Méthodes Agiles

CCI 36 - |Formation DISII 2016/2017

Veille thématique

Alexandre violeau, Cyril Touraine, Anthony Slimani

2016

Table des matières

[METHODES AGILES 2](#_Toc471079714)

[Processus de développement de logiciels 2](#_Toc471079715)

[Découpement des phases selon le schéma classique 3](#_Toc471079716)

[Modèles de Méthodes de développement significatifs 5](#_Toc471079717)

[Les Méthodes Agiles 9](#_Toc471079718)

[Manifeste pour le développement Agile de logiciels 9](#_Toc471079719)

[Principes sous-jacents au manifeste 10](#_Toc471079720)

[Principes sous-jacents au manifeste (suite) 11](#_Toc471079721)

[Méthodes Agiles par l’exemple 12](#_Toc471079722)

[SCRUM 12](#_Toc471079723)

[Plusieurs rôles définit par Scrum : 13](#_Toc471079724)

[Product Owner : 13](#_Toc471079725)

[Equipe de développement : 13](#_Toc471079726)

[Scrum Master : 13](#_Toc471079727)

[Vision du produit et product backlog : 13](#_Toc471079728)

[Démarrage : 14](#_Toc471079729)

[Phase 1 : « Le Quoi » 14](#_Toc471079730)

[Phase 2 : Le « Comment » 15](#_Toc471079731)

[Pendant un sprint : 15](#_Toc471079732)

[Mêlée quotidienne ou « Stand-up meeting » 16](#_Toc471079733)

[Graphique d’avancement (Burndown Chart) 17](#_Toc471079734)

[Revue de sprint 18](#_Toc471079735)

[Rétrospective de sprint 18](#_Toc471079736)

[Les pièges à éviter 18](#_Toc471079737)

[Table des liens (sources) 20](#_Toc471079738)

# **METHODES AGILES**

Avant d'entamer le sujet des méthodes dites "agiles", il conviendrait de décrire le support d'origine ; les méthodes de développement desquelles elles sont les émules. Ces méthodes trouvent leur origine dans une imitation de la gestion de projet telle qu'elle pouvait exister dans les secteurs du bâtiment par exemple.

## Processus de développement de logiciels

Leur nécessité s'est rapidement manifestée et découle directement de l'émergence de grandes entreprises ; inévitablement dotés de systèmes d'informations de plus en plus démesurés et sophistiqués, notamment lorsque l'on ajoute à l'équation ceux des conglomérats.

**SDLC** (Systems development life cycle) dans les années 1960 est généralement considéré comme étant le premier SDM (Software Development Methodology) opérationnel.

Conçu pour structurer idéalement un projet de système d'informations à l'origine ; il est imité et détourné pour divers domaines tels que le commerce. Il consiste en une succession de phases formant ce que l'on appelle "cycle de développement", ou plus généralement le "cycle de vie".

Son **objectif** a été de :

"(...) poursuivre le développement de systèmes d'informations d'une manière délibérée, structurée et méthodique, nécessitant que chaque étape du cycle – de la germination de l'idée à la livraison du système final – soit menée de manière rigide et séquentielle (...)"

"(...)développer des systèmes commerciaux fonctionnels de grande échelle dans l'âge des immenses conglomérats commerciaux. Les activités des systèmes d'information se résumaient à la manipulation et au traitement de lourdes bases de données, ainsi qu'à la réalisation de procédures de calcul."

Geoffrey Elliott (2004) *Global Business Information Technology: an integrated systems approach.*

## Découpage des phases selon le schéma classique

On met l'accent sur la **planification**, l'**analyse,** le **design** c'est-à-dire l'architecture/la conception et **l'implémentation.**

**Lors de la phase de planification** on établit une analyse préliminaire des tenants et aboutissants de l'organisation et on mène naturellement une étude du problème qui est posé et de son environnement direct ; menant ainsi à une compréhension plus fine du dit problème.

Cette analyse amène à prendre des décisions quant à la préservation, la suppression ou bien l'amélioration de l'existant ; avec ces dernières est dressée une évaluation chiffrée du projet qui commence à voir le jour, pour passer à une analyse approfondie.

**La phase d'analyse consiste principalement en une collecte d'informations** exhaustive, qui déterminera les axes du développement, ses spécifications fondamentales comme les plus anecdotiques. Une étude aussi large que possible est menée à travers la consultation de la documentation, l'observation mais surtout par l'échange avec le personnel, par les moyens voulus ; on peut ainsi préparer des questionnaires, par exemple.

