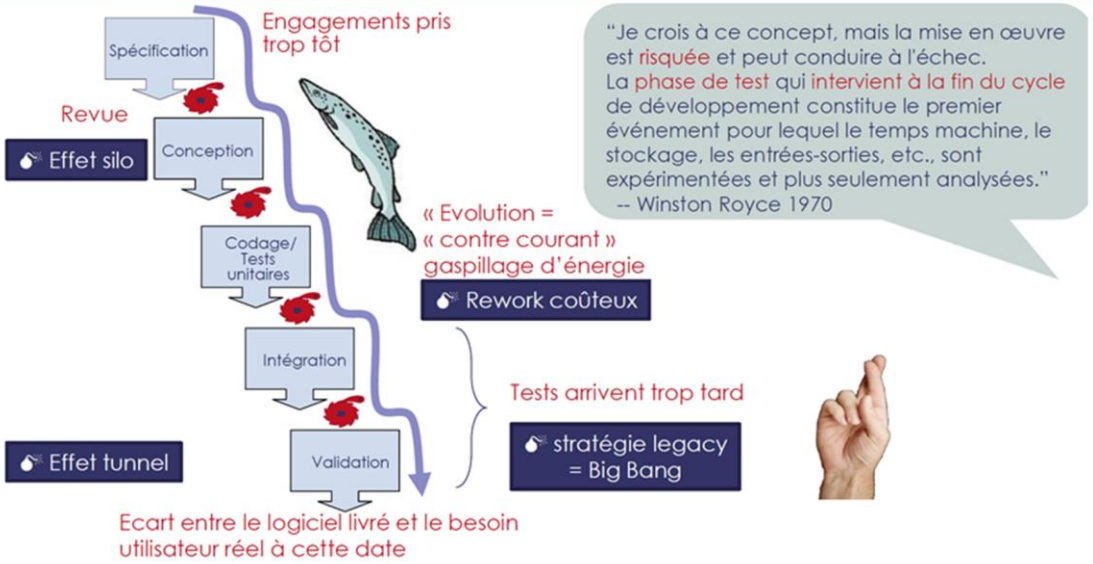
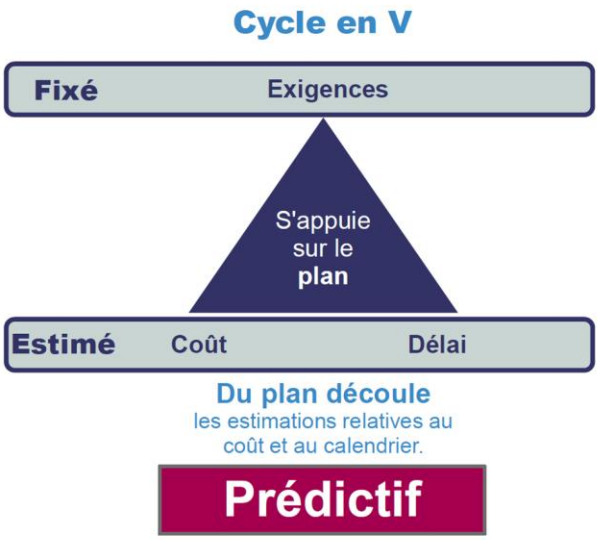
Formation aux méthodes Agiles

