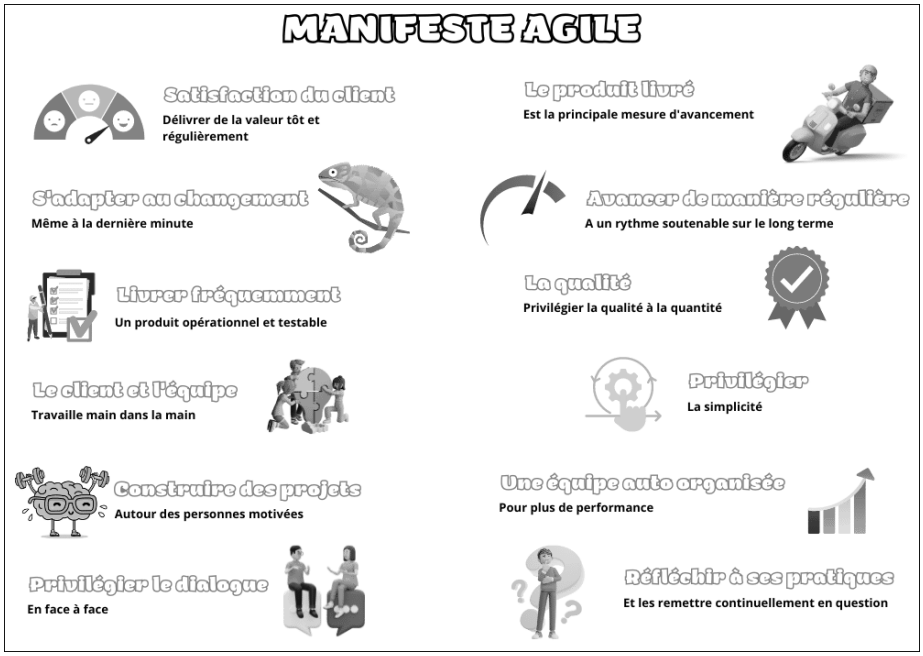
Chapitre 3
Découverte complète de Scrum

1. L'origine de Scrum

L'origine de Scrum remonte aux années 1980 et est profondément enracinée dans la philosophie du travail d'équipe et de la collaboration. Le terme "Scrum" lui-même, emprunté au rugby, symbolise la manière dont une équipe travaille ensemble pour atteindre un objectif commun. Cette métaphore a été introduite pour la première fois dans le monde des affaires dans un article écrit par Hirotaka Takeuchi et Ikujiro Nonaka en 1986. Intitulé *The New New Product Development Game*, cet article comparait les équipes de projet à haute performance à un Scrum dans le rugby et présentait une approche flexible et adaptable pour le développement de produits.

Dans les années 1990, Ken Schwaber et Jeff Sutherland ont commencé à formaliser la méthode Scrum. Ils ont fusionné leurs expériences et leurs idées avec les concepts de Takeuchi et Nonaka, créant ainsi un cadre de gestion de projet qui encourage la collaboration, l'adaptation et la livraison rapide. Cette période a été cruciale pour la mise en forme de Scrum tel que nous le connaissons aujourd'hui.

En 2001, le Manifeste agile a été publié, promouvant des concepts tels que la satisfaction du client grâce à la livraison rapide, l'adaptation au changement, et la collaboration étroite entre les équipes métier et de développement. Scrum, en tant que l'une des méthodologies agiles, adhère aux principes et valeurs définis dans ce manifeste, ce qui a contribué à sa popularité et son adoption.



Au fil des années, Scrum a été adopté dans divers secteurs d'activité, bien au-delà du domaine du développement logiciel. Sa flexibilité et son efficacité ont conduit à une large adoption dans de nombreux types d'organisations, allant des start-ups aux grandes entreprises.

L'origine de Scrum est le reflet d'une évolution dans la façon de penser et de gérer les projets. C'est un modèle qui rompt avec les méthodes traditionnelles en mettant l'accent sur les personnes, la collaboration, et la capacité à s'adapter rapidement aux changements. Le parcours de Scrum depuis son concept initial jusqu'à sa forme actuelle est une démonstration de l'ingéniosité et de l'innovation continues dans le domaine de la gestion de projet.

