1 Ingénierie de développement de produits technologiques

Il faut être conscient qu'il n'existe pas de produit technologique avancé parfait, et en particulier il n'existe pas de gros logiciels sans erreurs.

Un enjeu majeur du développement de produits technologiques est de limiter les risques liés à des erreurs de prédiction des fonctions du produit développé. Il s'agit de circonscrire ces erreurs aux fonctions non critiques, ou bien prévoir des « voies alternatives » comme dans le vivant pour les fonctions critiques.

Dans le cadre du développement de programmes, les buts du génie logiciel sont de proposer des méthodes et des techniques de développement de logiciels pour assurer la *fiabilité* des logiciels. Adopter une démarche d'ingénierie pour développer un produit logiciel implique une certaine lourdeur inhérente au développement de tout produit technologique. Cela a un coût : en informatique, le coût est équivalent au temps ingénieur car le prix du matériel informatique et du temps calcul est généralement négligeable par rapport au poids des salaires des ingénieurs, contrairement à la biologie « humide » où les expériences constituent le coût principal.

2 Les méthodes de développement

Ces différentes méthodes définissent ce que l'on appelle souvent le « Cycles de vie » des produits développés. Elles reposent sur des activités de différent types combinées entre elles durant le temps de développement et de maintenance. Il existe 4 types d'activités principaux :
- la spécification et la conception (parfois par prototypage),
- le développement, qui en logiciel correspond à l'étape de codage,
- la validation et le test,
- la maintenance et l'évolution du produit.

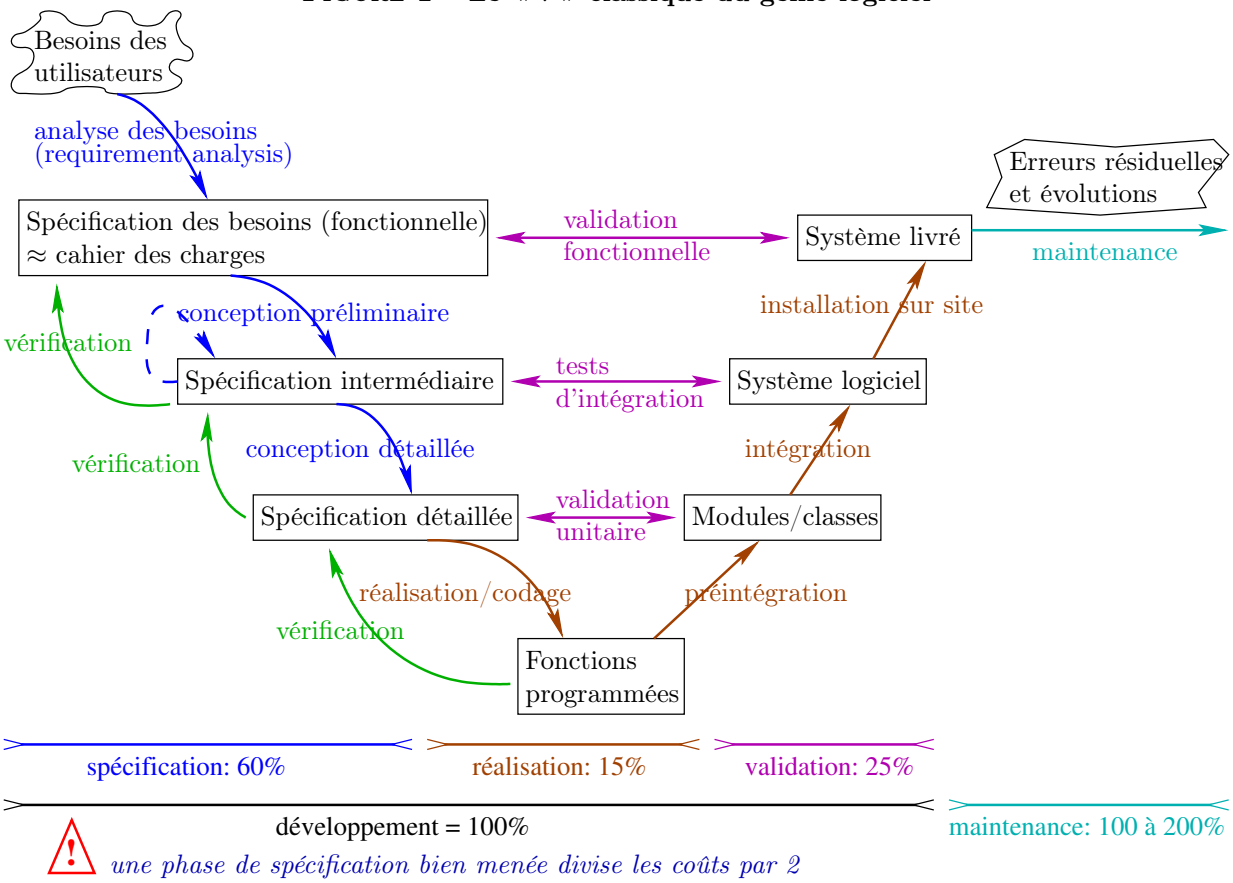
Le « cycle de vie » le plus classique en technologie (ici instancié pour l'informatique) est donné en figure 1.

