

Méthodes de développement

Développement itératif

1 - Introduction.....	2
2 -Développement séquentiel et développement itératif	2
2.1 Développement séquentiel	2
2.2 Développement itératif	2
2.3 Intérêts et problèmes	3
3 - Caractéristiques du développement itératif.....	4
3.1 Itérations et phases	4
3.2 Durée des itérations	5
3.3 Déroulement des activités	5
3.4 Planification du développement itératif et plan de développement	5
3.5 Revues	7
4 - Exemple de développement itératif	7
5 - Exemple de processus itératif simplifié pour prototype	9
6 - Exemple du RUP (Rational Uniform Process)	11
6.1 Contenu du RUP.....	11
6.2 Intérêts et problèmes	12
7 - Les activités à conduire.....	13
8 - Conduite des activités de développement	13
8.1 Analyse des exigences du logiciel (spécification).....	13
8.2 Conception de l'architecture du logiciel (conception préliminaire)	14
8.3 Conception détaillée et codage et essais	14
8.4 Intégration du logiciel	15
8.5 Validation du logiciel	15
9 - Conduite des activités transverses	16
9.1 Gestion du projet	16
9.2 Documentation	17
9.3 Gestion de la configuration	18
9.4 Assurance qualité	18
10 - Conduite des activités complémentaires	19
10.1 Rédaction du manuel d'utilisation	19
10.2 Achat et évaluation des outils.....	19
10.3 Achat et évaluation des matériels.....	19
10.4 Mise en place des plates-formes.....	19
10.5 Entretien des matériels et outils	20
10.6 Formation	20
10.7 Livraison.....	20
10.8 Installation des matériels et logiciels	20
10.9 Recette	21
11 - Conclusions.....	21

1 - Introduction

Le développement d'un logiciel nécessite de conduire les activités du cycle en V : analyse du besoin, conception de l'architecture, conception détaillée, codage et essais, intégration, validation.

Ces activités peuvent être réalisées consécutivement, ce qui est recommandé pour les logiciels critiques, mais il existe d'autres techniques qui permettent de réduire les risques et les délais de fourniture, et l'une de ces techniques est le développement itératif.

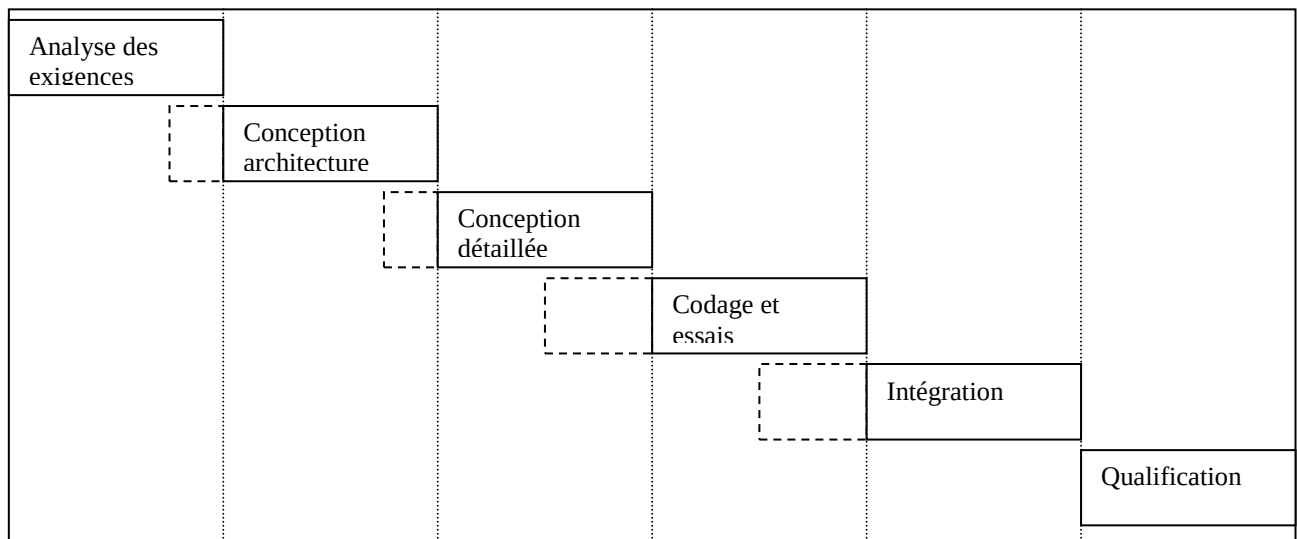
Nous allons donc présenter ce procédé de développement et son intérêt, puis détailler la conduite des activités.

2 -Développement séquentiel et développement itératif

2.1 Développement séquentiel

Dans un développement de type séquentiel, les activités du cycle en V sont exécutées consécutivement avec éventuellement de petits recouvrements. Cela permet de prendre en compte l'ensemble du problème à chaque activité et de vérifier les résultats d'une activité avant de passer à la suivante.

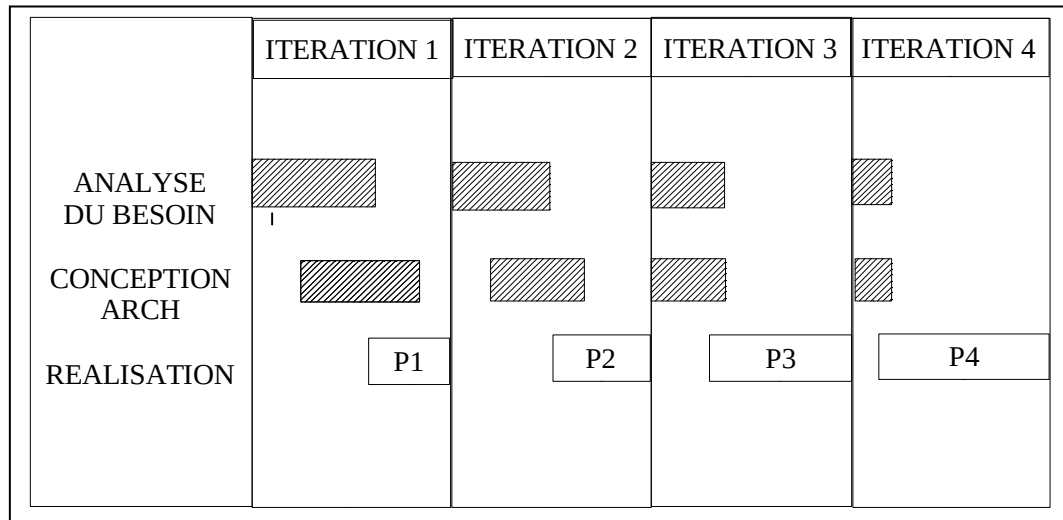
En contrepartie le produit n'est réalisé et montrable qu'en fin de projet. On risque de découvrir à ce moment des problèmes graves qui empêchent le produit de fonctionner, ou de présenter au client un produit qui ne lui convient pas. On peut certes réduire ces risques en réalisant des maquettes en début de projet, mais elles ne permettent pas de supprimer tous les risques.