Formation aux méthodes Agiles

Les problèmes inhérents au modèle en cascade

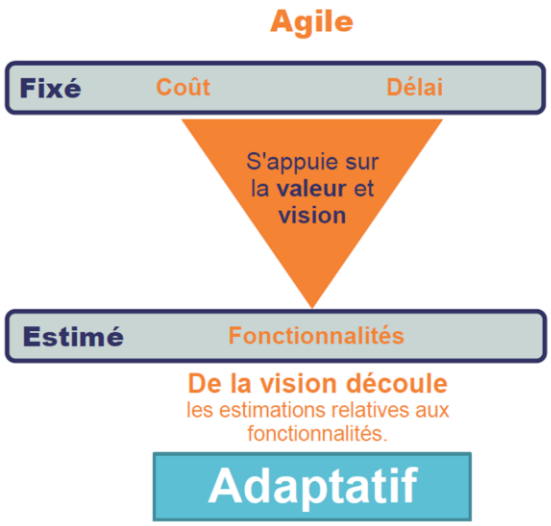


Modèle de gouvernance cycle en V

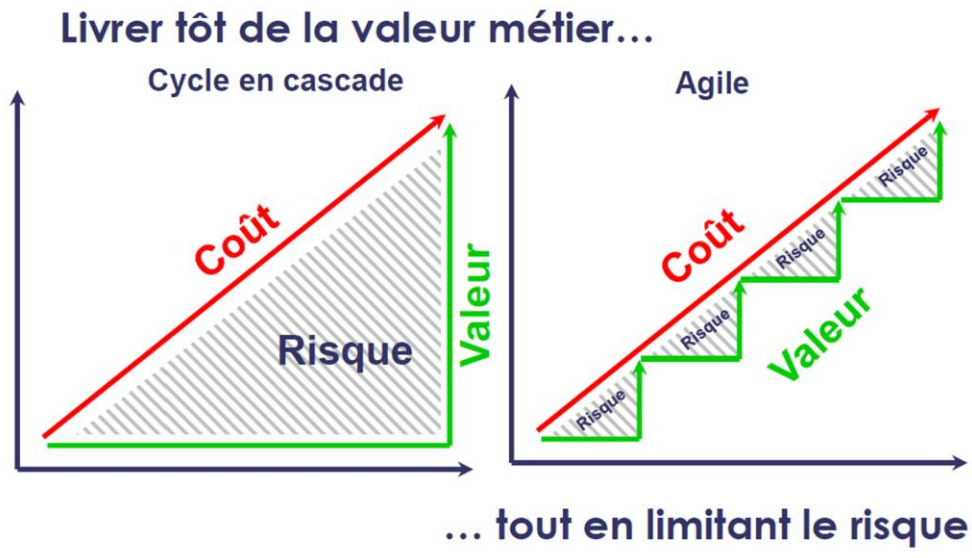


Définir l'agilité

Modèle de gouvernance Agile



Des livraisons petites et fréquentes



Manifeste Agile : l'humain est au centre

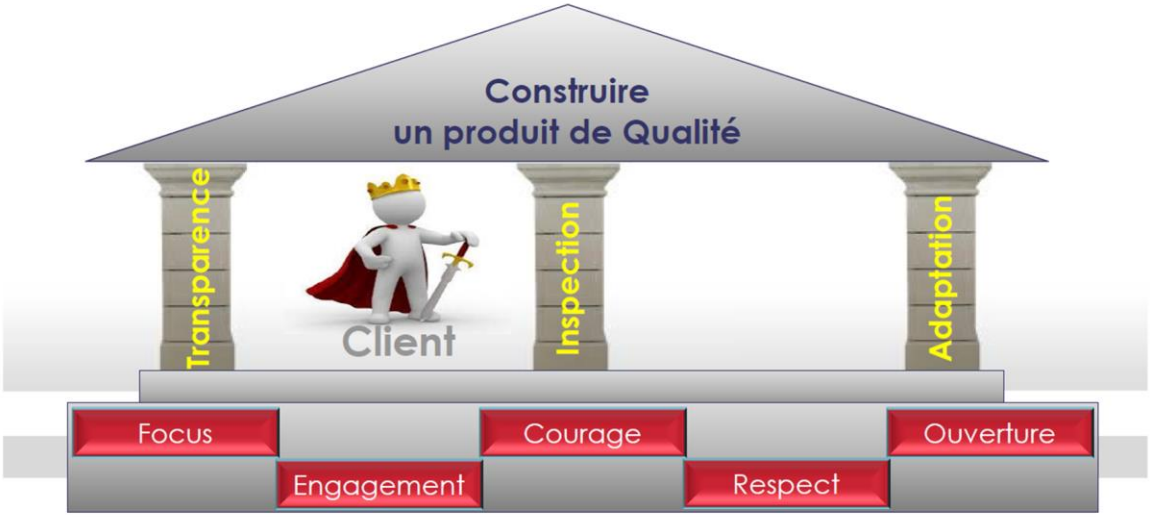
1. Notre plus haute priorité est de satisfaire le client en livrant rapidement et régulièrement des fonctionnalités à grande valeur ajoutée.
2. Accueillez positivement les changements de besoins, même tard dans le projet.
   Les processus Agiles exploitent le changement pour donner un avantage compétitif au client.
3. Livrez fréquemment un logiciel opérationnel avec des cycles de quelques semaines à quelques mois et une préférence pour les plus courts.
4. Les utilisateurs ou leurs représentants et les développeurs doivent travailler ensemble quotidiennement tout au long du projet.
5. Réalisez les projets avec des personnes motivées. Fournissez-leur l'environnement et le soutien dont ils ont besoin et faites-leur confiance pour atteindre les objectifs fixés.
6. La méthode la plus simple et la plus efficace pour transmettre de l'information à l'équipe de développement et à l'intérieur de celle-ci est le dialogue en face à face.
7. Un logiciel opérationnel est la principale mesure d'avancement.
8. Les processus Agiles encouragent un rythme de développement soutenable. Ensemble, les commanditaires, les développeurs et les utilisateurs devraient être capables de maintenir indéfiniment un rythme constant.
9. Une attention continue à l'excellence technique et à une bonne conception renforce l'Agilité.
10. La simplicité – c'est-à-dire l'art de minimiser la quantité de travail inutile – est essentielle.
11. Les meilleures architectures, spécifications et conceptions émergent d'équipes auto-organisées.
12. À intervalles réguliers, l'équipe réfléchit aux moyens de devenir plus efficace, puis règle et modifie son comportement en conséquence.

