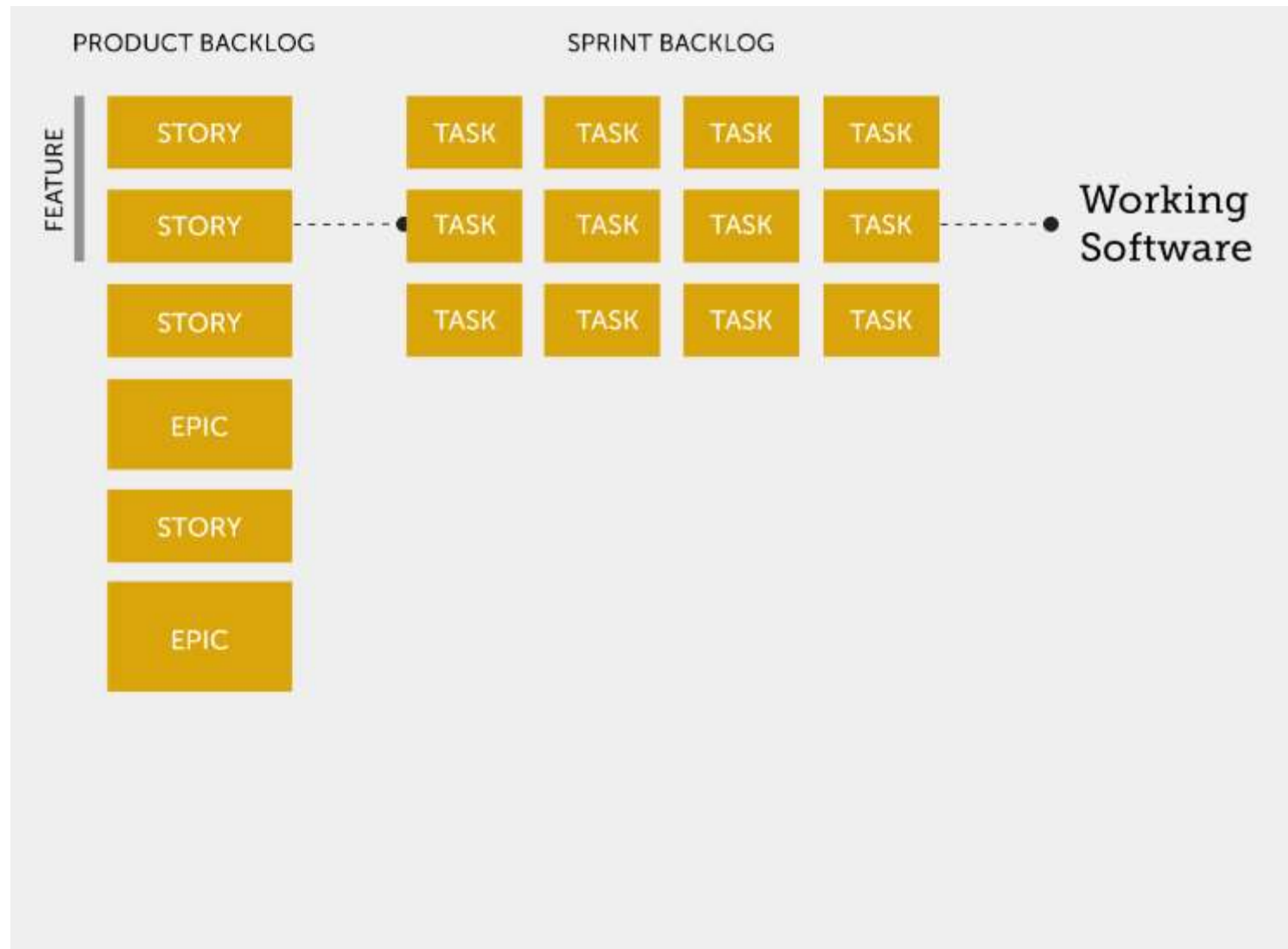
Userstory#3

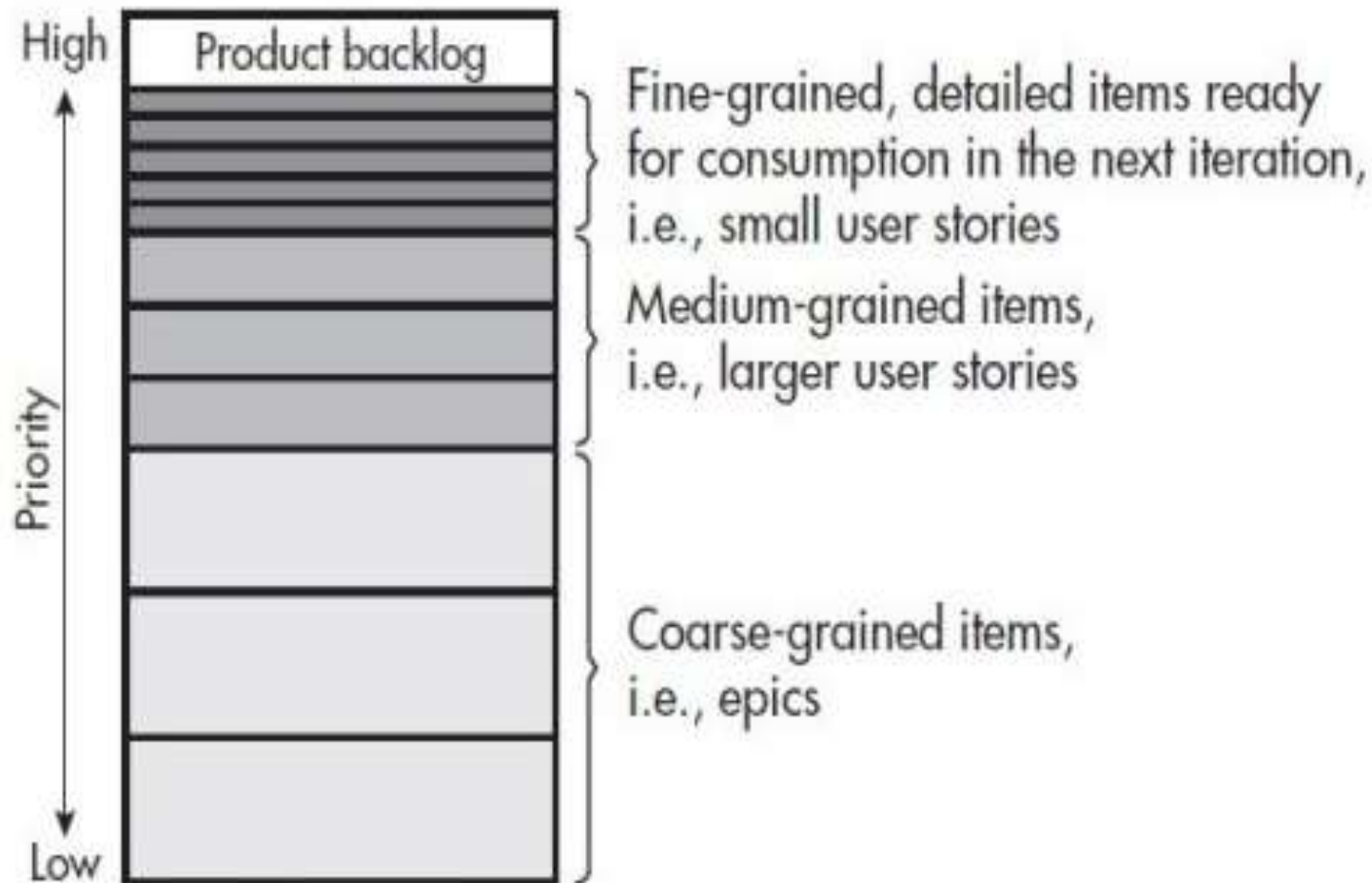
Sprint #2

Userstory#4

Sprint #3

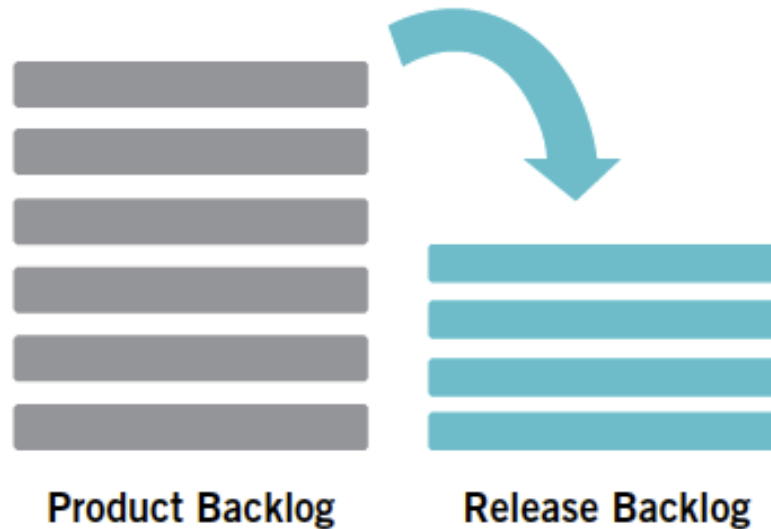
Emergent: Evolutif





Une équipe qualifie **une story d'epic** pour indiquer qu'elle devra être décomposée en plusieurs stories (on ne pourra pas l'intégrer dans un sprint dans sa taille actuelle).