2.2 Développement itératif

Dans un développement itératif, on réalise des prototypes successifs en enrichissant à chaque fois les fonctionnalités jusqu'à ce que le produit soit complet.

Le développement est découpé en itérations de durée limitée. Dans chaque itération on réalise un prototype en ajoutant des fonctions supplémentaires au prototype réalisé à l'itération précédente. On fait des itérations jusqu'à ce que le produit soit complet.



Les premières itérations peuvent être limitées à des travaux d'analyse du besoin et d'architecture et ne pas aller jusqu'à la réalisation du prototype.

Les activités du cycle en V du logiciel sont réalisées de façon incrémentale au cours des itérations. Par exemple la spécification technique de besoin est rédigée partiellement pendant la première itération, puis complétée à chacune des itérations suivantes.

Le premier prototype réalisé correspond généralement à une structure d'accueil ou base applicative (écran principal, réseau...). Dans les prototypes suivants on ajoute des fonctions de l'application jusqu'à ce que toutes les fonctions soient implémentées.

2.3 Intérêts et problèmes

Ce type d'approche présente plusieurs types d'intérêt :

- on peut s'assurer rapidement que le produit répond aux attentes du client en lui montrant les premiers prototypes, d'où une réduction des risques d'insatisfaction du client, et dans le cas contraire corriger le tir avant qu'il ne soit trop tard,
- on peut diminuer considérablement les risques d'échec du projet en traitant les points durs lors des premiers prototypes,
- les travaux d'intégration sont simplifiés, puisque chaque intégration ne porte que sur une partie limitée du logiciel, et on a toujours un produit réalisé (mais pas forcément complet) à l'expiration du délai du projet.

En contrepartie :

- la réalisation incrémentale de l'analyse du besoin, de l'architecture et de l'intégration peut conduire à des travaux supplémentaires par rapport à un développement séquentiel, d'une part en raison des multiples intégrations, d'autre part il y a des risques d'avoir à reprendre des codages suite à la progression des travaux d'architecture et d'analyse du besoin,
- la définition incrémentale de l'architecture et la prise en compte de la réalisation incrémentale peut conduire à des solutions moins optimisées que dans le cas d'une définition globale de l'architecture et une réalisation globale,

- le processus est mal adapté à la réalisation de logiciels critiques ou de logiciels sûrs où les résultats de chaque activité doivent faire l'objet de vérifications importantes qui peuvent les remettre en cause.

En raison de ses avantages, le processus itératif est recommandé chaque fois que l'on n'a pas des contraintes de sûreté de fonctionnement, de criticité ou d'optimisation importante.

Nota :

- *un logiciel doit être garanti sûr si son mauvais fonctionnement peut mettre en jeu des vies humaines ou causer des dégâts ou des pertes considérables (par exemple logiciel de pilote automatique d'avion),*
- *un logiciel critique est un logiciel dont le mauvais fonctionnement peut conduire à l'abandon de la mission du système ou à des dégâts ou des pertes importantes (par exemple logiciel de guidage d'un missile, logiciel de distributeur automatique de billet...).*

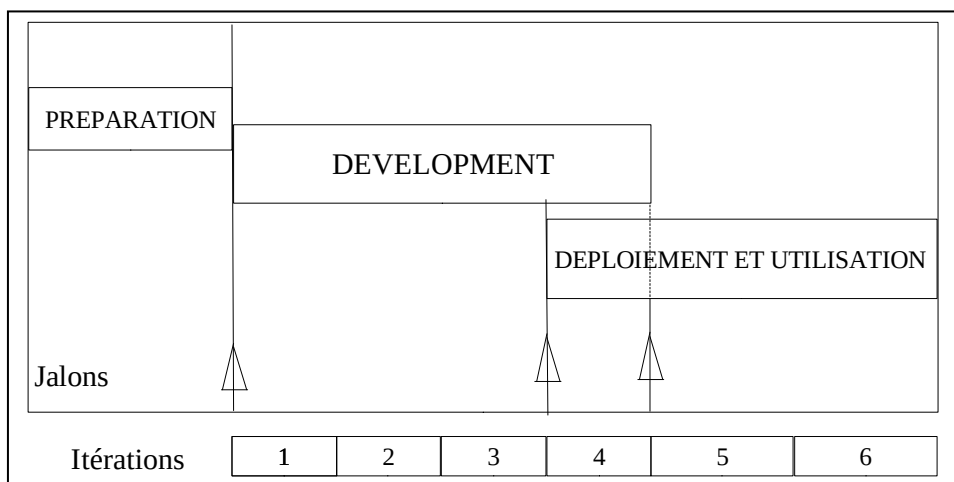
3 - Caractéristiques du développement itératif

3.1 Itérations et phases

La réalisation du projet est découpée en itérations qui se succèdent. Si l'on a distingué plusieurs phases dans le projet, chaque phase de réalisation du projet est découpée en itérations, et les jalons sont placés sur des fins d'itération.

Par exemple dans l'exemple ci-dessous de phasage simplifié de projet, on pourrait découper le développement proprement dit du logiciel en 4 itérations et placer le jalon lancement du déploiement en fin de 3^{-ième} itération. Le logiciel résultant de la 4^{-ième} itération sera utilisé pour les premières installations.

Lorsque le développement du logiciel sera terminé, il y aura des travaux complémentaires à prévoir pour corriger les problèmes relevés lors des installations ou des premiers mois d'utilisation. Cela donnera lieu à de nouvelles itérations (par exemple 2 itérations) qui permettront de livrer des produits corrigés.



3.2 Durée des itérations

Il est recommandé d'avoir la même durée pour toutes les itérations d'une phase du projet. Cependant il est admis que les premières itérations du projet, où il y a un travail de réflexion important pour l'analyse du besoin et la conception de l'architecture, soient plus longues.

La durée à adopter pour une itération dépend essentiellement de la taille de l'équipe et est à ajuster en fonction de l'expérience de l'équipe et des contraintes du projet.

