



Bienvenue

**POEC : Techniques
de développement
agiles**

**Passage certificat
Professional
SCRUM Developer
(PSD I)**



— Glodie Tshimini





Présentation

- Consultant Formateur et Développeur depuis 2017 **certifié SCRUM Professional Developer I**
- J'interviens dans la formation et développement sur les thèmes suivants
 - ▶ Gestion de projet
 - ▶ HTML & CSS & SASS & UX
 - ▶ JavaScript et son écosystème
 - ▶ PHP et son écosystème
 - ▶ CMS (WordPress et PrestaShop)
 - ▶ SEO
 - ▶ Versioning
 - ▶ Modélisation UML et Merise
 - ▶ Bases de données



**Avant de
commencer**

- Durée de la formation :
- Objectifs :
- Organisation
 - ▶ Horaires - 9h à 17h30
 - ▶ Pauses : 15 min en matinée et après midi (vers 10h30 et 15h30)
 - ▶ Déjeuner : 12h30 à 13h30





Avant de
commencer

— Tour de table :

1. Votre expérience sur **SCRUM** ou les méthodes agiles
2. Niveau de compréhension d'un texte en **Anglais**
3. Votre expérience dans le **développement informatique**



La certification PSD I de scrum.org

- **80 questions en anglais** à traiter en **1 heure**
- Pour obtenir la certification, il faut avoir un score de **85%** (soit **68 bonnes réponses**)
- Il n'y a pas de rattrapage, en cas d'échec, il faut repayer une nouvelle session
- Certification dispensée par **SCRUM.org**
- Valable **à vie**
- Valorisé par la profession (entreprises et en particulier **ESN**)



PLAN

- I. INTRODUCTION AUX MÉTHODES AGILES
- II. HISTORIQUE
- III. LA DÉFINITION DE SCRUM
- IV. LA THÉORIE DE SCRUM
- V. LES VALEURS DE SCRUM
- VI. L'ÉQUIPE SCRUM
- VII. LES ÉVÉNEMENTS SCRUM
- VIII. LES ARTÉFACTS DE SCRUM
- IX. BONNES PRATIQUES DU DÉVELOPPEMENT LOGICIELS ET AGILE

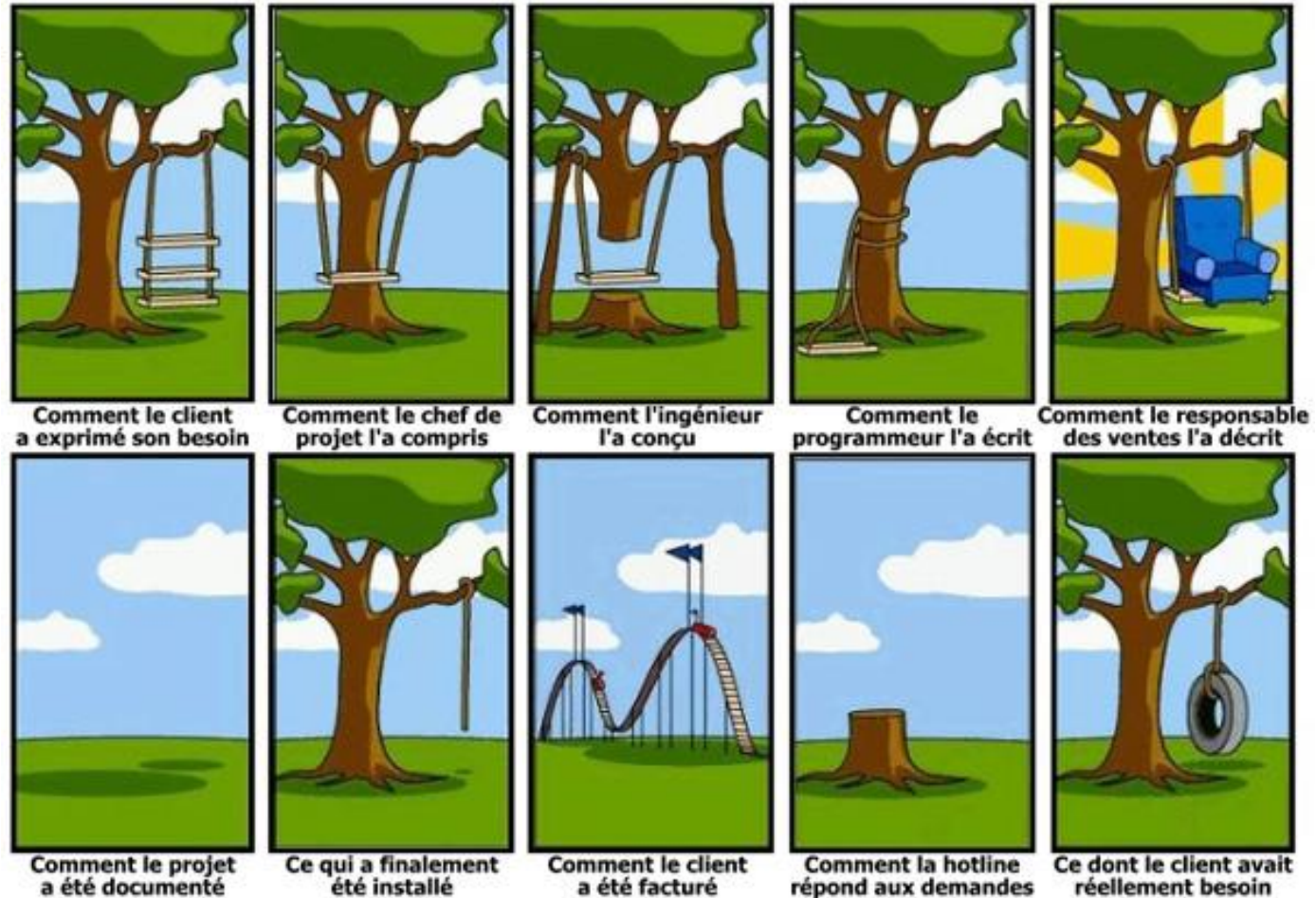


I. INTRODUCTION AUX Méthodes agiles



Vision d'un projet digitale

- Source image anyideas



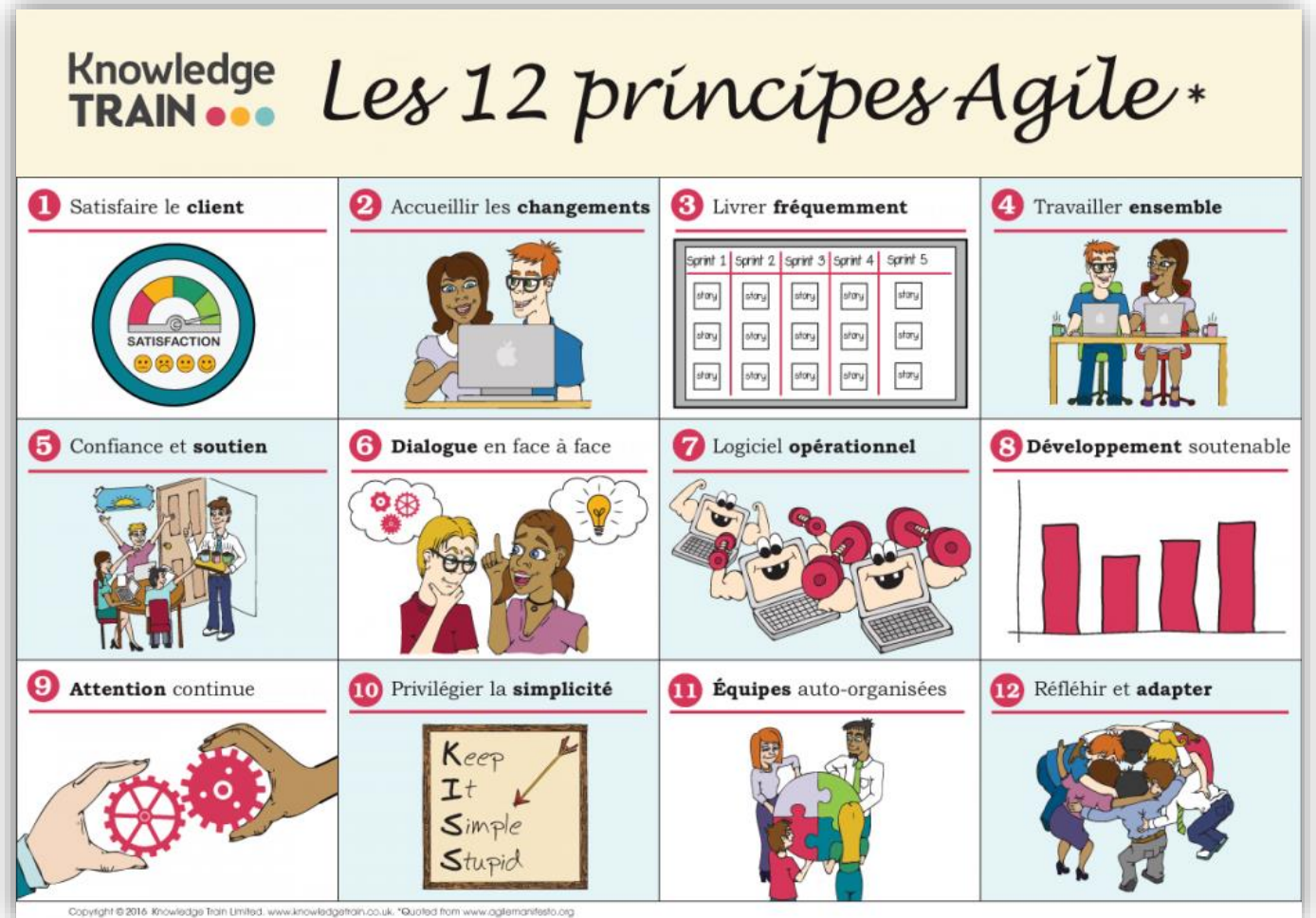
4 VALEURS AGILES

1. Les **individus** et les **interactions**, de préférence aux processus et aux outils.
2. Des **solutions opérationnelles**, de préférence à une documentation exhaustive.
3. La **collaboration** avec les clients, de préférence aux négociations contractuelles.
4. La réponse au **changement**, de préférence au respect d'un plan.

Précisément, même si les éléments à droite ont de la valeur, nous reconnaissons davantage de valeur dans les éléments à gauche.