Source : www.agilemanifesto.org (2001)

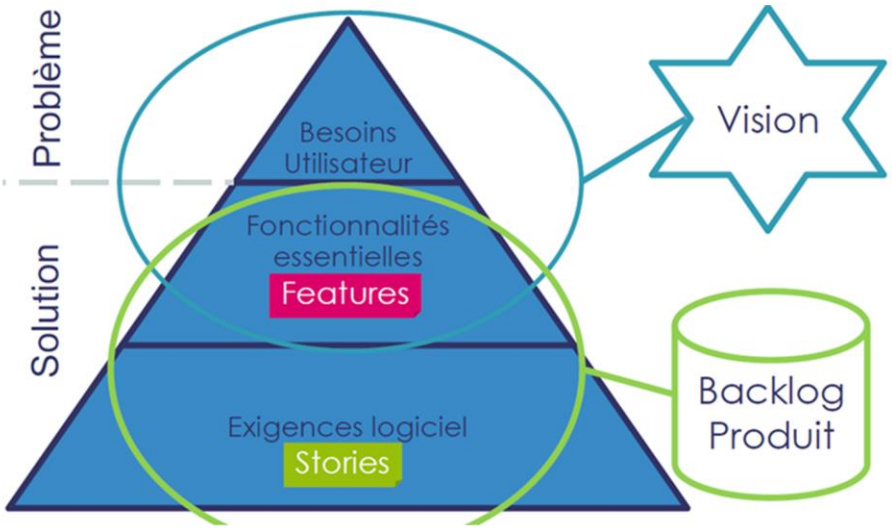
Les méthodes Agiles

Histoire de Scrum

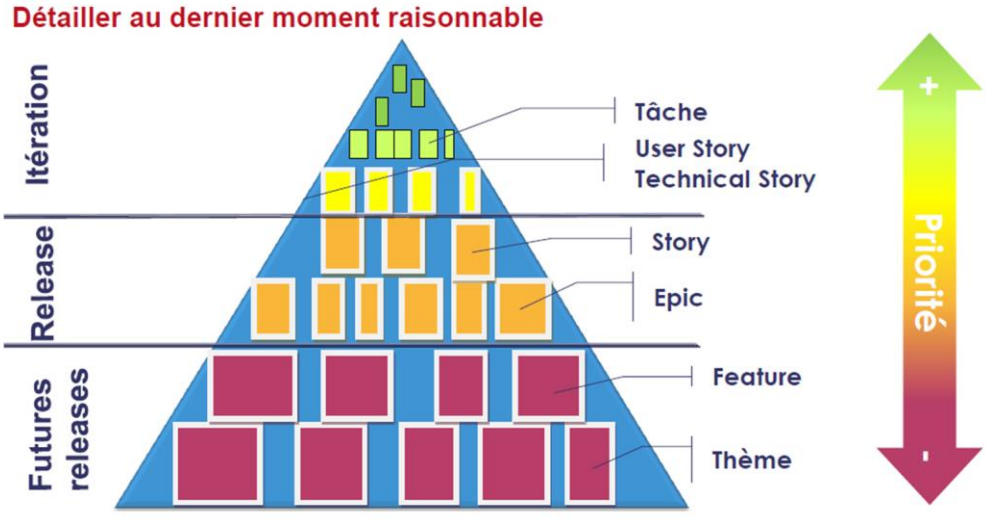
Les fondements de Scrum : 3 piliers et 5 valeurs



De la vision aux stories



Principe de l'iceberg et du juste à temps



Critères d'acceptation

Détails associés à une User Story qui permettent de la développer et de la valider

79	Ajout photo page perso	8

En tant qu'utilisateur enregistré, ***je veux*** pouvoir ajouter ma photo sur ma page personnelle ***afin que*** mes amis m'identifient sans ambiguïté

1. Vérifier que l'utilisateur peut uploader sa photo
2. Vérifier que la photo est affichée sur la page personnelle
3. Vérifier qu'un message de confirmation de chargement est affiché
4. Vérifier que la photo n'est visible que par les amis déclarés
5. Vérifier que l'on peut charger une nouvelle photo en écrasant l'ancienne
6. Vérifier que la photo précédente n'est pas gardée

Exemple de découpage

43	Login utilisateur	8

En tant qu'utilisateur je veux pouvoir utiliser mon identifiant et mot de passe d'employé pour accéder à l'application