Pour fixer les idées, IBM Rational recommande les durées suivantes dans le cadre du processus itératif RUP (Rational Uniform Process) qu'il commercialise :

Nb de lignes du logiciel	Nombre de développeurs	Durée d'itération
10000	5	1 semaine
50000	15	1 mois
500000	45	6 mois
1000000	100	1 an

Ces durées semblent optimistes pour les petites équipes.

3.3 Déroulement des activités

Toutes les activités peuvent être réalisées de façon incrémentale et les documents correspondants mis à jour et complétés à chaque itération.

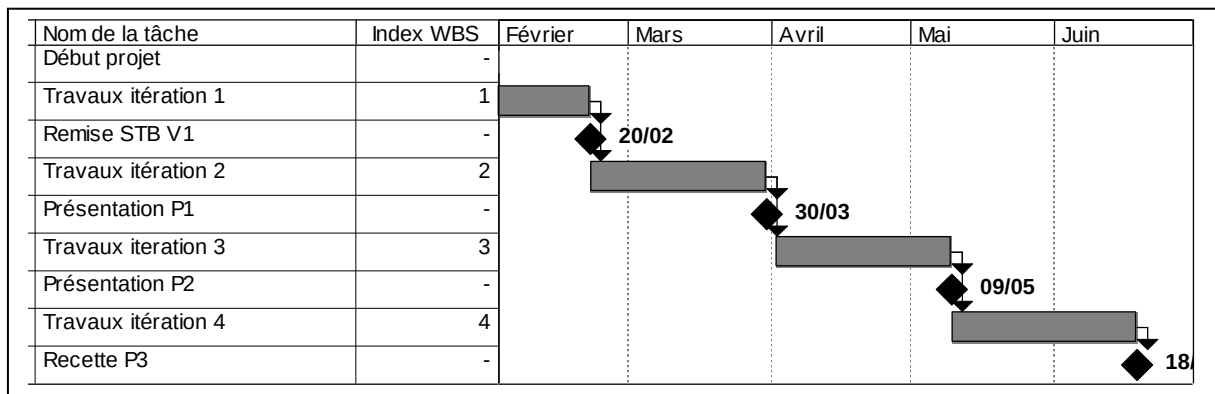
Par exemple pour la spécification technique de besoin, on peut lors de la première itération rédiger les parties générales et faire la liste des fonctions, puis à chaque itération, on met à jour si besoin les parties déjà écrites et on rédige un lot de fonctions. Une solution alternative possible est de rédiger complètement la spécification à la première itération, puis de ne faire que des mises à jour en fonction de remarques ou problèmes rencontrés lors des itérations suivantes.

Chaque itération donne en principe lieu à la réalisation d'un prototype exécutable, sauf éventuellement les premières itérations où on peut se limiter aux travaux d'analyse du besoin et de conception de l'architecture. Chaque prototype donne lieu à des travaux de codage, de tests et d'intégration. Il faut viser à ce que les objets codés pour un prototype puissent être réutilisés sans modifications dans le prototype suivant. De même pour les travaux d'intégration, l'objectif est de pouvoir partir du prototype de l'itération précédente, remplacer les objets modifiés (il est probable qu'il y en aura toujours), puis ajouter les nouveaux objets.

3.4 Planification du développement itératif et plan de développement

Lorsque la durée des itérations sera définie, le calendrier général du projet pourra être fait. Le premier niveau de découpage en tâches fait correspondre une tâche à chaque itération. Le calendrier des tâches de premier niveau montre la suite des itérations. Il faudra placer dans ce calendrier les principaux jalons tels que revues, présentations au client, fourniture de documents importants, qui devront être en principe placés à la fin d'une itération (cela peut conduire à ajuster la durée de certaines itérations).

Exemple de calendrier général

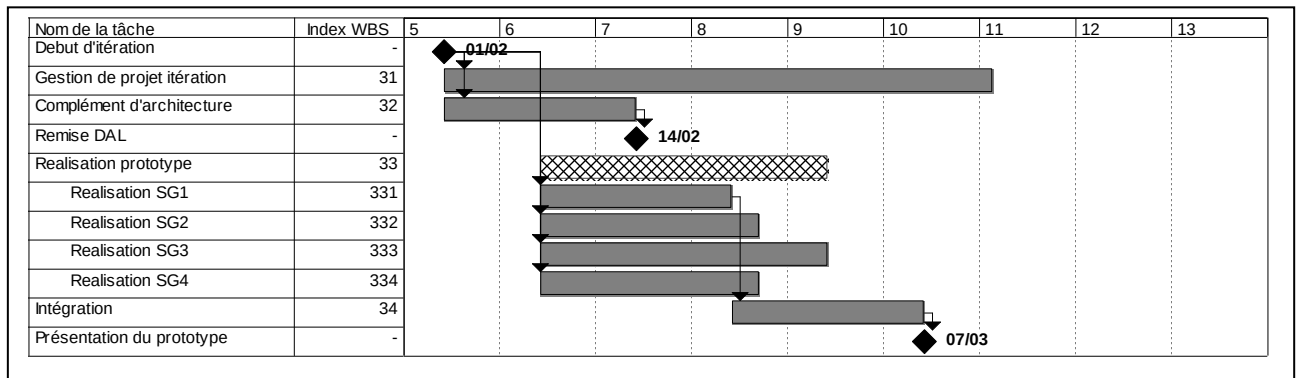


Pour chacune des itérations, il faudra définir l'objectif de l'itération au niveau de l'analyse du besoin, de l'architecture, et du contenu du prototype réalisé.

On pourra ensuite identifier les activités puis les tâches dans chaque itération (tâches de niveau 2), puis planifier les tâches dans chaque itération.

Il est donc préférable pour les plannings de faire un calendrier général puis un planning détaillé par itération. Il n'est pas nécessaire de faire le planning détaillé de toutes les itérations au début du projet, on pourra se limiter aux premières en début de projet et compléter en cours de projet.

Exemple de planning d'une itération



Le plan de développement devra lui aussi être organisé autour des itérations. Il présentera les itérations et l'objectif de chaque itération, et contiendra un plan d'itération pour chaque itération.

Le plan d'itération contient :

- tâches et calendrier,
- produits réalisés,
- contenu des tâches.

Il n'est pas nécessaire que tous les plans d'itération soient rédigés complètement au début du développement, on peut se limiter aux premières itérations.

3.5 Revues

Dans un développement logiciel, des revues placées aux étapes clé du développement sont fortement recommandées pour s'assurer du bon avancement des travaux et valider les travaux déjà faits et les documents produits. Certains clients demandent même à être associés à ces revues, et dans l'industrie spatiale à présider ces revues.