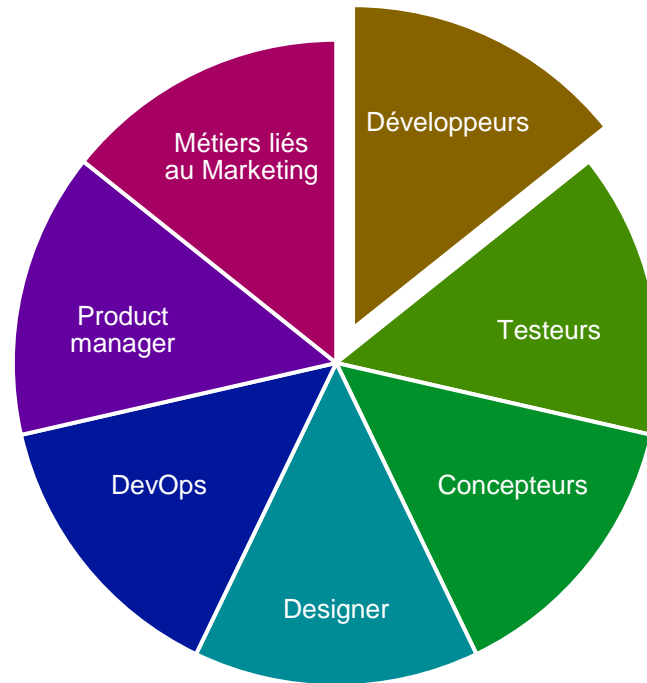


— source image Wikiagile





Quelques membres des équipes agiles



- Les équipes agiles sont **pluridisciplinaires**
- Les équipes **s'organisent**, travaillent et prennent des **décisions ensemble**
- L'accent est mis sur la **collaboration** des membres de l'équipe plutôt que des individualités (compétences spécifiques)



Lecture du guide SCRUM



Activité
30
minutes

Je reste disponible
pour toute question

— 0-exercices/README.md/QUIZ0



II. HISTORIQUE





HISTORIQUE

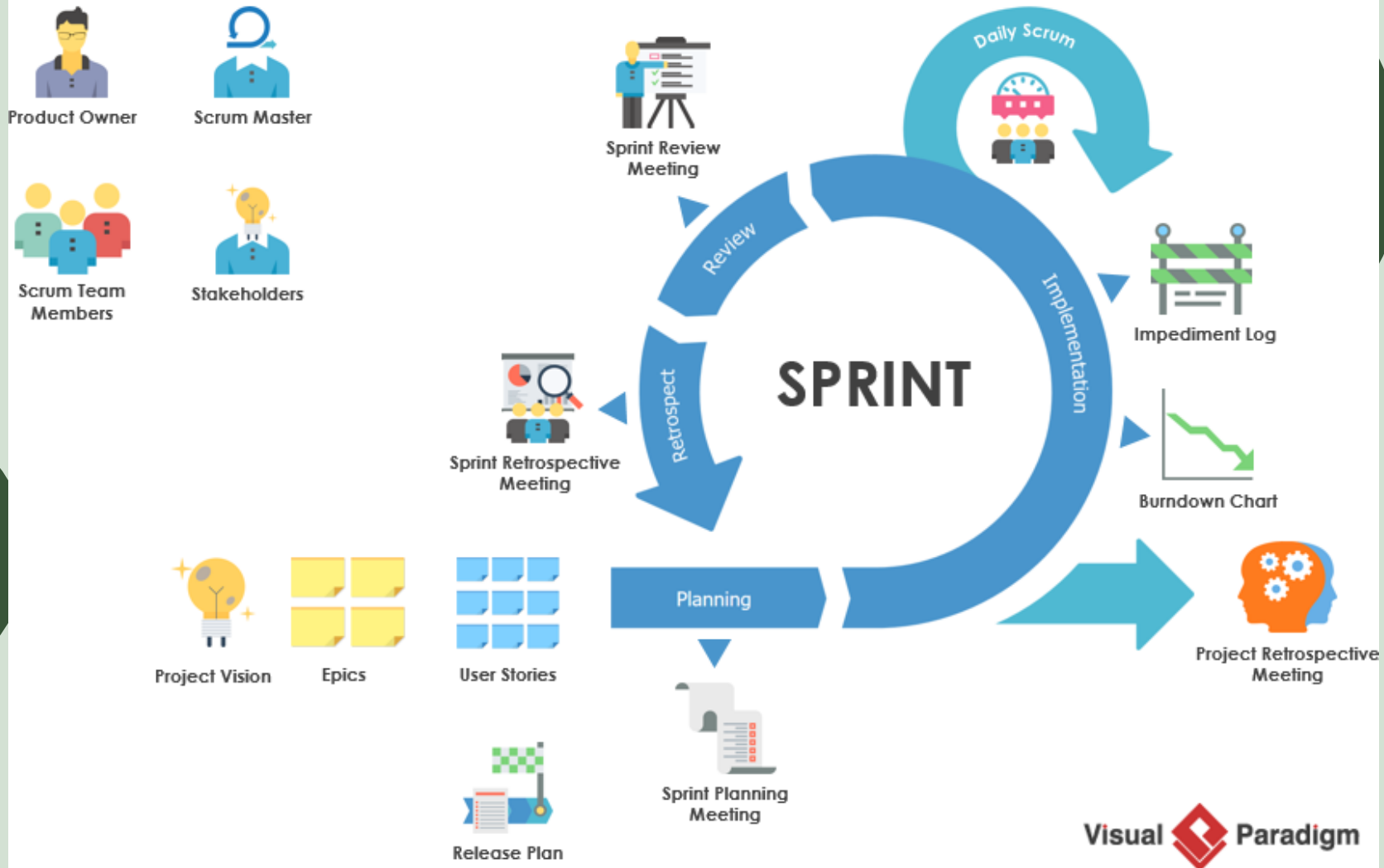
- Développé dans les années **90**
- Créateurs
 - [Ken Schwaber](#)
 - [Jeff Sutherland](#)

• SCRUM GUIDE

- Document officiel qui partage la vision de SCRUM par ses créateurs
- Les versions
 - Première en 2010
 - Précédente version 2017
 - Version actuelle sur lequel se portera la **certification PSD** est le **SCRUM GUIDE 2020**

Cycle SCRUM

The Agile – Scrum Framework





QUIZ 1

— 0-exercices/README.md/QUIZ1



15 min



III. LA DÉFINITION DE SCRUM



FRAMEWORK

- SCRUM est
 - Un **Framework léger**
 - Un ensemble des **règles** pour **guider les relations et les interactions** entre les individus participant au projet
 - **Volontairement incomplet** pour laisser la liberté à **l'intelligence collective** (capacité à résoudre des problématiques en travaillant ensemble avec des profils divers)
 - Un Framework Agile
 - Axé sur la **valeur du produit** et sa **maximisation**
 - **Adapté aux solutions innovantes et complexes**
 - Une aide pour les équipes à générer rapidement de la valeur aux produits et services à développer
 - Utilisé pour des projets dans divers domaines (n'est plus exclusivement réservé à l'IT)



IV. LA THÉORIE DE SCRUM

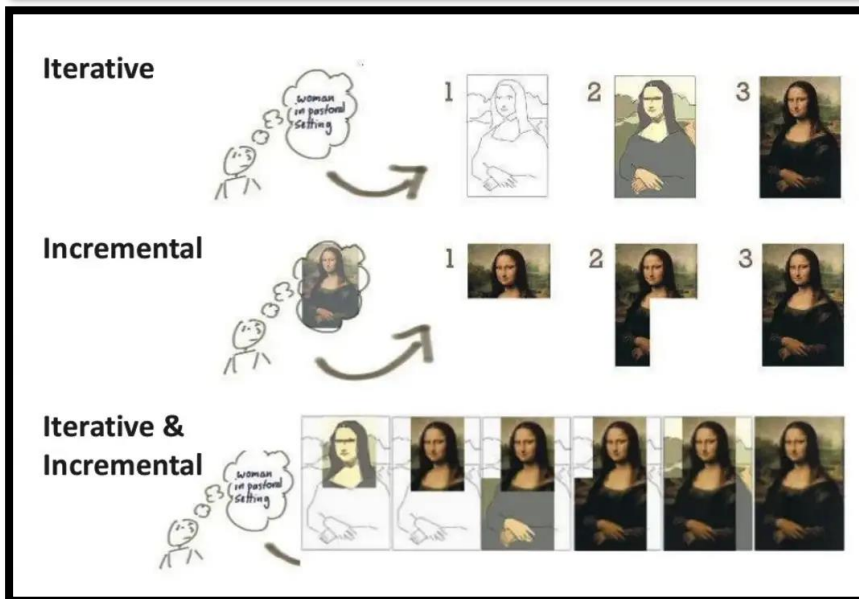
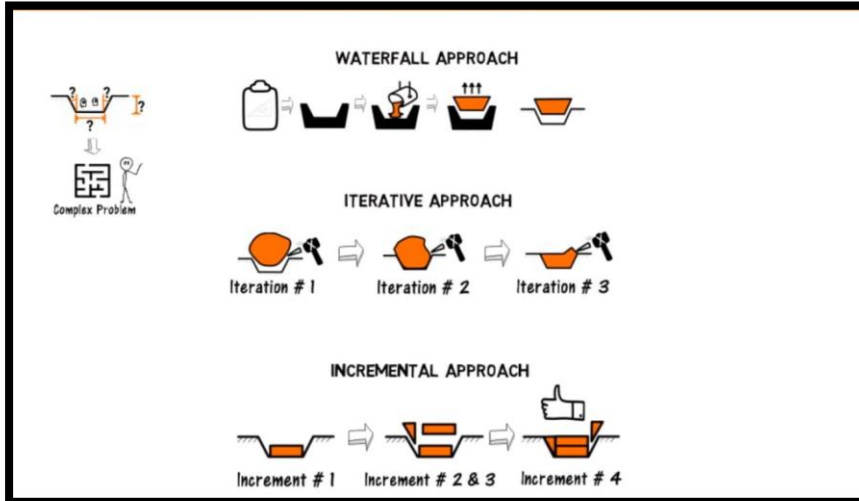


EMPIRISME ET LEAN

- La théorie de *SCRUM* se base sur l'**Empirisme** et la pensée **LEAN**
- Empirisme
 - Courant philosophique qui dit que toutes connaissances viennent de l'expérience
 - Appliqué dans le cadre SCRUM, cela signifie que les « **décisions à prendre doivent s'appuyer sur l'observation de faits** »
- Pensée du **LEAN**
 - **Réduction du gaspillage**, concentration sur l'essentiel et **discipline**
 - Meilleure performance
 - Meilleure rentabilité
 - Meilleure qualité