Grace aux résultats de l'analyse, le modèle en place sera mieux compris ; donc plus facilement comparable à celui du projet en cours, qui verra sa structure gagner en cohérence. Par ailleurs, les spécifications des utilisateurs ; comme les problèmes (parfois mineurs) rencontrés sur le système existant permettent d'anticiper l'apparition d'éventuels bugs ou soucis d'ergonomie.

**Le design du logiciel** commence alors, ses caractéristiques précises sont décrites, ainsi que ses fonctionnalités, une documentation est construite ; l'architecture globale, l'interface sont définies dans leur intégralité.

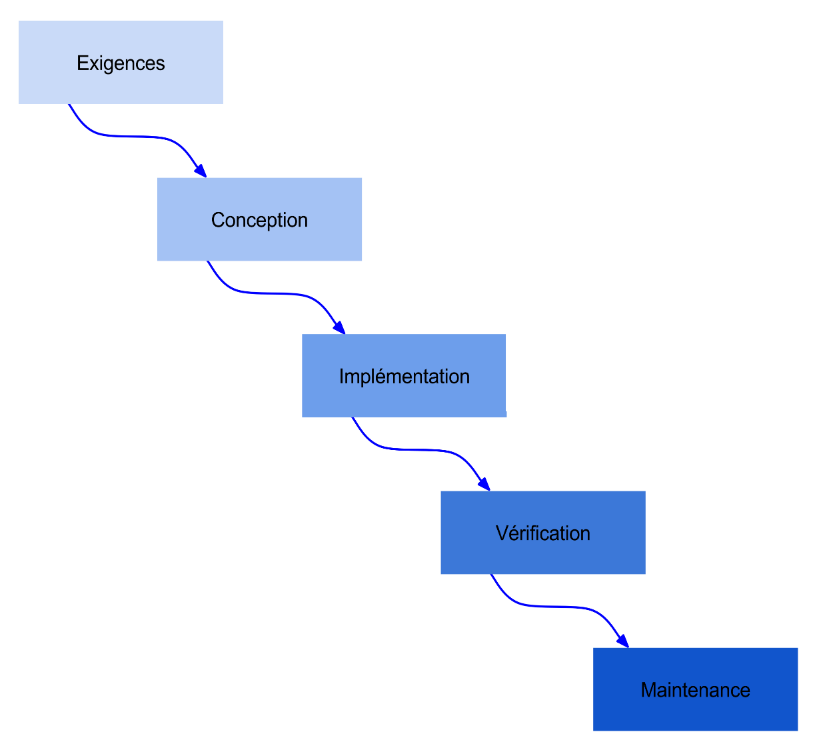
Le développement concret s'enclenche alors ; des étapes de tests et d'intégration prévues lors de l'analyse permettent de construire progressivement le logiciel ; de manière méthodique et relativement sûre.

**Le déploiement en fin de projet marque néanmoins le départ vers une phase de maintenance.** A partir de ce point, le logiciel est mis en production et est opérationnel ; il est par ailleurs évalué quant à ses performances et son adaptabilité.

La maintenance se découpe alors en branches, on parle de deux branches principales qui sont la maintenance corrective et la maintenance préventive. La maintenance corrective est elle-même scindée en deux : une branche dite "adaptative", l'autre dite "évolutive".

Ce dernier découpage lance une discussion sur la signification de la maintenance et sur ses champs d'applications, l'amalgame pouvant être parfois fait entre correction et amélioration.

La phase de maintenance peut comme on l'a vu connaître des opérations qualifiées d'évolutives, au cours desquelles le logiciel ou une part du système d'information peut être modifiée pour être améliorée ; toutefois ; certains besoins d'optimisation ne correspondant plus à ce qui a été prévu en maintenance peuvent faire l'objet d'une requête, et donc éventuellement celui du départ d'un nouveau cycle poursuivant le précédent.



https://fr.wikipedia.org/wiki/Cycle\_de\_d%C3%A9veloppement\_(logiciel)

## Modèles de Méthodes de développement significatifs

**Cycle en Cascade :**

C'est le modèle séquentiel par excellence, "waterfall" en anglais,sa méthode a gagnéen crédibilité depuis des dizaines d'années et conditionnerait aujourd'hui encore un tiers des projets.

