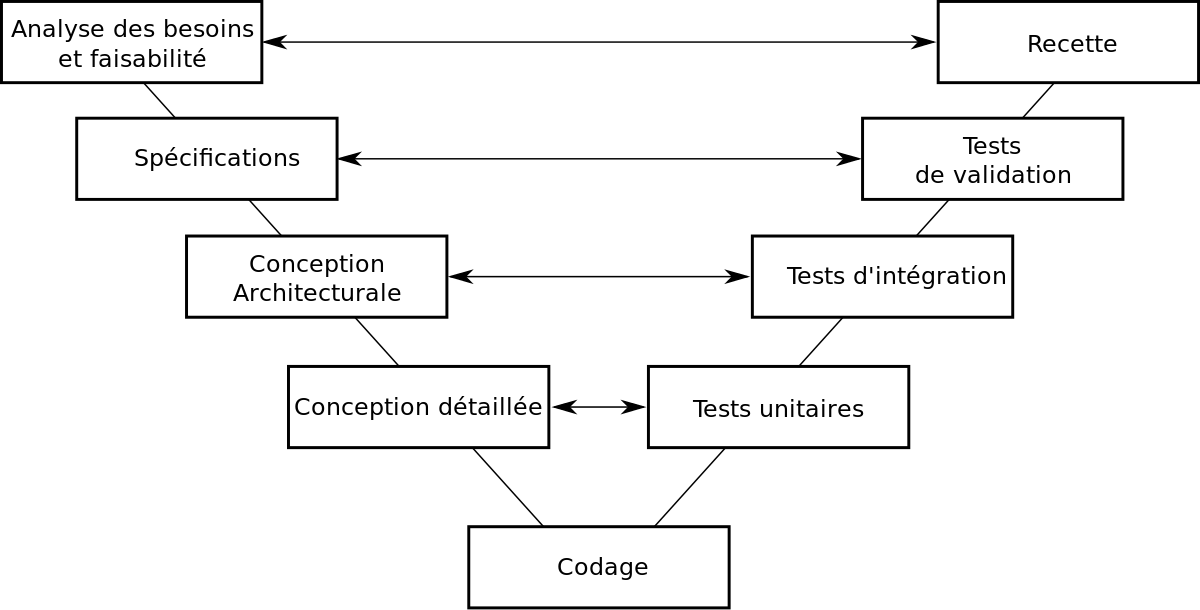
Tips



CdC : Un cahier des charges est très souvent un document très vague fournit par le client. Il est en lien direct avec la livrable (recette).

Spécifications générales : Pour respecter le CdC, il va falloir avoir des fonctions particulières à effectuer. Ici, on cherche à trouver les critères nécessaires au bon fonctionnement du projet. On définit à partir de cette partie les tests d’intégration.

Spécifications détaillées : On détaille d’autant plus les spécifications. Par exemple : comment le µC doit gérer telle ou telle sous-système… Il est en lien direct avec les tests unitaires. On fabrique des briques technologiques qui seront validé par quelqu’un d’autre de préférence à la suite de la programmation.   
  
Codage…

En réalité, un cycle en V est difficile à respecter. On passe beaucoup de temps à voyager en les différentes phases. On est parfois obligé de coder pour faire les spécifications (principe des méthodes agiles).

Avantage du cycle en V : lorsqu’il est respecté, et que si un jour on veut partir de son taff, et que si un gus arrive derrière et ne récupère que du code que l’on a fait, il risque d’être paumé. Sans documentation spécifiant et décrivant l’architecture des fonctions est nécessaire pour une bonne compréhension des systèmes.

Il est nécessaire de savoir prouver à notre client que les tests unitaires ont été effectués et que le travail effectué mène à des résultats qui fonctionnent. Egalement, il faut que les modules créés fonctionnent ensemble et mène aux résultats attendus, d’où l’intérêt des tests d’intégrations.

Notre projet doit donc débuté par des documents de spécification. Le cycle en V peut également s’appliquer sur différent petit module déjà présent dans le projet général afin de pouvoir mieux assimiler les différents éléments qui devront être pris en main.

Concernant notre module MSP430, si le µC est capable de communiquer en UART avec le PC, alors cette partie étudiée peut être considérée validée car le MSP sera capable de communiquer avec n’importe quel module en UART. Ce bloc technique, maintenant fonctionnel, peut donc être partagé avec n’importe quel personne travaillant sur le projet nécessitant cette fonction-ci.

MISRA : établissement d’un ensemble de règles qui permettent d’éviter des erreurs générales.

Exemple : GOTO : fonction permettant d’aller à un endroit choisi du code. Cette fonction crée de gros problèmes de sauvegarde de valeur de variables. Cette fonction est nulle. Clairement. « Il faut être au poil de cul de l’optimisation pour justifier l’utilisation des GOTO. »

Seront donc à fournir en se basant précisément sur les demandes du cahier des charges :

* Document de spécification
* Code sources
* Rapports de tests
* Livrable

i.e. :

🡪 2 rapports LDRA : nécessaire pour la validation du code.

🡪 Gestion de configuration logiciel (exemple : utilisation de **Git**) : nécessaire pour la gestion de fichiers avec utilisation/édition en multi-partage. Ceci permet de sauvegarder les différentes versions des codes ce qui offre la possibilité de faire des back-up pour retrouver des bugs éventuels.

🡪 .