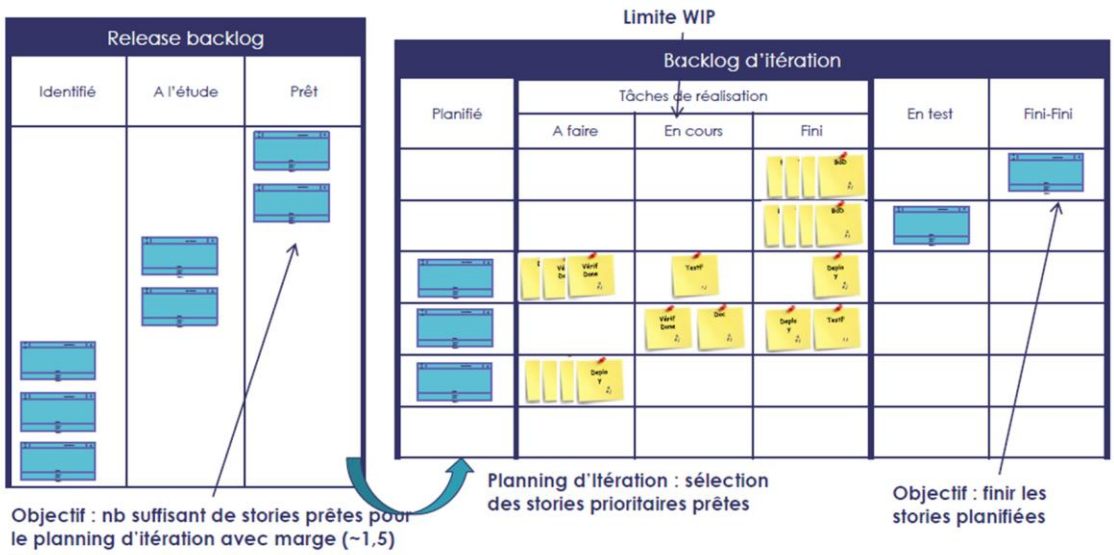
Etude 1 j | TestU 1,5 j | Dev Java 1,5 j | BdD 0,5 j

Doc 0,5 j | Vérif Done 0,5 j | TestF 1 j | Deploy 0,5 j

Critère 1 **Critère 2**

Critère 3 **Critère 4**

ScrumBan pour l'itération courante



Glossaire

Burndown Chart :
Représentation graphique de l'avancement des travaux au cours d'une itération. La courbe représente la prévision de l'évolution du reste à faire et l'évolution de celui-ci jour après jour.

Product Charter (Charte de Produit) :
L'équipe résume, dans un document très court, idéalement susceptible d'être affiché aux murs de l'espace de travail au format d'une feuille de « paperboard », les enjeux principaux du projet, son périmètre et les accords réciproques passés avec les commanditaires du projet.

Daily meeting (ou Scrum ou Stand-up meeting) :
Réunion quotidienne de 15 minutes maximum ou chaque membre de l'équipe répond à trois questions :
Ce que j'ai terminé depuis le dernier daily meeting ?
Ce que je vais terminer d'ici le prochain daily meeting ?
Quels problèmes je rencontre ?

Définition du Terminé (Definition of Done) :
La Définition du Terminé est la checklist des activités requises (écriture du code, des commentaires, de tests unitaires avec une couverture cible, des documents …) pour produire un logiciel et qui apportent une plus-value à ce logiciel.
Comme cette définition est co-élaborée par l'équipe, elle permet une mesure de l'avancement non ambiguë et responsabilisante pour chacun de ses membres.
Une définition du terminé spécifique est élaborée pour les fonctionnalités, les itérations et les versions.

Démonstration :
Voir Sprint Review

Dette technique :
Plus un logiciel a des imperfections techniques (architecture bancale, code mal écrit, standard de codage pas suivi), plus les étapes ultérieures de développement de ce logiciel couteront cher. Le code non testé et les tests non couvrants constituent souvent une part importante de la dette. Comme toutes les dettes, la dette technique devra être payée : plus elle sera payée tard, plus elle sera payée cher (intérêts).

Epic :
Une epic est une grosse story. Elle est en attente de décomposition en stories "normales". L'intérêt d'avoir des epics est de ne pas décomposer trop tôt : il est préférable de le faire au dernier moment raisonnable.

Facilitation :
Un facilitateur est une personne désignée pour animer une réunion. Il ou elle s'interdit généralement de participer au contenu de la réunion, mais se cantonne à créer les conditions dans lesquelles un groupe en réunion pourra parvenir à l'objectif qu'il s'est préalablement fixé.

La facilitation est une compétence à part entière, dont les tenants et les aboutissants dépassent largement le cadre des approches Agiles

Feature :
L'élaboration de la vision du produit passe par l'identification des features. Le terme features est abondamment utilisé dans le domaine de l'ingénierie des exigences. Une méthode agile l'utilise d'ailleurs dans son nom : Feature Driven Development ou FDD. OpenUp définit une feature comme un service fourni par le système, observable de l'extérieur, et qui remplit directement un besoin d'une partie-prenante.

Incrément :
L'incrément est le livrable d'une itération. Il doit respecter le niveau de qualité requis (définition du terminé) afin d'être potentiellement déployable auprès des utilisateurs finaux.

Intégration Continue :
Consiste à compiler, assembler, vérifier et tester l'ensemble du code source de façon automatisée dès qu'une modification est mise en configuration, ou, a minima, une fois par jour.

Itération (*Sprint* dans Scrum) :
Période de temps figée (entre 1 semaine et 4 semaines selon les projets) pendant laquelle un ensemble figé de fonctionnalités est développé, donnant un produit partiel complètement opérationnel.

Pair programming :
Deux programmeurs partagent un seul poste de travail (écran, clavier, souris), se répartissant les rôles entre un "conducteur" (aux commandes) et un "copilote" (qui surveille l'écran et challenge le pilote sur ses choix d'implémentation), intervertissant les rôles fréquemment dans le but d'être plus efficace en termes de conception, de revue de code et de acquisition de compétences.