Il reste dominant dans les secteurs de l'industrie et de la construction, ces derniers n'évoluant que peu ou moins rapidement que celui de l'informatique. Nous disposons par ailleurs ; bien évidemment ; d'un certain recul technologique permettant une préparation et un contrôle de projet autrement plus concrets et rigides sur ceux-ci, si on les compare à des domaines tels que la gestion de systèmes d'informations ou la programmation.

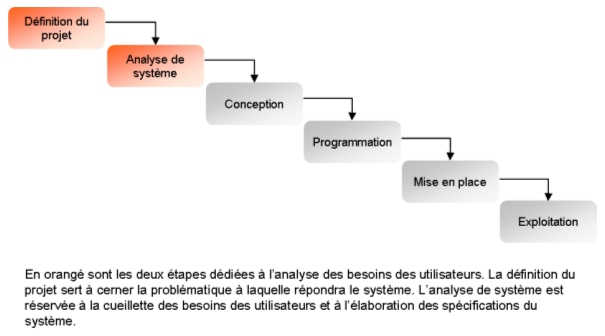
D'une manière générale, on peut noter qu'un certain équilibre essaye toujours d'être trouvé entre les aspects matériels et logiciels ; ce qui amènera d'ailleurs aux méthodes agiles, ainsi qu'aux horizons explorés par les "agilistes".

Suivant cette mouvance et la technique avançant, le modèle en cascade ; orienté machine à l'époque, a connu une adaptation orientée vers les logiciels.

C'est un cycle classique, suivant ce schéma on le voit progresser de manière stable vers le bas, chaque étape se devant d'être achevée pour pouvoir passer à la suivante.

On retrouve les grandes phases décrites précédemment :

* **Conception** : Etude des Pré-requis systèmes et logiciels, à partir de laquelle est produit un “Cahier des charges”.
* **Analyse**: fournit un "Cahier des spécifications".
* **Design :** Architecture, Conception du produit.
* **Construction :** développement.
* **Test**
* **Opérations :** Production/Implémentation, Maintenance.

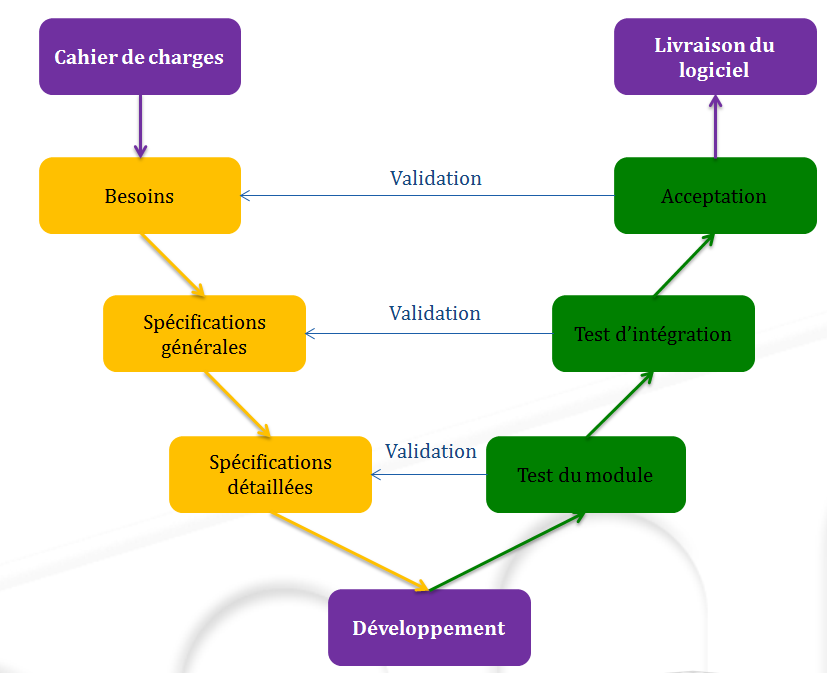
****

**http://theses.ulaval.ca/archimede/fichiers/22406/ch02.html**

Ce schéma séquentiel (phases bien délimitées dans le temps, dont la synchronicité même temporaire est “tolérée”) rend le processus de développement extrêmement rigide et manifestement hostile à toute nouvelle implémentation et encore plus à une révision quelconque de ce qui a déjà été fait et approuvé.

