

Le Cycle en V d'un Projet

Le cycle en V est une méthode d'organisation très connue dont l'origine remonte à l'industrie et qui a été adaptée à l'informatique dans les années 80. C'est l'une des premières méthodes qu'on apprend à l'école, et elle reste toujours d'actualité.



Vous ne connaissez pas le cycle en V? Le cycle en V est un paradigme du développement informatique.

Il décrit les étapes essentielles du développement d'un logiciel, le cycle de vie du projet. Il est représenté par un V dont la branche descendante contient toutes les étapes de la conception du projet, et la branche montante toutes les étapes de tests du projet. La pointe du V, quant à elle, représente la réalisation concrète du projet, le codage; on pourrait donc en déduire, de manière simpliste, que les deux branches montantes et descendantes ne sont que de la documentation! Certes, certes, mais pas que...

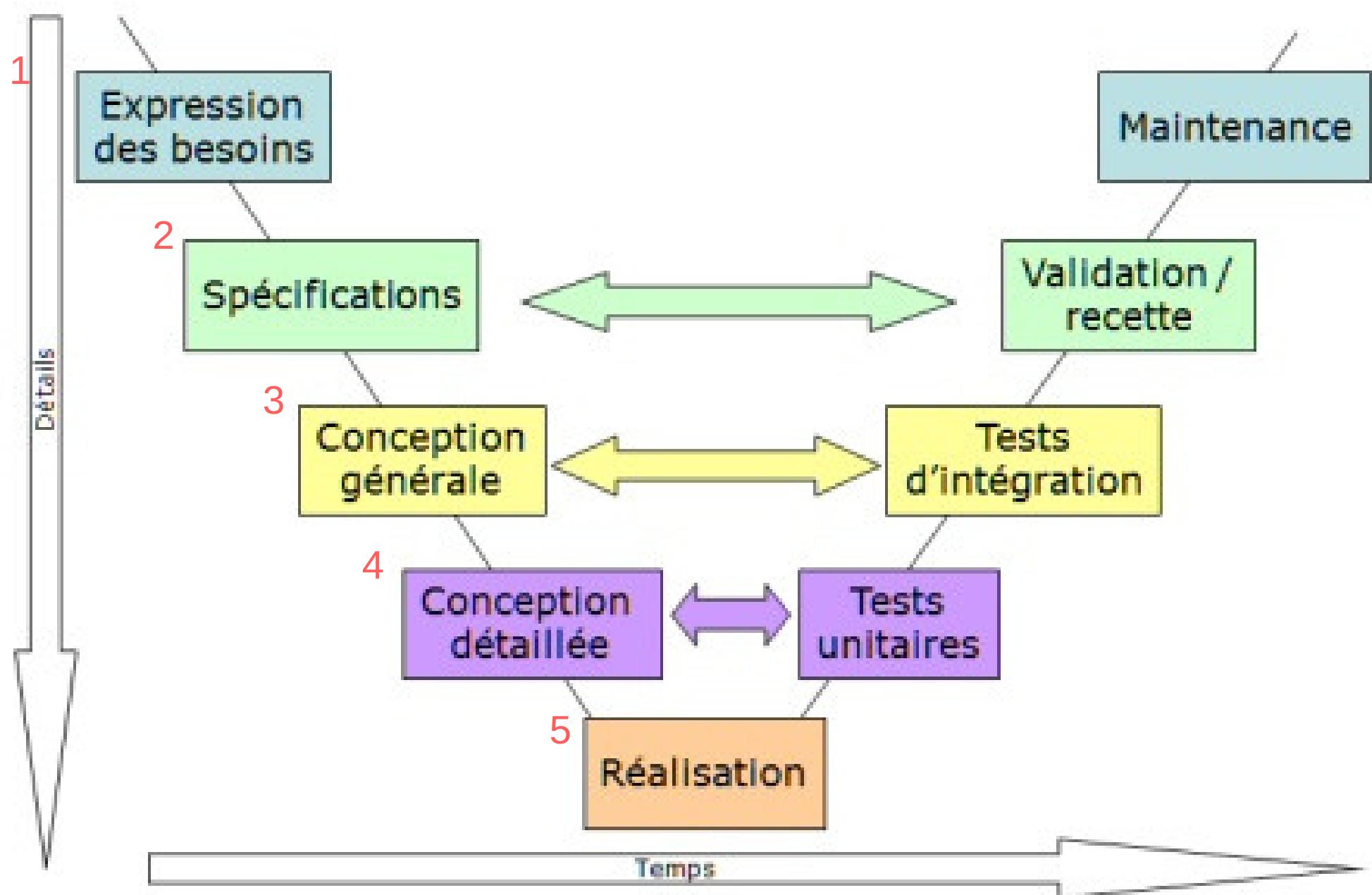
En effet, chaque étape d'une branche a son pendant dans l'autre branche, c'est à dire qu'une étape de conception correspond à une étape de test qui lui est spécifique. A tel point, d'ailleurs, qu'une étape de test peut être élaborée dès que la phase de conception correspondante est terminée, indépendamment du reste du projet.