2. Comprendre Scrum

2.1 Les valeurs fondamentales de Scrum

Comprendre Scrum nécessite une plongée profonde dans ses valeurs fondamentales, qui sont au cœur de cette méthodologie agile. Les valeurs de Scrum donnent un sens à la structure, aux règles et aux rôles qui le composent, et guident la manière dont les équipes travaillent ensemble.

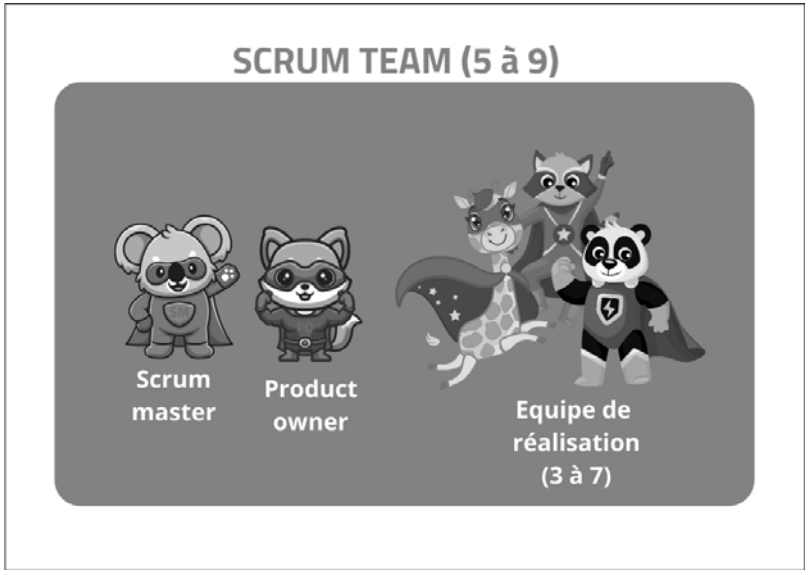
Les cinq valeurs fondamentales de Scrum sont les suivantes :

- **Engagement** : chaque membre de l'équipe est engagé envers les objectifs communs, les valeurs de l'équipe et le succès du projet. Cet engagement est essentiel pour garantir que chaque membre contribue pleinement et assume sa responsabilité dans le projet.
- **Courage** : la mise en œuvre de Scrum nécessite le courage de faire les choses différemment, de défier le statu quo, et de faire face aux obstacles et aux défis. Il s'agit également du courage de parler ouvertement et d'être transparent dans la communication au sein de l'équipe.
- **Concentration** : la concentration est primordiale pour l'efficacité de Scrum. Les équipes se concentrent sur un nombre limité de tâches à la fois, garantissant que l'effort est dirigé vers les priorités les plus importantes.
- **Respect** : dans un cadre Scrum, le respect mutuel est essentiel. Les membres de l'équipe respectent les compétences et les contributions des autres, et cette confiance et ce respect mutuels facilitent une collaboration plus efficace.
- **Ouverture** : l'ouverture implique une communication ouverte et honnête, ainsi qu'une transparence dans les processus et les décisions. Les membres de l'équipe sont ouverts aux idées et aux commentaires, ce qui favorise l'apprentissage continu et l'amélioration.

Ces valeurs ne sont pas simplement des mots; elles doivent être vécues et incarnées par l'équipe. Elles guident les interactions et les décisions et sont la base de la culture Scrum. Ensemble, elles créent un environnement où la collaboration, l'innovation et la livraison rapide sont non seulement possibles, mais encouragées et célébrées. Une compréhension profonde de ces valeurs est essentielle pour quiconque souhaite mettre en œuvre Scrum de manière efficace et authentique.

2.2 L'équipe Scrum

L'équipe Scrum est un élément clé de la méthodologie Scrum et elle se caractérise par sa structure unique et son fonctionnement interdépendant. Elle est composée de trois rôles principaux qui travaillent en étroite collaboration pour atteindre un objectif commun.



Le Product Owner : le Product Owner a la responsabilité de maximiser la valeur du produit. Il comprend les besoins des clients et des utilisateurs et traduit ces besoins en objectifs clairs pour l'équipe. Il gère le carnet de produits (*Product Backlog*) et priorise les éléments en fonction de leur valeur et de leur importance.