**Cycle en V :**

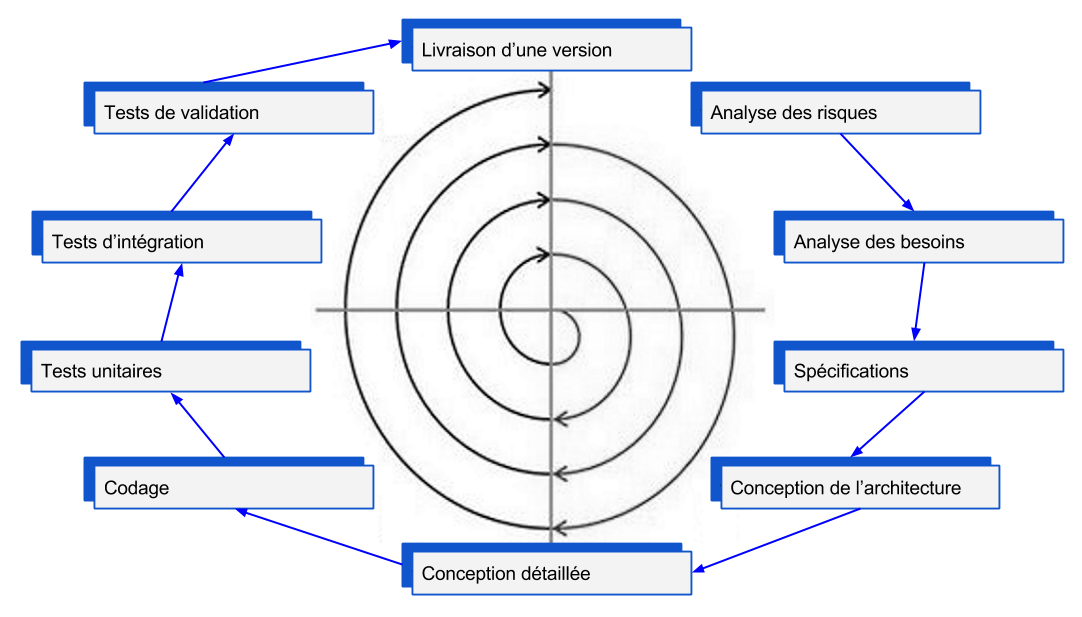
Celui est similaire au modèle en cascade, il profite de l’apport d’un système établissant une symétrie entre les parties projets et exécution des processus de développement, ce qui permet une meilleure adaptabilité, ainsi qu'une meilleure communication, rappelant la courbe du soleil merisienne, avec une montée vers l'abstraction et une redescente vers la "dureté" du réel où des tests pourront notamment être menés.

****

**http://www.utc.fr/~mastermq/public/publications/qualite\_et\_management/MQ\_M2/2011-2012/stages/flores\_soto/**

**Cycle en Spirale :**

Un modèle hybride plus souple se dessine avec répétition de phases de développement sur le modèle en V ; un ajout important survient : une évaluation des risques avant chacune des répetitions.

****

<http://www.responsive-mind.fr/cycles-developpement-informatique/>

Ce dernier modèle apporte le caractère itératif qui sera l'avantage et le trait distinctif de l'agilité.

## Les Méthodes Agiles

Les méthodes Agiles sont donc des cycles de développement itératifs, après le début des années 1960 et la popularisation de divers SDM, des méthodes qui seront plus tard nommées "agiles" émergent progressivement et occupent une place de plus en plus importante (SCRUM, RAD...).

En **2001** un manifeste "Agile Manifesto" définit quatre valeurs simples, formulées en quatre courtes phrases ; desquelles découlent une douzaine de principes.

**Manifeste pour le développement Agile de logiciels**

Nous découvrons comment mieux développer des logiciels  
par la pratique et en aidant les autres à le faire.  
Ces expériences nous ont amenés à valoriser :

**Les individus et leurs interactions** plus que *les processus et les outils*  
**Des logiciels opérationnels** plus qu’*une documentation exhaustive*  
**La collaboration avec les clients** plus que *la négociation contractuelle*  
**L’adaptation au changement** plus que *le suivi d’un plan*

Nous reconnaissons la valeur des seconds éléments,  
mais privilégions les premiers.