-
- **Le release backlog** est un sous-ensemble du Product backlog qui devrait être livré dans la prochaine Livraison,

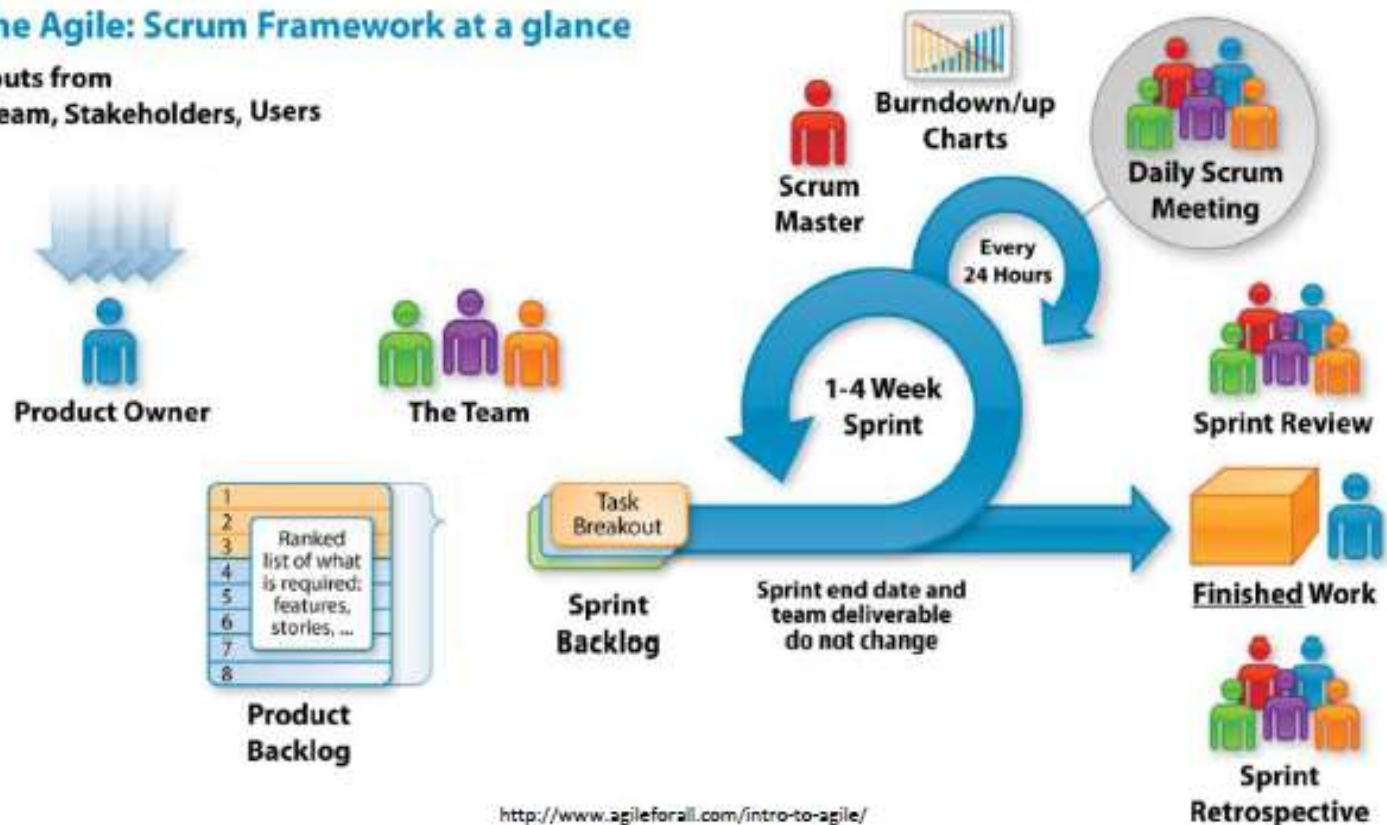


« Agile Development » : méthode

• SCRUM

The Agile: Scrum Framework at a glance

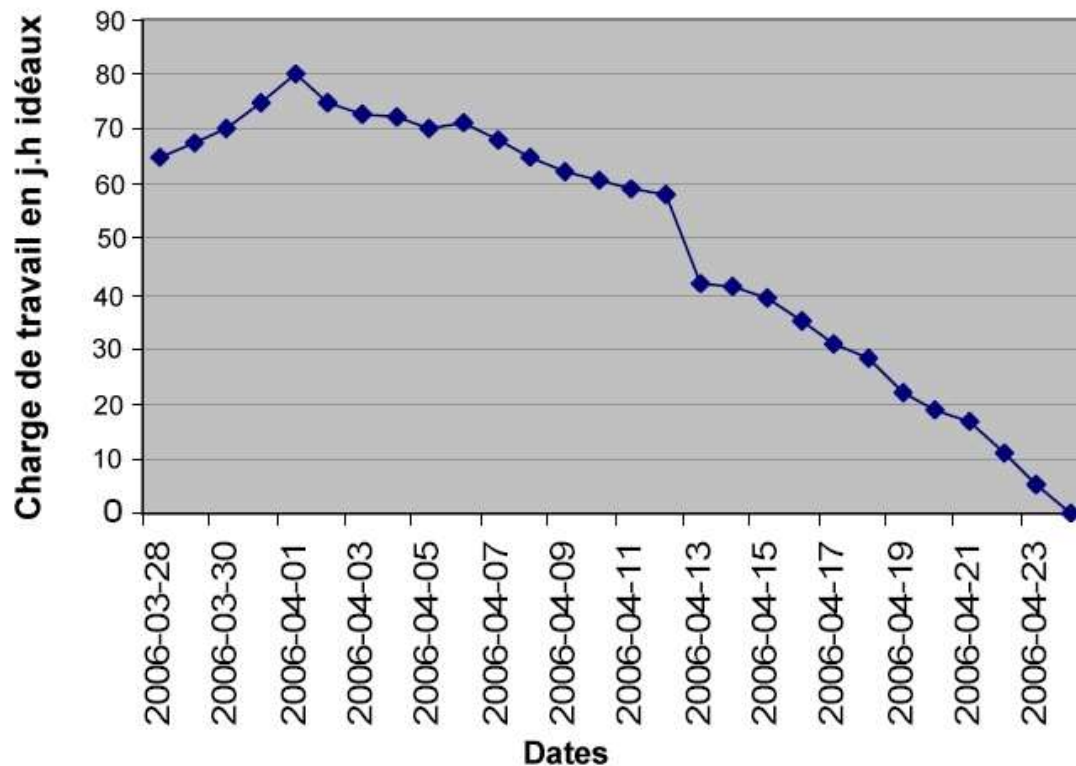
Inputs from
Team, Stakeholders, Users



<http://www.agileforall.com/intro-to-agile/>

Indicateurs de projet: Sprint burndown chart

- **Le burndown chart:** (graphique d'avancement) permettent de visualiser graphiquement l'avancement du travail durant un sprint.



Source : « Summary of Scrum », Signifikant Svenska A.B., 2007

Scrum – Principes clés

- ▶ Met l'accent sur:

- ▶ Auto-organisation de l'équipe
- ▶ Pouvoir de décision donné à l'équipe
- ▶ Réunions quotidiennes
- ▶ Livrer un logiciel fonctionnel - démonstration du résultat du sprint
- ▶ Planning adaptatif

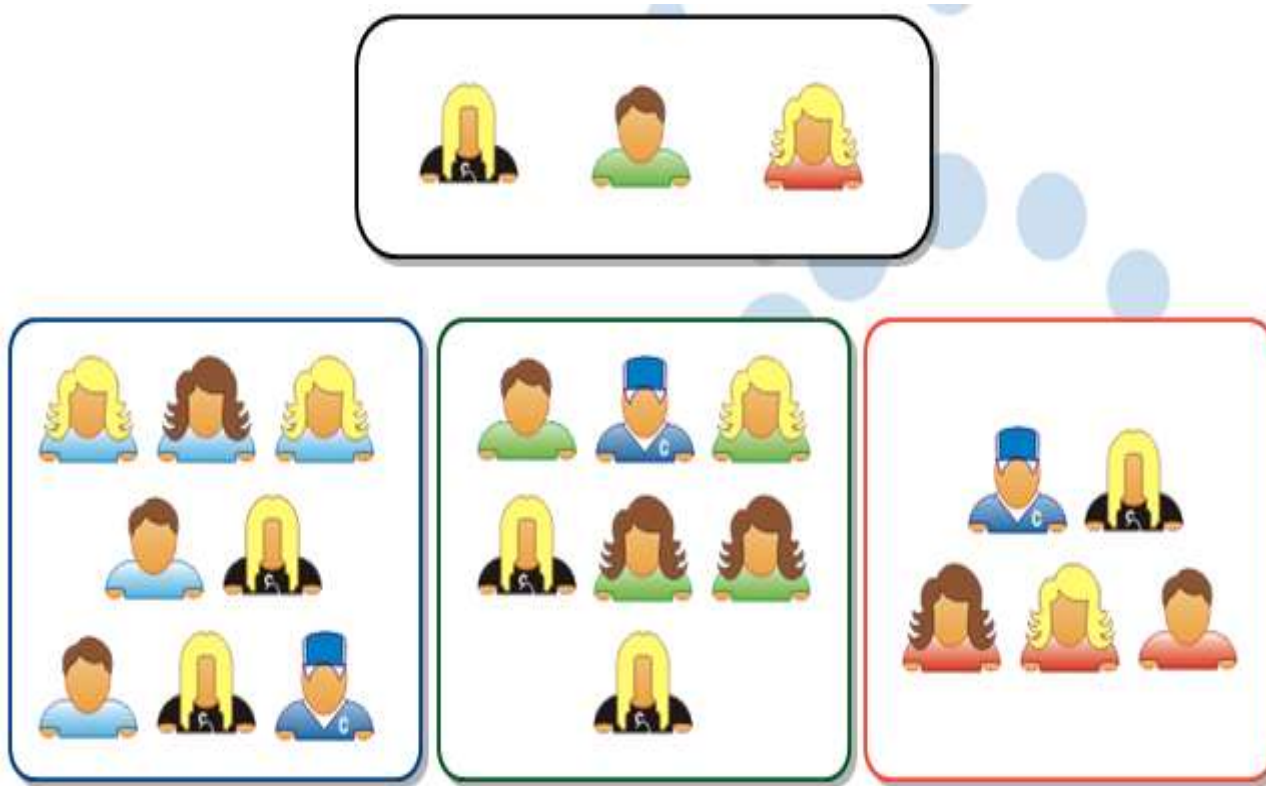


Scrum

- ▶ Adaptation de la méthode:
 - ▶ Commencer par une équipe Scrum standard
 - ▶ Création de plusieurs équipes: **Scrum des scrums**



Scrum des Scrum



Scrum a été utilisé sur des projets de 500 personnes.