definition: paradigme

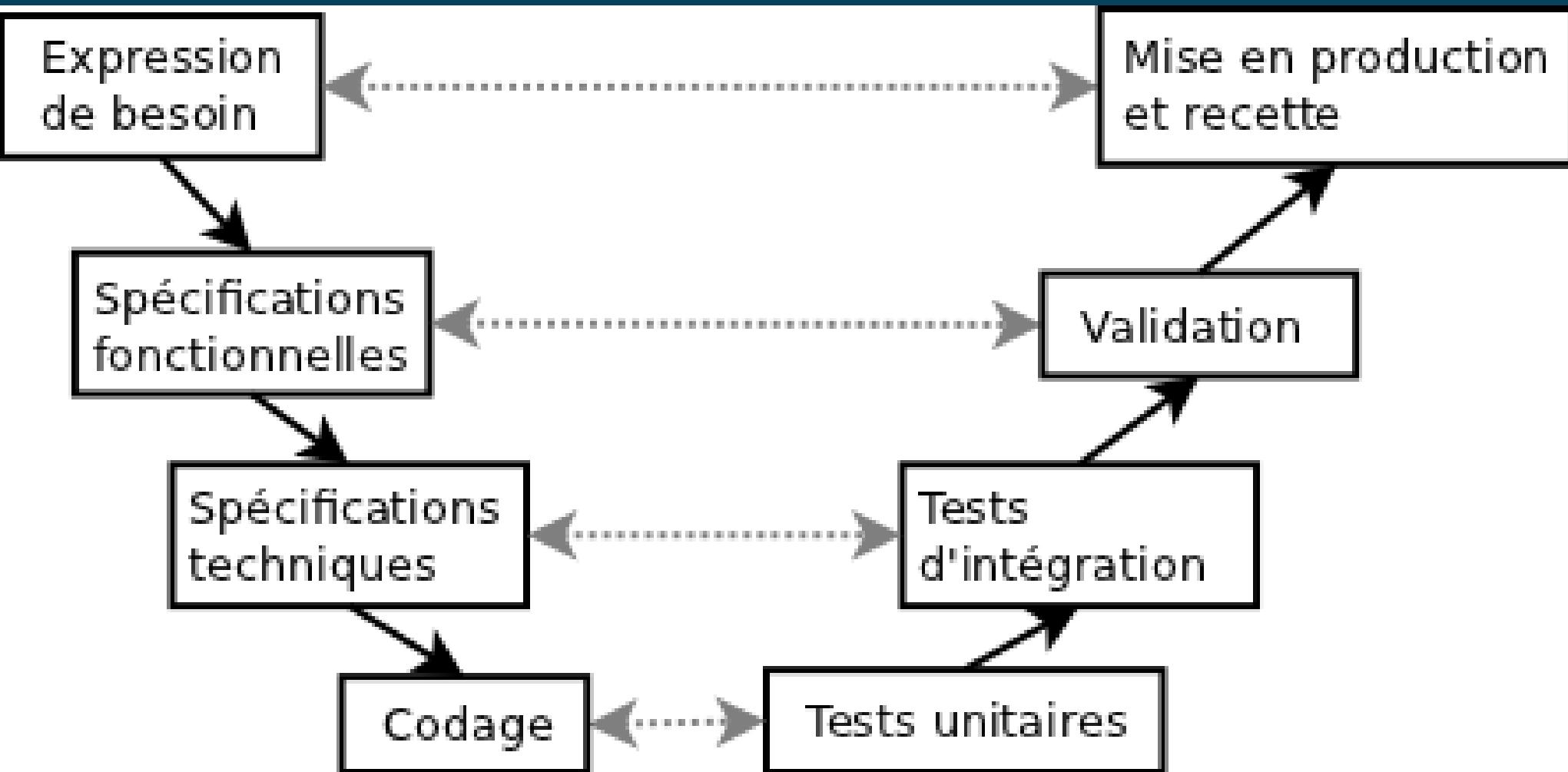
Un paradigme est une représentation du monde, une manière de voir les choses, un modèle cohérent de vision du monde qui repose sur une base définie (matrice disciplinaire, modèle théorique ou courant de pensée). C'est une forme de rail de la pensée dont les lois ne doivent pas être confondues avec celles d'un autre paradigme et qui, le cas échéant, peuvent aussi faire obstacle à l'introduction de nouvelles solutions mieux adaptées. Cette notion est rattachée à celle d'idéologie, au sens de la science des idées, des représentations.