La partie descendante du « cycle en V » est la *conception* du produit technologique.
- Partant de la formulation de ses besoins par le client (souvent informelle, incomplète, floue, voire contradictoire) il faut d'abord établir une *spécification des besoins* qui précise clairement sur papier ce que l'utilisateur attend du produit à créer. Cette spécification des besoins est à peu de choses près le futur mode d'emploi, si ce n'est que le document est sans doute un peu plus technique. En particulier la spécification des besoin peut être également vue comme un cahier des charges. L'activité de création de la spécification des besoins à partir de leur expression informelle par le client est appelée *L'analyse des besoins*.
- Il s'agit ensuite de commencer à structurer les différentes parties du futur prototype, qui seront utiles pour réaliser toutes les fonctionnalités demandées dans la spécification des besoins. Cela conduit à écrire une spécification qui commence à décrire, d'une part, comment combiner ces parties pour obtenir le produit global, et d'autre part, à spécifier proprement ce que devra faire chaque partie. De telles spécifications sont appelées des *spécifications intermédiaires* et l'activité de les concevoir est appelée le *raffinement de spécification* ou encore la *réification de spécification*. Si les parties mises en jeu doivent offrir des fonctionnalités elles-mêmes assez compliquées, on doit elles aussi les raffiner et cela peut donc donner lieu à plusieurs niveaux de spécifications intermédiaires si le produit à développer est particulièrement conséquent. Ces différents niveaux sont donc des étapes de conception intermédiaire.
- À chaque étape de raffinement, il est nécessaire de se convaincre que la nouvelle spécification « de bas niveau » réalise correctement la spécification précédente « de plus haut niveau ». Ces activités se nomment la *vérification*.
- Arrive enfin un niveau de détail où un technicien (ou un programmeur s'il s'agit de logiciel) n'aura pas de difficulté à réaliser les fonctions demandées dans chaque partie définie par la spécification. Cette dernière étape est appelée la conception détaillée et la dernière spécification est appelée la *spécification détaillée*.

En bas du V se trouve l'étape de *réalisation du prototype*. Il s'agit de produire les différentes pièces détachées du produit final. Cette étape est appelée le codage s'il s'agit d'un logiciel, qui produit donc des morceaux de programmes.

Il faut ensuite regrouper les pièces détachées (ou ces programmes) pour obtenir des éléments préassemblés (des *modules* informatiques) et cette étape est la *pré-intégration*, qui réalise chacune des parties de la spécification détaillée. Si le produit technologique est matériel, il s'agit simplement d'un préassemblage des plus petits composants. Les modules (ou parties matérielles préassemblées) sont ensuite interconnectés pour réaliser chacune des parties de la spécification

Figure 1 – Le « V » classique du génie logiciel



intermédiaire et ces différentes étapes (en nombre qui dépend du nombre de spécifications intermédiaires) constituent l'*intégration*.

Les étapes de pré-intégration et d'intégration doivent faire l'objet de *validations* par rapport aux spécification de même niveau, par des tests adaptés.

À ce stade, on a une version complète du prototype. Il reste à gérer la production effective du produit technologique. S'il s'agit d'un logiciel, il est définitif mais tourne sur les machines et l'environnement logiciel de l'équipe de réalisation du logiciel : il se peut que la présence de tout l'outillage logiciel à disposition sur le site de développement, qui n'est pas présent sur la/les machine/s cible/s du client, « cache » des dysfonctionnements potentiels. L'activité de « production » est donc de délivrer le logiciel sur son site cible.

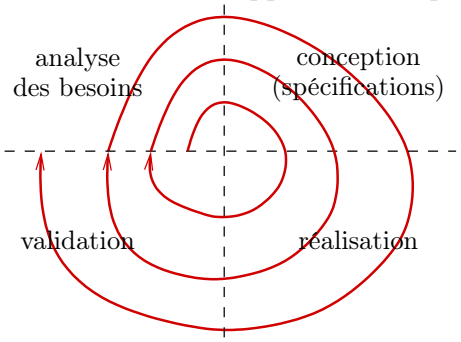
Ensuite commence l'exploitation du produit, la correction des bugs résiduels, l'adaptation des paramètres, *etc.* Cette étape est la *maintenance* et l'évolution des besoins conduira à modifier le produit pour le faire évoluer.