PDCA :
La roue de Deming, est une méthode d'amélioration continue qui comporte quatre étapes, chacune entraînant l'autre, et visant à établir un cercle vertueux.

- P = Plan : ce que l'on va faire
- D = Do : production
- C = Check : mesure, vérification
- A= Act : décision d'amélioration, de correction

PDSA :
L'acronyme PDSA signifie Plan, Do, Study, Act. Cette méthode est utilisée pour tester une idée en essayant d'introduire temporairement un changement et en mesurant l'impact de ce changement. Les quatre étapes sont :

- Plan –planification du changement devant être testé ou implémenté,
- Do –conduite du test ou du changement planifié,
- Study–étude des données avant et après l'introduction du changement. Qu'a-t-on appris ?
- Act–planification de la prochaine étape de test ou décision d'implémentation complète.

Planning Poker :
Un processus d'estimation collaboratif permettant à chacun de s'exprimer et à l'équipe de converger rapidement vers une estimation consensuelle. Cette méthode doit son nom à l'emploi de cartes à jouer dotées de nombre représentant une suite de Fibonacci modifiée.

Product Backlog :
La liste de fonctionnalités priorisées et estimées, jugées nécessaires et suffisantes pour la réalisation satisfaisante du produit :

- si une tâche contenue dans le backlog ne contribue pas aux objectifs du projet, il faut l'en retirer;
- à contrario, dès qu'on a connaissance d'une tâche qu'on pense nécessaire pour le projet, on doit l'inclure dans le backlog

Ces propriétés s'apprécient relativement à l'état des connaissances de l'équipe à un instant donné : l'élaboration du backlog peut se poursuivre tout au long du projet.
Il est le principal référentiel de l'équipe en matière d'exigences

Product Owner (ou Directeur de Produit) :
Il possède l'expertise fonctionnelle et est à même de réaliser les arbitrages nécessaires à la priorisation des développements. Il précise et affine le besoin fonctionnel d'itération en itération. Son rôle est absolument essentiel, et son respect des règles du jeu est la pierre angulaire du succès d'un projet agile.

Refactoring :
Remanier consiste à modifier un code source de façon à en améliorer la structure, sans que cela modifie son comportement fonctionnel. Cette pratique est encouragée en agilité pour maîtriser la dette technique.
Attention : A ne pas confondre avec le concept de rework qui dans un cycle en V consiste à refaire a postériori des portions de code dont les choix techniques initiaux ne permettent pas de tenir les exigences du projet.

Release Planning :
Cette réunion est destinée à partager une vision, définir ce que va apporter la release au produit. Les fonctionnalités de haut niveau sont estimées et une répartition prévisionnelle en sprint est opérée.

Rétrospective :
Réunit l'équipe et le Product Owner au terme de chaque sprint afin d'identifier ce qui a bien fonctionné et ce qui doit être amélioré et de définir un plan d'actions réduit qui est mis en œuvre par l'équipe dans le nouveau sprint. Cette réunion permet à l'équipe un apprentissage sur son processus.

Rework :
Travail effectué pour corriger des défauts. Parmi les causes de Rework en cycle en V :

- un changement tardif d'exigences,
- une conception mal définie.

Scrum Master :
Sa tâche principale est d'optimiser la capacité de production de l'Equipe en l'aidant à travailler de façon autonome et à s'améliorer constamment. Il est également le garant de la bonne application de Scrum.

Spike :
Un spike est une story qui nécessite pour être estimée un travail d'investigation de l'équipe durant un sprint. Le résultat du spike est l'estimation de la story d'origine.

Sprint Backlog :
Liste des tâches priorisées et estimées que l'Equipe s'est engagée à réaliser dans le sprint en cours.

Sprint Planning (ou Planification d'itération) :
Au cours de cette réunion l'équipe s'engage sur les fonctionnalités réalisées dans le sprint et décompose les activités retenues du backlog de produit en tâches qui constituent le backlog de sprint.

Sprint Review (ou Revue d'itération) :
La revue de sprint consiste, au terme de chaque itération, à faire une démonstration publique du résultat du sprint et de recueillir un retour des commanditaires permettant d'ajuster le contenu du backlog de produit.
La revue de sprint est également l'occasion de présenter l'avancement du projet aux parties prenantes.

Sustainable pace (ou rythme soutenable)[1] :
L'équipe vise un rythme de travail tel qu'il pourrait être soutenu indéfiniment. Ceci implique en général de refuser ce qui est parfois considéré comme un « mal nécessaire » : heures supplémentaires, travail le week-end.

[1] Définition issue de http://referentiel.institut-agile.fr

L'approche Agile considère que le recours autre qu'exceptionnel à des mesures telles que les heures supplémentaires est rarement productif, et contribue en fait à masquer des défauts de planification, de management ou de qualité interne. Il est préférable de mettre au jour ces défauts et d'en traiter la cause profonde plutôt que de leur appliquer un traitement symptomatique.