Classiquement dans un développement logiciel, les principales revues étaient :

- la revue de spécification logiciel pour valider la spécification technique de besoin,
- la revue de conception préliminaire pour valider l'architecture,
- la revue de qualification pour s'assurer que le logiciel est prêt à être utilisé.

Dans un développement itératif, on pourra placer une revue à la fin de chaque itération ou à la fin de certaines itérations (si la tenue d'une revue à chaque itération apparaît être une charge trop lourde).

La spécification technique de besoin et le document d'architecture étant rédigés de façon incrémentale, ils ne seront en principe complet qu'à la fin du développement, et les revues concernant ces documents ne pourraient avoir lieu qu'à la fin du développement ce qui ôterait une grande partie de leur intérêt.

Dans la pratique on peut organiser une revue de spécification lorsque la spécification technique de besoins sera considérée comme assez complète (par exemple à la fin de la deuxième itération), et de même au niveau de la conception (par exemple troisième ou quatrième itération), et faire la revue de qualification à la fin de la dernière itération du développement.

On définira donc des revues lors des fins d'itérations pour s'assurer du bon avancement des travaux, et certaines revues auront en plus des objectifs particuliers.

4 - Exemple de développement itératif

Nous allons définir sommairement un exemple et montrer les différents aspects du processus itératif sur cet exemple.

Un client a lancé une consultation pour le développement et le déploiement d'un système dont il fournit le cahier des charges avec les contraintes suivantes :

- recette du logiciel au bout de 10 mois, puis 2 mois pour le déploiement, puis 12 mois de garantie,
- tenue d'une revue de spécification au bout de 3 mois, d'une revue de conception préliminaire avec présentation de maquettes au bout de 5 mois, puis d'une revue de qualification lors de la recette.

Pour être concret, prenons comme système un système informatique permettant de jouer à la belote ou au tarot.

En tant que fournisseur potentiel, vous êtes ravi d'être consulté, et vous commencez à préparer le projet et évaluer la charge de travail, et vos premières conclusions :

- la charge de travail pour les 10 mois du développement correspond à une équipe de 10 personnes,

- les fonctions du produit peuvent être décomposées en 3 groupes de fonctions : fonctions communes aux jeux de cartes (F1), fonctions spécifiques de la belote (F2), fonctions spécifiques des tarots (F3),
- un développement itératif semble la meilleure méthode, sous réserve de possibilités d'ajustement pour les revues et les maquettes.

Vous définissez d'abord la durée des itérations. Pour le développement vous choisissez des itérations d'un mois et demi, les deux premières étant allongées à 2 mois, soit 6 itérations pour 10 mois. Pour le déploiement, l'équipe sera réduite et les itérations réduites à un mois, soit 2 itérations supplémentaires.

Pour la garantie, vous prévoyez la livraison de 2 versions et vous estimez que les modifications seront minimales et que vous pouvez vous limiter à une itération pour chaque version, soit 2 itérations de 6 mois. Si les modifications s'avèrent plus importantes il faudra prévoir plusieurs itérations pour chaque version.

Développement						Déploiement		Garantie	
1	2	3	4	5	6	7	8	9	10
Itérations									

Vous fixez ensuite des objectifs à chaque itération. Le tableau ci-dessous donne les objectifs fixés :

Itération	Analyse du besoin - STB	Architecture	Réalisation du prototype
1	Parties générales + F1	Grandes lignes	-
2	id + F2	Structure d'accueil	Structure d'accueil
3	Complète	id + F1 + F2	id + F1
4	Mise à jour	Complète	id + F2
5	-	Mise à jour	id + F3
6	-	-	Mise à jour + validation
7	-	-	Mise à jour + validation
8	-	-	Mise à jour + validation
9	Complément	Complément	Modification + validation
10	Complément	Complément	Modification + validation

Le prototype de l'itération 2 sera limité à une structure d'accueil (un écran sur chaque poste et le réseau), le prototype de l'itération 3 comprendra en plus les fonctions de base des jeux de cartes (distribuer, montrer la main, jouer la carte...), le prototype de l'itération 4 permet de jouer à la belote, et le prototype de l'itération 5 à la belote et au tarot. A partir de l'itération 6, on n'ajoute plus de fonctions mais on fait des corrections ou des modifications.

On peut en déduire les activités à mener dans chaque itération, en ajoutant aux activités propres de développement les activités transverses (par exemple gestion de projet, assurance qualité...) et les activités complémentaires (par exemple évaluation des outils, achats de matériels...) qui seront détaillées plus loin :

Itération	1	2	3	4	5	6	7	8	9	10
Analyse besoin	X	X	X	X					X	X
Architecture	X	X	X	X	X				X	X
Fabrication		X	X	X	X	X	X	X	X	X
Validation						X	X	X	X	X
Act. transverses	X	X	X	X	X	X	X	X	X	X
Act. compl.	X	X	X	X	X	X	X	X	X	X

Nota

La fabrication recouvre les activités de conception détaillée, codage et essais et intégration.

La revue de spécification logiciel pourra être placée en fin de l'itération 2 au bout de 4 mois, la STB ne sera pas complète mais bien avancée. La revue de conception préliminaire sera proposée à la fin de la 4^{ème} itération. La revue de qualification sera tenue à la fin de la 6^{ème} itération.

A la place des maquettes, on proposera au client de lui montrer les prototypes successifs réalisés à partir du 4^{ème} mois.

5 - Exemple de processus itératif simplifié pour prototype

Reprenons le même problème, mais plaçons maintenant dans le cas où le client a seulement demandé de réaliser un prototype du logiciel en vue de faire des démonstrations ou des expérimentations.

Dans ce cas :

- on peut supprimer l'activité de validation,
- les activités de test peuvent être allégées, les erreurs laissées dans le produit ayant moins d'importance,
- les fonctionnalités peuvent être allégées, certaines fonctionnalités jugées secondaires pouvant être supprimées,
- enfin les performances peuvent être moindres, et on peut utiliser des techniques de réalisation plus faciles à implémenter mais moins performantes.

La charge de travail étant réduite, on peut réduire la taille de l'équipe de réalisation et/ou le délai de réalisation. Le client va demander un délai de réalisation de 7 mois.

La préparation du projet va montrer qu'on peut réaliser le prototype avec une équipe de 8 personnes, et que 4 itérations semble une bonne solution.

Nous ferons 2 itérations de 2 mois et 2 itérations de 1,5 mois.