Le paradigme au sens collectif est un système de représentations largement accepté dans un domaine particulier. Cela dit, les paradigmes tendent à différer selon les groupes sociaux et à changer dans le temps en fonction de l'évolution des connaissances (cas notamment des paradigmes en sciences).

Exemple d'un projet en V



La grande force du cycle en V, c'est qu'il définit assez précisément la manière dont les choses devraient se passer.



On peut y distinguer 3 grandes parties : La phase de conception, la phase de réalisation (codage) et la phase de validation. Les phases de conception et de validation se découpent en plusieurs parties. Chaque étape ne peut être réalisée qu'une fois que l'étape précédente est terminée, ce qui diminue les prises de risque sur le projet.

Ce qui est bien visible sur le diagramme, c'est que chaque étape de conception possède son alter ego de validation.

Il devient alors assez aisément de valider un projet, car le référentiel de test est connu très précisément. Ajouter quelques lignes dans le corps du texte

Les différentes étapes

Le cycle en V est constitué de 8 étapes qui ont toutes leur importance.

Expression de besoin :

Le client exprime son besoin, en décrivant les usages correspondant au produit fini tel qu'il peut l'imaginer.

Cela doit répondre aux questions « Que veut-on ? » et « À quel coût ? ».

Spécifications fonctionnelles :

C'est le cahier des charges exact du produit final, tel que le désire le client. Il doit couvrir l'intégralité des cas d'utilisation du produit, en expliquant ce qu'il doit faire et non pas comment il va le faire.

Spécifications techniques :

C'est une traduction des spécifications fonctionnelles en termes techniques. C'est durant l'élaboration des specs techniques que sont choisies les technologies à mettre en oeuvre pour développer le produit, et qu'est conçue l'architecture logicielle du produit.

Codage :

C'est la phase de réalisation à proprement parler, pendant laquelle sont développées des briques qui sont ensuite assemblées pour créer le produit fini.

Tests unitaires :

Ces tests interviennent à un niveau « atomique ». Chaque brique logicielle a été modélisée puis codée durant les étapes précédentes. Les tests unitaires assurent que ces briques respectent de manière individuelle leur cahier des charges.

Tests d'intégration :

Ce sont là les premiers tests grandeur nature du produit fini. On s'assure qu'il suit les indications des spécifications techniques.

Validation :

Le produit est à ce moment testé en regard de la spécification fonctionnelle. Toutes les utilisations qui y ont été définies doivent pouvoir se vérifier dans les faits.

Mise en production et recette :

Le produit est vérifié une dernière fois en pré-production, avant d'être mis en production.

Le client procède à la recette, pour vérifier que son expression de besoin est respectée.

definition: recette

En informatique, la recette (ou test d'acceptation) est une phase de développement des projets, visant à assurer formellement que le produit est conforme aux spécifications (réponse donnée à un instant t aux attentes formulées). Elle s'inscrit dans les activités plus générales de qualification.

Cette étape implique, en la présence effective des différents acteurs du projet, maîtrise d'œuvre et maîtrise d'ouvrage, le déroulement rigoureux de procédures de tests préalablement décrits, et l'identification de tout écart fonctionnel ou technique.

SPECIFICATION

La spécification est souvent le document d'entrée du projet. Elle est même parfois rédigée en commun avec le client. Et quelle quantité de sang elle peut faire couler! Et du sang avec plusieurs ADN!