TDD (Test Driven Development) :
Ce terme désigne une technique de développement qui entremêle la programmation, l'écriture de tests unitaires et l'activité de remaniement. Elle consiste à écrire les programmes de tests unitaires avant de programmer les fonctions elles-mêmes, puis d'adapter le code source testé unitairement jusqu'à obtenir un code de qualité.

Thème :
Un thème est une collection de stories. L'intérêt des thèmes est évident quand il s'agit de définir les priorités : c'est bien plus facile à faire sur 5 à 10 thèmes que sur des dizaines de stories. Un autre intérêt est de définir un domaine fonctionnel dont on pourra suivre l'avancement.

User Stories :
L'intégralité du travail à réaliser est découpée en incréments fonctionnels et les activités de développement s'organisent autour de ces incréments appelés "User Stories".

Adopter la pratique des User Stories implique de tenir pour généralement vraies un certain nombre d'hypothèses sur ces incréments : on peut les réaliser dans un ordre arbitraire, et chacun indépendamment des autres peut avoir une valeur pour l'utilisateur.

Pour rendre ces hypothèses très concrètes, on privilégie une représentation de ces User Stories sous une forme bien précise : la fiche cartonnée ou le Post-It, qui en renforcent le caractère atomique et manipulable par tous les intervenants du projet.

Valeur Métier :
Mesure dans une unité propre à un client de la valeur apportée par une fonctionnalité à un produit. L'objectif des méthodes agile et de piloter par la valeur métier en maximisant la production de celle-ci le plus tôt possible. On parle de ROI en valeur métier.

Velocity (Vélocité) :
La quantité d'effort du backlog de produit qu'une équipe est capable de traiter en un sprint. Cette valeur est déduite en moyennant les valeurs constatées sur les sprints précédents (avec l'hypothèse que l'équipe est restée stable).
La vélocité est utilisée pour prévoir les dates de fin de release.

Version (Release ou Livraison dans Scrum) :
Livraison comprenant un certain nombre d'incréments du produit apportant suffisamment de valeur perçue par les utilisateurs finaux pour compenser le coût du déploiement.

Whole Team (Equipe) :
Une « équipe » au sens Agile est un petit groupe de personnes affectées à un seul projet, pour la plupart à temps plein (un membre d'une équipe peut avoir par ailleurs des responsabilités opérationnelles, c'est-à-dire qui ne constituent pas un "projet" à proprement parler).
La notion d'équipe implique le partage des responsabilités : bon ou mauvais, le résultat obtenu est le fait de l'équipe plutôt que de tel ou tel individu.
L'équipe réunit l'ensemble des compétences nécessaires : fonctionnelles et techniques. L'accent est mis sur les résultats à obtenir plus que sur les rôles et responsabilités de chacun : un développeur peut tester, analyser ou découvrir des exigences, même si celles-ci restent soumises à la validation du client; un testeur peut développer si nécessaire, etc.

Rôles

Product Owner (P.O) :
Personne représentant le client et l'utilisateur auprès du Scrum Master et de l'équipe de développement. Il définit le produit et priorise les fonctionnalités voulues.

Scrum Master :
C'est le facilitateur, le garant du processus agile. Il n'est PAS un chef de projet mais un leveur d'obstacles qui empêcheraient l'équipe de développement d'avancer. Il protège et guide l'équipe des interférences extérieures pendant le sprint.

***Team** (équipe de développement) :*
Autogérée et multidisciplinaire (développeurs, testeurs, architectes, etc), les membres travaillent idéalement dans une seule et même pièce. Elle livre un produit utilisable à la fin de chaque sprint.

Concepts

User Story (U.S) :
Description d'une fonctionnalité du point de vue utilisateur. Elle prend le formalisme "En tant que... Je veux... afin de...". Une user story peut être divisée en tâches si elle est complexe.
Un exemple de représentation d'une user story sur une carte.

Story points (points d'histoire) :
Outil d'estimation de l'effort nécessaire pour développer des fonctionnalités. Les points d'histoire permettent de se soustraire du concept de jour/homme. Les points sont attribués à une user story relativement à d'autres user stories. Par exemple, une user story estimée à deux points demandera deux fois plus d'effort pour la terminer qu'une user story estimée à un point, ceci sans indication de la durée en jour.

Velocity :
L'effort, exprimé en nombre de points d'histoire, que l'équipe de développement peut fournir dans un sprint.

La valorisation en points des user stories permet de déterminer le panier de fonctionnalités absorbable par l'équipe de développement en un sprint.

Definition of Done (Fini) :
La "définition de fini" est la liste de critères qu'une user story doit remplir pour être considérée comme ayant l'état "fini", donc livrable. Cette liste de critères peut inclure, par exemple, une couverture de test minimum, une revue de code d'un autre membre de l'équipe, une javadoc suffisante, etc. Il est important d'avoir une DoD déterminée de façon claire et conjointe entre l'équipe de développement et le Product Owner. Ce dernier exprime son acceptation d'une user story via des tests d'acceptance

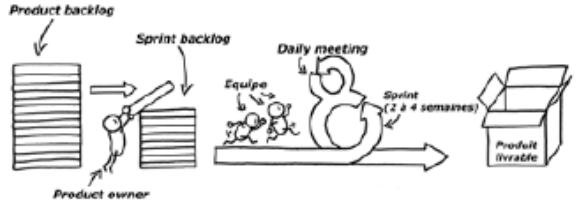
Rituels

Sprint : Période de 2 à 4 semaines dédiée au développement des user stories du backlog, et permettant d'avoir un produit potentiellement livrable à la fin de celle-ci.