Mois	1	2	3	4	5	6	7
Itérations	1	2	3	4			

On fixe alors les objectifs des 4 itérations :

Itération	Analyse du besoin - STB	Architecture	Réalisation
1	STB complète	Ebauche	-
2	Mise à jour	DAL complet	Structure d'accueil
3	-	Mise à jour	Id + F1
4	-	-	Id + F2 + F3

On notera que :

- l'analyse du besoin n'est plus faite de façon incrémentale, mais entièrement faite à la première itération avec une simple mise à jour après,
- l'architecture est définie à la deuxième itération après une ébauche à la première itération,
- il n'y a pas de réalisation à la première itération, à la deuxième itération on réalise une structure d'accueil (par exemple l'écran principal et le réseau) sans fonctionnalités applicatives, et les fonctionnalités applicatives sont progressivement ajoutées à la troisième et quatrième itération pour avoir le prototype complet à la fin de la quatrième itération,
- l'activité de validation n'est pas prévue.

On en déduit les activités à réaliser lors de chaque itération :

Itération	1	2	3	4
Analyse du besoin	oui	Mise à jour		
Architecture	oui	oui	Mise à jour	
Fabrication		oui	oui	oui
Activités transverses	oui	oui	oui	oui
Activités complémentaires	oui			oui

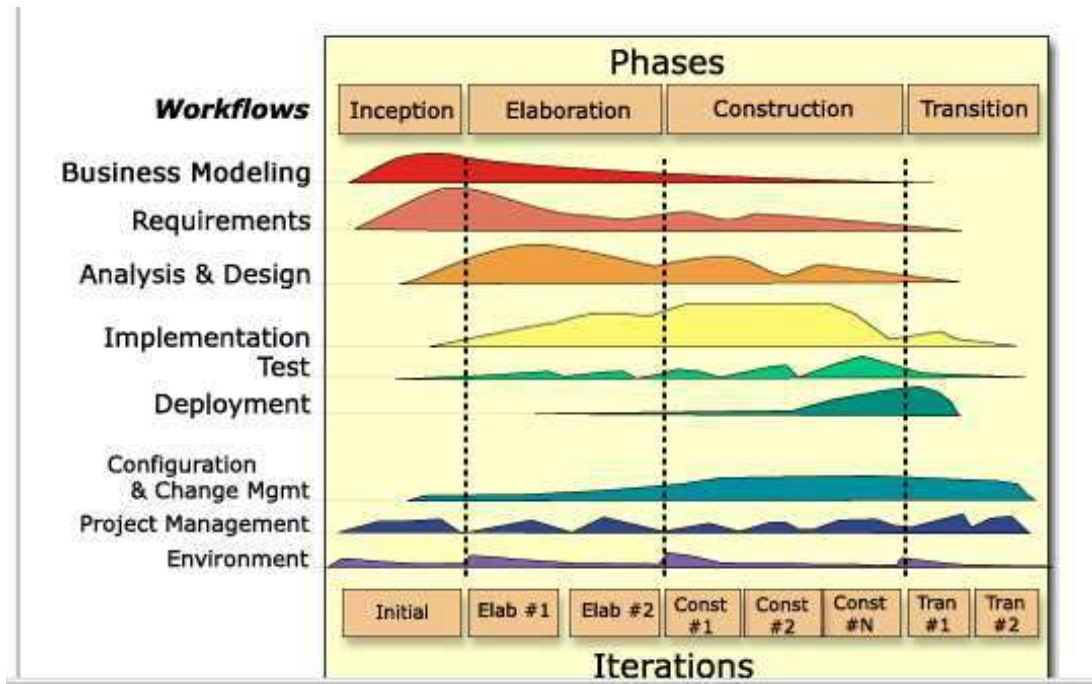
On pourra faire une revue de spécification logiciel à la fin de la première itération, une revue de conception préliminaire à la fin de la deuxième itération, et une revue de validation (bien qu'il n'y ait pas de validation) lors de la recette du prototype à la fin de la quatrième itération.

Les prototypes issus des deuxième et troisième itérations pourront être montrés au client. Le prototype issu de la quatrième itération fera l'objet d'une recette et d'une livraison au client.

6 - Exemple du RUP (Rational Uniform Process)

Un exemple de processus itératif est le RUP (Rational Uniform Process) construit autour de la méthode de conception objet UML (Uniform Method Language) et commercialisé par IBM Rational.

6.1 Contenu du RUP



Ce schéma fait apparaître un phasage du projet :

- phase "inception" où l'on fait le "business modeling", qui correspond en gros au cahier des charges pour une application de gestion, et on commence à rédiger la spécification technique de besoins ("requirements" pour les américains),
- phase "elaboration" où le produit est conçu, et dans laquelle la spécification technique de besoin est complétée et l'essentiel de la conception réalisé,
- phase "construction" où on fabrique le produit de façon itérative jusqu'à obtenir le produit complet,
- phase "transition" où l'on prépare la commercialisation et la diffusion du produit.

Ce phasage semble calqué sur les pratiques des équipes de Rational dont le métier est de réaliser des outils logiciels pour les vendre en quantité par la suite. Il ne distingue pas la préparation du projet de la réalisation, et est assez différent des phasages utilisés habituellement dans un contexte client - fournisseur.

Chacune des phases est découpée en une ou plusieurs itérations en fonction de la durée de la phase et des durées d'itération retenues (liées essentiellement à la taille de l'équipe de développement).

Enfin les "workflows" correspondent aux activités :

- "business modeling" correspond en gros à l'établissement du cahier des charges, qui est généralement considéré comme une activité de préparation de projet et non une activité du projet,
- "requirements" correspond à l'analyse du besoin,
- "analysis and design" correspond à la conception de l'architecture,
- "implementation" regroupe les activités de fabrication : conception détaillée, codage et essais, intégration;
- "tests" correspond principalement à l'activité de validation,
- "configuration and change management" et "project management" correspondent aux activités transverses de gestion de configuration et de gestion de projet,
- "deployment" et "environment" correspondent à des activités complémentaires telles que les installations, les matériels...

Pour chacun de ces "work flow" (activités), le RUP fournit des guides pour la conduite de l'activité et définit les documents à rédiger. Pour chaque document à rédiger il fournit un guide et un canevas à compléter.

De plus, Rational propose avec le RUP un ensemble cohérent d'outils pour la réalisation des activités :

- outil de gestion des exigences de la spécification technique de besoin (Requisite Pro),
- ROSE, outil de modélisation UML et d'aide à la génération de code,
- outils d'aide au test (Purify, Visual Quantify, Visual Pure Coverage, Team test),
- outils de gestion de configuration (Clear Case, Clear Quest),
- outil de documentation (SoDa),
- gestion de projet (MS Project).