L'équipe de développement : l'équipe de développement est composée des professionnels qui font le travail de création et de livraison du produit. Elle est auto-organisée et interdisciplinaire, et les membres de l'équipe collaborent étroitement pour s'assurer que le produit est développé selon les standards de qualité requis. L'équipe de développement est responsable de la livraison de produits potentiellement livrables à la fin de chaque Sprint.

Le Scrum Master : le Scrum Master est le gardien de la méthodologie Scrum au sein de l'équipe. Il facilite les réunions, aide à résoudre les obstacles et soutient l'équipe dans son utilisation de Scrum. Le Scrum Master travaille avec l'équipe pour s'assurer que Scrum est compris et appliqué correctement et qu'il sert l'objectif du projet.

Ces rôles travaillent en symbiose, et la réussite de l'un dépend de la réussite des autres. La collaboration, la communication ouverte et le respect mutuel sont essentiels pour le bon fonctionnement de l'équipe Scrum. Les responsabilités et les attentes sont clairement définies, et chaque membre de l'équipe comprend son rôle et la manière dont il contribue à l'ensemble.

L'équipe Scrum n'est pas seulement un groupe de personnes travaillant sur un projet, c'est une unité cohésive qui partage une vision et un engagement communs envers l'excellence et l'innovation. La dynamique et la composition de l'équipe Scrum sont conçues pour encourager la créativité, la responsabilité et l'efficacité, et elles sont au cœur de la puissance et de la flexibilité de la méthodologie Scrum.

Présentation de l'équipe

Nous allons maintenant vous présenter l'équipe que nous allons suivre à travers ce livre et qui vous permettra de comprendre le rôle de chaque membre d'une équipe Scrum.

Cette équipe symbolise une mise en œuvre typique de la méthodologie Scrum dans un contexte réel.

Thomas, le Scrum Master : Thomas est le gardien de la méthodologie Scrum au sein de l'équipe. Passionné et déterminé, il veille à ce que chaque membre de l'équipe comprenne et applique les principes et les pratiques de Scrum. Il facilite les réunions, aide à éliminer les obstacles et assure que le processus Scrum se déroule sans heurt. Thomas est un leader serviteur qui met l'équipe en premier et travaille en étroite collaboration avec le Product Owner et l'équipe de développement.

Dans le contexte de Scrum, un "leader serviteur" est un type de leadership où le rôle principal du leader est de servir son équipe. Il s'agit d'une approche qui renverse le modèle traditionnel de leadership hiérarchique en mettant l'accent sur les besoins des membres de l'équipe, plutôt que sur ceux du leader.

Thomas, en tant que leader serviteur, se concentre principalement sur :

– **Mettre l'équipe en premier** : Thomas donne la priorité aux besoins de l'équipe, en veillant à ce que tous les membres disposent des ressources, du soutien et de l'environnement nécessaires pour réussir. Il s'assure que l'équipe est motivée, engagée et qu'elle a une bonne dynamique de travail.
– **Travailler en étroite collaboration avec le Product Owner et l'équipe de développement** : en tant que leader serviteur, Thomas facilite la communication et la collaboration entre le Product Owner, qui détient la vision et les exigences du produit, et l'équipe de développement, qui met en œuvre cette vision. Il s'assure que les objectifs, les priorités et les retours d'information circulent efficacement entre tous les intervenants.

L'approche de leader serviteur dans Scrum est souvent incarnée par le rôle du Scrum Master, bien que d'autres membres de l'équipe puissent également adopter ce style de leadership. Le leader serviteur dans Scrum se concentre sur l'élimination des obstacles, l'encouragement de la croissance personnelle et professionnelle de chaque membre de l'équipe, et la promotion des valeurs et principes agiles au sein de l'équipe. Ce faisant, le leader serviteur aide à créer un environnement où l'équipe peut être autonome, auto-organisée et hautement performante, tout en travaillant de manière cohérente vers les objectifs communs du projet.

Clara, le Product Owner : Clara est la visionnaire derrière le produit. Elle comprend les besoins du marché et travaille en étroite collaboration avec les clients et les utilisateurs pour identifier ce qui doit être réalisé. Sophie priorise le carnet de produits et collabore étroitement avec l'équipe de développement et Thomas pour s'assurer que le bon travail est fait au bon moment.