Scrum – Ingénierie logicielle

- ▶ **Scrum** est une méthode **de gestion de projet**
- ▶ Doit être complétée par **des techniques d'ingénierie logicielle**
- ▶ Complémentaire avec Extreme Programming :
 - ▶ Test Driven Development
 - ▶ Intégration continue



Questions

- ▶ Quels sont les artefacts au niveau de la méthode scrum?

Scrum a été utilisée par:

- Microsoft
- Yahoo
- Google
- Electronic Arts
- High Moon Studios
- Lockheed Martin
- Philips
- Siemens
- Nokia
- Capital One
- BBC
- Intuit
- Intuit
- Nielsen Media
- First American Real Estate
- BMC Software
- Ipswitch
- John Deere
- Lexis Nexis
- Sabre
- Salesforce.com
- Time Warner
- Turner Broadcasting
- Oce





eXtreme Programming (XP)



Qu'est-ce que XP ?

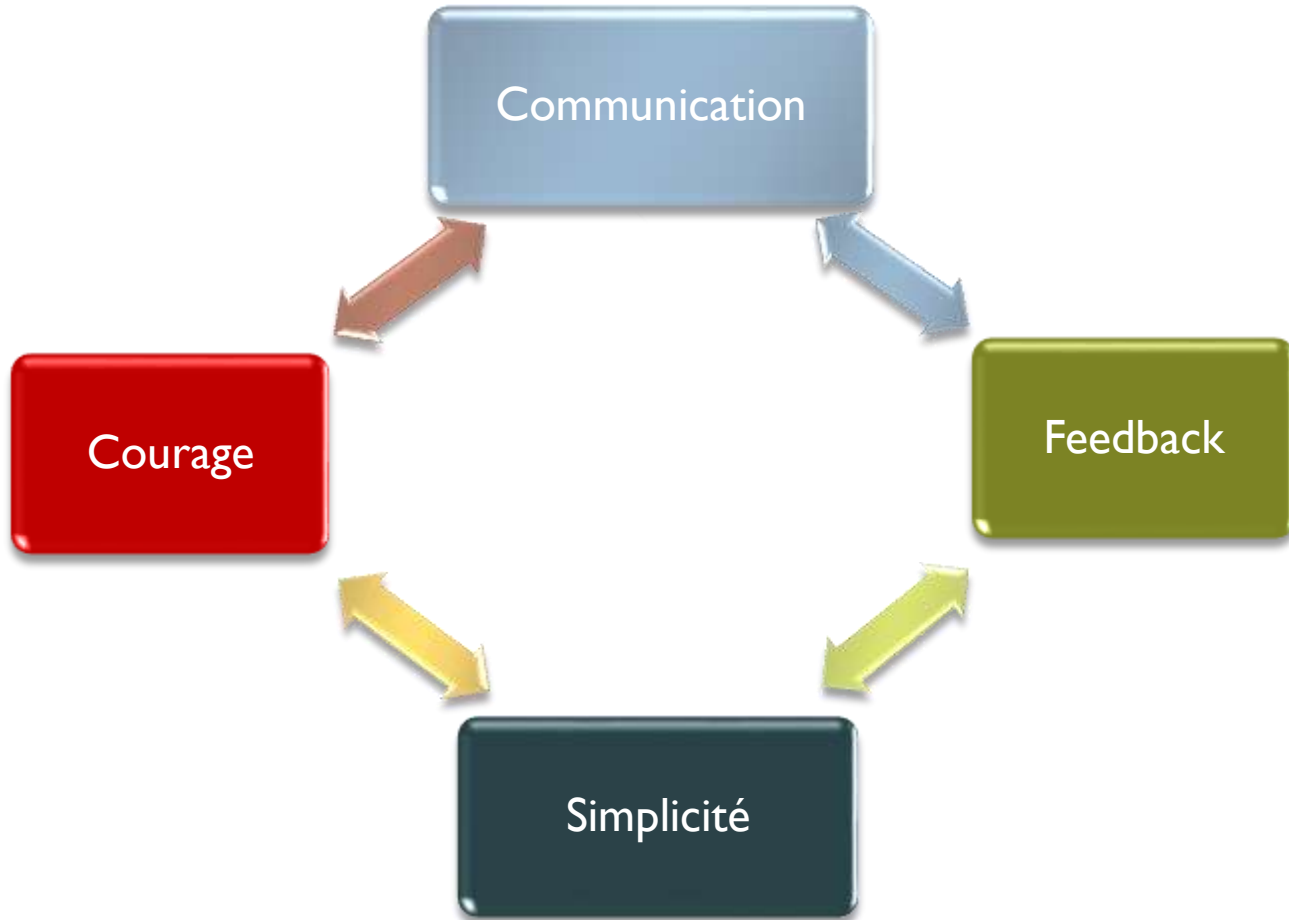
- ▶ Méthode de **développement**
- ▶ Rassemblement **de bonnes pratiques** déjà connues et utilisées
- ▶ Une des méthodes agiles
- ▶ Méthode proposée par **Kent Beck en 1999**

eXtreme Programming

- ▶ Méthodologie de développement basée sur des valeurs, principes et pratiques
- ▶ Equipe entre 2 et 10
- ▶ Propose des pratiques d'ingénierie comme:
 - ▶ Le développement en binôme
 - ▶ Test Driven Development
 - ▶ Intégration continue
- ▶ Propose des pratiques de gestion de projet



Les quatre valeurs d'XP



Mise en œuvre dans **les pratiques** au quotidien

eXtreme Programming : Valeurs

► Communication

- **Verbale**
- Dans l'équipe
- Avec le client
- Facilitée **par la colocalisation** de l'équipe



eXtreme Programming : Valeurs

- ▶ Simplicité
 - ▶ « You're not gonna need it »
 - ▶ Cherche la solution la plus simple qui convient au problème du jour.
- ▶ Feedback
 - ▶ Des tests unitaires, fonctionnels
 - ▶ Du client
 - ▶ Revue avec le client toutes les deux à trois semaines
 - ▶ De l'équipe
 - ▶ Grace à la communication continue

eXtreme Programming : Valeurs

- ▶ **Courage**

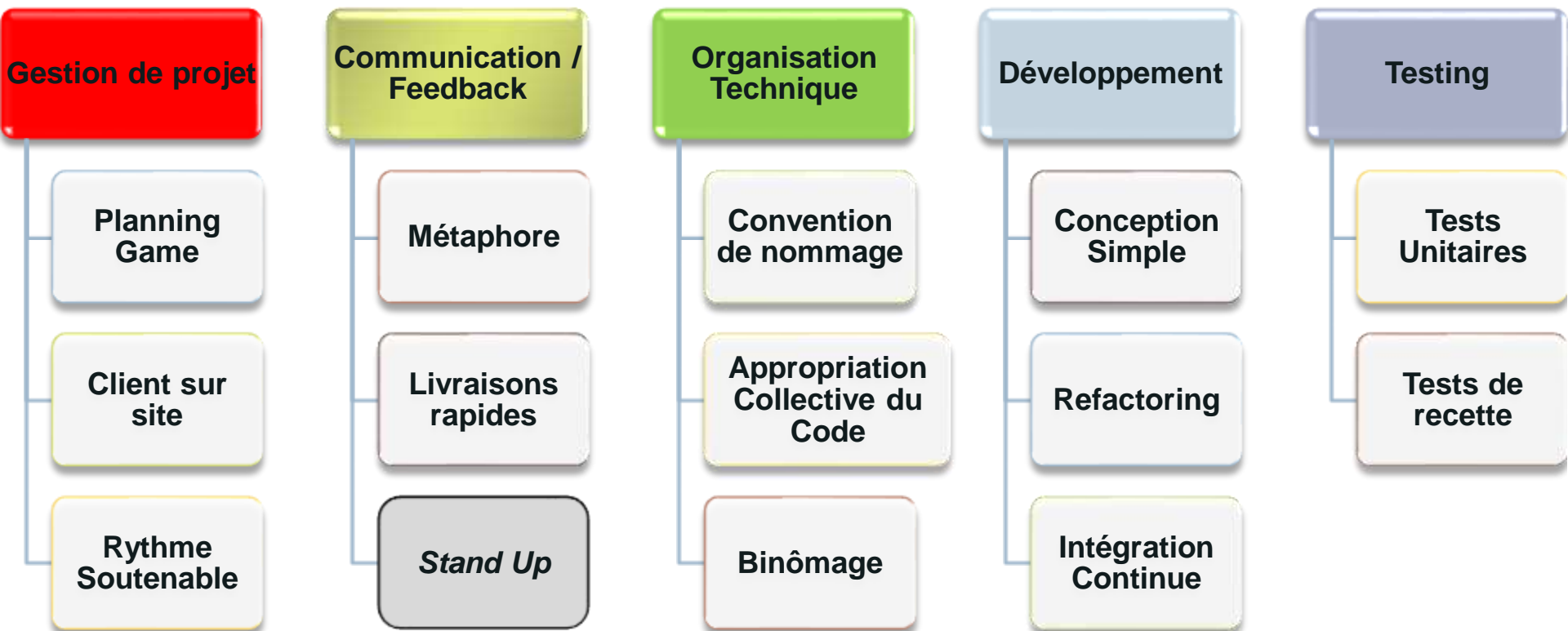
- ▶ De s'attaquer aux problèmes tout de suite
- ▶ D'appliquer les valeurs XP
- ▶ De jeter du code lorsque nécessaire



Modèle de développement

- ▶ Modèle de développement **Itératif et INCREMENTAL**
- ▶ **Petites livraisons** : entre 1 et 2 mois
- ▶ Itérations : entre 1 à 4 semaines

