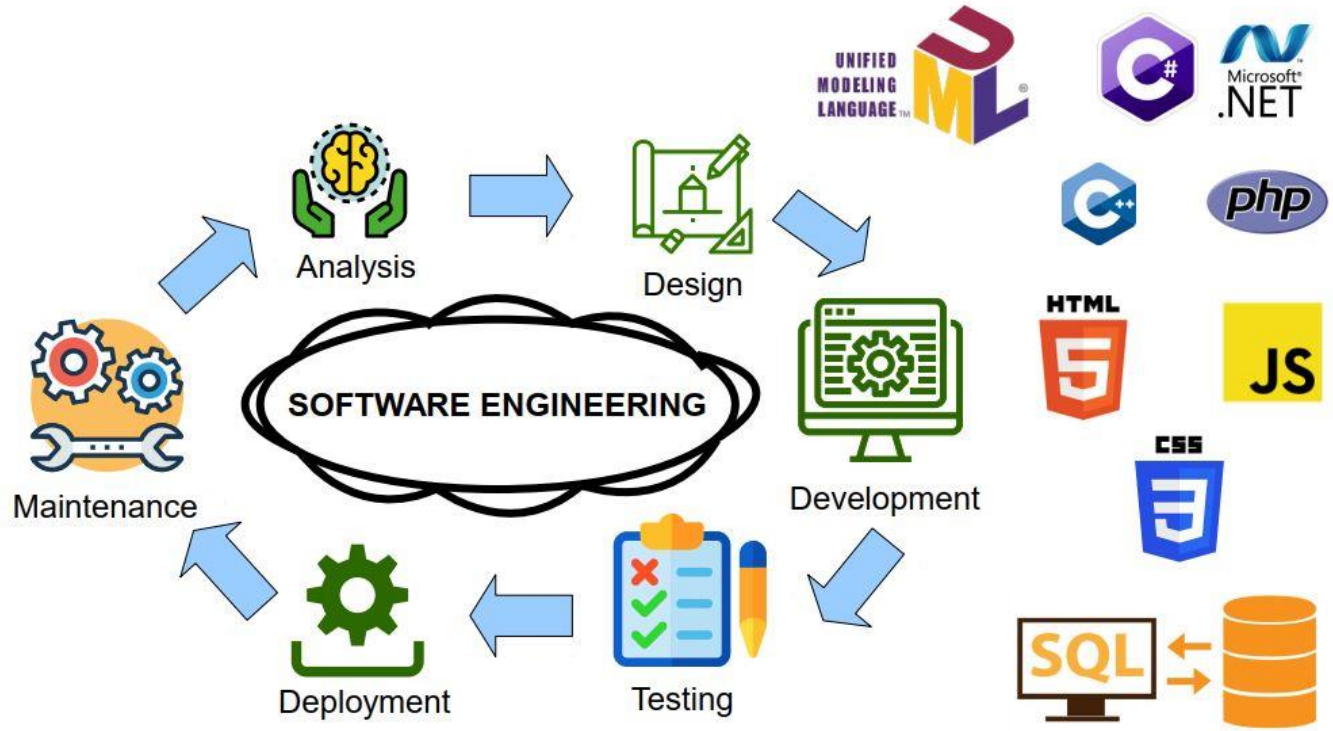
Ingénierie des Processus de Développement Logiciel (IPDL)



Dr. El Hadji Bassirou TOURE

Ecole Supérieure Polytechnique (E.S.P)