6.2 Intérêts et problèmes

Le RUP peut être intéressant si l'on veut adopter un processus itératif. Il présente l'avantage de l'homogénéité puisque toutes les activités sont basées sur UML, tant pour la spécification que la conception.

De plus sont associés au RUP les grands outils de génie logiciel de Rational.

Cependant il y a aujourd'hui un certain nombre de problèmes :

- le RUP est prévu uniquement dans un contexte de type fournisseur investisseur, les relations client - fournisseur ne sont pas abordées,
- le phasage est particulier et n'est pas adapté à toutes les situations, il ne sépare pas nettement la préparation du projet de la réalisation du projet,
- le plan documentaire et les guides de rédaction de documents sont prévus pour des très gros logiciels et sont très complexes.

Le RUP peut donc difficilement être utilisé tel quel et demande pour chaque équipe de développement un gros travail pour l'adapter à son contexte de travail et à la taille de ses projets.

Nota

Les processus itératifs et un certain nombre de guides documentaires ou de plans de documents du présent cours s'inspirent du RUP, mais avec un gros effort de simplification.

7 - Les activités à conduire

Nous allons maintenant aborder la conduite des différentes activités dans un processus itératif.

Nous distinguons 3 types d'activité :

Activités de développement	Analyse du besoin Conception de l'architecture Conception détaillée Codage et tests unitaires Intégration Validation
Activités transverses	Gestion de projet Documentation Gestion de la configuration Assurance qualité
Activités complémentaires	Rédaction du manuel d'utilisation Achat et évaluation des outils Achat et évaluation des matériels Mise en place des plates-formes Entretien des matériels et outils Formation Livraison Installation des matériels et logiciels Recette

Les activités de développement sont les activités du cycle en V logiciel et se retrouveront dans tout ou partie des itérations pour réaliser le logiciel.

Les activités transverses doivent être menées dans toutes les itérations pour gérer le projet et assurer la qualité des produits réalisés.

Les activités complémentaires sont des activités qu'on peut rencontrer couramment dans des projets et qui ne concernent que certains projets et certaines itérations.

Toutes ces activités ne donneront pas lieu chacune à une tâche par itération. Certaines activités ne concernent qu'une ou quelques itérations, et on opérera des regroupements pour ne pas avoir trop de tâches à gérer.

8 - Conduite des activités de développement

8.1 Analyse des exigences du logiciel (spécification)

L'objectif de cette activité est d'approfondir et compléter les exigences du cahier des charges pour arriver aux exigences que devra respecter le logiciel.

Ces travaux conduisent à la rédaction de la spécification technique de besoin (STB) qui présente la mission du logiciel et définit de façon détaillée les besoins des fonctions à assurer.

Dans une méthode itérative ces travaux sont menés de façon incrémentale sur les premières itérations, et la STB correspondante rédigée de façon incrémentale.

Par exemple :

- à la première itération on rédigera les parties "mission du logiciel" et "exigences complémentaires" et on se limitera au plan de la partie détail des fonctions,
- à la deuxième itération, on rédigera quelques fonctions ou cas d'utilisation (méthode UML),
- aux itérations suivantes, on rédige des fonctions ou cas d'utilisation supplémentaires jusqu'à ce que la STB soit complète.

Il faut cependant qu'une fonction soit spécifiée avant d'en effectuer le codage. Il faut donc dans la définition des objectifs des itérations faire en sorte que la spécification d'une fonction soit faite dans une itération antérieure à celle où elle doit être codée, ou au plus tard dans la même itération.

Une solution alternative si le volume des travaux de spécification n'est pas trop important est de rédiger entièrement la STB dans la première itération.

Une nouvelle version de la STB pourra être fournie pour remarques au client ou à de futurs utilisateurs à la fin de chacune des itérations où elle est complétée ou modifiée. On pourra organiser une revue de spécification logiciel à la fin de l'itération où la STB est jugée suffisamment complète.

8.2 Conception de l'architecture du logiciel (conception préliminaire)

L'objectif de cette activité est de trouver une bonne solution d'architecture et découper la réalisation en sous-systèmes réalisables en parallèle de façon quasi indépendante.

La conception de l'architecture peut être faite de façon incrémentale si l'on prend quelques précautions au niveau de l'architecture. Il faudra commencer par définir une structure d'accueil ou base applicative sur laquelle on pourra ajouter progressivement les fonctions d'application. Cette structure d'accueil ou base applicative comprendra par exemple l'affichage d'un écran de base, la prise en compte des actions opérateurs, les échanges avec le réseau, la gestion de la base de données...

On pourra ainsi à la première itération définir l'organisation générale du programme et l'architecture de la structure d'accueil, et compléter aux itérations suivantes avec les fonctions applicatives. Le document d'architecture correspondant sera alors rédigé de façon incrémentale.

Une solution alternative est de consacrer l'essentiel de la première itération à l'analyse des besoins et de n'y faire qu'une ébauche d'architecture, et de consacrer la deuxième itération à l'architecture et la réalisation d'une structure d'accueil.

8.3 Conception détaillée et codage et essais

L'objectif de ces activités est d'identifier les constituants ou objets de chaque sous-système, préciser leur définition, les coder et faire les tests unitaires.

Pour ces travaux, l'équipe de projet est divisée en sous-groupes de réalisation, et chaque sous-groupe se voit confier ces activités de réalisation pour un ou plusieurs sous-systèmes.

Dans un processus itératif, ces activités sont faites à chaque itération, sauf éventuellement lors des premières itérations où l'on ne prévoit pas toujours la réalisation d'un prototype.

Ainsi à chaque itération le sous-groupe responsable d'un sous-système :

- corrige les problèmes identifiés dans les constituants ou objets déjà codés et reprend les tests unitaires de ces constituants ou objets,
- définit, code et fait les tests unitaires des nouveaux constituants ou objets permettant d'assurer les fonctions supplémentaires prévues dans le prototype pour l'itération,
- définit et code les constituants complémentaires nécessaires pour assurer le fonctionnement du prototype (simulation de fonctions absentes).

Il faut se fixer comme objectif qu'une fois qu'un constituant ou objet est codé, il doit pouvoir être réutilisé tel quel (aux corrections d'erreur près) dans toutes les itérations. Sinon cela augmenterait beaucoup la charge de travail.

8.4 Intégration du logiciel

L'intégration du logiciel est à faire à chaque itération où un prototype est réalisé.

L'intégration consiste à rassembler tous les constituants ou objets du prototype et à vérifier le bon fonctionnement de l'ensemble.