La spécification, bien rédigée, est une suite d'exigences, si possible regroupées par thèmes, pour permettre d'esquisser un début d'architecture et aussi un plan de test, mais bon on peut toujours rêver.

Chaque exigence devrait être testable, mesurable, etc., donc on devrait vivre sur la planète des bisounours. Ne rêvons pas, la spécification est la pomme de discorde de tout projet digne de ce nom: "Ah oui mais là, ce que vous demandez est en dehors du périmètre, ça met en péril le prochain jalon, on va devoir faire un avenant", "Votre couverture de test est incomplète, la livraison est refusée". Ca vous fait pleurer, vous aussi? Ca vous fait penser à vos matrices de traçabilité pas à jour? Bienvenue au club!

SPECIFICATION-bis

Avant de démarrer un projet, la spécification doit absolument être bétonnée, figée, cristallisée, statufiée, protégée, et si possible dans un outil de gestion de configuration, sans cela point de salut: les entrants instables garantissent à 100% la dérive de votre projet, avec bonne ambiance garantie avec votre (futur ex-) client lors des réunions d'avancement et autres comités de pilotage. Il faut absolument borner ce document, comme tous les documents d'entrée d'ailleurs. Au début de tout projet, je préconise de lever automatiquement ce premier risque: instabilité des entrants, avec tous les indicateurs au rouge. Je vous aurai prévenus.

CONCEPTION

La conception regroupe les activités d'étude qui suivent la spécification, et ce jusqu'au codage. Le codage n'est que la matérialisation de la conception.

La conception englobe:

la modélisation: facultative, assistée ou non d'un outil, elle permet de définir et de visualiser le système ou une partie de celui-ci, aussi bien de manière statique que dynamique. La modélisation n'est pas une étape en soi, elle peut être utilisée à tout endroit du projet (pourquoi ne pas modéliser les tests, par exemple?). Outre ses qualités techniques, l'aspect visuel de cette pratique provoque un fort impact chez le client, qui peut plus facilement imaginer son produit fini.

CONCEPTION-bis

l'architecture, ou conception préliminaire: indispensable à partir d'une certaine taille de projet, elle définit l'ensemble des briques constitutives de l'application et leurs interfaces. C'est souvent le point faible d'un logiciel, car un défaut à ce niveau, donc probablement détecté en phase d'intégration, sera toujours pénible à corriger de par ses conséquences: débogage long, reprise d'intégration dans un outil de gestion de configuration, régénération de l'exécutable, traitement de la demande de correction, etc. C'est au niveau des API que le bâblesse. A soigner, donc.

La conception est donc une étape à ne pas négliger, elle n'est pas plus importante qu'une autre au niveau contractuel, mais techniquement une conception bâclée plantera un projet de manière sûre et certaine.

CODAGE

Le codage consiste à mettre en forme la conception détaillée, donc transformer du pseudo-code en code. La tâche paraît simple et faire figure de parent pauvre du développement.

Et pourtant: le codeur doit savoir lire une conception détaillée (ne riez pas), et surtout doit connaître le langage utilisé. Pour l'assembleur cela paraît aller de soi, et ce serait rapidement, enfin on l'espère, détecté en cas d'incompétence, mais quid des langages dits évolués? Êtes-vous sûr que votre développeur maîtrise le C/C++, Java, etc.? Connait-il les règles de programmation en vigueur dans la société? Comment gère-t-il les réservations/libérations de mémoire? les constructeurs? les destructeurs? les pointeurs? et la mise à jour de la conception détaillée...(c'est la question qui tue). Donc attention de ne pas confier cette tâche à n'importe qui, une faille à ce niveau ne mettra probablement pas le projet en péril, mais par contre les délais risquent d'en pâtir.

TESTS UNITAIRES

Les tests unitaires semblent être à la phase de test, branche remontante du cycle en V, ce qu'est la conception détaillée dans la branche descendante. Ca "embête" tout le monde. Et pourtant... Quel plaisir de pouvoir être sûr du comportement d'une fonction et donc d'être en mesure de pouvoir dire que seuls les paramètres d'appel de la fonction peuvent être incriminés! Que de temps gagné! Et quel plaisir de rejouer les tests unitaires d'un seul clic! Et pan dans la régression!

Le seul problème est le temps mis pour développer un test unitaire, temps que l'on peut empiriquement évaluer comme étant pratiquement identique au temps mis pour développer la fonction testée..