Les 13 pratiques d'eXtreme Programming

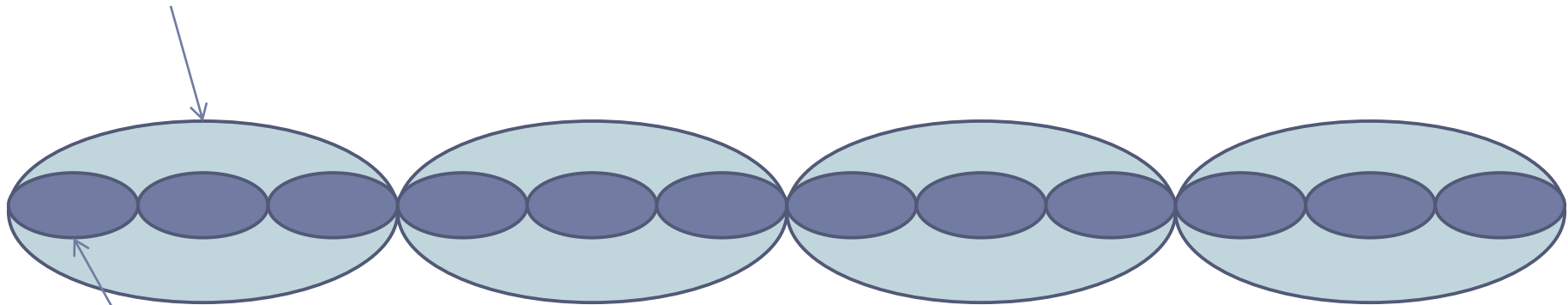


Planification itérative: Planning Game

Itération: 1 à 4 semaines

Livraison: toutes les deux ou trois itérations

Livraisons



Itérations

Décomposition du projet en livraisons et itérations

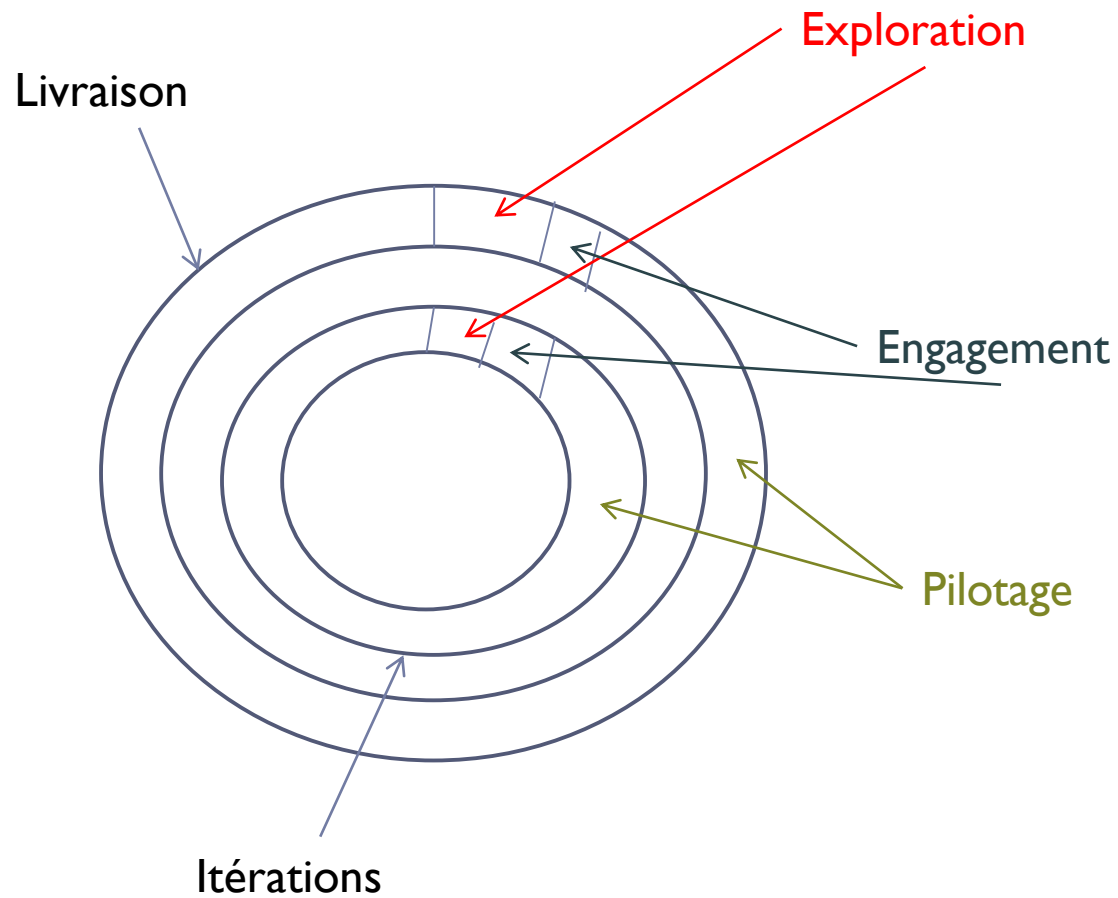
Pratiques: Planification itérative

Rythme

- Environ 2 ou 3 itérations entre deux livraisons
- Pour un projet de six ou sept mois
 - Itérations de deux semaines environ

Séances de planification (« Planning Game »)

- Livraisons et itérations sont des cycles imbriqués
- Deux niveaux de granularité pour le pilotage
 - Livraisons → Fonctionnalités
 - Itérations → Tâches à réaliser par les développeurs
- Ces deux cycles sont découpés en trois phases
 - ❶ Exploration, identifier le travail et estimer son coût
 - ❷ Engagement, sélectionner le travail pour le cycle en cours
 - ❸ Pilotage, contrôler la réalisation de ce qui est demandé



Les trois phases des cycles de livraisons et d'itérations

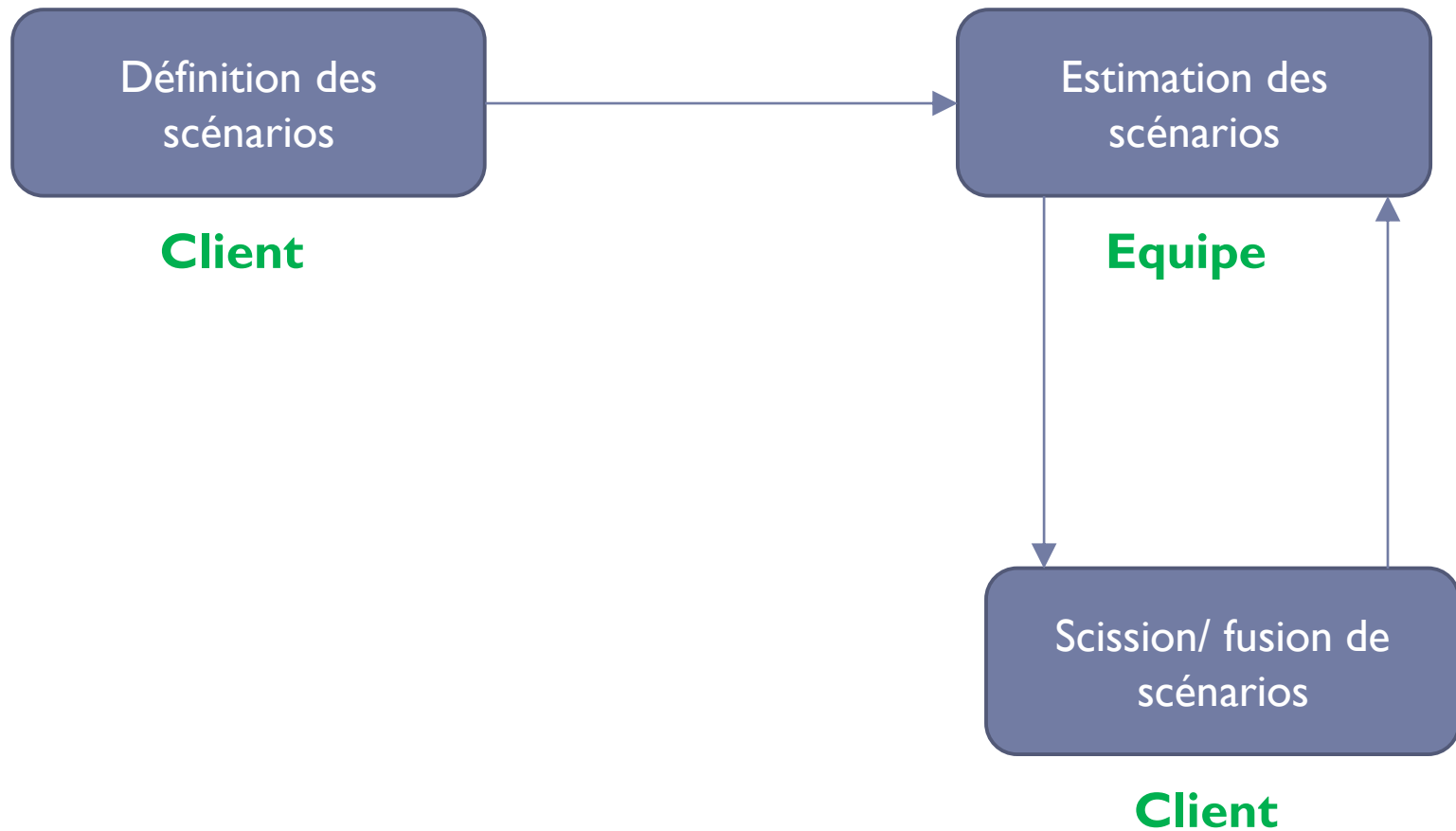
Planning Game



Planification des livraisons

Exploration (de plusieurs jours à quelques heures)

- Définir les scénarios client
 - Granularité fine
 - Doit être testable
 - Généralement notés sur des fiches A5
 - Plus simples à manipuler
- Estimer les scénarios client
 - Attribution d'un nombre de « points »
 - Tenir compte du codage des tests et de la validation
 - En cas d'inconnue technique → prototypage **rapide**
 - Scinder/Fusionner des scénarios si nécessaire