Daily standup meeting :

Réunion faite debout pour ne pas durer trop longtemps et à heure fixe (généralement le matin), lors de laquelle chaque participant répond aux trois questions:
"Qu'ai-je fait hier ?",
"Que vais-je faire aujourd'hui ?" et
"Ai-je un point de blocage ?"

***Sprint review** (démonstration de fin de sprint) :*
Réunion tenue en fin de sprint durant laquelle l'équipe de développement montre le travail accompli pendant le sprint (i.e. les fonctionnalités, les user stories demandées par le Product Owner).



Planning poker :

Séance d'estimation menée par l'équipe de développement qui évaluent ensemble l'effort nécessaire pour traiter les user stories du backlog.
Pour cela, ils utilisent chacun un jeu de carte sur lesquelles sont inscrit des nombres de points d'histoires dont les valeurs suivent généralement la suite de Fibonacci :
0, 1, 2, 3, 5, 8, 13...
Les estimations sont faites face cachée et dévoilées en même temps pour éviter d'influencer les autres membres de l'équipe

Retrospective :

Réunion permettant à l'équipe de faire un bilan du sprint qui vient de se terminer.
On y note ce qui fait avancer le projet et ce qui le ralentit. Dans ce dernier cas, l'équipe cherche des actions pour lever les obstacles.
Elle est généralement menée par le Scrum Master et s'organise en 5 étapes :

- set the stage, prendre la température des participants via un vote (de confiance et/ou d'itération);
- gather data, liste le ressenti de l'équipe, les problèmes, les points positifs, les émotions qui l'ont marquée pendant le sprint qui vient de se terminer;
- generate insights, permet une réflexion de groupe sur la perception et les causes des obstacles évoqués précédemment;
- decide what to do, est l'étape qui permet de générer des actions à appliquer lors du sprint suivant pour tenter de lever les obstacles évoqués;
- close the retrospective, marque la fin de cette réunion. On y fait en général un vote nommé ROTI (Return On Time Invested) pour indiquer le degré de satisfaction sur le temps consacré à la rétrospective.

Artefacts

Product backlog :

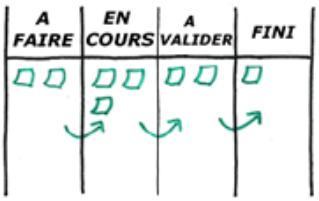
Ensemble des caractéristiques (fonctionnalités ou besoins techniques) qui constituent le produit souhaité. Il doit être priorisé pour permettre de développer les éléments de plus haute importance en premier.

Sprint backlog :

Sous-ensemble des éléments du backlog de produit.
Les éléments constituent les user stories à développer au cours du sprint et sont préalablement détaillés pour pouvoir être estimés par l'équipe de développement. Il est également priorisé.

Task board (tableau des tâches) :

Tableau physique ou logiciel reprenant les éléments du backlog de sprint. Il possède plusieurs colonnes (ex. à faire, en cours, à valider, validée) permettant de suivre l'avancement des user stories affichées via des post-it ou des cartes.



Burn down chart :

Graphique permettant de suivre le "reste à faire" durant le sprint. Il possède en abscisse le temps et en ordonnée les points d'histoire. La courbe indique le nombre de points d'histoire abattus pendant le sprint.

Elles sont mises à jour en continu. Cela permet d'anticiper les dérives et les ruptures de charge. L'idéal étant bien sûr d'arriver à zéro point le dernier jour du sprint.

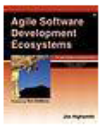
Réalisé à partir du memento à destination des équipes Scrum proposé par :
Hing CHAN
http://hingchanscrum.blogspot.com
@HingCChan
Et
Thierry LERICHE
http://icauda.com
@thierryleriche

Ouvrages généraux sur l'agilité

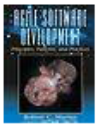
Agile Estimating and Planning
Auteur(s) : Mike Cohn
Editeur : Prentice Hall (2005)
ISBN 13 : 9780131479418

A Practical Guide to Feature Driven Development
Auteur(s) : Stephen Palmer, John Felsig
Editeur : Prentice Hall (2002)
ISBN 13 : 9780130676153

Agile and Iterative Development: A Manager's Guide
Auteur(s) : Craig Larman
Editeur : Addison-Wesley Professional (2003)
ISBN 13 : 9780131111554



Agile Software Development Ecosystems
Auteur(s) : Jim Highsmith
Editeur : Addison-Wesley Professional (2002)
ISBN 13 : 9780201760439



Agile Software Development: Principles, Patterns and Practices
Auteur(s) : Robert C. Martin
Editeur : Prentice Hall (2002)
ISBN 13 : 9780135974445