ITÉRATIVE ET INCRÉMENT ALE

- [Source image Vscrum Team](#)
- [Source image bootcamp uxdesign](#)

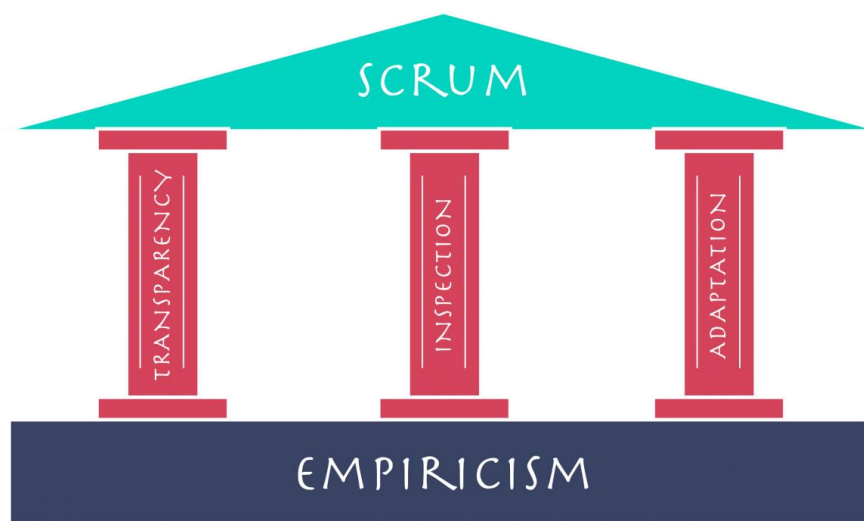


- « *Approche à la fois itérative et incrémentale pour optimiser la prédictibilité et le contrôle de risque.* »
- Les équipes SCRUM ont toutes les **compétences** et **expertises** pour travailler sur le projet
- Les membres de l'équipe **partagent** leurs compétences et expertises au sein de l'équipe

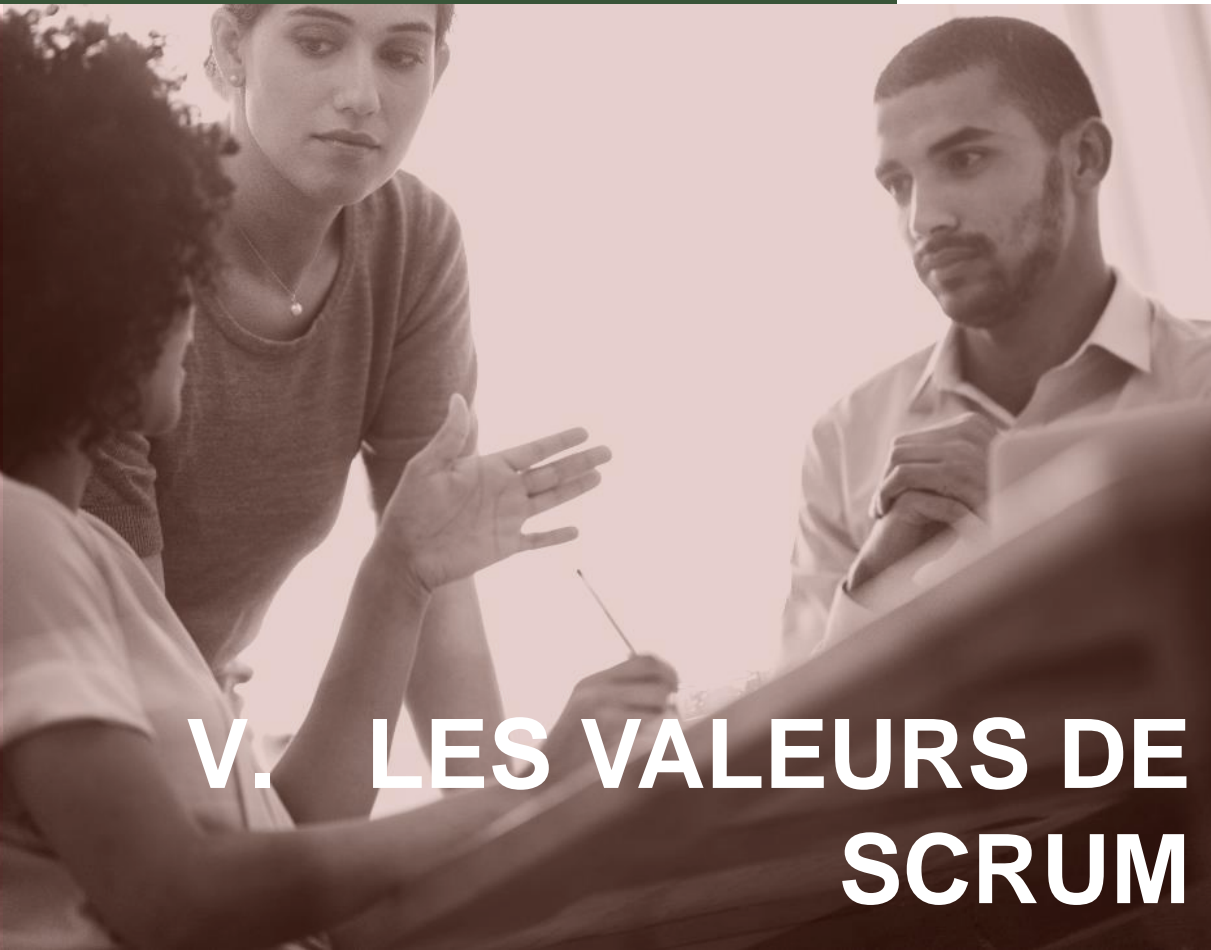


LES 3 PILLIERS DE SCRUM

— [Source image wikiagile](#)



- Transparence
 - Tout ce qui concerne le produit doit être visible par toutes les personnes impliqués
 - Peu de transparence augmente le risque et diminue la valeur du produit
 - La transparence permet l'inspection
- Inspection
 - Inspection fréquente du travail en cours pour détecter rapidement les écarts
 - L'inspection permet l'adaptation
- Adaptation
 - S'ajuster dès que les écarts deviennent inacceptables pour atteindre l'objectif fixé



V. LES VALEURS DE SCRUM

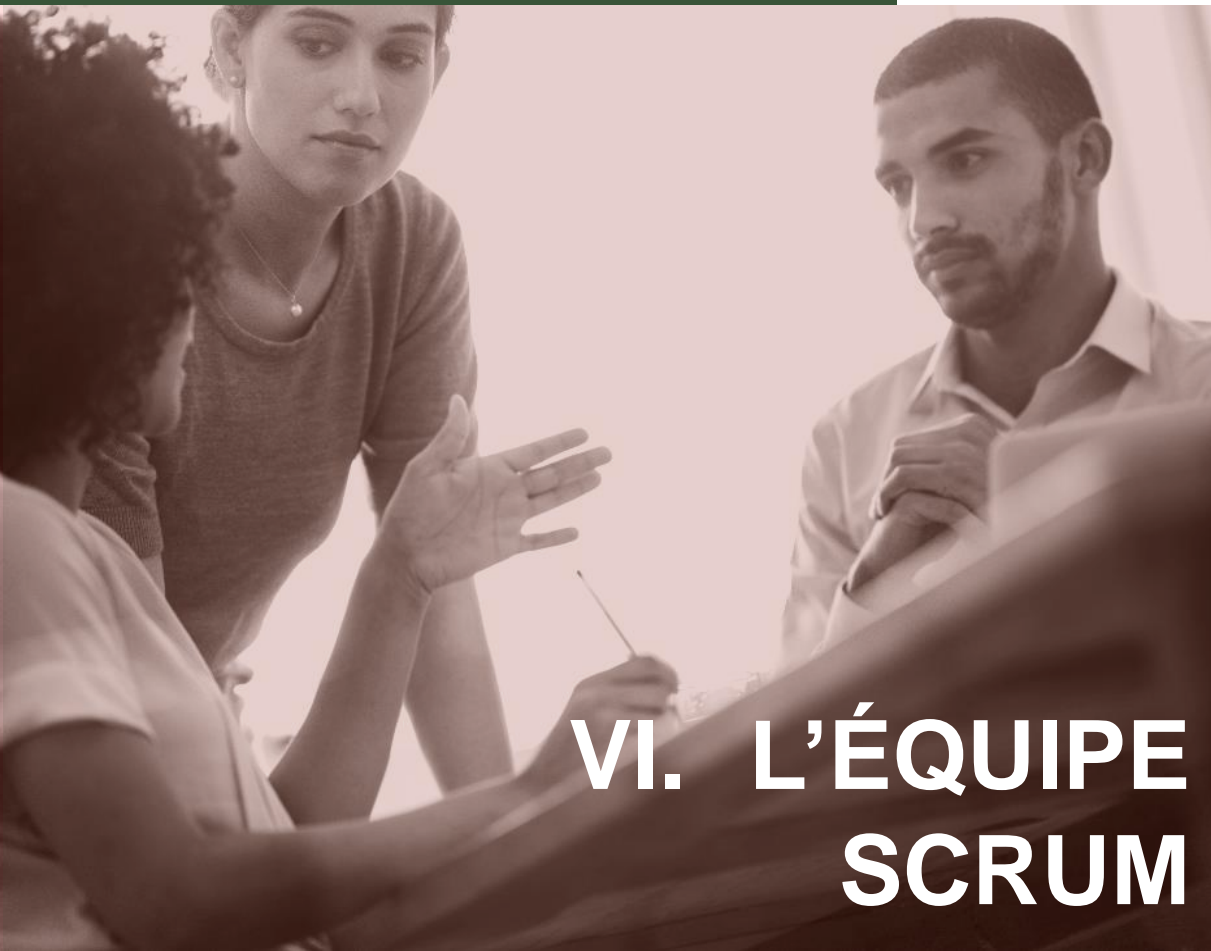


LES 5 VALEURS DE SCRUM

- [Source image blog Scrum.org](https://blog.scrum.org)