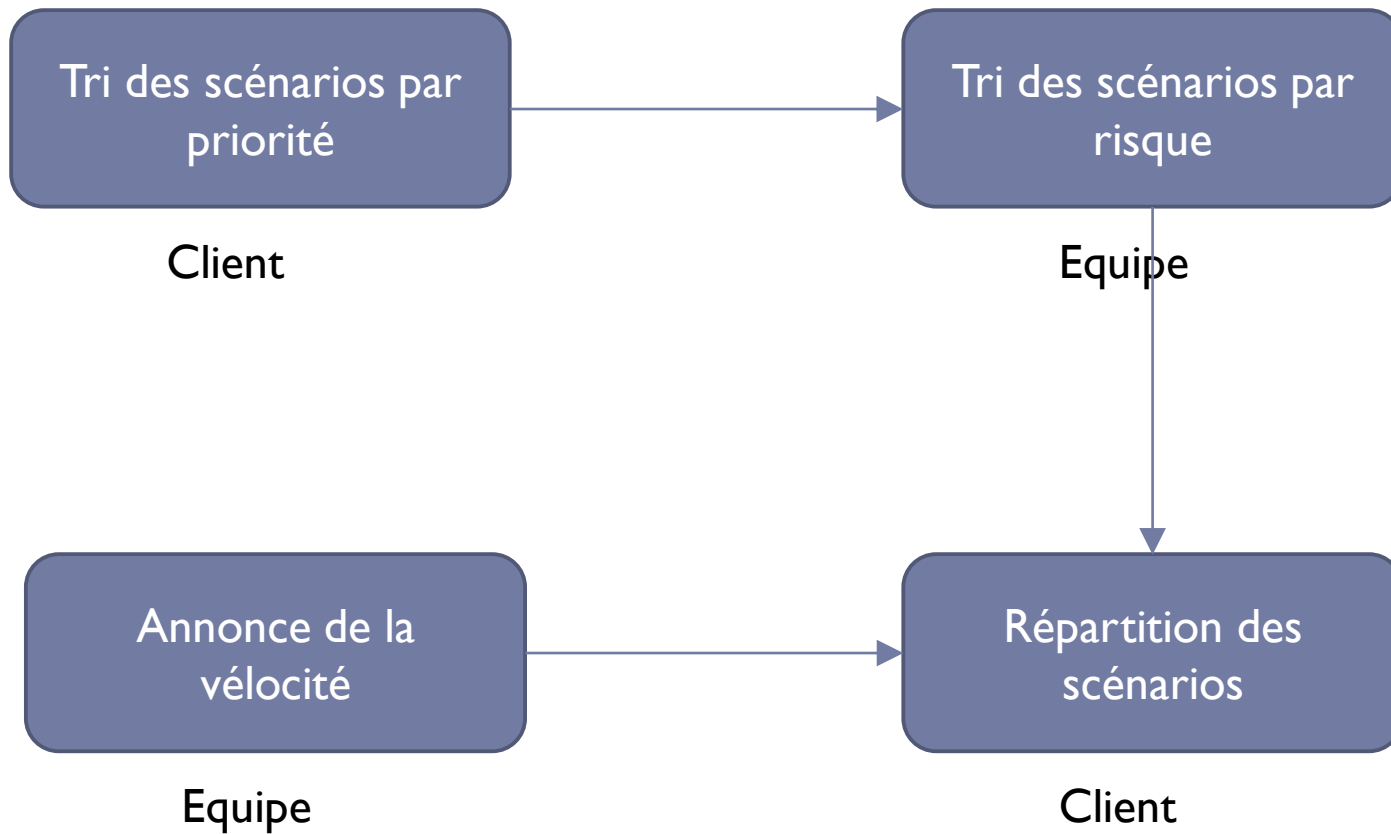


La phase d'exploration du cycle des livraisons

Planification des livraisons

Engagement (quelques heures seulement)

- ▶ Trier les scénarios (9 tas)
 - ▶ Par priorité (client) : Indispensable, Essentiel, Utile
 - ▶ Par risque (programmeurs) : Fort, Moyen, Faible
- ▶ Annoncer la « vélocité » de l'équipe
 - ▶ Nombre de points que peut traiter l'équipe en une itération
 - ▶ Pour la première itération le coach fourni sa propre estimation
- ▶ Répartir les scénarios parmi les livraisons à venir
 - ▶ Le client « achète » les scénarios en fonction de la vélocité
 - ▶ Création du plan de livraison *provisoire*



La phase d'engagement du cycle des livraisons

Planification des livraisons

Pilotage (jusqu'à la date de livraison)

- ▶ Suivre les tests de recette
 - ▶ Nombre de tests réalisés par le client et les testeurs
 - ▶ Nombre de tests validés
- ▶ Gérer les défauts
 - 1 Intégrer au mécanisme général de planification
 - 2 Priorité absolue

Planification des Itérations

Exploration

- Analyser les scénarios pour les scinder en tâches
- Questions au client et discussion en sa présence
- Tâche : réalisable par un binôme en une ou deux journées

Engagement

- Choisir et estimer des tâches
- Equilibrage des choix dans l'équipe
- Fusion/Scission de tâches (si nécessaire)
- Avancement/Report de scénarios (si nécessaire)
- Production du plan d'itération

Planification des Itérations

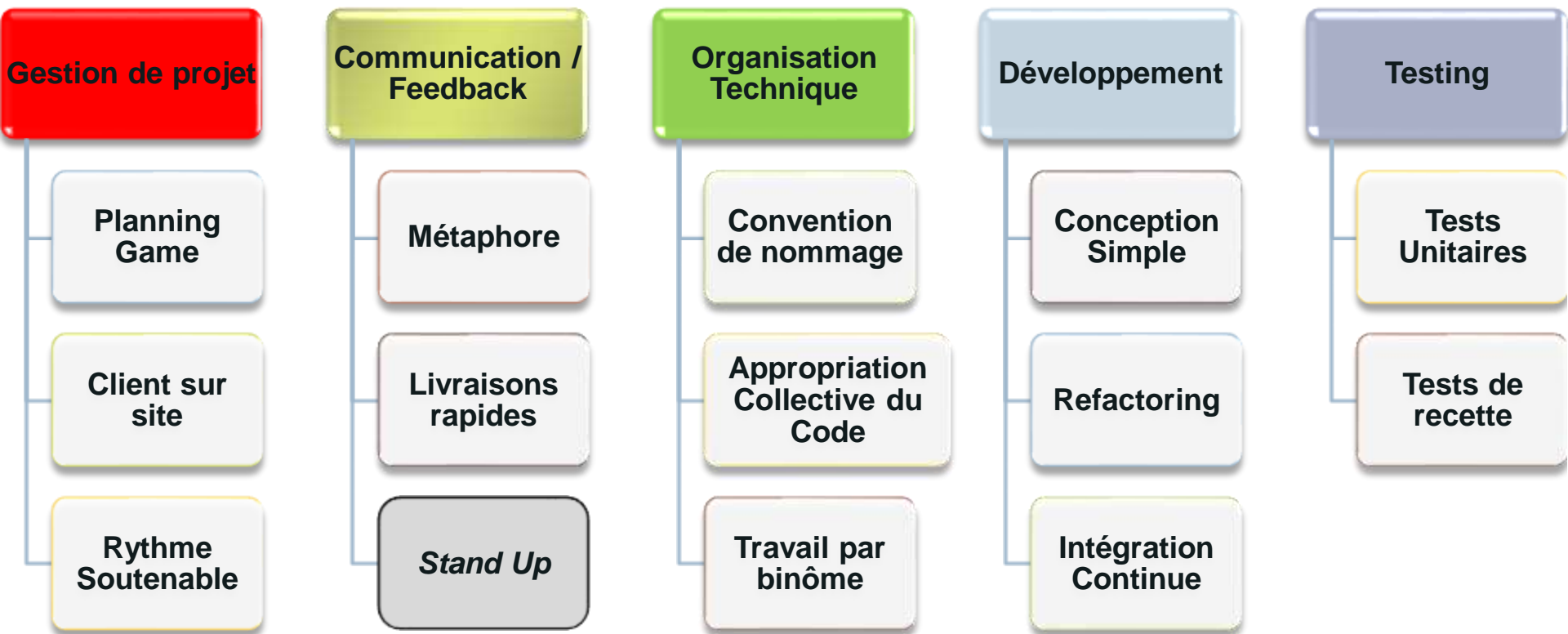
Pilotage

- Réalisation des tâches (mise à jour du plan d'itération)
- Tournée du Tracker (au moins deux fois par semaine)
 - Déceler tout dérapage au plus tôt
 - Prendre des mesures correctives (alléger une charge...)
- Stand-up meetings
 - Chaque matin un point sur la situation ou les difficultés
 - Permet d'organiser la journée (binômes...)

Et ensuite ?