L'équipe de développement : composée de développeurs, testeurs, designers et autres professionnels, l'équipe de développement collabore pour créer le produit. Ils sont auto-organisés et interdisciplinaires, et ils travaillent en étroite collaboration avec Thomas et Sophie pour s'assurer que le produit est développé conformément aux spécifications.

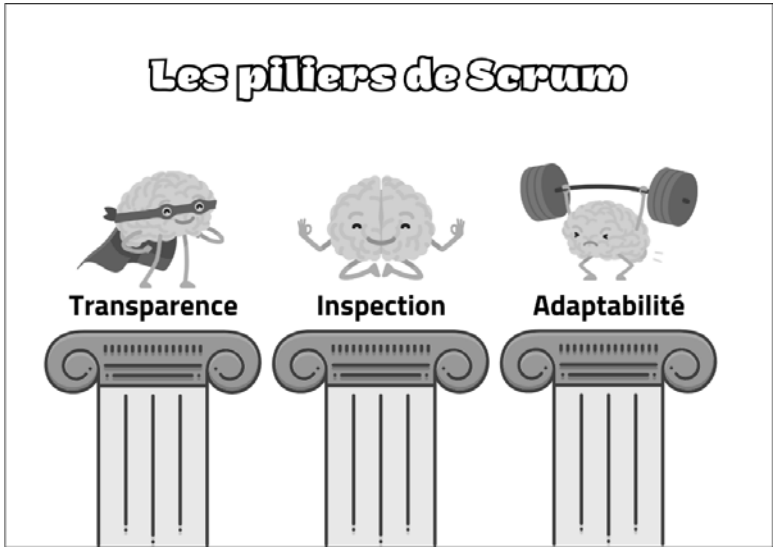
L'équipe de Thomas incarne les valeurs fondamentales de Scrum, telles que la collaboration, l'engagement, la concentration, l'ouverture et le respect. Chaque membre apporte son expertise et ses compétences uniques, et ensemble, ils forment une équipe soudée qui s'efforce d'atteindre un objectif commun.

La réussite de cette équipe dépend de la manière dont ils communiquent, collaborent et se font confiance. Thomas, en tant que Scrum Master, joue un rôle crucial dans la facilitation de cette dynamique et assure que l'équipe reste fidèle aux principes de Scrum.

L'exemple de l'équipe de Thomas sera utilisé tout au long des chapitres suivants pour illustrer divers aspects et concepts de Scrum, ce qui permettra une compréhension plus profonde et plus contextualisée de cette méthodologie agile.

2.3 Les trois piliers de Scrum

Les trois piliers de Scrum sont des principes fondamentaux qui soutiennent l'ensemble de la structure de la méthodologie. Ils sont essentiels pour comprendre comment Scrum fonctionne et comment il peut être appliqué avec succès dans n'importe quel projet. Reprenons l'exemple de l'équipe de Thomas pour illustrer ces concepts.



2.3.1 La transparence

La transparence est le premier pilier de Scrum. Elle implique que toutes les informations relatives au projet soient visibles et comprises par tous les membres de l'équipe. Dans l'équipe de Thomas, cela signifie que chaque personne, qu'il s'agisse de développeurs, de testeurs ou de gestionnaires, a accès aux informations pertinentes. Thomas, en tant que Scrum Master, veille à ce que les processus soient transparents et que les informations soient partagées ouvertement. La transparence aide à construire la confiance et assure que tout le monde est sur la même longueur d'onde.

2.3.2 L'inspection

Le deuxième pilier, l'inspection, concerne la vérification régulière des progrès et des processus pour s'assurer qu'ils sont en ligne avec les objectifs du projet. L'équipe de Thomas utilise diverses réunions Scrum, comme les mêlées quotidiennes et les rétrospectives, pour inspecter leur travail et leur progression. L'inspection permet de détecter les problèmes tôt et de s'assurer que l'équipe est sur la bonne voie. Cela favorise également l'apprentissage continu et l'amélioration.

2.3.3 L'adaptabilité

L'adaptabilité est le troisième pilier de Scrum. Cela signifie que l'équipe doit être prête à changer et à s'ajuster en fonction de ce qu'elle apprend pendant l'inspection. Dans l'équipe de Thomas, l'adaptabilité se produit à plusieurs niveaux. Si un obstacle est identifié lors d'une mêlée quotidienne, l'équipe peut adapter son plan pour le surmonter. Si une rétrospective révèle des domaines d'amélioration, des changements peuvent être apportés au processus pour le sprint suivant. L'adaptabilité assure que l'équipe reste agile et réactive aux changements et aux défis.