- Les 5 valeurs que doivent partager les membres de l'équipe
- 1. **Commitment**
 - Engagement et soutien de chacun pour atteindre les objectifs
- 2. **Courage**
 - Travail sur les problématiques difficiles et trouver les bonnes actions
- 3. **Focus**
 - Concentration sur les objectifs fixés
- 4. **Openness**
 - « Ouverture sur le travail et les défis à relever »
- 5. **Respect**
 - Respect mutuel de chaque individu et du travail

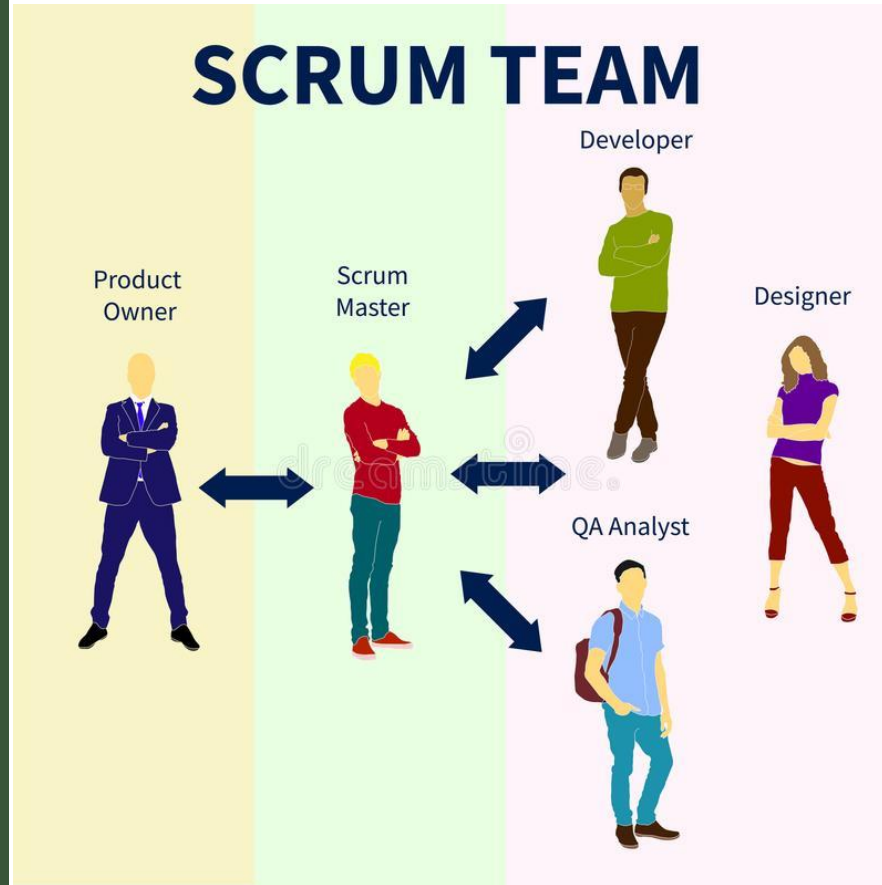


VI. L'ÉQUIPE SCRUM



L'ÉQUIPE SCRUM

— [Source image dreamstime](#)



- 1 Scrum Master
- 1 Product Owner
- Developers (3 à 9 individus)
- Pas de hiérarchie
- Pas de sous-équipe
- Pas d'autres rôles que les 3 ci-dessus

COMPÉTENCES ET EXPERTISES DE L'ÉQUIPE

- Les équipes plus petites ont tendances à mieux s'organiser et à être plus productif (c'est une généralité et non une vérité absolue)
- Plusieurs équipes SCRUM peuvent travailler sur le même produit
- Plusieurs teams
 - Avoir le **même Product Goal** (Objectif produit)
 - Avoir le **même Product Backlog** (Carnet de produit)
 - **1 Product Owner (PO)**
 - **1 Scrum Master (SM)** par équipe

- Les équipes SCRUM sont
 - **Cross-functional** (Multifonctionnel ou transversal)
 - Les membres ont toutes les compétences requises pour produire de la valeur à chaque itération de la réalisation du produit
 - **Self-managing** (Auto-gestion)
 - Les membres décident de qui fait quoi, quand et comment
- Les équipes sont soutenues par leur direction dans leurs prises de décision
- Tout le monde est responsable de la création de valeur.

DEVELOPERS

- [Source image Letsscrumit](#)

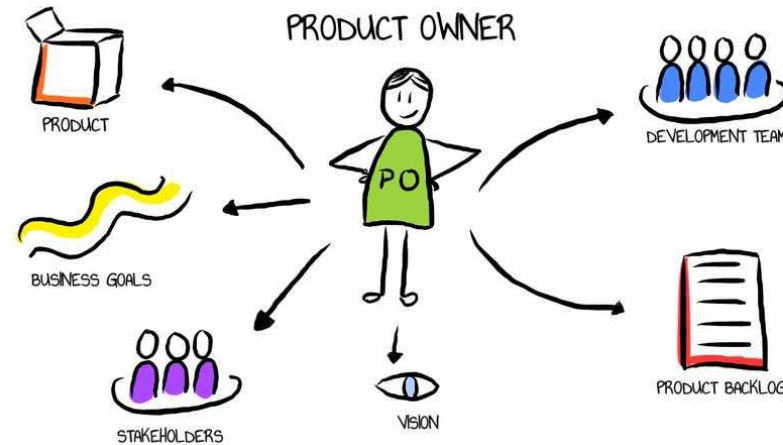


- Le rôle **developers** regroupent toutes les personnes responsables de la création d'un incrément de valeur à chaque sprint. (équipe de réalisation)
- Taille maximale 9 individus
- Responsabilités
 - Réaliser **le Sprint Backlog**
 - Établir un plan de Sprint (plan d'action)
 - Adhérer à la **Definition of Done**
 - Adapter leur plan chaque jour pour atteindre le **Sprint Goal** (objectif du sprint)
 - Estimer la complexité des éléments du Product Backlog
 - Être professionnels



PRODUCT OWNER

— [Source image Letsscrumit](#)

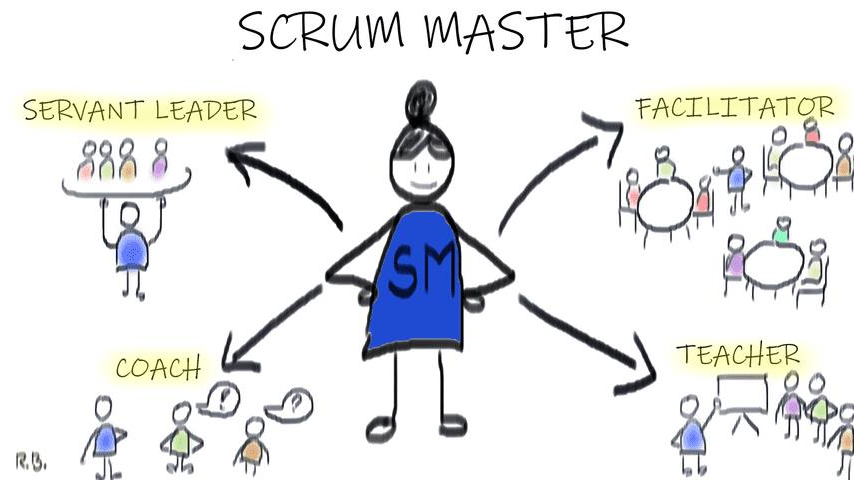


R.B.

- Une seule personne
- Représente les autres parties prenantes (client, utilisateurs finaux etc.) au sein de l'équipe
- Responsabilités
 - Maximiser la valeur du produit
 - Gérer efficacement le **Product Backlog (PB)**
 - Développer et communiquer le **Product Goal**
 - Créer, communiquer et prioriser les **Product Backlog Items**
 - Rendre le PB transparent, visible et compréhensible
 - Peut déléguer une partie de son travail aux autres membres de l'équipe cependant il demeure le seul responsable et le dernier mot lui revient.

SCRUM MASTER

— [Source image Letsscrumit](#)



Responsabilités

- S'assurer de la bonne application de **SCRUM**
 - Tous les événements ont lieu avec le respect du temps (**timebox**) et que les membres concernées y participent
- Aide l'équipe et l'organisation à comprendre **SCRUM**
 - Efficacité de l'équipe
 - Aider à l'amélioration des pratiques
- Au service de l'équipe
 - En tant que **facilitateur**
 - Celui qui **lève les obstacles**
 - **Coach** l'équipe dans l'autogestion et partage des compétences
 - Aide l'équipe à se concentrer sur la maximisation de la valeur

SCRUM MASTER

- Collaboration avec le Product Owner
 - Aide à trouver des techniques plus efficace pour communiquer le *Product Goal*
 - Aide à gérer efficacement le *Product Backlog*
 - Aide à comprendre et à décrire clairement les *Product Backlog Items*
 - Facilite la collaboration avec les parties prenantes du projet
- Collaboration avec l'organisation (entreprise)
 - Guide, entraîne et coach l'organisation à adopter *SCRUM*
 - Planifier et conseil l'organisation à mettre en place *SCRUM*
 - Supprime les barrières entre les parties prenantes et l'équipe *SCRUM*



QUIZ 2

— 0-exercices/README.md/QUIZ2



15 min



VII. LES ÉVÉNEMENTS SCRUM



SPRINT

- Les événements dans SCRUM sont une occasion **d'inspecter** et **d'adapter** les artefacts
- Le sprint est le **conteneur** de tous les événements SCRUM
- But
 - **Transformer les idées en valeur**
 - Développer un **incrément opérationnel** (livrable utilisable)
 - Une brique supplémentaire qui vient s'ajouter à l'existant
 - Un incrément qui nous rapproche davantage du produit final
- Aucun changement ne doit nuire à l'objectif du Sprint
- Les objectifs de qualité ne doivent pas baisser
- Le périmètre peut être clarifié et renégocié entre les *developers* et le *Product Owner*
- Le prochain sprint commence dès que le précédent se termine, **il n'y a pas de pause entre deux sprint**

SPRINT

- Les sprints **plus courts limitent les risques** liés aux **coûts** et à l'effort car le feedback est plus rapide et donc **l'adaptation est faite plus tôt**.
- Attention **le Sprint 0 n'existe pas dans la théorie de SCRUM**, c'est une mauvaise pratique qu'on retrouve dans les entreprises qui consistent à ne pas produire de la valeur au premier sprint mais mettre en place l'architecture, la base de données, l'organisation de l'équipe etc. c'est contre l'esprit de SCRUM qui est de produire de la valeur à chaque Sprint.
- **Le Sprint peut être annulé uniquement par le Product Owner si l'objectif du Sprint dévient obsolète.**
- On ne présente pas et par extension, on ne livre pas un incrément non fini.
Les éléments non finis du *Sprint Backlog* retournent dans le *Product Backlog*