2021 - 2022

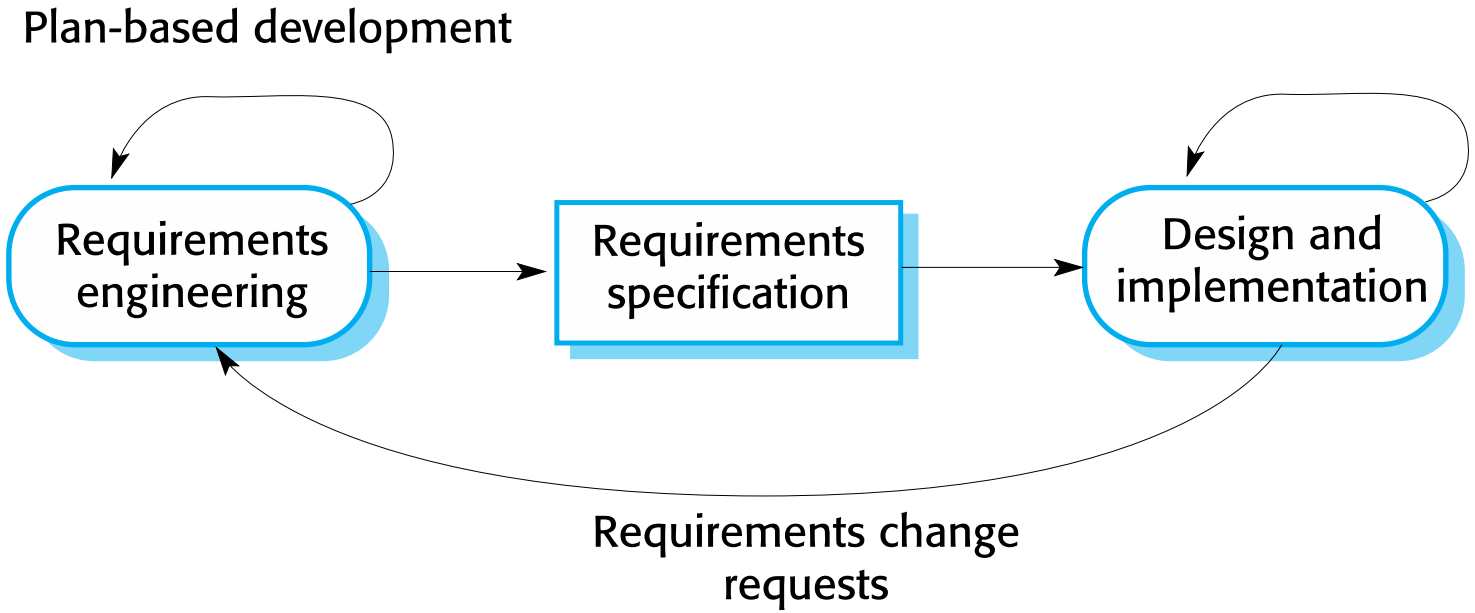
Développement rapide de logiciels

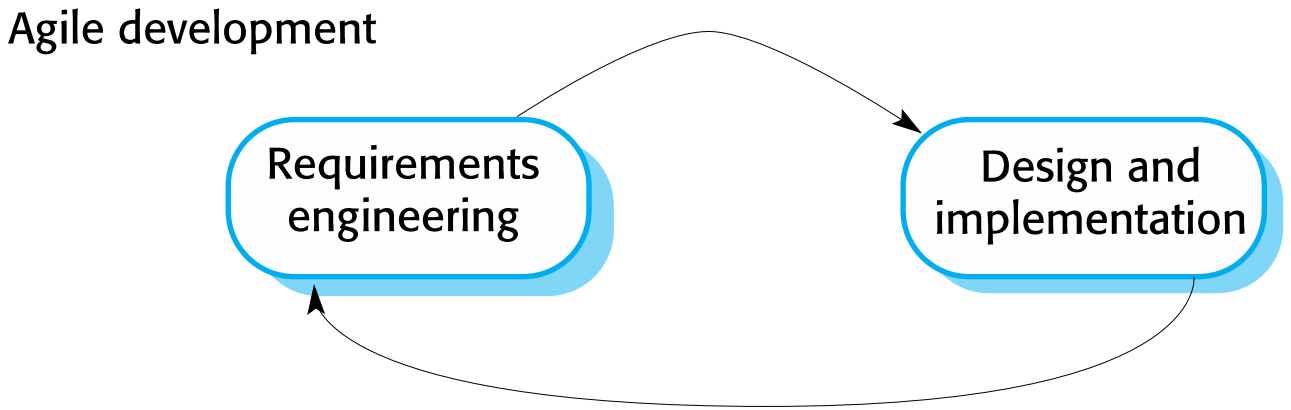
- Le développement et la livraison rapides sont devenus la plus importante des exigences pour les systèmes logiciels.
  - Les besoins métiers changent rapidement et il est pratiquement impossible de définir un ensemble fixe de besoins
  - Le logiciel doit évoluer pour s'accorder à ces changements métiers.
- Le développement planifié est essentiel pour certains types de systèmes mais n'est pas en mesure de remplir ces exigences.
- Les méthodes de développement agiles ont émergé à la fin des années 90 et ont pour but de réduire drastiquement les durées de livraison des systèmes logiciels de qualité.