A la première itération où un prototype est réalisé, le nombre de constituants ou objets est limité et l'intégration devrait être simple.

Pour les itérations suivantes, on part du prototype qui a été intégré à l'itération précédente, on commence par remplacer les constituants ou objets qui ont été modifiés et on repasse des tests d'intégration pour vérifier que le produit fonctionne toujours correctement. On ajoute alors progressivement les nouveaux constituants ou objets et on passe de nouveaux tests d'intégration, puis on termine par les tests globaux du prototype.

Le nombre de constituants ou objets ajoutés étant limités à chaque itération, l'opération est beaucoup plus simple qu'une intégration globale et les risques d'échec de l'opération bien moindres.

Les anomalies relevées lors des tests d'intégration ne seront corrigées avant de reprendre l'intégration que si elles apparaissent graves ou susceptibles d'empêcher le fonctionnement du prototype de l'itération. Sinon elles seront corrigées à l'itération suivante.

8.5 Validation du logiciel

L'objectif est de vérifier que le logiciel réalisé remplit bien toutes les exigences de la spécification technique de besoins.

La validation est faite lorsque le prototype est complet, en principe à la dernière itération du développement. Par la suite on fera une validation réduite à toutes les itérations qui aboutissent à la livraison du produit en repassant un échantillon des tests de validation.

En fin de développement, il restera des anomalies non corrigées qui donneront lieu à des réserves dans le logiciel livré. On fera des itérations supplémentaires pour réaliser de nouvelles versions corrigeant ces anomalies, soit pendant la phase de déploiement si elle est prévue, soit au titre de la garantie ou de la maintenance.

9 - Conduite des activités transverses

9.1 Gestion du projet

9.1.1 Organisation du projet

En début de projet, à la première itération, on fait l'organisation du projet :

- organisation détaillée de l'équipe (composition de l'équipe de projet et des sous-groupes, responsabilités déléguées),
- définition détaillée de la méthode de développement,
- définition des objectifs et de la durée des itérations,
- planification des travaux,
- choix des méthodes, techniques, outils, moyens...

On inclut aussi en général dans l'organisation du projet la définition des dispositions qualité (plan qualité), la définition de la gestion de configuration, et la définition de la documentation et de sa gestion qui sont présentés plus loin.

Une partie de ces travaux a normalement été faite pendant la préparation du projet. Il faudra les mettre à jour et les compléter.

Si un plan de développement a été rédigé lors de la préparation du projet, il sera mis à jour, sinon il sera rédigé.

Le plan de développement comprend un plan d'itération pour chaque itération précisant les tâches, le calendrier et les produits réalisés. Ces plans d'itération pourront être rédigés au fur et à mesure des itérations.

Le calendrier des tâches est fait itération par itération dans le plan d'itération correspondant. Comme calendrier général, on pourra se limiter à un calendrier montrant les itérations et les principaux jalons.

9.1.2 Suivi de projet et bilan de projet

Les activités de suivi de projet doivent être prévus à chaque itération.

Il s'agit en particulier :

- de suivre l'avancement des tâches et tenir à jour les calendriers,
- d'enregistrer les problèmes rencontrés (techniques ou d'avancement ou de qualité), les analyser, définir des solutions et s'assurer qu'elles sont appliquées,
- de s'assurer que les fournitures sont livrées au client dans les délais prévus.

A la fin du projet on fera un bilan du projet.

On peut également inclure dans le suivi de projet les travaux d'assurance qualité, de gestion de documentation et de gestion de configuration qui sont présentés par la suite.

9.2 Documentation

9.2.1 Plan documentaire

Le développement logiciel donne normalement lieu à de nombreux documents. Chacun de ces documents peut être complété ou mis à jour à chaque itération. Cela fait exploser le nombre de documents. Il faut donc essayer de limiter les documents à produire et bien gérer les documents.

La liste des documents à produire est à définir pour chaque projet en fonction des demandes du client, des caractéristiques de criticité et de sûreté du logiciel, de la durée de vie escomptée du produit, et des objectifs de qualité.

Nous allons donner un plan documentaire assez complet en indiquant dans quelles itérations ces documents sont rédigés (R), complétés (C) ou mis à jour (MAJ) :

Document	It 1	It 2		It		It finale
Plan de développement	R	C		C		C
Plan qualité	R					
Plan de gestion de configuration	R					
Plan de gestion de documentation	R					
Spécification technique de besoin	R	C		MAJ		MAJ
Document d'architecture		R		C		MAJ
Documents de conception détaillée		R		C		
Procédé de génération du code exécutable		R		MAJ		MAJ
Manuel d'utilisation				R		C
Plan de validation		R				
Plan d'intégration		R		C		C
Procédure des tests unitaires		R		C		C
Procédure des tests d'intégration		R		C		C
Procédure des tests de validation				R		
Procédure de livraison						R
Bordereau de livraison						R
Procédure de recette						R

Nous donnons maintenant un plan minimal, les projets courants devraient se situer entre les deux :

Document	It 1	It 2		It		It finale
Calendriers	R	C		C		C
Plan qualité	R					
Spécification technique de besoin	R	MAJ				
Document d'architecture		R		MAJ		
Procédure de recette						R
Manuel d'utilisation				R		C

9.2.2 Gestion de la documentation

Le plan documentaire et les règles de gestion des documents doivent être définis au début du projet lors de la première itération. Selon l'importance du projet, cela peut faire l'objet d'un plan de gestion de la documentation, d'un chapitre du plan de gestion de la configuration ou d'un chapitre du plan qualité.

Ces règles recouvrent l'identification des documents, l'approbation des documents, la présentation des documents, la gestion des éditions successives...

L'activité de gestion de documentation est à prévoir à chaque itération où l'on produit de nouveaux documents ou de nouvelles éditions de documents. Elle peut être regroupée avec les activités de suivi de projet dans des tâches de gestion de projet.

9.3 Gestion de la configuration

La gestion de configuration comprend principalement la gestion des versions des constituants et produits logiciels, et le contrôle des modifications induites par l'équipe de développement ou demandées par le client.

Les règles de gestion de la configuration doivent être définies en début de projet à la première itération. Selon l'importance du projet elles sont consignées dans un plan de gestion de la configuration ou dans un chapitre du plan qualité.

L'activité de gestion de la configuration est une activité permanente pendant toute la durée du projet et se retrouve à toutes les itérations. Elle peut être regroupée avec les activités de suivi de projet dans des tâches de gestion de projet.

9.4 Assurance qualité

L'assurance qualité recouvre un ensemble de dispositions préétablies et systématiques mises en œuvre pour donner des garanties suffisantes que le produit satisfera aux exigences du client.