SPRINT PLANNING

- **Événement d'ouverture du Sprint**
- L'équipe peut inviter des personnes extérieures pour avoir des conseils et leurs expertises
- Le Sprint Planning doit **répondre à 3 questions** :
 1. ***Pourquoi ?***
 2. ***Quoi ?***
 3. ***Comment on travail ?***
- A la fin du Sprint Planning, on doit avoir l'artefact Sprint Backlog
- Sprint Backlog = Sprint Goal + Product Backlog items sélectionnés + Plan de livraison

1. **Pourquoi** ce Sprint est-il important ?
 - Le Product Owner expose la valeur qu'apporte l'objectif de ce Sprint au produit
 - L'équipe finalise l'objectif du Sprint avant la fin du meeting (Sprint Goal)
2. Que peut-on faire durant ce Sprint (**quoi**) ?
 - Les Developers sélectionnent les éléments du Product Backlog sur lesquelles ils ont travailler pour atteindre l'objectif
 - Ils affinent et s'engagent à les réaliser durant le Sprint. Pour cela, ils s'appuient sur les performances passées, leur projection sur les capacités à venir et la Definition of Done.
3. **Comment** le travail choisi sera-t-il réalisé ?
 - Pour chaque élément du Product Backlog sélectionné, l'équipe des Developers définit un plan d'action qui répond à la Definition of Done
 - Les Developers sont les seuls à décider de la manière de travail sur un Sprint.

DAILY SCRUM

- Objectifs
 - **Inspecter** les travaux en cours vers l'objectif de Sprint et **adapter** le Sprint Backlog si nécessaire
 - Améliorer la communication entre les Developers
 - Identifier les obstacles et les lever
 - Éliminer la nécessité d'avoir recours à d'autres réunions (réduire ne va pas dire supprimer entièrement cette possibilité)
- **Se tient à la même heure et au même endroit pour réduire la complexité**
- Participation
 - **Obligatoires pour les Developers**
 - Le Scrum Master et Product Owner peuvent participer uniquement dans le cas où ils travaillent activement sur un élément du Sprint Backlog. Ils participent avec le rôle de Developers



SPRINT REVIEW

- [Source image letsscrumit](#)



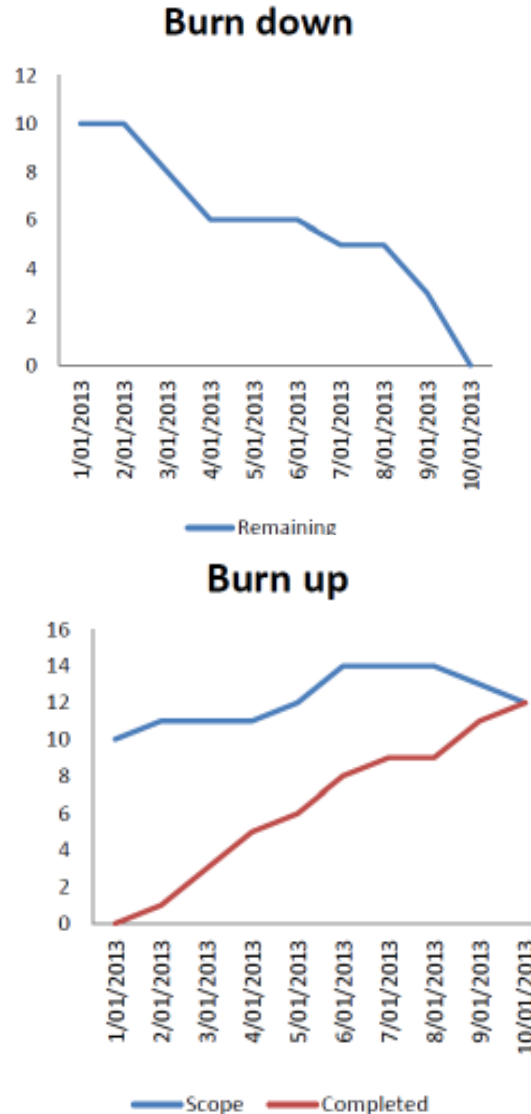
- Session de travail à part entière (ne se limite pas à une présentation du travail réalisé)
- Objectifs
 - Inspecter le travail réalisé durant le Sprint
 - Présenter ce travail aux parties prenantes du projet et la progression vers l'objectif du produit
 - **Récolter du feedback** pour **adapter la direction** à prendre pour les prochains Sprints

SPRINT RETROSPECTIVE

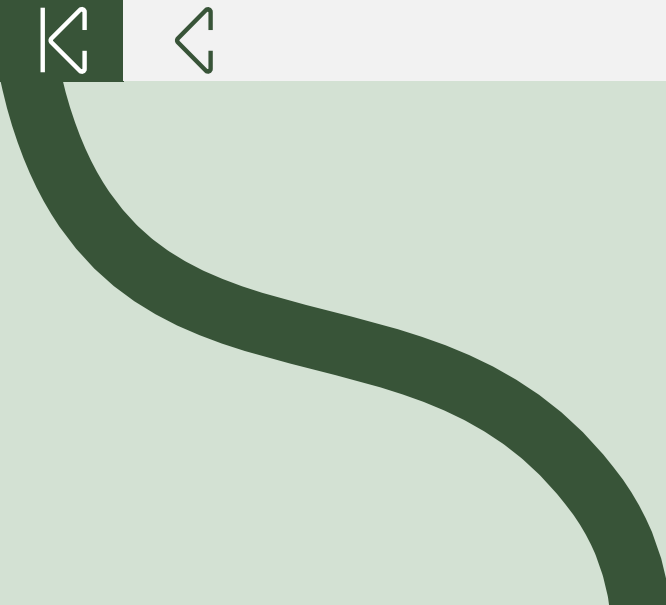
- Objectifs
 - **Réflexion sur les pistes pour améliorer la qualité et l'efficacité de l'équipe pour les prochains Sprints**
 - **Inspecter le déroulement du Sprint en cours** au niveau
 - Des individus
 - Interactions
 - Processus
 - Outils
 - Definition Of Done (DoD)
 - Identifier les éléments qui ont fait dévier l'équipe de son plan initial et leurs origines
 - Identifier les obstacles, problèmes et comment ils ont été résolus ou non
 - Les éléments d'amélioration les plus impactant peuvent être ajoutés au Product BackLog
 - L'équipe peut décider durant cette événement de changer le DoD
 - Événement qui conclut le Sprint

EVALUER LA PROGRESSION D'UN SPRINT

— [Source image Clariostechology](#)



- **Burn Down**
 - Indique le travail restant à produire dans un périmètre fixe et stable
- **Burn Up**
 - Indique le travail réalisé dans un périmètre qui évolue
- La *vélocité* est la quantité de travail terminé. Souvent, on se réfère à la complexité que l'équipe a réussi à résoudre durant le sprint pour quantifier la vélocité
- La prise de décision repose toujours sur l'empirisme, les diagrammes sont justes un moyen d'afficher la progression de l'équipe
- Les diagrammes sont destinés à l'équipe



TIME-BOXING DE TOUS LES ÉVÉNEMENTS SCRUM

- Sprint
 - **1 mois au maximum** (entre 1 et 4 semaines)
- Sprint Planning
 - Pour un Sprint d'un mois, c'est **8 heures maximum**
 - Pour un Sprint de moins d'un mois, c'est moins de 8 heures
- Daily Scrum
 - **15 minutes maximum**
- Sprint Review
 - Pour un Sprint d'un mois, c'est **4 heures maximum**
 - Pour un Sprint de moins d'un mois, c'est moins de 4 heures
- Sprint Retrospective
 - Pour un Sprint d'un mois, c'est **3 heures maximum**
 - Pour un Sprint de moins d'un mois, c'est moins de 3 heures



QUIZ 3

— 0-exercices/README.md/QUIZ3



20 min



VIII. LES ARTÉFACTS SCRUM

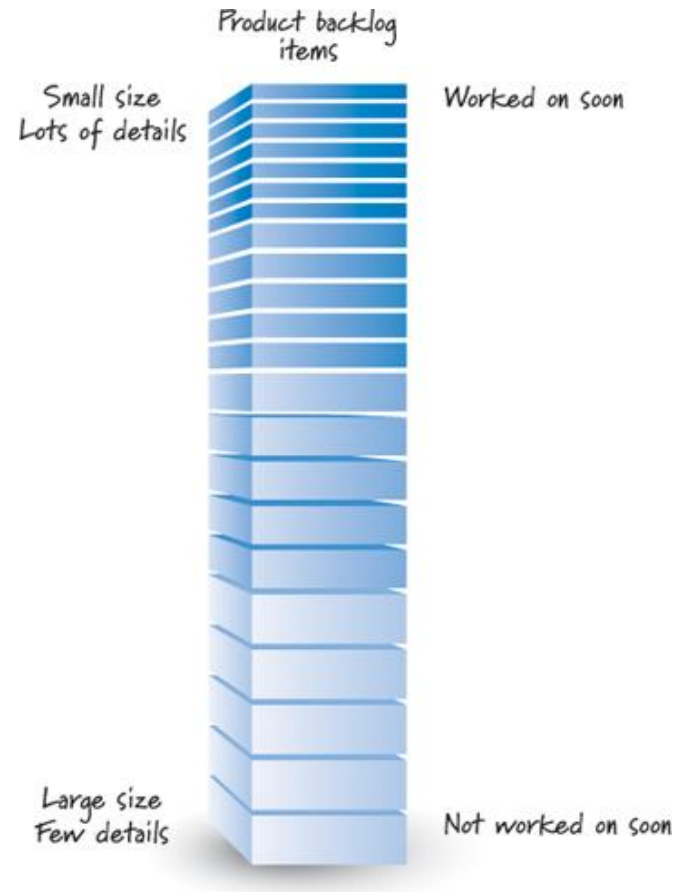


ARTEFACTS DE SCRUM

- Artefact = travail ou valeur
- Pour
 - Maximiser la **transparence** des informations
 - Permettre l'**inspection**
 - Permettre l'**adaptation**
- 3 artefacts SCRUM
 1. **Product Backlog**
 2. **Sprint Backlog**
 3. **Increment**
- Chaque artefact est associé à un engagement de transparence
- Le **product Backlog** est associé à l'engagement du **Product Goal**
- Le **Sprint Backlog** à l'engagement du **Sprint Goal**
- L'**incrément** à l'engagement de la **Definition of Done**

PRODUCT BACKLOG

— [Source image Informit](#)



- Liste émergente et ordonnée
- **Product Backlog** contient
 - Besoins fonctionnels
 - Besoins non-fonctionnels
 - Bugs
 - Etc.
- Les éléments constituant le **Product Backlog** peuvent être déclinés en
 - *Epic* (macro fonctionnalité)
 - *User Story* (fonctionnalité)
 - Tâche (plus petite unité de travail)
 - **NFR (Nonfunctional Requirements)**
 - Améliorations à faire issu de la restrospective
 - Etc.