<http://agilemanifesto.org>

Les quatre valeurs en contiennent en réalité huit ; cette comparaison construit alors une "philosophie" qui n'exclut pas totalement les méthodes classiques, mais qui privilégie :

* Itérativité.
* Dialogue.
* Travail d'équipe.
* Performance.
* Adaptabilité.

en opposition à :

* Planification rigoureuse.
* Suivi rigide, resistance au changement (progression exponentielle voir cours Thierry).
* Négociations éloignées, fixées.
* "Matérialisme".

**Principes sous-jacents au manifeste**

*Nous suivons ces principes:*

Notre plus haute priorité est de satisfaire le client  
en livrant rapidement et régulièrement des fonctionnalités  
à grande valeur ajoutée.

Accueillez positivement les changements de besoins,  
même tard dans le projet. Les processus Agiles  
exploitent le changement pour donner un avantage  
compétitif au client.

Livrez fréquemment un logiciel opérationnel avec des  
cycles de quelques semaines à quelques mois et une  
préférence pour les plus courts.

Les utilisateurs ou leurs représentants et les   
développeurs doivent travailler ensemble quotidiennement  
tout au long du projet.

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.

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.

**Principes sous-jacents au manifeste (suite)**

Un logiciel opérationnel est la principale mesure d’avancement.

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.

Une attention continue à l'excellence technique et  
à une bonne conception renforce l’Agilité.

La simplicité – c’est-à-dire l’art de minimiser la   
quantité de travail inutile – est essentielle.

Les meilleures architectures, spécifications et   
conceptions émergent d'équipes autoorganisées.

À intervalles réguliers, l'équipe réfléchit aux moyens  
de devenir plus efficace, puis règle et modifie son  
comportement en conséquence.

[http://agilemanifesto.org](http://agilemanifesto.org/)

Ces principes définissent plus spécifiquement les axes à prendre quant aux étapes de développement d'un logiciel, ils décrivent également les conditions de travail ainsi que son organisation globale.

L'accent est mis sur :

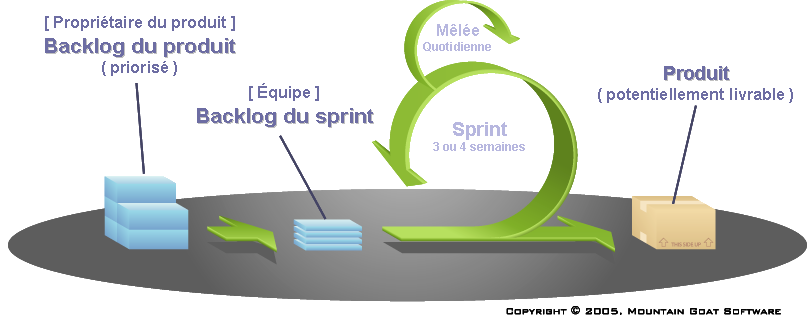
* + l'optimisme, notamment la motivation et l'**accueil au changement.**
  + la cohésion, le travail collectif, la collaboration, plein d'autres synonymes.
  + la qualité du produit, sa fonctionnalité.
  + le caractère cyclique inhérent à l'agilité.

# **Méthodes Agiles par l’exemple**

# **SCRUM**

Nous prendrons l’exemple ici de SCRUM (à traduire par mêlée au rugby). Méthode expérimentée en 1993, elle est la méthode la plus populaire.

Elle se présente sous forme d’itérations (appelées sprints) de très courtes durées (entre 2 à 4 semaines).



Nous reviendrons sur des termes plus précis dans le déroulement de cette méthode.

## Plusieurs rôles définit par Scrum :

### Product Owner :

Expert du domaine métier du projet. Il est en étroit collaboration avec l’équipe de développement

### Equipe de développement :

Cette équipe est chargée de transformer les besoins exprimés par le Product Owner en fonctionnalités utilisables. Elle accueille différents cœurs de métiers (développeur, graphiste, architecte logiciel, analyste).

### Scrum Master :

Le Master a le rôle de coach pour l’ensemble de l’équipe et doit s’assurer que la méthode est correctement appliquée. Nous pouvons l’assimilé à un chef de projet. Ce Scrum Master devra appliquer une méthode de management participatif à son équipe.

## Vision du produit et product backlog :

Cette vision permet de décrire les principaux objectifs, les jalons à poser, les utilisateurs visés.