2.3.4 Combinaison

En combinant ces trois piliers, l'équipe de Thomas crée une approche de travail solide et flexible qui lui permet de répondre aux besoins changeants de leur projet. La transparence, l'inspection et l'adaptabilité se renforcent mutuellement et chacun est nécessaire pour le bon fonctionnement de Scrum. La compréhension et l'application de ces principes sont essentielles pour le succès de toute équipe Scrum.

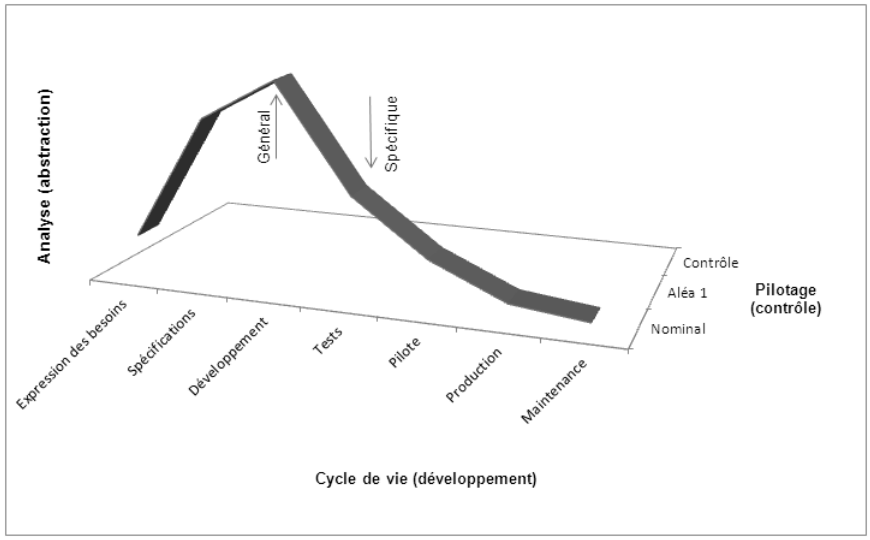
Chapitre 3
La prise en compte du risque

1. Les trois axes

Le chef de projet en devenir est parfois perplexe face au nombre des éléments qu'il doit organiser pour aboutir. Dans quelle direction partir ? Par quoi commencer et comment continuer ?

Pour répondre à ces questions, il faut d'abord se représenter le processus du projet tel que nous l'avons décrit au chapitre Démarrer un projet informatique. Cette fois-ci, nous choisissons un espace structuré par trois axes, le temps (cycle de vie), l'analyse et le pilotage.

Chaque dimension de cet espace correspond elle-même à une méthode plus ou moins standard, qui donnera lieu à un processus que le chef de projet devra suivre.



Le premier axe (en abscisse horizontale) est celui qui décrit les phases du projet, tel que l'on peut le préfigurer. Le schéma ci-dessus propose un modèle de développement assez standard, partant de l'expression des besoins, évoluant jusqu'à la livraison (production) et la maintenance du projet. En toute rigueur, on devrait introduire la mort du projet, car nous savons que le cadre support n'est pas infini. Nous le verrons par la suite, d'autres modèles de développement existent et proposent des organisations différentes.

Vient ensuite l'axe d'analyse (en ordonnée, vertical). Il indique le niveau d'abstraction du projet. Plus l'ordonnée est haute, plus grande est l'abstraction. Inversement, une faible ordonnée équivaut à un niveau de détail très important. Dans l'exemple figurant sur le schéma, on voit nettement l'abstraction grandir au début du projet avant de redescendre progressivement. Ce type de courbe est assez caractéristique des méthodes d'analyse basées sur UML (*Unified Modeling Language*) et Merise. Là encore, d'autres schémas sont envisageables et ils traduisent d'autres façons de procéder.

Sur le dernier axe sont représentés les aléas, les points de contrôles, les prises de décision. Il s'agit de l'axe de pilotage (cycle de décision). Par rapport aux deux autres axes, c'est celui qui réserve le plus de « surprises », mais il ne faut pas non plus le laisser totalement libre. Des méthodes se sont forgées au fil des projets pour organiser le pilotage de manière méthodique et efficace.