PRODUCT BACKLOG REFINEMENT

- N'est pas un événement, il s'agit d'un processus d'affinage des Product Backlog Items (PBIs) pour les rendre plus clairs et transparents pour tout le monde
- Lorsqu'un item est suffisamment détaillé, on dit qu'il est « Ready », c'est-à-dire prêt à être potentiellement sélectionné dans le Sprint Backlog et être réalisé en un sprint.
- L'équipe peut définir une Definition of Ready (liste d'éléments permettant de dire qu'un élément est prêt pour être embarqué dans un Sprint)



Estimation des Product Backlog Items

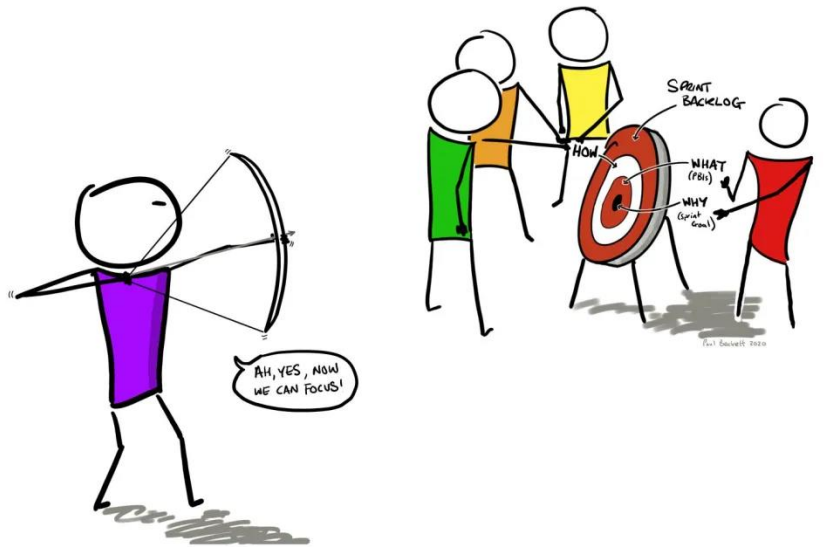
— [Source image Cybermedian](#)



- Les éléments du Product Backlog doivent idéalement être
- **Suffisamment détaillé et transparent**
- **Émergente**
- **Priorisé**
 - Classé dans l'ordre
 - Ceux qui ont le plus de valeurs et/ou les plus affinés sont en haut de la pile
 - Ceux qui ont les moins de valeurs et non détaillé en bas de la pile
- **Estimé uniquement par les Developers**
- Plusieurs techniques d'estimation
 - Planning Poker (Fibonacci, size, day, Et)
 - Extreme Quotation

SPRINT BACKLOG

- [Source image Edinburgh Agile](#)



- [cf. chapitre Sprint Planning](#))
- Est mis à jour tout au long du Sprint au fur à mesure que les **Developers** apprennent des nouveaux éléments sur l'incrément en cours de réalisation
- Le périmètre du *Sprint Backlog* peut varier mais elle ne doit pas affecter le Sprint Goal ou réduire les exigences de qualité.

INCREMENT ET DEFINITION OF DONE

— Increment

- **Bout de produit opérationnel** (qui est utilisable) qui tend vers le Product Goal.
- Chaque nouvel Increment s'ajoute à la somme de tous les increments précédents
- **Un sprint donne lieu à la création d'un ou plusieurs Increments**
- A la fin du Sprint, lorsqu'un Increment n'est pas opérationnel, il ne doit pas être présent à la Sprint Review, il est éventuellement réestimé et obligatoirement replacé dans le Product Backlog.
- Un Increment peut être délivré dès qu'il satisfait à la Definition of Done.
On ne doit pas attendre le Sprint Review pour le délivrer

— Definition of Done (DoD)

- **Apporter de la transparence sur la définition du travail fini ou terminé**
- Généralement des **standards de l'organisation** (SO) que la Scrum Team doit suivre
 - S'il n'y a pas de SO, la Scrum Team doit créer sa propre DoD
 - S'il y a plusieurs équipes sur le même produit, la DoD doit être commune
- **Un increment doit satisfaire au DoD** pour être présenté au Sprint Review et être potentiellement livrable en production
- **Il n'y a pas l'obligation de livrer les Increments en production à chaque fin de Sprint**
- La stratégie de livraison est déterminée par l'équipe SCRUM.



QUIZ 4

— 0-exercices/README.md/QUIZ4



10 min



IX. BONNES PRATIQUES DU DÉVELOPPEMENT LOGICIELS ET AGILE

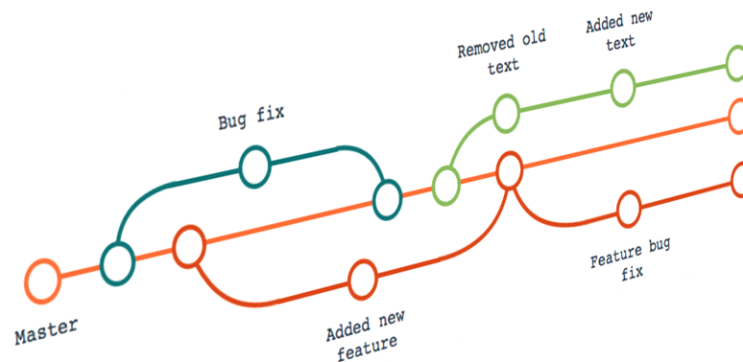


VERSIONING



VERSIONING

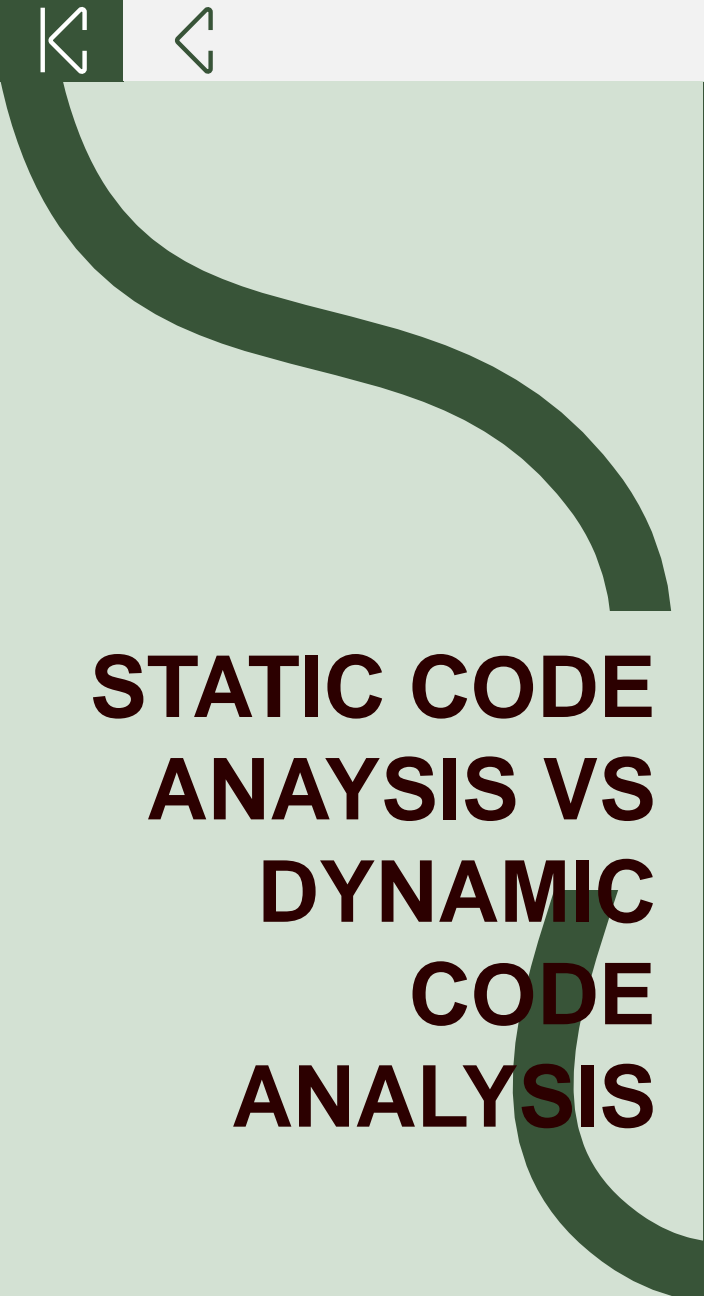
- [Source image blog Planethoster](#)



- Gestion des branches
 - Bonne gestion du versioning de son code n'empêche pas les conflits lors de l'intégration des travaux des équipes
 - Avoir une branche principale nommée **main**, **mainline**, **master** ou **trunk** qui sert de base à la création des autres branches de travail