- Petite livraison informelle au client de l'équipe
- Mise à jour des coûts des scénarios réalisés

Les 13 pratiques d'eXtreme Programming



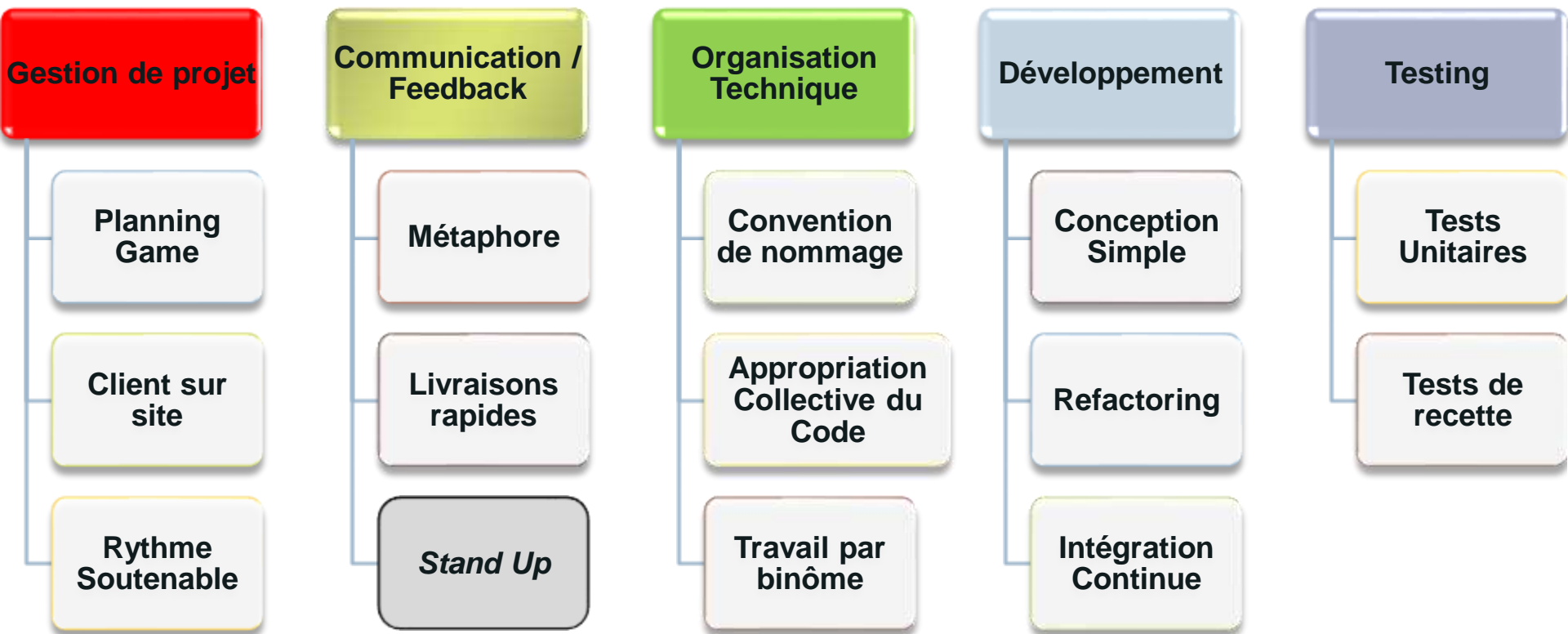
Client sur site

- ▶ Présence de représentant du client **pendant toute la durée du projet** pour répondre aux questions de l'équipe.
- ▶ Il doit avoir les connaissances de l'utilisateur final et avoir une vision globale du résultat à obtenir

Rythme soutenable

- ▶ Rythme régulier (vélocité= nombre de points traités durant une itération)
- ▶ L'équipe maintient sa capacité à rester efficace en ne surchargeant pas le planning et en s'aménageant des horaires raisonnables.
- ▶ Pas d'heures sup deux semaines de suite

Les 13 pratiques d'eXtreme Programming



Les équipes XP emploient des métaphores.

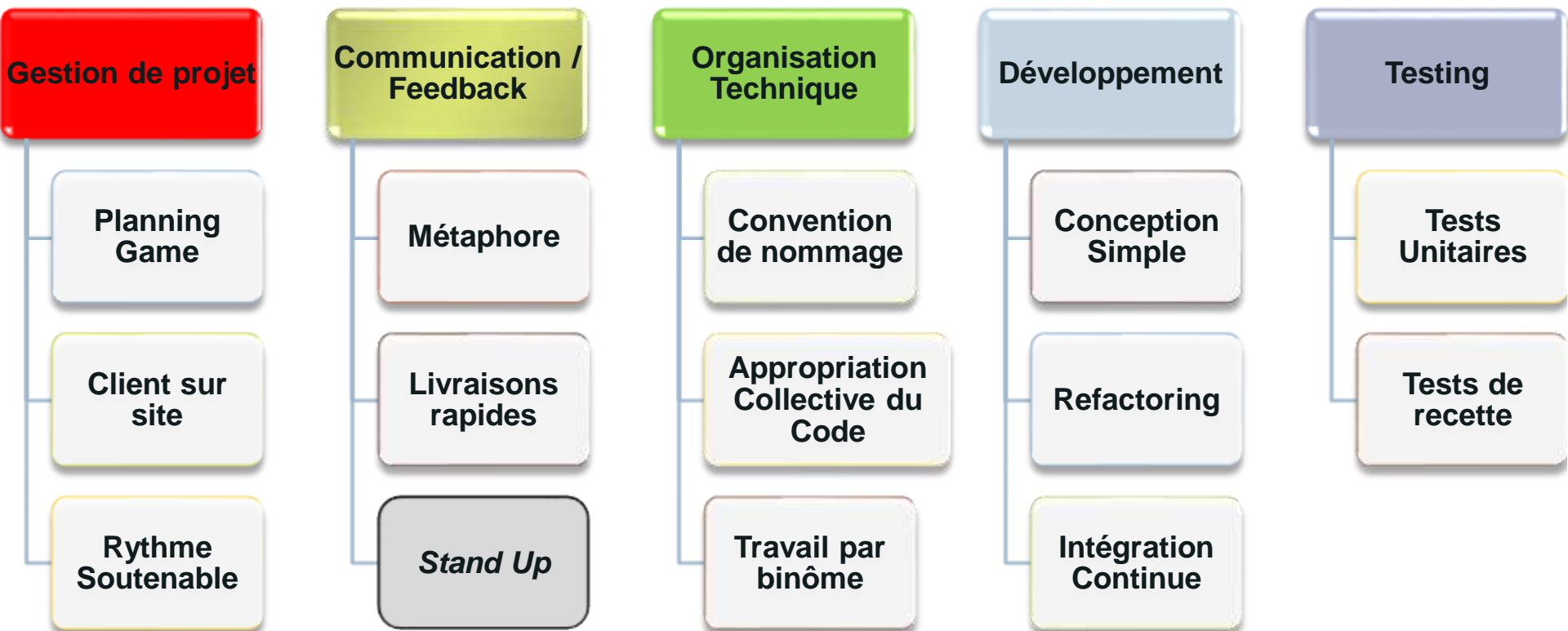
- La métaphore est une abstraction de niveau élevé.
- La métaphore ne doit pas être une simplification abusive du concept initial, mais doit permettre de simplifier l'expression d'une chose complexe comme, par exemple, clarifier des fonctionnalités.
- Dans XP, la plupart des métaphores concernent le fonctionnement de l'application ou l'architecture.
- La métaphore est utilisée pour décrire le système et son fonctionnement avant la livraison de fonctionnalités réelles, afin de le rendre compréhensible par les non informaticiens et les utilisateurs impliqués dans le développement.

Exemple:

Une description d'un système de récupération d'information à base d'agents.

« Ce programme fonctionne comme une ruche d'abeilles, qui font sortir du pollen et le ramènent à la ruche »

Les 13 pratiques d'eXtreme Programming



Convention de codage

Le code doit respecter une convention de nommage et de présentation afin d'être lisible par tous les membres de l'équipe.

Propriété Collective du Code

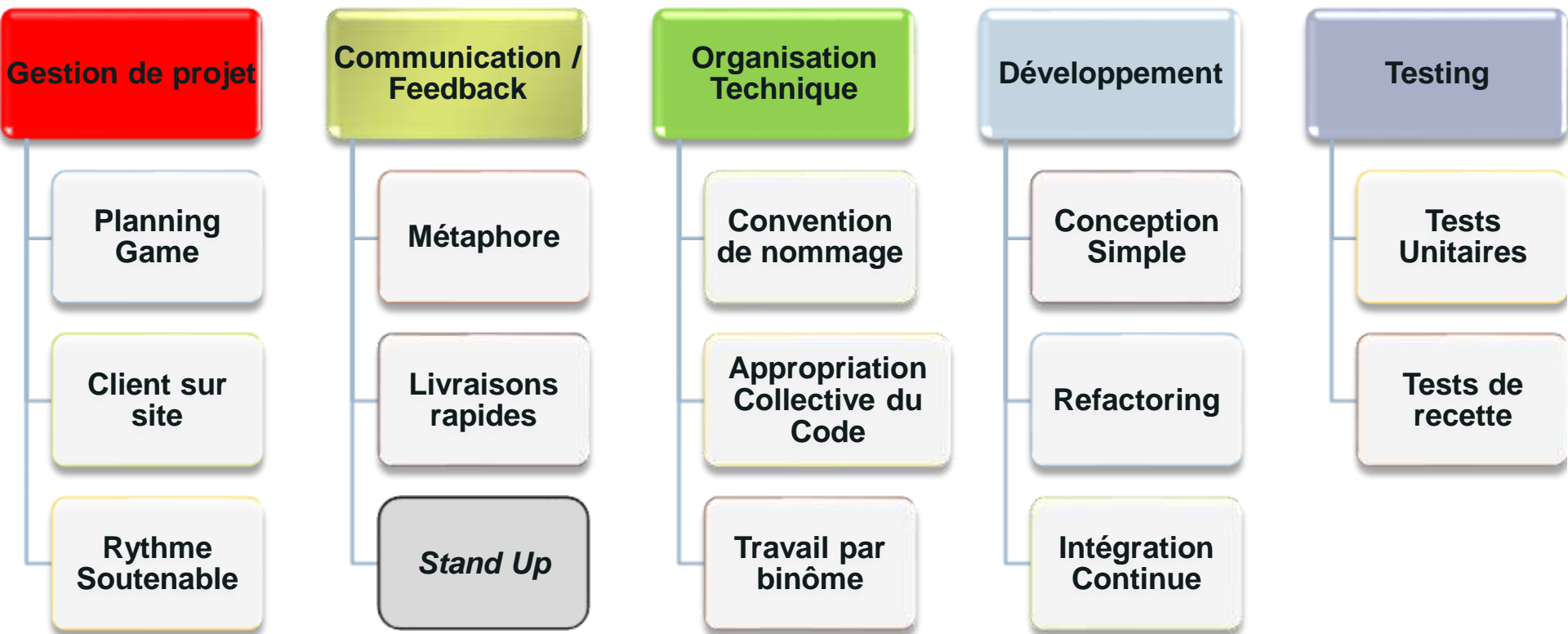
L'application n'est pas découpée en zones réservées à chaque développeur (IHM, BD, etc): chacun est susceptible de travailler sur toutes les parties de l'application.