Tout est affaire de stratégie: dans moult projets, les tests unitaires sont ignorés pour gagner du temps. A charge du chef de projet d'évaluer s'il préfère gagner du temps à court terme ou bien gagner du temps et beaucoup d'argent sur la maintenance.

TESTS D'INTEGRATION

Les tests d'intégration consistent à tester le comportement de l'application en intégrant progressivement toutes les briques du logiciel. C'est une étape laborieuse qui, non seulement en butte aux problèmes d'interface des composants logiciels, est confrontée à des problèmes autres que purement fonctionnels: compilation, compatibilité de versions, etc. Bon courage à l'intégrateur, qui doit connaître aussi bien le système que l'outil de gestion de configuration. Sans compter le relationnel qu'il doit mettre en oeuvre pour concilier les intérêts des responsables des différentes briques logicielles mises en cause.

Devant la complexité de la tâche, ces tests peuvent difficilement être automatisés, et souvent ils sont longs à dérouler. Tout problème détecté à ce niveau peut être lourd de conséquence, aussi bien en terme de délai qu'en terme de coût (ou de ressources), car on approche de la fin du projet, donc du jalon final. Les samedi ouvrés ne sont pas loin...

TESTS DE VALIDATION

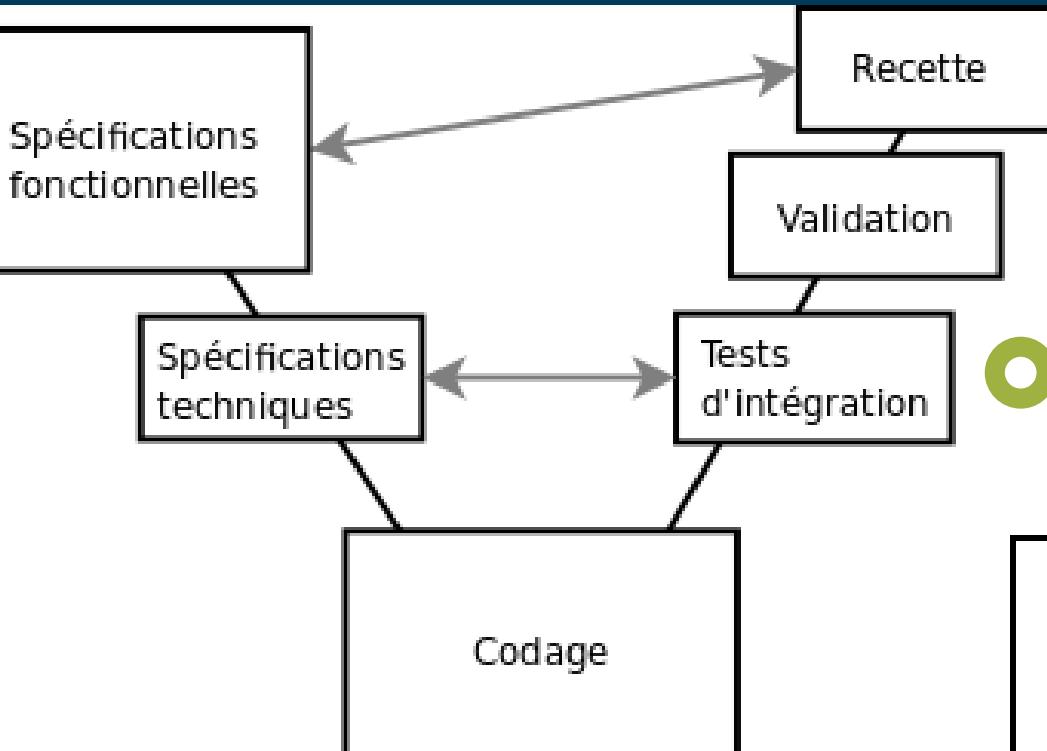
Les tests de validation consistent à dérouler les tests permettant d'affirmer que le logiciel répond aux exigences de la spécification.

Ce sont souvent des scénarios déroulés automatiquement. Et heureusement, car une campagne de test est souvent très longue, plusieurs heures voire plusieurs jours.