ANALYSE DU CODE





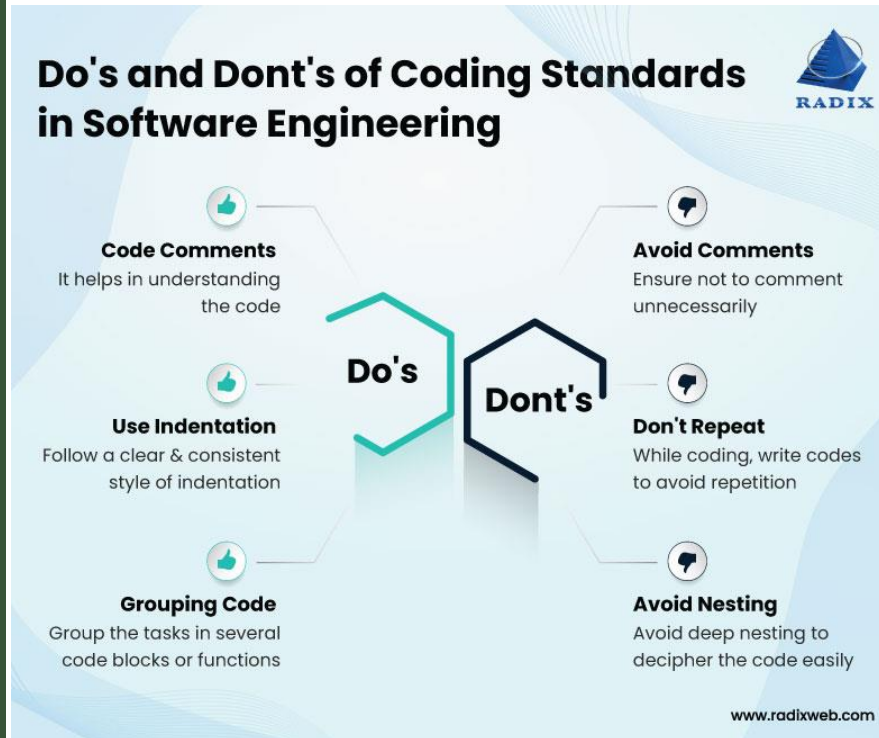
STATIC CODE ANAYSIS VS DYNAMIC CODE ANALYSIS

- Static Code analysis
 - Inspection sans exécuter le code
 - Inspection des failles dans le code
 - Code malveillant
 - Portée dérobée
 - Failles
 - Ce test permet d'inspecter le code du programme pour détecter les failles de codage, les portes dérobées et le code potentiellement malveillant.

- Dynamic Code Analysis
 - Inspection en exécutant le code
 - Inspection des fonctionnalités
 - Temps de réponse
 - Performance globale
 - Ce test permet de contrôler le comportement fonctionnel, le temps de réponse et les performances générales du système.



— [Source image Radixweb](#)



- Indentation
- Usage pointer et référence
- Nommage fichier, organisation des dossiers
- Convention de nommage
- Commentaires et documentations
- Utilisations des classes, fonctions et interfaces
- Bénéfice
 - Tout le monde code de la même manière
 - Code plus facile à lire, comprendre, maintenir et partager



QUIZ 5

— 0-exercices/README.md/QUIZ5



6 min

LES TESTS



Definition du testing (test)

- Ecrire du code permettant de vérifier que son code fonctionne correctement, c'est-à-dire donne le résultat attendu
- Plusieurs types et niveaux de tests existent, allant de la plus petite unité de code à tester (bout de code, par exemple, une condition), l'intégration de deux unités de code ensemble (par exemple une fonction), en passant par la réalisation bout en bout d'une fonctionnalité ou encore des performances attendues.
- Il existe également des tests doublés, c'est-à-dire des tests qui simulent le plus souvent des traitements lourds (une fonction complexe, une connexion à une base de données, l'appel d'une API extérieure, etc.) en fournissant directement le résultat attendu

Les différents tests

- Mots-clés (keyword testing)
- *Regression*
 - Vérifier qu'une mise à jour n'a pas induit des erreurs sur les autres fonctionnalités
- *Data-Driven Testing*
 - Test piloté par les données
- *Black Box Testing*
 - Test à l'aveugle sans connaître et voir le code source
- *White Box testing*
 - Test en connaissant le fonctionnement d'une fonction ou en ayant le code source sous les yeux
- **Unit** : isole une unité d'une fonctionnalité ([Acronyme First](#))
- **Intégration** : intégration d'au moins 2 unités de code
- **Fonctionnels** : teste une fonctionnalité de bout en bout
- **Performance**
 - À faire durant le développement pour détecter au plus tôt les problèmes
 - De préférence à ne pas faire en production pour éviter de perturber le fonctionnement de l'application

Les tests doublés

- Tests doublés
 - Repose sur la simulation des dépendances extérieures à une fonction.
 - Vérifier le bon fonctionnement d'une fonctionnalité sans utiliser les ressources extérieures
 - Les tests
 - *Mock*
 - *Spy*
 - *Stub*
 - *Article sur les différences entre Mock, Spy, Stub, etc.*
- Smoking testing (Build verification testing)
 - Des tests qui vérifient qu'une fonctionnalité fonctionne correctement avant d'entreprendre éventuellement des tests plus poussés



QUIZ 6

— 0-exercices/README.md/QUIZ6



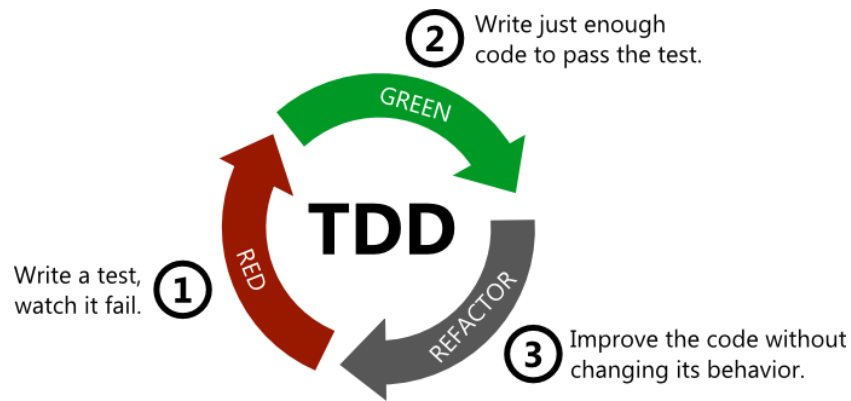
15 min

STRATÉGIE DE DEVELOPPEMENT



TEST DRIVEN DEVELOPMENT (TDD)

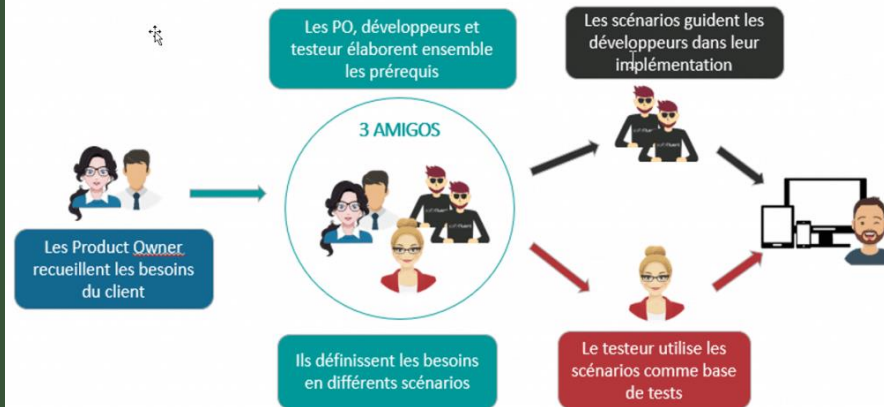
— [Source image Radixweb](#)



- Développement **piloté par les tests unitaires**
- **Écrire les tests en premier** avant d'implémenter le code
- Réduit les bugs, mais ne les empêchent pas
- N'est pas un moyen prévisible ou garanti de développer un code fonctionnel et bien organisé
- Résultat = suite des tests de régression automatisée

BEHAVIOUR DRIVEN DEVELOPMENT (BDD)

— [Source image Softfluent](#)



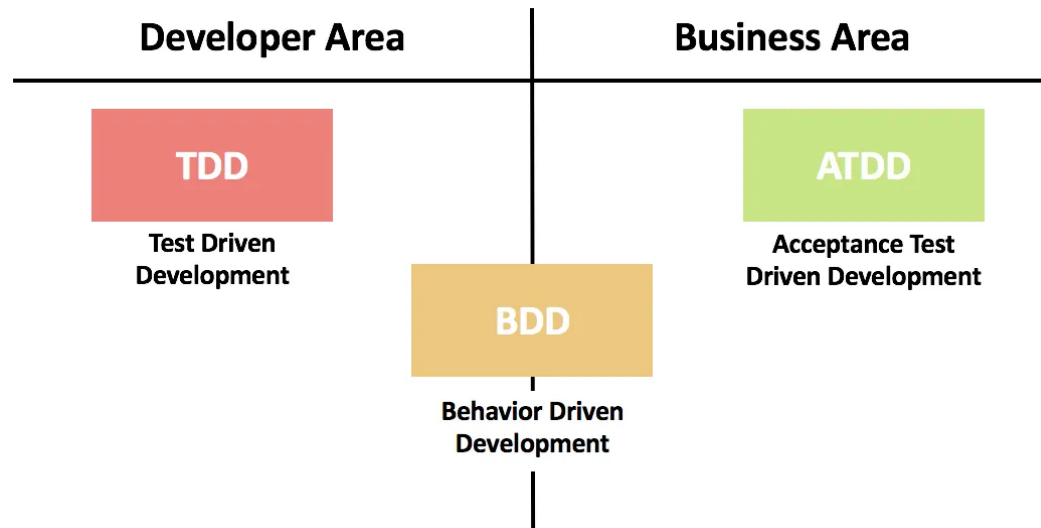
- Développement **piloté par le comportement des utilisateurs finaux**
- Des scénarios de cas d'utilisation sont rédigés en langage naturel
- Ces scénarios de cas d'utilisation servent de base d'abord à rédiger des tests puis à développer
- Avantages
 - Développer que ce qui est essentiel (LEAN)
 - Utilisation du langage naturel compris par tous
- Le BDD est généralement combiné au TDD

ACCEPTANCE DRIVEN DEVELOPMENT (ATDD)

- Développement piloté par les recommandations du client et du marché, règles métiers.
- À partir de ces recommandations
 - Les tests fonctionnels sont rédigés
 - Le développement est piloté par les tests fonctionnels
- Autrement dit, on écrit les tests fonctionnels orientés métiers avant d'écrire les autres types de tests et le code adéquat.

DIFFÉRENCES TDD, BDD, ATDD

- [Source image Widyan Ivan](#)
- [Source image Damia Mutlu](#)



Parameters	TDD	BDD	ATDD
Definition	TDD is an approach of development that focuses on implementation of features	BDD is an approach of development that focuses on system behaviors.	ATDD is a technique that focuses defining accurate requirements.
Participants	Developers	Developers, Customers, QAs	Developers , Customers, QAs
Language	Testing language of used programming language	Simple native language	Simple native language
Main Focus	Unit test	Understanding requirements	Defining acceptance criteria.