Développement agile

- La spécification, la conception et l'implémentation sont indissociées
- Le système est développé comme une série de versions ou d'incréments avec les acteurs impliqués dans la spécification des versions ainsi que leur évaluation.
- Livraison fréquente de nouvelles versions pour évaluation
- Utilisation extensive d'outils tels que les test automatisés
- Peu de documentation (se concentre sur du code qui marche)

Développements planifié et agile





Développements planifié et agile

- Développement planifié
  - Une approche planifiée est basée sur un ensemble de phases de développement avec des résultats, à produire à chacune de ces phases, planifiés à l'avance.
  - Les itérations se produisent dans les activités
- Développement agile
  - La spécification, la conception, l'implémentation et les tests sont indissociés et les résultats de ces phases sont décidés à partir de compromis pendant le processus de développement du logiciel.

Méthodes agiles

Méthodes agiles

- L'insatisfaction due aux nombreuses considérations générales des méthodes de développement des années 80 et 90 a conduit à la création des méthodes agiles :
  - Se concentrent sur le code plus que sur la conception
  - Se basent sur une approche itérative de développement de logiciels
  - Doivent livrer rapidement des logiciels qui fonctionnent et évoluent rapidement pour satisfaire les changements des exigences.
- Le but des méthodes agiles est de réduire les lourdeurs des processus logiciels (en limitant par exemple la documentation) et d'être capable de répondre rapidement aux changements des exigences tout en minimisant les reprises.

Manifest agile

- *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
  - *Individuals and interactions over processes and tools*
  - *Working software over comprehensive documentation*
  - *Customer collaboration over contract negotiation*
  - *Responding to change over following a plan*
- *That is, while there is value in the items on the right, we value the items on the left more(http://agilemanifesto.org/) .*

Principes des méthodes agiles

Principe	Description
Implication du client	Les clients doivent être très fortement imliqués durant tout le processus de développement. Leur rôle est d'identifier et prioriser les nouvelles exigences, et d'évaluer chaque itération du système.
Livraison incrémentale	Le logiciel est développé en incréments, avec le client qui spécifie les exigences à inclure dans chaque incrément.
Les personnes, non les processus	Les aptitudes des équipes de développement doivent être identifiées et exploitées. Les membres des équipes peuvent employer leurs propres techniques de développement sans avoir à suivre, à la lettre, un processus prescriptif.
Inclusion des changements	Il est attendu que les exigences du système changent, et donc que la conception du système change pour satisfaire ces changements.
Maintenir la simplicité	Se concentre sur la simplicité à la fois du système et du processus de développement. Eliminer autant que possible toute complexité.