Programmation en binôme

- ▶ La programmation se fait par deux:
 - ✓ **Driver (pilote)** : tient le clavier, travaille sur la portion du code à écrire .
 - ✓ **Partner (copilote)** : pour l'aider en suggérant des nouvelles possibilités ou en décelant d'éventuels problèmes
 - ✓ Partage des connaissances
- ▶ Les développeurs changent fréquemment de partenaire ce qui permet d'améliorer **la connaissance collective** de l'application et **d'améliorer la communication** au sein de l'équipe.



Les 13 pratiques d'eXtreme Programming



Une conception simple

- ▶ Le système répond aux scénarios du client.
- ▶ Aucune anticipation des problèmes futurs
- ▶ Conception rudimentaire: squelette de conception

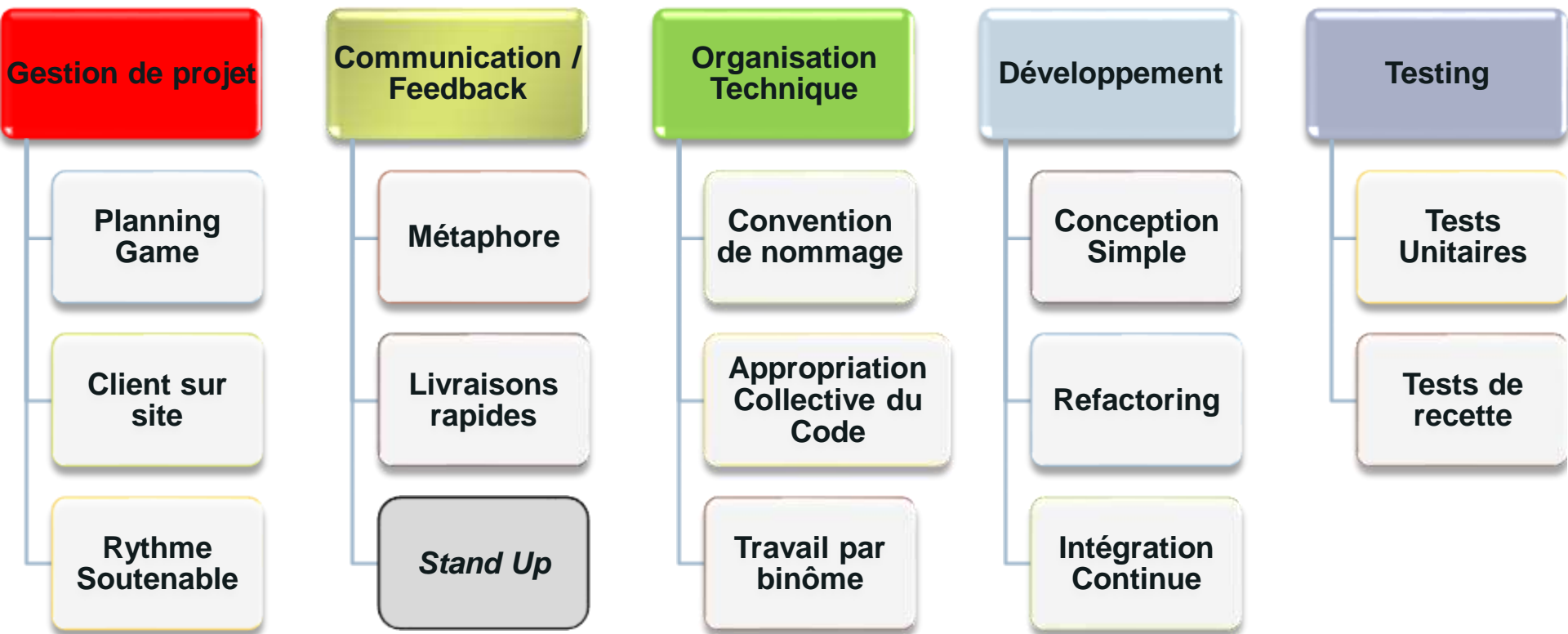
Remaniement Continu (Refactoring)

- ▶ La qualité du code est systématiquement améliorée
 - ▶ Renommer les variables les méthodes les classes
 - ▶ Décomposer les grosses méthodes en des méthodes + petites...
- ▶ Améliorer le code en le rendant plus clair
- ▶ Correction continue du code

Intégration continue

- ▶ Plusieurs fois par jour pour chaque binôme
- ▶ Tâches d'une granularité de quelques heures
- ▶ Tous les jours une version stable du logiciel est disponible.

Les 13 pratiques d'eXtreme Programming

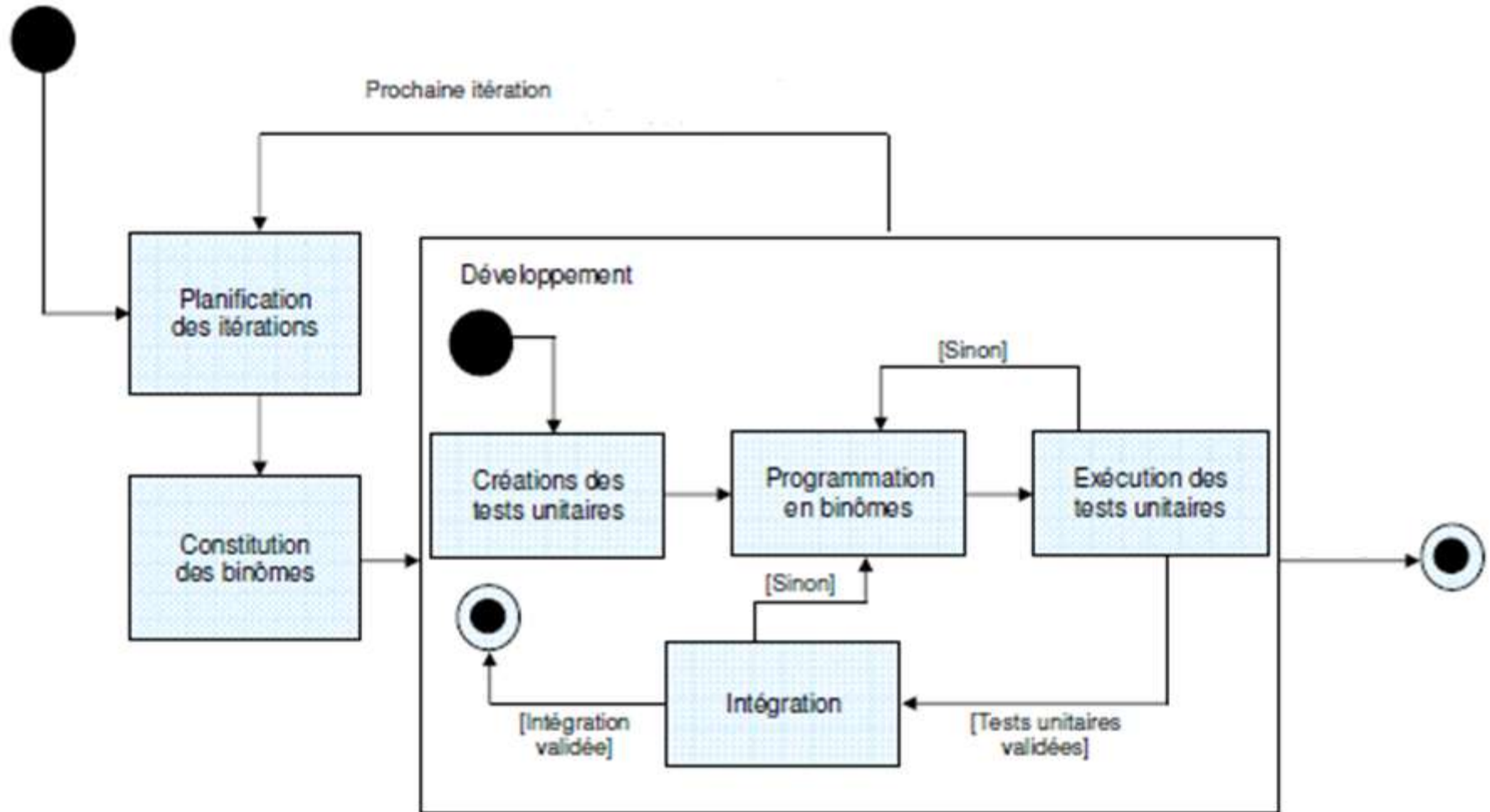


Refactoring = Revoir, corriger et améliorer le code



Pratiques XP

Itération



Pratique: Tests unitaires

- ▶ **Écriture systématique de tests unitaires**
 - ▶ Priorité numéro 1 du programmeur
 - ▶ Écrits avant le code par les programmeurs
 - ▶ Tests automatisés: environnements de développement de tests tels que Junit, cppUnit,...
 - ▶ *TDD: Test Driven Development ou Test-Driven Approach*
 - ▶ Cela garantit qu'une fonctionnalité peut être constamment vérifiée à l'issue de son implémentation.