Pour un logiciel ces dispositions peuvent être l'exécution de contrôles sur le logiciel et sa documentation, la rédaction de certains documents, la tenue de revues...

Ces dispositions doivent être définies au début du développement à la première itération dans un plan qualité.

Ces dispositions s'appliquent pendant tout le projet et donneront des travaux à chaque itération. Cependant certains travaux font partie du développement (par exemple rédaction d'une spécification technique de besoin) et se retrouveront dans les tâches de développement. Les autres travaux, en particulier l'activité de l'ingénieur qualité, pourront être regroupés avec le suivi du projet dans les tâches de gestion de projet.

10 - Conduite des activités complémentaires

10.1 Rédaction du manuel d'utilisation

Le manuel d'utilisation fournit à l'utilisateur toutes les informations nécessaires pour réinstaller son logiciel et l'utiliser.

Le manuel comporte des informations qui ne seront connues que lors des dernières itérations telles que certains dialogues, certains messages d'erreur...

Pour que le client ne découvre pas le manuel d'utilisation en tout fin de projet, il est recommandé de procéder en deux étapes :

- faire d'abord un projet de manuel lors d'une itération dès que la STB est complète et le soumettre pour avis au client,
- mettre à jour et compléter le manuel à la dernière itération de développement.

Le travail de rédaction du manuel d'utilisation est usuellement rattaché à l'activité d'analyse du besoin ou à l'activité de conception de l'architecture.

10.2 Achat et évaluation des outils

Un projet logiciel fait appel à de nombreux outils logiciel : outil de planification, outil de modélisation, outil d'aide au test, générateurs d'interface homme machine, ateliers logiciels et bibliothèques de logiciel ...

Si l'équipe de projet n'a pas encore utilisé ces outils, il faut prévoir en début de projet lors des premières itérations une évaluation de ces outils pour vérifier qu'ils rendent bien les services qu'on attend d'eux. Cela peut être fait en réalisant des petites maquettes avec ces outils.

Enfin si l'on ne dispose pas de licences de ces outils ou bibliothèques, il faudra prévoir leur achat.

10.3 Achat et évaluation des matériels

Si le projet comporte la fourniture de matériels avec le logiciel, il faut planifier les achats de matériel en tenant compte des temps de passage de commande et de livraison. Même quand le projet ne demande pas de livrer des matériels, il peut y avoir besoin d'acheter des matériels pour les besoins du développement.

Compte tenu de l'évolution très rapide des matériels informatiques, il est recommandé de faire les achats informatiques le plus tard possible compte tenu du calendrier du projet, et de n'acheter en début de projet que les matériels nécessaires pour les premières tâches du projet.

Lorsque les matériels achetés seront livrés, on pourra prévoir des travaux d'évaluation pour vérifier que le matériel répond bien aux besoins du projet.

10.4 Mise en place des plates-formes

Le développement d'un logiciel nécessite des plates-formes de production (pour les activités de codage et essais), une ou plusieurs plates-formes d'intégration pour les activités d'intégration, et éventuellement des plates-formes distinctes pour la validation.

La mise en place des plates-formes de production (matériels et outils logiciels) est à prévoir à la première itération en début de projet.

La mise en place des plates-formes d'intégration est à prévoir dès la première itération où un prototype est réalisé.

Les plates-formes de validation, si elles sont particulières doivent être mis en place avant le début des travaux de validation.

10.5 Entretien des matériels et outils

Pendant la durée du projet les matériels peuvent tomber en panne, de nouvelles versions d'outils logiciels peuvent apparaître...

Il ne faut donc pas oublier dans le projet de prendre en compte les problèmes de maintenance et de mise à jour.

10.6 Formation

Si le personnel n'est pas formé aux techniques ou outils retenus, il faudra prévoir leur formation au début du projet.

10.7 Livraison

Le projet demande en général la livraison du logiciel en fin de développement.

Il faut alors définir le support de livraison en accord avec le client et, lors de la dernière itération de développement, rédiger et valider la procédure de livraison.

En fin de l'itération, le support de livraison sera généré. Il servira pour la recette et les installations, et sera fourni au client.

Le support de livraison sera accompagné d'un bordereau de livraison qui donnera l'identification du logiciel et quelques informations sur le logiciel.

10.8 Installation des matériels et logiciels

Certains projets comportent l'installation chez le client de logiciels avec éventuellement les matériels associés.

Le projet comportera alors une phase de déploiement pendant laquelle seront faites ces installations.

L'installation du logiciel sera préparée pendant la dernière itération du développement en rédigeant puis validant la procédure d'installation.

Il faudra aussi préparer un support de livraison qui sera fourni aux installateurs et au client par la suite. Il faudra donc procéder comme indiqué ci-dessus.

Des anomalies restent dans le logiciel en fin de développement, et de nouvelles peuvent être découvertes lors des installations. Il est donc recommandé de prévoir pendant le déploiement de prévoir des itérations courtes pour corriger les anomalies découvertes et laisser au client en fin de déploiement des logiciels ayant un minimum de défauts.

10.9 Recette

On prévoit en général une recette du logiciel en fin de développement et à chaque installation pour permettre les paiements des lots du contrat correspondant à ces travaux.

Pendant une recette, on ne fait qu'un nombre limité d'essais et cela ne suffit pas à démontrer que le logiciel répond au besoin. La garantie que le logiciel répond au besoin est fournie par les dispositions qualité et les essais de validation.

Une procédure de recette doit être rédigée avant la recette et soumise au client pour accord.

La recette se passe en présence du client et consiste à charger le logiciel à partir du support de livraison et suivant la procédure d'installation ou les consignes d'installation du manuel d'utilisation, puis à effectuer les essais prévus dans la procédure de recette en consignait les résultats.

Selon les demandes du client, la recette peut être faite chez le développeur ou chez le client, ou partagée entre les deux. Dans tous les cas il est fortement recommandé que l'équipe de développement valide le déroulement de la recette lors d'une pré-recette avant de présenter la recette au client.

11 - Conclusions

Le développement itératif est une solution intéressante et recommandée pour développer les logiciels lorsqu'on n'a pas de contraintes de criticité, de sûreté ou d'optimisation poussée.

Il permet de réduire considérablement les risques d'échec du projet et de dérapage des délais.

Cependant c'est une technique délicate, et il faut prendre beaucoup de précautions dans la mise en œuvre pour diminuer effectivement les risques et ne pas augmenter sensiblement la charge de travail (par exemple si on en arrive à réécrire tout le logiciel à chaque itération).