Utilisation des méthodes agiles

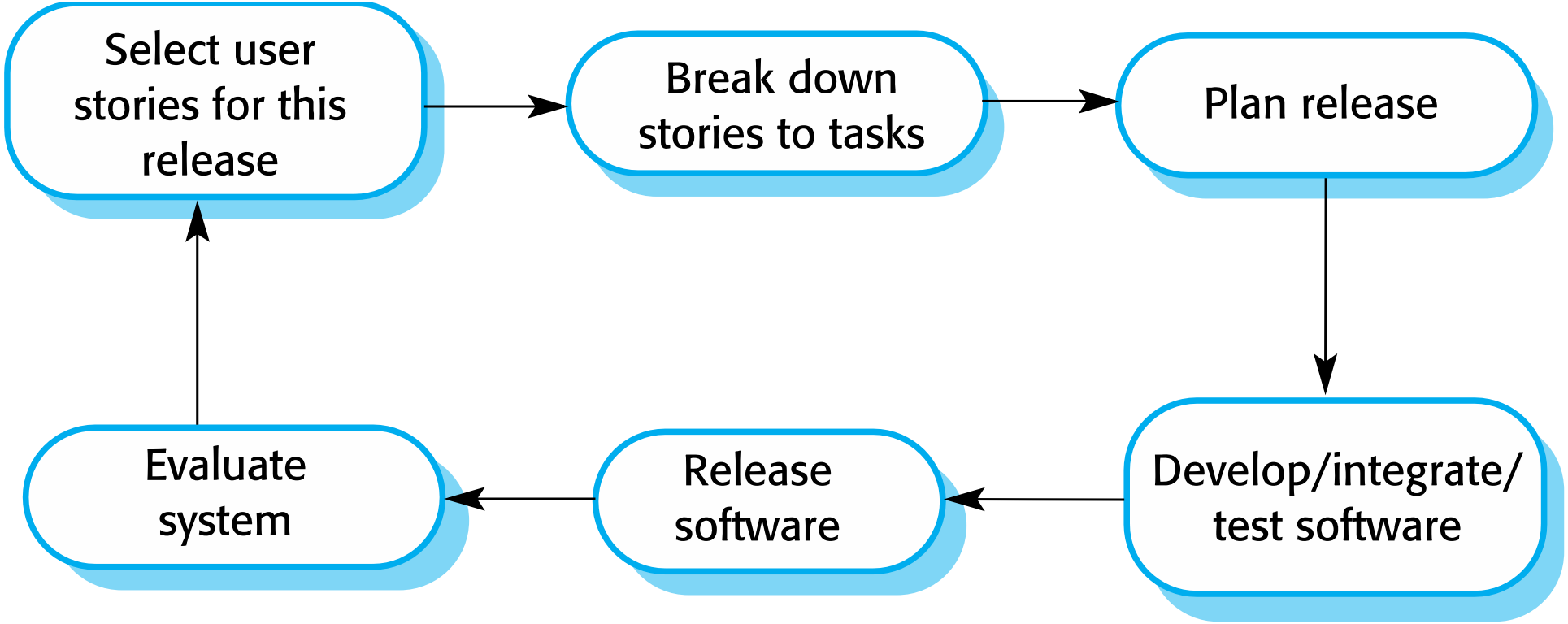
- Développement de produits, de petite ou moyenne taille, à vendre.
  - Presque tous les produits et applications sont maintenant développés en utilisant une approche agile.
- Développement de produits personnalisés,
  - où il y a une implication claire du client dans le processus de développement
  - et où il n'y a que très peu de règles et de considérations externes pouvant affecter le logiciel.
- Leur succès est dû à une communication constante entre l'équipe de développement et le client ou le gestionnaire du produit.

Techniques de développement agile :
Exemple XP

Extrem programming

- Une méthode agile très influente, dévelopée à la fin des années 90, qui a inspiré un grand nombre de techniques de développement agile.
- Extrem Programming (XP) prend une approche “extreme” de développement itératif :
  - De nouvelles versions peuvent être construites plusieurs fois/jour;
  - Les incréments sont livrés au client toutes les 2 semaines;
  - Tous les codes doivent être testés lorsqu’un nouveau code est integré dans le système.

Extrem programming : cycle de livraison



Pratiques d'Extrem Programming (1)

Principe ou pratique	Description
Planification incrémentale	Les exigences sont enregistrées dans des cartes de scénario. Les scénarios à inclure dans une livraison sont définies selon leur disponibilité et leur priorité. Les développeurs découpent ces scénarios en tâches.
Release	Un ensemble minimal de fonctionnalités qui fournissent une valeur ajoutée au dévelopement du système est développé en premier. Les releases sont fréquentes et ajoutées de manière incrémentale à la première.
Conceptions simples	Une conception suffisante, et pas plus, est définie pour prendre en compte les exigences courantes.
Tests en premier(test-first development)	Un cadre de tests unitaires automatisé est mis en place pour écrire des tests pour de nouvelles fonctionnalités avant que ces dernières ne soient implémentées.
Révision de codes (Refactoring)	Il est attendu des dévelopeurs qu'ils révisent leurs codes de manière continue dès que des améliorations du code sont possibles. Cela permet d'avoir un code simple et maintenable.