Ainsi, le « secret » du démarrage d'un projet résiderait-il dans le choix des méthodes associées à chaque dimension ? Ce n'est pas tout à fait suffisant, car il faut encore s'assurer d'un mélange harmonieux de ces trois composantes, d'une communication efficace, bref d'un sens pratique sans lequel les meilleures recommandations du monde resteront vaines. Autrement dit, l'application isolée, cloisonnée, d'une méthode sur un seul axe ne donne pas de meilleurs résultats qu'un projet dans lequel on oublierait l'existence des autres axes.

Le tableau suivant donne un aperçu du résultat lorsque les méthodes sont appliquées seules (indépendamment les unes des autres) ou lorsqu'elles font défaut :

Méthode/axe	Application seule ou indépendante	Non-application (absence)
Développement	En général, la méthode d'analyse est censurée ou ne donne aucun résultat probant.	Non-conformité (l'expression des besoins est prise en compte tardivement). Dérive du projet (difficulté à stabiliser la solution par manque de tests).
Analyse	Vision beaucoup trop conceptuelle ; le projet devient techniquement risqué car les tests d'intégration sont oubliés.	Non-conformité (focus sur des fonctions mineures, alors que les fonctions majeures manquent). Code non optimal entraînant des difficultés à le faire évoluer (pas de factorisation).

Pilotage	Inadéquation du pilotage par rapport aux caractéristiques du projet.	Non tenue des objectifs, manque de contrôle. Explosion de l'équipe projet en cas de montée en charge. Conduite du changement inexistante : rejet de l'application livrée.

Ce tableau récapitule donc six constructions (*antipatterns*) qu'il convient absolument d'éviter. Et pourtant, les occasions de se tromper ne manquent pas, soit parce que les habitudes ont la vie dure, soit parce qu'une nouvelle méthode n'est pas toujours à même de s'intégrer facilement dans le paysage d'un référentiel de conduite de projets.

Remarque

Les exemples d'achoppement sont légion. Le chef de projet doit toujours garder en tête que les trois composantes sont indispensables et qu'elles doivent coopérer pour être efficaces.

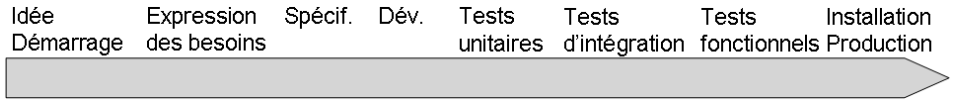
2. Le modèle de développement

Le modèle de développement constitue l'un des premiers choix que doit effectuer le chef de projet. Il façonne de manière explicite l'organisation temporelle du projet et a une grande influence sur le résultat final.

Il n'y a pas de bon ou de mauvais modèle, du moins parmi les grandes classes dont l'énumération suit. Cependant, chaque projet a des caractéristiques qui rendent l'application d'un modèle judicieuse ou au contraire inefficace.

2.1 Le modèle cascade

On l'appelle aussi modèle linéaire ou modèle nominal. C'est certainement le modèle le plus simple, ce qui ne le déprécie pas pour autant !



Dans ce modèle, chaque étape suit la précédente, sans autre nécessité que « d'attendre » son issue ; dès que l'expression des besoins est terminée, les spécifications sont écrites. Au terme de ce travail, le développement est réalisé jusqu'à son achèvement. Les tests unitaires s'enchaînent, et ainsi de suite.

Avant d'expliquer l'appellation cascade, intéressons-nous aux différentes étapes prévues par ce modèle.

Idée - démarrage	C'est le point de départ du projet, le moment où les conditions matérielles sont remplies et où la vision du projet est partagée par l'équipe.
Expression des besoins	Après la décision d'enclencher un projet, on refait le tour des utilisateurs pour recueillir plus en détail leurs exigences fonctionnelles. Ces requêtes sont consignées dans un classeur (papier ou tableur), identifiées, et finalement regroupées ou recoupées.
Spécifications	L'ensemble des requêtes est considéré selon différentes approches dictées par le modèle d'analyse : importance, priorité, difficulté technique, risques… Il ressort des spécifications un découpage technico-fonctionnel de l'application.
Développement	Les spécifications sont traduites en code, en projetant le découpage modulaire (composants) sur la plate-forme choisie par l'architecte.