Elle contribuera à guider et fédérer les acteurs du projet. La suite consiste à établir la liste des **exigences fonctionnelles** et **non fonctionnelles** du produit. Chaque exigence est ensuite estimée par l’**équipe de développement** avec la technique de **Planning Poker**. Une fois ces estimations terminées, la liste ainsi complétée est ordonnancée. Les exigences seront converties en fonctionnalités utilisables selon cet ordonnancement. Le principe étant de convertir en premier les exigences qui apportent le plus de valeur ajoutée (ou ROI) au commanditaire. Il s’agit donc de faire remonter les exigences fonctionnelles de la plus haute valeur ajoutée (ou dont le ROI est le plus élevé) en haut de la liste. Cette liste est appelée le **Product Backlog**. Le Product Backlog servira à piloter l’équipe de développement et pourra évoluer tout au long du projet. **Le changement est non seulement autorisé mais encouragé afin de pouvoir éliminer les idées de départ qui s’avéreront mauvaises et de prendre en compte les nouvelles idées qui arriveront en cours de route.** Cette activité de construction du Product Backlog est collaborative, elle implique le Product Owner et l’équipe de développement.

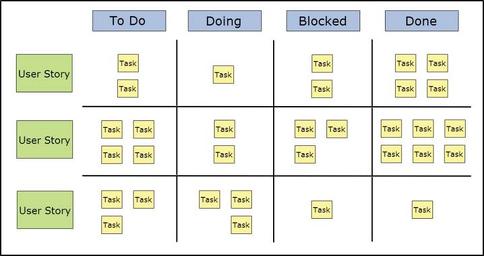
## Démarrage :

Pour commencer, il faudra déterminer la durée des sprints (4 semaines maximum).  
Cette durée devra être une base commune à l’ensemble du projet afin de maintenir un rythme régulier.

Le sprint 0 est généralement dédié aux travaux de préparations. Plus haut, nous avons dû déterminer une durée de sprint générale. Cependant pour la phase préparatoire, il n’est pas nécessaire de respecter cette durée.

### Phase 1 : « Le Quoi »

Lorsque le Product Backlog est dans sa globalité complet et ordonnancé, un sprint peut-être planifier. L’équipe entière vérifient les estimations et confirment qu’elles sont exactes.



<http://www.sparkplugagility.com/single-post/2016/05/13/Blog-coming-soon>

### Phase 2 : Le « Comment »

Un inventaire des tâches est créé par l’équipe de développement.

Les tâches de développement sont centralisées dans le Sprint Backlog et ajoutées au tableau des tâches (physique – aussi appelé Kanban)



[http://www.agiliste.fr/guide-de-demarrage-scrum/#Les-R-les-en-bref](http://www.agiliste.fr/guide-de-demarrage-scrum/%23Les-R-les-en-bref)

## Pendant un sprint :

L’équipe de développement au cours d’un sprint se concentre sur l’accomplissement des tâches.

En cas de retard, des tâches seront alors retirées tout en préservant l’objectif du sprint. Dans le cas contraire, si l’équipe prend de l’avance, des tâches seront ajoutées.

Les développements sont verticaux et non horizontaux par couche. Afin d’éviter un mini cycle en V.

Le tableau des tâches physique qui est rempli de post-it permet une vision très claire des tâches à réaliser, en cours et accomplies.

## Mêlée quotidienne ou « Stand-up meeting »

Avec une durée maximum de 15 minutes, cette réunion se fait debout. Elle permet aux membres de l’équipes de remonter les informations importantes, obstacles rencontrés, vérifier l’avancement.

Ces mêlées permettent la cohésion de groupe et créer un esprit d’équipe.

3 questions auxquelles chaque membre doit répondre :

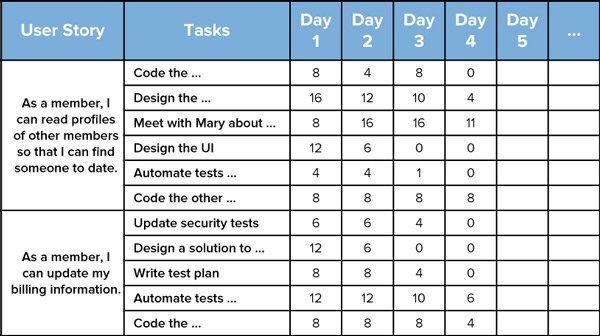
* Qu'ai-je fait hier qui a aidé l'équipe de développement à atteindre l'objectif Sprint ?
* Que vais-je faire aujourd'hui pour aider l'équipe de développement à atteindre l'objectif Sprint ?
* Est-ce que je vois des obstacles susceptibles de m'empêcher ou d'empêcher l'équipe de développement d'atteindre l'objectif du Sprint ?