Pratiques d'Extrem Programming (2)

Pair programming (Programmation par binôme)	Les développeurs travaillent par pairs, chacun vérifiant le travail de l'autre. Ce qui apporte de la qualité au travail effectué.
Propriété collective	Les binômes travaillent dans toutes les parties du système, tous prennent en charge la responsabilité totale et non partielle du code. N'importe qui peut modifier n'importe quoi.
Intégration continue	Dès que le travail sur une tâche est achevé, elle est intégrée dans le système. Après cela, tous les tests unitaires du système doivent réussir. Le code est toujours testé, il est donc parfait. On code, on teste, on incorpore au projet.
Rythme maitrisé	Les heures supplémentaires ne sont pas acceptables car ont pour effet de réduire la qualité du code et, à moyen terme, la productivité. En effet, on ne code pas plus de 40h par semaine, car un développeur code mieux quand il est en bon état de forme.
Client sur place	Un représentant de l'utilisateur final du système (client) doit faire partie de l'équipe XP à temps plein. Les clients ont pour rôle d'apporter les exigences à l'équipe de développement.

Principes « agiles » ➔ XP

- Développement incrémental ➔
  - Livraison de petites et fréquentes versions du système.
- Implication du client ➔
  - Engagement du client avec l'équipe, à temps plein.
- Personnes plutôt que processus ➔
  - Programmation par pairs + Responsabilité collective (tout en évitant les trop longues heures de travail).
- Prise en compte des changements ➔
  - Production régulière de versions.
- Maintenance de la simplicité ➔
  - Révision constante du code.

Influences des pratiques XP

- XP a une préoccupation technique et ce n'est pas aisé de la mettre en oeuvre dans la plupart des organisations.
- En effet, même si le développement agile utilise des pratiques XP, cette dernière comme définie originellement n'est pas largement utilisée.
- Pratiques clés
  - Scénarios pour la spécification
  - Refactoring du code
  - Développement de tests en premier
  - Programmation par pairs ou en binôme

Scénarios (user stories)

- En XP, un client ou utilisateur fait partie de l'équipe XP et c'est lui qui détermine quels sont les besoins du système.
- Ces besoins sont traduits en scénarios.
- Ces derniers sont écrits sur des cartes et les équipes de développement les découpe en plusieurs tâches d'implémentation.
- Ces tâches permettent de déterminer une estimation d'un calendrier et des coûts.
- Le client choisit les scénarios à inclure dans la prochaine version en se basant sur les priorités et le calendrier défini.

Refactoring (révision ou refonte)

- Les équipes de programmation cherchent à améliorer le code dès que possible et pas seulement si ces améliorations deviennent un besoin.
- Ceci améliore la compréhension du logiciel et réduit le besoin de documentation.
- Les changements sont faciles à effectuer parce que le code est bien structuré et clair.
- Cependant, certains changements nécessitent un refactoring de l'architecture. Ce type de changements est plus couteux.

Exemples de refactoring

- Réorganisation d'une hiérarchie de classe pour enlever du code dupliqué.
- Ordonner et renommer les attributs et les méthodes pour les rendre plus compréhensifs.
- Remplacement d'un code en ligne avec des appels de méthodes inclues dans une librairie.

Développement dirigé par les tests
Test-driven development (TDD)

- Le test est au coeur de XP et XP a développé une approche telle que le programme est testé après chaque changement effectué.
- Caractéristiques des tests XP :
  - TFD (testing – coding – refactoring).
  - Développement de test incrémental à partir des scénarios.
  - Implication du client dans le développement et la validation des tests.
  - Des tests automatisés sont utilisés pour exécuter tous les tests de composant à chque fois qu'une nouvelle version est créée.

Développement de tests en premier
Test-first development (TFD)

- Ecrire des tests avant le code clarifie les besoins à implémenter.
- Les tests sont écrits comme des programmes et non comme des données, ils pourront être exécutés.
- Le test englobe un vérificateur qui statue si le test est correct ou non.
- Généralement on utilise un cadre (framework) de test tel que Junit.
- Tous les tests précédents et les nouveaux tests sont exécutés automatiquement, lorsqu'une nouvelle fonctionnalité est ajoutée.

Implication du client

- Le rôle du client dans le processus de test est d'aider à écrire des tests pour les scénarios qui seront développés dans la prochaine version du système.
- Le client qui fait partie de l'équipe définit les tests sous la forme de buts.
- Tout nouveau code est validé pour assurer que c'est ce dont le client veut.
- Cependant, les clients ont des temps limités et ne peuvent pas faire partie de l'équipe à temps plein.