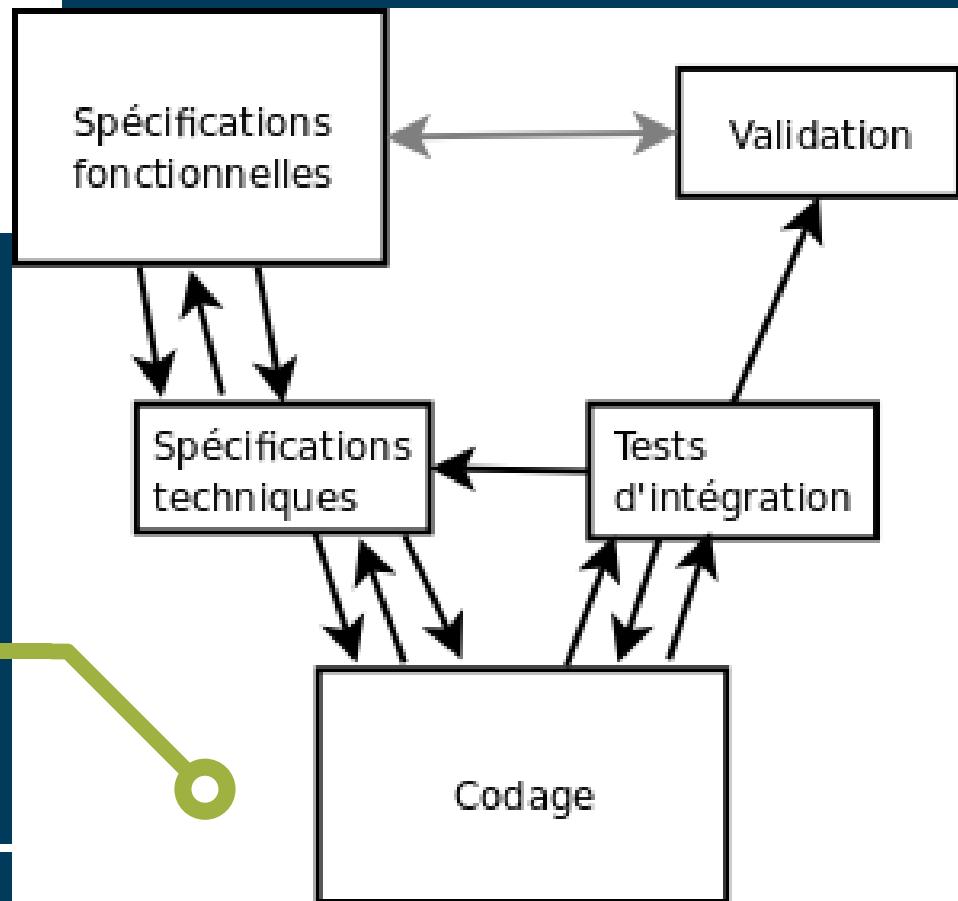
A ce stade, tout problème détecté est potentiellement catastrophique, car il est susceptible de remettre en cause la spécification, donc tout le développement.

Et dans ce cas là, vous êtes très content d'avoir le dimanche pour vous reposer de vos longues soirées passées au bureau.

Le plus gros problème avec le cycle en V, c'est que dans la réalité il est quasiment impossible d'obtenir des spécifications fonctionnelles (ou techniques) qui se suffisent à elles-mêmes et qui ne seront pas impactées par la suite. C'est souvent pendant l'implémentation du produit que l'on identifie les cas limites et les problèmes conceptuels. Dans ce cas, une stricte application de cette méthode forcerait à reprendre le cycle depuis le début, en intégrant au niveau fonctionnel les remontées d'ordre technique, ce qui implique des délais conséquents..



Malheureusement, si le cycle en V est limpide d'un point de vue théorique, son application réelle est très difficile. Dans une grande majorité de cas, on voit des organisations qui ressemblent plutôt à ce schéma :



Une telle organisation possède encore des avantages structurels assez convaincants, car elle définit toujours les différentes étapes de la réalisation d'un projet. Mais bien souvent, on se retrouve en réalité avec une organisation qui ressemble plutôt à quelque chose comme ça :



ET AU FINAL, C'EST BIEN ÇA LE PROBLÈME DE CETTE MÉTHODE : SON MANQUE DE SOUPLESSE. C'EST POUR CELA QUE D'AUTRES TYPES D'ORGANISATIONS SONT APPARUES, ET NOUS VERRONS CELA TRÈS BIENTÔT.

made
with
love