Innovation Games: Creating Breakthrough Products Through Collaborative Play
Auteur(s) : Luke Hohmann
Editeur : Addison Wesley (2006)
ISBN 13 : 9780321437297

Practices of an Agile Developer: Working in the Real World
Auteur(s) : Venkat Subramaniam, Andy Hunt
Editeur : Pragmatic Bookshelf (2005)
ISBN 13 : 9780974514086

The Art of Agile Development
Auteur(s) : James Shore, Shane Warden
Editeur : O'Reilly Media (2007)
ISBN 13 : 9780596527679

Working Effectively with Legacy Code
Auteur(s) : Michael Feathers
Editeur : Prentice Hall (2004)
ISBN 13 : 9780131177055

Lean et kanban

Lean Software Development: An Agile Toolkit
Auteur(s) : Mary Poppendieck, Tom Poppendieck
Editeur : Addison-Wesley Professional (2003)
ISBN 13 : 9780321150783

KANBAN POUR l'IT: Une nouvelle méthode pour améliorer les processus de développement
Auteur(s) : Laurent Morisseau
Editeur : Dunod (2012)
ISBN 13 : 9782100578672

Lean from the Trenches
Auteur(s) : Henrik Kniberg
Editeur : Pragmatic Bookshelf (2012)
ISBN 13 : 9781934356852

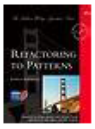
Gestion de projet agile

Gestion de projet : vers les méthodes agiles
Auteur(s) : Véronique Messager-Rota, Jean Tabaka
Editeur : Eyrolles (2009)
ISBN 13 : 9782212125184

Managing Agile Projects
Auteur(s) : Sanjiv Augustine
Editeur : Prentice Hall (2005)
ISBN 13 : 9780131240711

Agile Project Management: Creating Innovative Products (2nd Edition)
Auteur(s) : Jim Highsmith
Editeur : Addison-Wesley Professional (2009)
ISBN 13 : 9780321658395

Refactoring



Refactoring to Patterns
Auteur(s) : Joshua Kerievsky
Editeur : Addison-Wesley Professional (2004)
ISBN 13 : 9780321213358

Refactoring: Improving the Design of Existing Code
Auteur(s) : Martin Fowler, Kent Beck, John Brant
Editeur : Addison-Wesley Professional (1999)
ISBN 13 : 9780201485677

Le cadre Scrum

SCRUM
Auteur(s) : Claude Aubry
Editeur : Dunod (2010)
ISBN 13 : 9782100540181

Agile Retrospectives: Making Good Teams Great
Auteur(s) : Ester Derby
Editeur : Pragmatic Bookshelf (2006)
ISBN 13 : 9780977616640

Agile Software Development with Scrum
Auteur(s) : Ken Schwaber, Mike Beedle
Editeur : Prentice Hall (2001)
ISBN 13 : 9780130676344

Les tests en agile

Continuous Integration: Improving Software Quality and Reducing Risk
Auteur(s) : Paul Duvall, Steve Matyas, Andrew Glover
Editeur : Addison-Wesley Professional (2007)
ISBN 13 : 9780321336385

FIT for Developing Software
Auteur(s) : Rick Mugridge, Ward Cunningham
Editeur : Prentice Hall (2005)
ISBN 13 : 9780321269348

Test Driven Development: By Example
Auteur(s) : Kent Beck
Editeur : Addison-Wesley Professional (2002)
ISBN 13 : 9780321146533

eXtreme Programming

Gestion de projet, eXtreme Programming
Auteur(s) : Jean-Louis Bénard, Laurent Bossavit, Régis Médina
Editeur : Eyrolles (2004)
ISBN 13 : 9782212115611

Extreme Programming Applied: Playing To Win
Auteur(s) : Ken Auer
Editeur : Addison-Wesley Professional (2001)
ISBN 13 : 9780201616408

Blogs Francophones

Des liens vers des blogs d'agilistes chevronnés :

Claude Aubry : aubryconseil.com

Alexandre Boutin : agilex.fr

Thierry Cros : etreagile.thierrycros.net

Fabrice Aimetti : agilarium.blogspot.com

L'Institut Agile de Laurent Bossavit (plein de ressources !) : blog.institut-agile.fr

Emmanuel Chenu (de Thales Avionics à Valence) : emmanuelchenu.blogspot.com

Des liens vers des groupes de discussion et d'associations agiles :

L'Esprit Agile à Marseille : esprit-agile.com

Le CARA (Club Agile Rhône-Alpes) : clubagilerhonealpes.org

Agile Toulouse : agiletoulouse.fr

En Bretagne : agilebreizh.org

Sites de référence

Agile Alliance : agilealliance.org

Scrum Alliance : scrumalliance.org

French Scrum User Group : frenchsug.org

Jeff Sutherland : jeffsutherland.com

Mike Cohn : mountaingoatsoftware.com

Evénementiel France et Monde

Agile Tour : agiletour.org

Agile France : conf.agile-france.org

Agile Grenoble : agile-grenoble.org

Lean Kanban : leankanban.fr

Scrum Day : scrumday.fr