Le défaut principal du développement en V est qu'il n'est pas incrémental : toutes les fonctionnalités du produit final doivent être prévues dès le départ et réalisées ensemble, sans réelle considération de priorité entre les fonctionnalités. Bien souvent, lorsque le produit n'est pas soumis à des réglementations drastiques (souvent en raison de potentielles lourdes conséquences de la moindre erreur) on peut se permettre de repousser à plus tard la conception et la réalisation des fonctionnalités « secondaires » et de commencer par faire un « noyau » du produit en cours, ne contenant que les fonctionnalités principales, puis de l'enrichir par étapes.

Chaque étape d'enrichissement donne lieu aux activités du V : analyse des besoins, spécification, réalisation et validation et l'on parle alors souvent de *développement en spirale* représenté en figure 2. Le premier « tour de spirale » effectue une analyse des besoins restreinte aux fonctionnalités indispensables du produit, les spécifie, les réalise et les valide. On obtient ainsi le « noyau » sus-mentionné. Chaque tour supplémentaire détermine quelles fonctionnalités doivent être ajoutées (analyse des besoins), les spécifient de manière cohérente avec les spécifications préexistantes du « tour d'avant », les réalise et revalide l'ensemble du produit.

Comme on l'a souligné, ces méthodes incrémentales ne sont pas toujours applicables, en particulier quand les conséquences des erreurs peuvent être lourdes et/ou la réglementation impose des normes précises (santé, transports, nucléaire, aérospatiale, *etc*).

Figure 2 – Le développement en spirale



3 Quelques chiffres pour les produits logiciels

3.1 sur la nécessité d'appliquer des techniques d'ingénierie

Voici des chiffres résultant des développements artisanaux qui étaient de mise au début des années 1980, avant que le génie logiciel s'impose comme incontournable. Ils sont donnés en pourcentage du coût total, sur une étude ayant porté sur 7M$ durant le début des années 1980.

Cinq « niveaux de réussite » ont été considérés :
- les logiciels livrés au client et utilisés sans modifications majeures représentent seulement **2%**,
- les logiciels livrés et utilisés après modifications majeures représentent seulement **3%**,
- les logiciels livrés, ayant fait l'objet de modifications majeures, mais rapidement abandonnés représentent **19%**,
- les logiciels livrés mais jamais utilisés par le client représentent **48%**
- enfin ceux jamais livrés, donc non payés par le client, représentent **28%**.

Comme on le voit, au moins 95% des coûts de cette étude ont été dépensés en pure perte avec des approches artisanales. Il est donc d'appliquer des méthodes professionnelles d'ingénierie pour éviter de grosses pertes pour l'entreprise.

3.2 sur la négociation des délais

À produit fixé, le développement par plusieurs personnes ou plusieurs équipes multiplie automatiquement par au moins 3 le coût du logiciel par rapport à une seule personne ou une seule équipe car cela s'accompagne par un alourdissement des spécifications et des procédures de communication. Par conséquent mettre 3 personnes sur un logiciel ne permet pas de diminuer le temps de développement du produit ; il en faut plus...

Il est donc très rentable de négocier la date de rendu d'un produit technologique : plus le temps de conception/développement est grand, moins le nombre de personnes à affecter au projet est grand, et moins les procédures de spécification détaillées et de synchronisation des équipes sont coûteuses.

3.3 sur les jugements de valeur

Il faut totalement bannir les jugements de valeur au sein d'une équipe de conception/développement. C'est infondé et contre-productif.

Par exemple, en moyenne, 90% des développeurs d'un logiciel produisent 15% du logiciel final (donc 10% des développeurs produisent 85% du logiciel final). Cela signifie-t-il que seuls 10% des développeurs travaillent vraiment ? Non. En fait, la productivité d'une personne dépend, de manière presque intrinsèque, de son rôle dans l'équipe. L'expérience suivante a été menée à plusieurs reprise, avec systématiquement le même résultat :
- on constitue 10 équipes de bonne taille pour développer chacune un logiciel ;
- on observe invariablement dans chacune de ces équipes la répartition 90%-15% sus-mentionnée ;
- on « mélange » ces équipes de telle sorte que l'une des équipes est constituée des 10% les plus productifs des équipes précédentes ;
- non seulement on constate que cette équipe n'est pas notablement plus performante que les autres sur de nouveaux projets,
- mais de plus on observe de nouveau la répartition 90%-15% sus-mentionnée au sein de cette équipe.