Automatisation des tests

- Signifie que les tests sont écrits comme des des composants exécutables avant que la tâche associée ne soit implémentée.
  - Ces composants de test doivent être autonomes, prendre les entrées à tester et vérifier que le résultat produit satisfait les spécifications. Un cadre de test automatisé comme Junit est un système facilitant l'écriture de tests exécutables et pouvant prendre un ensemble de tests à exécuter.
- Vu que les tests sont automatisés, il y a toujous un ensemble de tests pouvant être exécutés rapidement et facilement.
  - Dès qu'une fonctionnalité est ajoutée au système, les tests sont exécutés et les problèmes que le nouveau code a engendré pourront être rapidement détectés.

Problèmes du TFD

- Les programmeurs préfèrent programmer que tester. Ils peuvent prendre des raccourcis pour écrire les tests (incomplets : ne prenant pas en compte toutes les exceptions pouvant survenir).
- Certains tests ne sont pas évidents à écrire de manière incrémentale.
- Difficile pour un test d'être complet.

Programmation par pairs

- Elle implique que les programmeurs travaillent par pairs.
- Elle favorise la propriété commune et le partage de connaissance à travers l'équipe.
  - Réduit les risques d'échec du projet si un membre venait à quitter l'équipe.
- Cela peut également être vu comme un processus de contrôle informel car le code est analysé par plus d'une personne.
- Elle encourage aussi le refactoring, l'équipe entière pourra bénéficier d'une amélioration de code.

Méthodes de Gestion de projet agile :
Exemple Scrum

Gestion de projet agile

- La principale responsabilité d'un gestionnaire de projet est de faire en sorte que le logiciel soit livré à temps et selon le budget imparti au projet.
- L'approche standard de gestion de projet est planifiée.
  - Le gestionnaire établit, pour le projet, un plan représentant ce qui doit être livré, quand et ceux qui vont les développer.
- La gestion de projet "agile" exige une approche différente adaptée au développement incrémental et aux pratiques utilisées par les méthodes agiles.

Scrum

- Scrum est une méthode itérative et incrémentale permettant la gestion effective de produits ou de développements logiciels.
- Il est composé de trois phases :
  - La phase initiale est celle où sont établis les objectifs généraux pour le projet et où est conçue l'architecture logicielle.
  - Elle est suivie par une série de cycles de sprint, telles que chaque cycle développe un incrément du système.
  - La phase de clôture du projet conclut le projet, complète la documentation nécessaire (par exemple le manuel d'utilisateur) et évalue le projet dans sa globalité.

Terminologie Scrum

Terme	Définition
Equipe de développement	Un ensemble de développeurs logiciels, pas plus de 7, responsables du développement du logiciel et de l'écriture d'autres documents importants pour le logiciel.
Incréments potentiellement livrables (Potentially shippable product increment)	Un incrément découle d'un sprint. L'idée est que ce dernier doit pouvoir être livré au client, ce qui signifie qu'il est dans un état fini et il ne reste qu'à l'inclure dans le produit final. Dans la pratique, ce n'est pas toujours évident.
Product backlog	Il s'agit d'un ensemble d'éléments " à faire " pour l'équipe de développement. Ces éléménts peuvent aller de la définition des exigences, à celle de l'architecture ou du manuel d'utilisateur.
Product owner	Un individu (ou un petit groupe) dont le travail consiste à identifier les caractéristiques ou les exigences du produit, les classe par ordre de priorité, et effectue une vérification continue du product backlog pour assurer que le produit satisfait toujours les exigences métiers. Il peut s'agir du client ou du gestionnaire de produit.

Terminologie Scrum

Terme	Définition
Scrum	Une reunion journalière de l'équipe Scrum qui vérifie les progrès, et classe par priorité les tâches à effectuer ce jour là. Idéalement, il s'agit d'un court face à face entre les membres de toute l'équipe.
ScrumMaster	Assure que le processus Scrum est suivi rigoureusement par l'équipe de développement. Il joue le role d'intermédiaire entre l'équipe de développement et le reste de l'équipe.
Sprint	Un iteration de développement, d'une durée comprise entre 2 à 4 semaines.
Velocité	Une estimation de l'effort de développement que nécessitera un sprint particulier.

Cycle de sprint Scrum