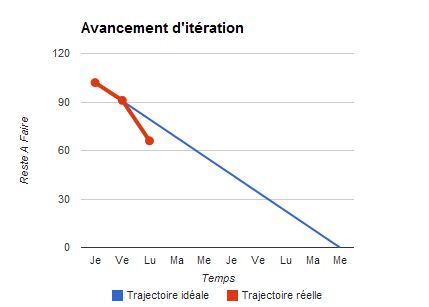
<https://www.youtube.com/watch?v=kKIc1NFO-AU>

## Graphique d’avancement (Burndown Chart)

Pour connaître votre avancement, vous allez avoir besoin de tracer le **Burndown Chart** du sprint en cours. Il s’agit du tracé de la charge de travail restante (en heures) en fonction du temps (en jour).



<http://www.mountaingoatsoftware.com/agile/scrum/sprint-backlog>

Lors de chaque standing-up meeting, le graphique sera mis à jour. Il peut être intéressant de créer un indicateur d’avancement d’itération.

Source : http://www.agiliste.fr/guide-de-demarrage-scrum/#Les-R-les-en-bref

## Revue de sprint

Avec une durée maximum d’1 heure par semaine de sprint (chaque fin de sprint), son objectif est d’inspecter, faire le point sur le projet et adapter au besoin. Cette revue est effectuée en présence de tous les acteurs intéressés. Le Product Owner donne un retour (feedback) à l’équipe, qui accepte ou refuse les fonctionnalités présentées.

## Rétrospective de sprint

Avec une durée maximum de 45 minutes par semaine de sprint (chaque fin de sprint)

Le Scrum Master dirige cette réunion. Chacun apporte des idées visant à d’améliorer le processus de développement. Tous les domaines peuvent être abordés en rétrospective : humain, organisationnel, pratiques, processus, outillage, qualité de vie au travail, conflits, interactions avec le métier.

## Les pièges à éviter

« Scrum est simple mais difficile »

Scrum est un **cadre de gestion de projet** qui laisse à d’autres **méthodes agile** complémentaires, le soin d’apporter les pratiques de développement appropriées. C’est le cas de **eXtreme Programming** et **software craftsmanship**. Il est donc primordial de ne pas se contenter d’utiliser Scrum sans s’assurer que les **pratiques de développement Agile** sont ou seront maîtrisées in fine. Constatant de nombreuses dérives liées à cette erreur courante, les créateurs de Scrum alertent :

*« Début 2009, davantage d’organisations utilisaient des processus agiles plutôt que des processus en cascade. Cependant, moins de 50% de celles utilisant Scrum développaient les itérations de façon incrémentale, ce qui est le cœur de Scrum. Un des plus grands défis de l’utilisation de Scrum a toujours été la courbe d’apprentissage des développeurs de l’équipe Scrum. »*

Plusieurs équipes de développement Scrum

Dans l’idéal, afin d’être le plus efficace, une équipe de développement Scrum de doit pas dépasser 9 membres. Le Scrum Master et le Product Owner ne sont pas compter dans ces 9 personnes.

# **Table des liens (sources)**

* <https://fr.wikipedia.org/wiki/M%C3%A9thode_agile>
* <http://agilemanifesto.org/>
* <http://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/techniques/2517/2517-gestion-de-projet-agile.pdf>
* <https://www.unow.fr/le-coin-des-experts/gestion-de-projet-et-agilite/tour-horizon-differentes-methodes-agiles/>
* <http://less.works/less/framework/index.html>
* <http://www.access-dev.com/access-dev/la-gestion-de-projet-methodes-classiques-vs-methodes-agiles/>
* <https://en.wikipedia.org/wiki/Software_development_process>
* <https://fr.wikipedia.org/wiki/Extreme_programming>
* <https://fr.wikipedia.org/wiki/Manifeste_agile#R.C3.A9sum.C3.A9_de_la_mise_en_pratique>
* <http://www.agiliste.fr/guide-de-demarrage-scrum/#Les-R-les-en-bref>