Tests fonctionnels ou tests de recette

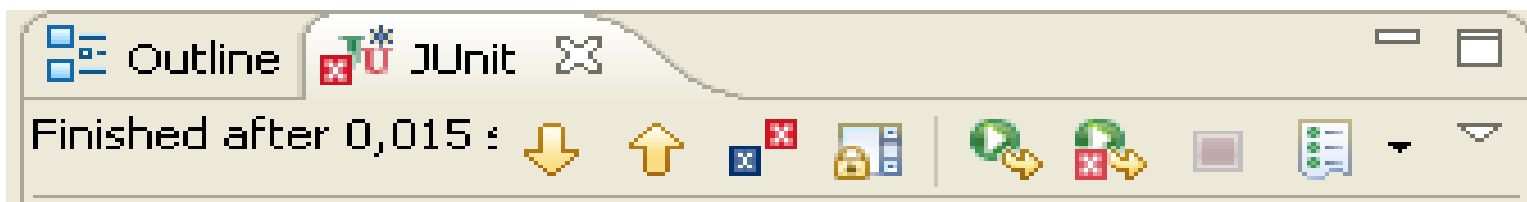
► Écriture de tests fonctionnels

- Ecrits par le client avec l'aide d'un testeur
- Les tests fonctionnels sont appelés tests de recette dès lors qu'ils sont validés par le client

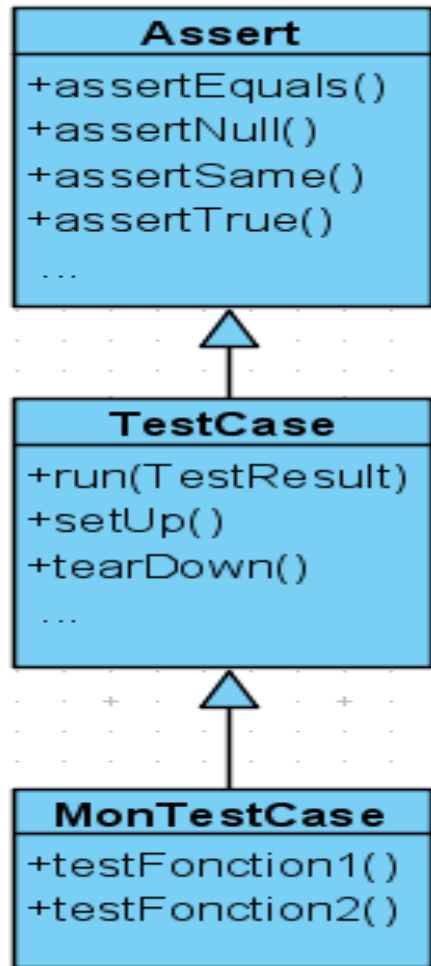


JUNIT

- ▶ JUnit désigne un framework de rédaction et d'exécutions de tests unitaires.
- ▶ Imaginé et développé en Java par **Kent Beck** et Erich Gamma.
- ▶ IL représente une instance de l'architecture « **xUnit** »
- ▶ C'est un outil gratuit et open source
- ▶ De nombreux IDE intègrent Junit : Eclipse, NetBeans, Jbuilder,...



Architecture



Ecrire une classe de test consiste à:

- hériter de la classe TestCase
- implémenter plusieurs méthodes nommées test<f>()
- utiliser des assertions (assertXXX()):
 - assertTrue(l > 0);
 - assertEquals(7,3+4);

Remarque :

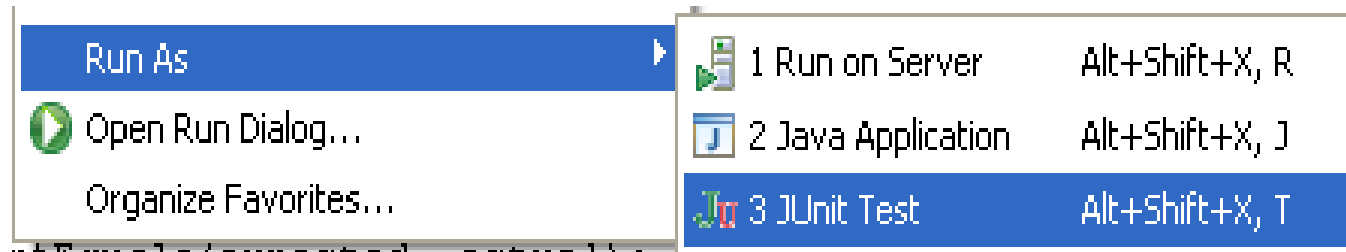
Teardown = interrompre
Assert: vérifier

Junit

- Méthodes d'assertion

Méthode	Description
<code>assertEquals</code>	vérifie l'égalité entre deux objets
<code>assertFalse</code>	vérifie si l'expression est fausse
<code>assertNotNull</code>	vérifie que l'objet n'est pas <code>null</code>
<code>assertNotSame</code>	vérifie que deux références sont différentes
<code>assertNull</code>	vérifie que l'objet est <code>null</code>
<code>assertSame</code>	vérifie que deux références sont les mêmes
<code>assertTrue</code>	vérifie que l'expression est vraie
<code>fail</code>	provoque l'échec du test

JUnit



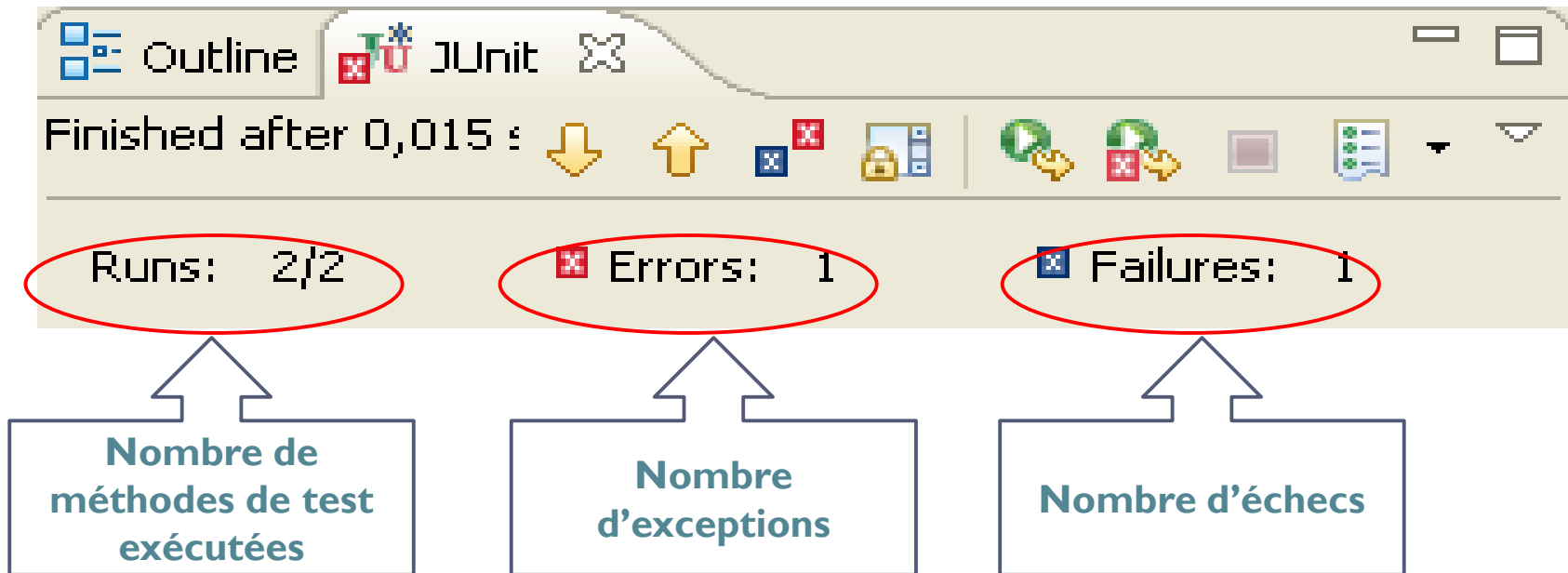
- Exemple de test

- le test peut-être lancé sous Eclipse par
Run As... JUnit Test

```
public class CalculatriceTestSimple_1 extends
TestCase
{
    public void testAddition()
    {
        int a = 3;
        int b = 4;
        int r = a+b;
        Calculatrice calc = new Calculatrice();
        assertTrue(r==calc.additionner(a, b));
    }
}
```

Terminologie : Errors et Failure

JUnit distingue entre les erreurs et les échecs.



- ▶ « Failure » = assertion (assertXXX) qui échoue.
- ▶ « Error » = Exception non attendue.
- Lors d'une détection d'une « Failure » ou « Error » dans une méthode de test, Junit:
 - Interrompt l'exécution de cette méthode.
 - Lance l'exécution de la méthode de test suivante.

Rôles dans XP

Rôles

- ▶ Coach (scrum master dans scrum)
 - ▶ Mise en œuvre de la démarche XP
 - ▶ Amélioration du fonctionnement de l'équipe: facilitateur
- ▶ Tracker
 - ▶ Responsable des indicateurs (nombre de scénarios développés, nombre de tests qui passent,..)
 - ▶ Mesure l'avancement des programmeurs.
- ▶ Manager (chef de projet)
 - ▶ Interface avec le reste de l'organisation
 - ▶ S'occupe des problèmes matériels de l'équipe, recrutement,..
 - ▶ Vérifie que les résultats sont là
 - ▶ Il se tient informé des plans de livraisons et d'itération et veille à leur respect.
- ▶ Client
- ▶ Développeur
- ▶ Testeur
 - ▶ Responsable des tests fonctionnels et le suivi des métriques de tests.
 - ▶ Responsable de tests de technologies

Adopter XP – Comment ?

Appliquer les recommandations de la méthode **par petites touches**.

- Le jeu du planning
- Tests unitaires et intégration continue
- ...

Adopter XP – Difficultés

Implication forte du client nécessaire



Scrum Master (H/F)

Massy (91)

Type de contrat



...Rédiger les User Story ; - Animer les cérémonies Scrum, échanger avec les équipes tout au long du projet...
Technique/Fonctionnel : * Connaissance du mode Scrum Kanban * Une expérience en gestion de projet (suivi...

Détails ▼

COACH AGILE F/H

CDI • Paris (75)

Le 9 octobre



New

...Agile. Vous formez les Product Owner et Scrum Master et les accompagnez lors du lancement du projet. Vous assurez ensuite un suivi et un support, intervenant en tant que coach au sein de l'équipe si nécessaire... outils liés à l'Agilité * Maîtrise des méthodes Scrum, XP, Lean, Kanban * Maîtrise d'un outil de gestion...

Détails ▼

COACH AGILE - ITIM/LEO

CDI • Île-de-France

Le 7 octobre



...moins 5 ans dans le monde projet ainsi que d'une pratique confirmée des approches agiles (lean, scrum, kanban, XP, management visuel, innovation games...) en tant que coach. Vous présentez également une...

Détails ▼





Agile Dev Survey

Agile Methods & Practices

AGILE METHODOLOGY USED

Scrum or Scrum variants (72%) are still the most popular agile methodologies being used. Kanban and Kanban variants nearly doubled this year, mostly due to an uptick in Scrumban use.