PAIR PROGRAMMING ET CODE REFACTORING

- Pair programming 2 rôles
 - **Driver** (écrit le code) et **navigator** (supervise, observe)
 - Le **Driver** se concentre sur la tactique (implémentations concrètes à court terme)
 - **Navigator** sur la stratégie (vision et solution à long terme)
 - Avantages
 - Moins d'erreurs donc une meilleure qualité de code.
 - Un moyen efficace de partager les connaissances
 - Résoudre les problèmes plus vite et plus rapidement
 - Former de nouvelles personnes et améliorer les compétences interpersonnelles et sociales.
- Code Refactoring
 - Clarification, simplification du code sans modifier le comportement
 - **Rendre le code plus lisible et maintenable**

Principes DRY ET YAGNI

– DRY

- *Don't Repeat Yourself*
- Éviter la duplication (répétition) du code
- Économiser du temps et des efforts de développement
- Faciliter la maintenance
- Réduire les risques de bugs.

– YAGNI

- *You Ain't Gonna Need It*
- Développer que ce qui sera utilisé
- Très utilisé dans le **XP** ([*Extreme Programming*](#))
 - Le XP est une application extrême des bonnes pratiques du développement logiciels



QUIZ 7

— 0-exercices/README.md/QUIZ7



15 min

DEVOPS ET METRICS





DEVOPS

— Vision

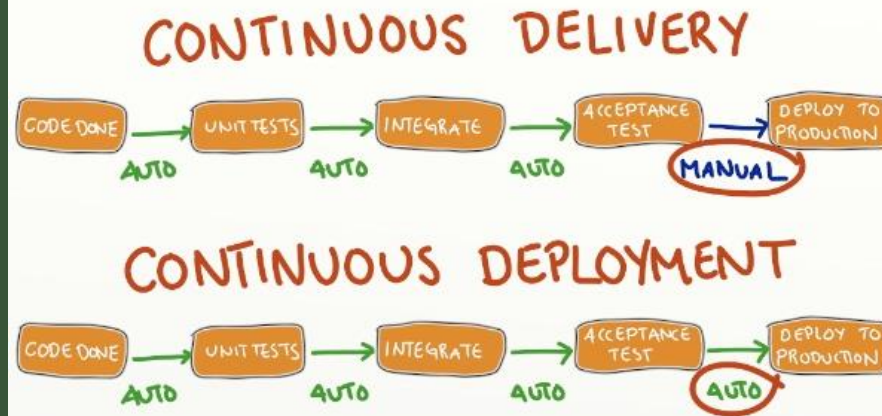
- Accélérer le flux du travail en automatisant les ***builds*** et ***release pipelines***
- Un *build* est la compilation de l'application à partir de son code source en fichier (s) exécutable (s)
- Un *release pipeline* est un ensemble d'étapes pour déployer une application dans un environnement (*preprod*, *test*, *prod*, etc.)
- Vision d'un produit qui peut être amélioré et optimisé en continu

— CI/CD

- Prévenir les problèmes d'intégration en détectant rapidement les *builds* défectueux et en réduisant les efforts et les risques lors de l'intégration des changements.
- **CI : *Continuous Integration***, intégrer régulièrement les modifications du code source en utilisant par exemple un logiciel comme [Git](#). Autrement dit, chaque modification du code doit être intégrée régulièrement sur Git.
- Dans le processus du CI, on met en place une à plusieurs pipelines dédiées aux tests automatisés et s'assurant qu'ils passent tous avant d'intégrer les modifications effectuées. Dans le cas où au moins un test échouerait, toutes les modifications sont rejetées et ne sont pas intégrées

DEVOPS

- [Source image blog Crisp](#)



- CD pour ***Continuous Delivery***
 - Automatisation du déploiement d'une application nécessitant à un moment **une intervention humaine**
- CD pour ***Continuous Deployment***
 - Automatisation totale du déploiement en continu **sans intervention humaine**

MÉTRIQUES CODE ET MÉTRIQUES QUALITÉ DE CODE

— Code metrics

• **Code coverage**

- Quantité de code source exécuté par les tests automatisés

• **Lignes de code** (Line of Code (LOC) ou Source Lines of Code (SLOC))

- Nombre de ligne d'un fichier, programme ou projet dans sa globalité

— Code Quality metrics

• **Depth of inheritance** (profondeur de l'héritage)

• **Class coupling** (nombre de classes utilisées par une classe donnée)

• **Cyclomatic complexity** (nombre de sortie possible avec par exemple les if, elseif...else)

• **Efferent coupling**

• **Afferent coupling**

— Bénéfices

- Identifier plus facilement les bugs
- Réduire risque de bug
- Réduire les coûts de développement
- Maintenir plus facilement

AFFERENT, EFFERENT COUPLING

```
<?php

class Computer {
    private $_os;
    public function __construct()
    {
        $this->_os = new OS();
    }
}

class OS {
    public function __construct()
    {
    }
}
```

- La classe **Computer** est efférente, elle dépend (utilise) de la classe **OS**
- La classe **OS** est afférente car elle est utilisée par la classe **Computer**
- Les modifications de la classe **OS** peuvent affecter le fonctionnement de la classe **Computer**
- Efferent Coupling (couplage efférent) mesure le nombre de classes dont dépend une classe donnée
 - ▶ C'est une **métrique de qualité** de code
 - ▶ Une classe avec un **nombre élevé** de couplage **efférent** peut indiquer **une certaine instabilité** de cette classe puisqu'elle dépend de la stabilité des autres classes



QUIZ 8

— 0-exercices/README.md/QUIZ8



15 min

ARCHITECTURE



ARCHITECTURE

- Dans le cadre SCRUM

- L'équipe doit **consacrer du temps à la réflexion de l'architecture** avant l'implémentation d'une fonctionnalité
- L'architecture est **émergente**, elle **n'est pas figée ou prévue à l'avance**. Elle émane de la **compréhension de plus en plus précise du produit dans le temps**.
- L'équipe doit mettre en place des **bonnes pratiques** à appliquer et à respecter par tout le monde

- Architecture SPIKE

- Prototype d'architecture visant à **tester rapidement** une architecture pour vérifier une hypothèse
- Prototype est généralement **"sale"**, implémenté rapidement sans respecter les standards à la lettre
- La **finalité** du prototype est **d'être supprimé** après avoir répondu ou donné une orientation de la direction à prendre

CONCEPTION

– Design Pattern

- Solution générale et réutilisable à un problème courant dans un contexte donné de conception logiciels
- Pas une conception finie qui peut être transformé directement en code
- Description ou modèle de résolution d'un problème

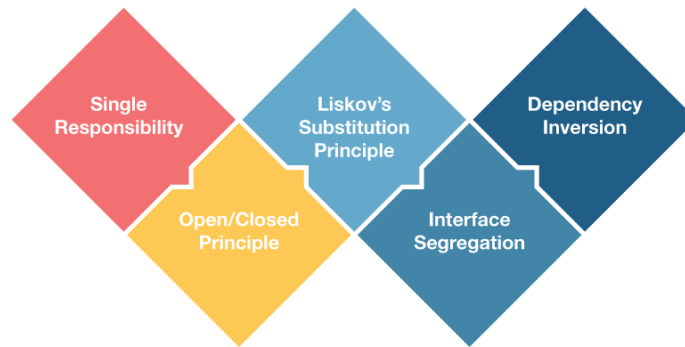
– EDUF

- ***Enough Design Up Front***
- Une conception **suffisante** dès le départ constitue une bonne base et permet d'atténuer les risques, sans perte de temps inutile.
- Il est fortement conseillé de dédier des heures à la conception lors du Sprint
- Utiliser les bonnes pratiques, Design Pattern qui sont compris et suivi par tous les Developers



— [Source image Brahmatechnolab](#)

S.O.L.I.D.



- Principe de programmation logicielle qui permet d'avoir des solutions plus faciles à maintenir et à faire évoluer
- 1. **Single Responsibility Principle**
 - Empêche d'avoir des classes trop surchargées
- 2. **Open-Closed Principle**
 - Permettre l'ajout d'une nouvelle fonctionnalité sans modifier le code déjà existant
- 3. **Liskov Substitution Principle**
 - Les sous-classes sont interchangeables sans affecter le programme
- 4. **Interface Segregation Principle**
 - Réduit le couplage entre les classes en utilisant plusieurs interfaces spécifiques
- 5. **Dependency Inversion**
 - Dépendre des abstractions (regroupement des caractéristiques communes mais qui seront spécifiés par les classes filles à leurs manières) et non des implémentations (réalisation)
- [Article sur le sujet](#)



QUIZ 9

— 0-exercices/README.md/QUIZ9



10 min

DETTE TECHNIQUE ET BUG



DETTE TECHNIQUE

— Conséquences

- **Augmente le coût de la maintenance**
- **Réduit la performance de l'équipe** à produire des nouvelles fonctionnalités
- Conséquence à long terme des mauvaises solutions pris pour résoudre un problème à court terme
- Reflète le travail de développement supplémentaire qui survient lorsque du code facile à mettre en œuvre à court terme est utilisé au lieu d'appliquer la meilleure solution globale.

— Préventions

- Code Reviews
- Analyse statique du code
- Réduire la dette technique
- Amélioration du DoD
- Standards de code



— [Source image DevriX](#)



1. Titre clair et approprié au langage technique
2. Un bug par report, simple et reproductible
3. En attente de résultats et d'observations des résultats
4. Version du build ou le bug a été trouvé
5. Des captures d'écran des interfaces concernés par le bug



QUIZ 10

— 0-exercices/README.md/QUIZ10

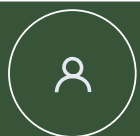


15 min



Simulation exams

— 2-exams/README.md



60 min
pour chaque

Conseils pour le passage de la certification

- Soyez **frais** et **bien reposé**
- Idéalement passez votre certification dans la **matinée** avant 11H
- Avant le passage de la certification, **économisez vos forces**, c'est trop tard pour réviser
- Evitez de refaire des sessions d'une heure juste avant
- Depuis votre ordinateur, gardez uniquement l'onglet du passage du test et le SCRUM GUIDE
- Supprimez et enlevez toutes les autres distractions (autres pages web, SmartPhone, Discord, Slack, etc)

FIN

BON COURAGE !
GLODIE TSHIMINI
CONTACT@TSHIMINI.FR