Tests unitaires	Chaque composant est testé indépendamment des autres. Son comportement doit être conforme aux spécifications, dans la plage de fonctionnement prévue.
Tests d'intégration	Les composants sont le plus souvent assemblés sur une plateforme techniquement hétérogène. La lecture d'enregistrements peut provoquer des problèmes de sécurité, de charge, de robustesse (gestion des valeurs NULL par exemple…). Les tests d'intégration s'assurent que l'édifice final est stable dans toutes les occasions prévues par le cahier des charges.
Tests fonctionnels	Toutes les fonctionnalités de l'application sont testées pour s'assurer du respect du cahier des charges (logique, résultat d'exécution, temps de réponse...).
Installation	L'application est déployée sur sa plateforme définitive (cette étape peut être précédée d'une phase d'industrialisation pour les logiciels packagés). Des ajustements de sécurité sont parfois nécessaires, des chaînes de connexion sont à redéfinir, etc.

Ce modèle est nominal ; chacun est libre d'y ajouter d'autres étapes ou de le remanier, c'est une base de travail. Il est aussi linéaire, aucune étape n'est traitée parallèlement à une autre.

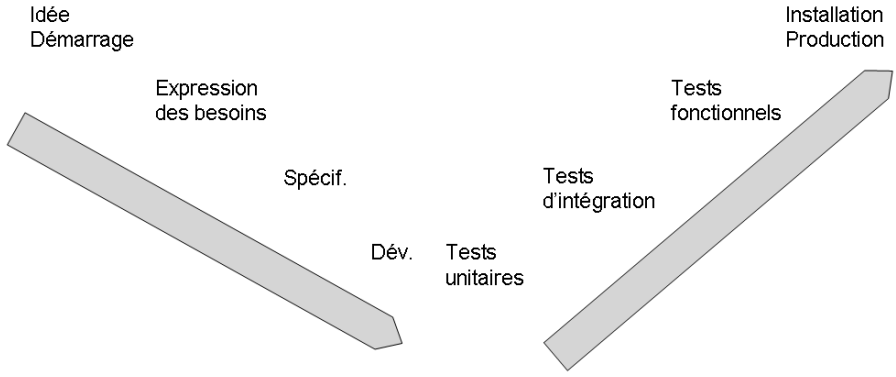
D'où vient l'appellation cascade ? La remontée est difficile, comme pour des poissons qui cherchent à gagner les sources d'une rivière. Les écluses, barrages, ou cascades sont difficilement franchissables. Dans le modèle cascade, un coût unitaire « à la descente » est décuplé s'il faut faire machine arrière. Autrement dit, une erreur de conception se paye de manière exponentielle ! Quelque chose qui a oublié d'être spécifié coûtera 10 fois plus à développer. Si l'oubli est réparé en phases de tests unitaires, c'est 100 fois plus…

2.2 Le modèle en V

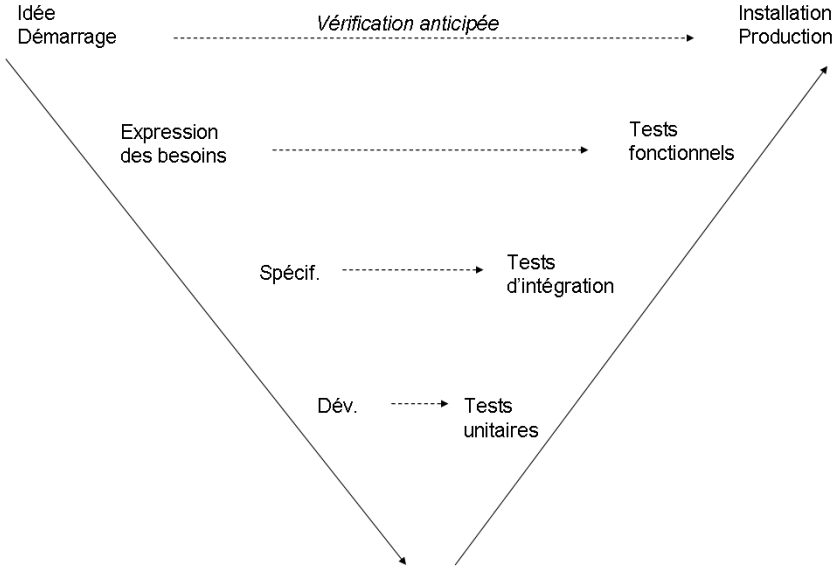
À regarder de plus près le modèle cascade, on remarque une symétrie entre les différentes étapes qui jalonnent le processus. Cette symétrie est justement l'un des points faibles du modèle, puisque les phases de validation sont finalement extrêmement éloignées des phases de spécifications. Chaque acteur du projet se trouve de ce fait piégé dans un tunnel, il est forcé d'attendre la fin du processus pour vérifier la conformité du produit.

Ainsi, l'utilisateur espère que le produit final répondra à ses attentes. Le concepteur ne peut pas vérifier ses hypothèses avant la réalisation des tests fonctionnels. L'architecte ne peut pas intégrer des composants qui n'existent pas encore… Le développeur n'est pas mieux loti puisqu'il y a de grandes chances, vu le manque de recul et de perspective d'ensemble, que ses réalisations soient à reprendre ou à abandonner.

Au lieu de différer au maximum les tests fonctionnels, le cycle en V les anticipe en déroulant le processus de telle manière que ces tests sont sous-traités, réalisés en parallèle, et finalement appliqués au plus tôt.



La sous-traitance des tests est une nécessité, sans quoi leur objectivité serait discréditée. Le parallélisme vise à vérifier « sur le papier » que la composition coïncide avec les besoins exprimés. De cette façon, on anticipe une erreur flagrante de couverture fonctionnelle et on épargne un travail inutile ou défectueux aux développeurs, voire aux analystes.



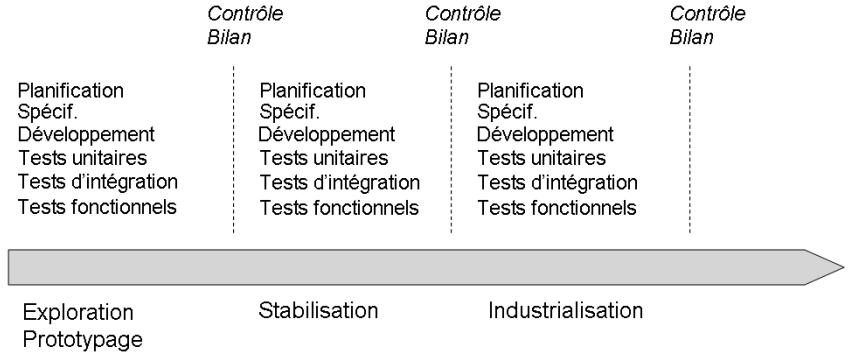
Quels sont les impacts de ce modèle sur la durée du projet ? Le logiciel n'est pas disponible deux fois plus rapidement ! Mais la durée est au contraire rallongée d'environ 30 à 40 %. En effet, les tests sont joués deux fois. La première, de manière fictive, à des fins de contrôle. La seconde, en réel, sur la base d'un logiciel opérationnel.

Le modèle en V constitue donc une amélioration du modèle cascade sur le plan des tests fonctionnels, au prix il est vrai d'une augmentation de la durée du projet.

2.3 Le modèle itératif

À l'inverse du modèle en V, le modèle itératif va anticiper les tests d'intégration, c'est-à-dire s'attacher à diminuer le risque technique. Ce cycle est une succession de cascades dans lesquelles les périmètres fonctionnels et techniques évoluent constamment.

On commence par les parties les plus difficiles pour évacuer le risque d'abandon du projet. Le développement du produit se poursuit par des phases de stabilisation et d'industrialisation.



2.4 Le modèle RAD

RAD est l'acronyme de *Rapid Application Development*. Ce modèle fonctionne sur le principe de l'exploration fonctionnelle, avec une prédominance du maquettage graphique. Le client (ou le demandeur) est fortement impliqué dans les phases de délimitation du périmètre, un peu moins dans celles relevant de la stabilisation et de l'industrialisation.

Pour fonctionner, ce modèle nécessite l'emploi d'une technologie robuste et mature, dotée d'outils de développement de haut niveau. Une plateforme standard est tout à fait à même de supporter le développement et la production de logiciels conçus en mode RAD. Dans ce cas de figure, le risque technique est maîtrisé au travers de l'emploi de technologies simples mais éprouvées.